Studies in Computational Intelligence 1141

Hocine Cherifi
Luis M. Rocha
Chantal Cherifi
Murat Donduran *Editors*

Complex Networks & Their Applications XII

Proceedings of The Twelfth International Conference on Complex Networks and their Applications: COMPLEX NETWORKS 2023 Volume 1

Springer

Studies in Computational Intelligence **1141**

Series Editor

Janusz Kacprzyk, *Polish Academy of Sciences, Warsaw, Poland*

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

Indexed by SCOPUS, DBLP, WTI Frankfurt eG, zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

Hocine Cherifi · Luis M. Rocha ·
Chantal Cherifi · Murat Donduran
Editors

Complex Networks & Their Applications XII

Proceedings of The Twelfth International Conference on Complex Networks and their Applications: COMPLEX NETWORKS 2023
Volume 1

Springer

Editors
Hocine Cherifi
University of Burgundy
Dijon Cedex, France

Chantal Cherifi
IUT Lumière - Université Lyon 2
University of Lyon
Bron, France

Luis M. Rocha
Thomas J. Watson College of Engineering
and Applied Sciences
Binghamton University
Binghamton, NY, USA

Murat Donduran
Department of Economics
Yildiz Technical University
Istanbul, Türkiye

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-031-53467-6 ISBN 978-3-031-53468-3 (eBook)
https://doi.org/10.1007/978-3-031-53468-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Preface

Dear Colleagues, Participants, and Readers,

We present the 12th Complex Networks Conference proceedings with great pleasure and enthusiasm. Like its predecessors, this edition proves complex network research's ever-growing significance and interdisciplinary nature. As we navigate the intricate web of connections that define our world, understanding complex systems, their emergent properties, and the underlying structures that govern them has become increasingly crucial.

The Complex Networks Conference has established itself as a pivotal platform for researchers, scholars, and experts from various fields to converge, exchange ideas, and push the boundaries of knowledge in this captivating domain. Over the past twelve years, we have witnessed remarkable progress, breakthroughs, and paradigm shifts highlighting the dynamic and complex tapestry of networks surrounding us, from biological systems and social interactions to technological infrastructures and economic networks.

This year's conference brought together an exceptional cohort of experts, including our keynote speakers:

- Michael Bronstein, University of Oxford, UK, enlightened us on "Physics-inspired Graph Neural Networks"
- Kathleen Carley, Carnegie Mellon University, USA, explored "Coupling in High Dimensional Networks"
- Manlio De Domenico, University of Padua, Italy, introduced "An Emerging Framework for the Functional Analysis of Complex Interconnected Systems"
- Danai Koutra, University of Michigan, USA, shared insights on "Advances in Graph Neural Networks: Heterophily and Beyond"
- Romualdo Pastor-Satorras, UPC, Spain, discussed "Opinion Depolarization in Interdependent Topics and the Effects of Heterogeneous Social Interactions"
- Tao Zhou, USTC, China, engaged us in "Recent Debates in Link Prediction"

These renowned experts addressed a spectrum of critical topics and the latest methodological advances, underscoring the continued expansion of this field into ever more domains.

We were also fortunate to benefit from the expertise of our tutorial speakers on November 27, 2023:

- Tiago de Paula Peixoto, CEU Vienna, Austria, guided "Network Inference and Reconstruction"
- Maria Liakata, Queen Mary University of London, UK, led us through "Longitudinal language processing from user-generated content"

We want to express our deepest gratitude to all the authors, presenters, reviewers, and attendees who have dedicated their time, expertise, and enthusiasm to make this event successful. The peer-review process, a cornerstone of scientific quality, ensures

that the papers in these proceedings have undergone rigorous evaluation, resulting in high-quality contributions.

We encourage you to explore the rich tapestry of knowledge and ideas as we dive into these four proceedings volumes. The papers presented here represent not only the diverse areas of research but also the collaborative and interdisciplinary spirit that defines the complex networks community.

In closing, we extend our heartfelt thanks to the organizing committees and volunteers who have worked tirelessly to make this conference a reality. We hope these proceedings inspire future research, innovation, and collaboration, ultimately helping us better understand the world's networks and their profound impacts on science, technology, and society.

We hope that the pleasure you have reading these papers matches our enthusiasm for organizing the conference and assembling this collection of articles.

Hocine Cherifi
Luis M. Rocha
Chantal Cherifi
Murat Donduran

Organization and Committees

General Chairs

Hocine Cherifi	University of Burgundy, France
Luis M. Rocha	Binghamton University, USA

Advisory Board

Jon Crowcroft	University of Cambridge, UK
Raissa D'Souza	Univ. of California, Davis, USA
Eugene Stanley	Boston University, USA
Ben Y. Zhao	University of Chicago, USA

Program Chairs

Chantal Cherifi	University of Lyon, France
Murat Donduran	Yildiz Technical University, Turkey

Lightning Chairs

Konstantin Avrachenkov	Inria Université Côte d'Azur, France
Mathieu Desroches	Inria Université Côte d'Azur, France
Huijuan Wang	TU Delft, Netherlands

Poster Chairs

Christophe Crespelle	Université Côte d'Azur, France
Manuel Marques Pita	Universidade Lusófona, Portugal
Laura Ricci	University of Pisa, Italy

Special Issues Chair

Sabrina Gaito	University of Milan, Italy

Publicity Chairs

Fabian Braesemann	University of Oxford, UK
Zachary Neal	Michigan State University, USA
Xiangjie Kong	Dalian University of Technology, China

Tutorial Chairs

Luca Maria Aiello	Nokia-Bell Labs, UK
Leto Peel	Maastricht University, Netherlands

Social Media Chair

Brennan Klein	Northeastern University, USA

Sponsor Chairs

Roberto Interdonato	CIRAD - UMR TETIS, France
Christophe Cruz	University of Burgundy, France

Sustainability Chair

Madeleine Aurelle	City School International De Ferney-Voltaire, France

Local Committee Chair

Charlie Joyez	Université Côte d'Azur, France

Publication Chair

Matteo Zignani	University of Milan, Italy

Submission Chair

Cheick Ba	Queen Mary University of London, UK

Web Chairs

Stephany Rajeh	Sorbonne University, France
Alessia Galdeman	University of Milan, Italy

Program Committee

Jacobo Aguirre	Centro de Astrobiología (CAB), Spain
Luca Maria Aiello	ITU Copenhagen, Denmark
Esra Akbas	Georgia State University, USA
Sinan G. Aksoy	Pacific Northwest National Laboratory, USA
Mehmet Aktas	Georgia State University, USA
Tatsuya Akutsu	Kyoto University, Japan
Reka Albert	Pennsylvania State University, USA
Alberto Aleta	University of Zaragoza, Spain
Claudio Altafini	Linkoping University, Sweden
Viviana Amati	University of Milano-Bicocca, Unknown
Frederic Amblard	Université Toulouse 1 Capitole, IRIT, France
Enrico Amico	EPFL, Switzerland
Yuri Antonacci	University of Palermo, Italy
Alberto Antonioni	Carlos III University of Madrid, Spain
Nino Antulov-Fantulin	ETH Zurich, Switzerland
Mehrnaz Anvari	Fraunhofer SCAI, Germany
David Aparicio	Zendesk, Portugal
Nuno Araujo	Univ. de Lisboa, Portugal
Panos Argyrakis	Aristotle University of Thessaloniki, Greece
Oriol Artime	University of Barcelona, Spain
Malbor Asllani	Florida State University, USA
Tomaso Aste	University College London, UK
Martin Atzmueller	Osnabrück University & DFKI, Germany
Konstantin Avrachenkov	Inria Sophia-Antipolis, France

Giacomo Baggio	University of Padova, Italy
Franco Bagnoli	Università di Firenze, Italy
James Bagrow	University of Vermont, USA
Yiguang Bai	Xidian University, China
Sven Banisch	Karlsruhe Institute of Technology, Germany
Annalisa Barla	Università degli Studi di Genova, Italy
Nikita Basov	The University of Manchester, UK
Anais Baudot	CNRS, AMU, France
Gareth J. Baxter	University of Aveiro, Portugal
Loredana Bellantuono	University of Bari Aldo Moro, Italy
Andras Benczur	SZTAKI, Hungary
Rosa M. Benito	Universidad Politécnica de Madrid, Spain
Ginestra Bianconi	Queen Mary University of London, UK
Ofer Biham	The Hebrew University, Israel
Romain Billot	IMT Atlantique, France
Livio Bioglio	University of Turin, Italy
Hanjo D. Boekhout	Leiden University, Netherlands
Anthony Bonato	Toronto Metropolitan University, Canada
Anton Borg	Blekinge Institute of Technology, Sweden
Cecile Bothorel	IMT Atlantique, France
Federico Botta	University of Exeter, UK
Romain Bourqui	University of Bordeaux, France
Alexandre Bovet	University of Zurich, Switzerland
Dan Braha	New England Complex Systems Institute, USA
Ulrik Brandes	ETH Zürich, Switzerland
Rion Brattig Correia	Instituto Gulbenkian de Ciência, Portugal
Chico Camargo	University of Exeter, UK
Gian Maria Campedelli	Fondazione Bruno Kessler, Italy
M. Abdullah Canbaz	University at Albany SUNY, USA
Vincenza Carchiolo	DIEEI, Italy
Dino Carpentras	ETH Zürich, Switzerland
Giona Casiraghi	ETH Zürich, Switzerland
Douglas Castilho	Federal Inst. of South of Minas Gerais, Brazil
Costanza Catalano	University of Florence, Italy
Lucia Cavallaro	Free University of Bozen/Bolzano, Italy
Remy Cazabet	University of Lyon, France
Jianrui Chen	Shaanxi Normal University, China
Po-An Chen	National Yang Ming Chiao Tung Univ., Taiwan
Xihui Chen	University of Luxembourg, Luxembourg
Sang Chin	Boston University, USA
Daniela Cialfi	Institute for Complex Systems, Italy
Giulio Cimini	University of Rome Tor Vergata, Italy

Matteo Cinelli	Sapienza University of Rome, Italy
Salvatore Citraro	University of Pisa, Italy
Jonathan Clarke	Imperial College London, UK
Richard Clegg	QMUL, UK
Reuven Cohen	Bar-Ilan University, Israel
Jean-Paul Comet	Université Côte d'Azur, France
Marco Coraggio	Scuola Superiore Meridionale, Italy
Michele Coscia	ITU Copenhagen, Denmark
Christophe Crespelle	Université Côte d'Azur, France
Regino H. Criado Herrero	Universidad Rey Juan Carlos, Spain
Marcelo V. Cunha	Instituto Federal da Bahia, Brazil
David Soriano-Paños	Instituto Gulbenkian de Ciência, Portugal
Joern Davidsen	University of Calgary, Canada
Toby Davies	University of Leeds, UK
Caterina De Bacco	Max Planck Inst. for Intelligent Systems, Germany
Pietro De Lellis	University of Naples Federico II, Italy
Pasquale De Meo	University of Messina, Italy
Domenico De Stefano	University of Trieste, Italy
Fabrizio De Vico Fallani	Inria-ICM, France
Charo I. del Genio	Coventry University, UK
Robin Delabays	HES-SO, Switzerland
Yong Deng	Univ. of Electronic Science and Tech., China
Mathieu Desroches	Inria Centre at Université Côte d'Azur, France
Carl P. Dettmann	University of Bristol, UK
Zengru Di	Beijing Normal University, China
Riccardo Di Clemente	Northeastern University London, UK
Branco Di Fátima	University of Beira Interior (UBI), Portugal
Alessandro Di Stefano	Teesside University, UK
Ming Dong	Central China Normal University, China
Constantine Dovrolis	Georgia Tech, USA
Maximilien Dreveton	EPFL, Switzerland
Ahlem Drif	University of Setif, Algeria
Johan L. Dubbeldam	Delft University of Technology, Netherlands
Jordi Duch	Universitat Rovira i Virgili, Spain
Cesar Ducruet	CNRS, France
Mohammed El Hassouni	Mohammed V University in Rabat, Morocco
Frank Emmert-Streib	Tampere University, Finland
Gunes Ercal	Southern Illinois University Edwardsville, USA
Alejandro Espinosa-Rada	ETH Zürich, Switzerland
Alexandre Evsukoff	Universidade Federal do Rio de Janeiro, Brazil
Mauro Faccin	University of Bologna, Italy

Max Falkenberg	City University, UK
Guilherme Ferraz de Arruda	CENTAI Institute, Italy
Andrea Flori	Politecnico di Milano, Italy
Manuel Foerster	Bielefeld University, Germany
Emma Fraxanet Morales	Pompeu Fabra University, Spain
Angelo Furno	LICIT-ECO7, France
Sergio Gómez	Universitat Rovira i Virgili, Spain
Sabrina Gaito	Università degli Studi di Milano, Italy
José Manuel Galán	Universidad de Burgos, Spain
Alessandro Galeazzi	Ca' Foscari university of Venice, Italy
Lazaros K. Gallos	Rutgers University, USA
Joao Gama	INESC TEC—LIAAD, Portugal
Jianxi Gao	Rensselaer Polytechnic Institute, USA
David Garcia	University of Konstanz, Germany
Floriana Gargiulo	CNRS, France
Michael T. Gastner	Singapore Institute of Technology, Singapore
Alexander Gates	University of Virginia, USA
Alexandra M. Gerbasi	Exeter Business School, UK
Fakhteh Ghanbarnejad	Potsdam Inst. for Climate Impact Res., Germany
Cheol-Min Ghim	Ulsan National Inst. of Science and Tech., South Korea
Tommaso Gili	IMT School for Advanced Studies Lucca, Italy
Silvia Giordano	Univ. of Applied Sciences of Southern Switzerland, Switzerland
Rosalba Giugno	University of Verona, Italy
Kimberly Glass	Brigham and Women's Hospital, USA
David Gleich	Purdue University, USA
Antonia Godoy Lorite	UCL, UK
Kwang-Il Goh	Korea University, South Korea
Carlos Gracia	University of Zaragoza, Spain
Oscar M. Granados	Universidad Jorge Tadeo Lozano, Colombia
Michel Grossetti	CNRS, France
Guillaume Guerard	ESILV, France
Jean-Loup Guillaume	Université de la Rochelle, France
Furkan Gursoy	Bogazici University, Turkey
Philipp Hövel	Saarland University, Germany
Meesoon Ha	Chosun University, South Korea
Bianca H. Habermann	AMU, CNRS, IBDM UMR 7288, France
Chris Hankin	Imperial College London, UK
Yukio Hayashi	JAIST, Japan
Marina Hennig	Johannes Gutenberg University of Mainz, Germany

Takayuki Hiraoka	Aalto University, Finland
Marion Hoffman	Institute for Advanced Study in Toulouse, France
Bernie Hogan	University of Oxford, UK
Seok-Hee Hong	University of Sydney, Australia
Yujie Hu	University of Florida, USA
Flavio Iannelli	UZH, Switzerland
Yuichi Ikeda	Kyoto University, Japan
Roberto Interdonato	CIRAD, France
Antonio Iovanella	Univ. degli Studi Internazionali di Roma, Italy
Arkadiusz Jędrzejewski	CY Cergy Paris Université, France
Tao Jia	Southwest University, China
Jiaojiao Jiang	UNSW Sydney, Australia
Di Jin	University of Michigan, USA
Ivan Jokifá	Technology University of Delft, Netherlands
Charlie Joyez	GREDEG, Université Côte d'Azur, France
Bogumil Kamiński	SGH Warsaw School of Economics, Poland
Marton Karsai	Central European University, Austria
Eytan Katzav	Hebrew University of Jerusalem, Israel
Mehmet Kaya	Firat University, Turkey
Domokos Kelen	SZTAKI, Hungary
Mohammad Khansari	Sharif University of Technology, Iran
Jinseok Kim	University of Michigan, USA
Pan-Jun Kim	Hong Kong Baptist University, Hong Kong
Maksim Kitsak	TU Delft, Netherlands
Mikko Kivelä	Aalto University, Finland
Brennan Klein	Northeastern University, UK
Konstantin Klemm	IFISC (CSIC-UIB), Spain
Xiangjie Kong	Zhejiang University of Technology, China
Onerva Korhonen	University of Eastern Finland, Finland
Miklós Krész	InnoRenew CoE, Slovenia
Prosenjit Kundu	DA-IICT, Gandhinagar, Gujarat, India
Haewoon Kwak	Indiana University Bloomington, USA
Richard La	University of Maryland, USA
Josè Lages	Université de Franche-Comté, France
Renaud Lambiotte	University of Oxford, UK
Aniello Lampo	UC3M, Spain
Jennifer Larson	Vanderbilt University, USA
Paul J. Laurienti	Wake Forest, USA
Anna T. Lawniczak	University of Guelph, Canada
Deok-Sun Lee	KIAS, South Korea
Harlin Lee	Univ. of North Carolina at Chapel Hill, USA
Juergen Lerner	University of Konstanz, Germany

Lasse Leskelä	Aalto University, Finland
Petri Leskinen	Aalto University/SeCo, Finland
Inmaculada Leyva	Universidad Rey Juan Carlos, Spain
Cong Li	Fudan University, China
Longjie Li	Lanzhou University, China
Ruiqi Li	Beijing Univ. of Chemical Technology, China
Xiangtao Li	Jilin University, China
Hao Liao	Shenzhen University, China
Fabrizio Lillo	Università di Bologna, Italy
Giacomo Livan	University of Pavia, Italy
Giosue' Lo Bosco	Università di Palermo, Italy
Hao Long	Jiangxi Normal University, China
Juan Carlos Losada	Universidad Politécnica de Madrid, Spain
Laura Lotero	Universidad Nacional de Colombia, Colombia
Yang Lou	National Yang Ming Chiao Tung Univ., Taiwan
Meilian Lu	Beijing Univ. of Posts and Telecom., China
Maxime Lucas	CENTAI, Italy
Lorenzo Lucchini	Bocconi University, Italy
Hanbaek Lyu	UW-Madison, USA
Vince Lyzinski	University of Maryland, College Park, USA
Morten Mørup	Technical University of Denmark, Denmark
Leonardo Maccari	Ca'Foscari University of Venice, Italy
Matteo Magnani	Uppsala University, Sweden
Maria Malek	CY Cergy Paris University, France
Giuseppe Mangioni	University of Catania, Italy
Andrea Mannocci	CNR-ISTI, Italy
Rosario N. Mantegna	University of Palermo, Italy
Manuel Sebastian Mariani	University of Zurich, Switzerland
Radek Marik	CTU in Prague, Czech Republic
Daniele Marinazzo	Ghent University, Belgium
Andrea Marino	University of Florence, Italy
Malvina Marku	INSERM, CRCT, France
Antonio G. Marques	King Juan Carlos University, Spain
Christoph Martin	Hamburg University of Applied Sciences, Germany
Samuel Martin-Gutierrez	Complexity Science Hub Vienna, Austria
Cristina Masoller	Universitat Politecnica de Catalunya, Spain
Rossana Mastrandrea	IMT School for Advanced Studies, Italy
John D. Matta	Southern Illinois Univ. Edwardsville, USA
Carolina Mattsson	CENTAI Institute, Italy
Fintan McGee	Luxembourg IST, Luxembourg
Matus Medo	University of Bern, Switzerland

Ronaldo Menezes	University of Exeter, UK
Humphrey Mensah	Epsilon Data Management, LLC, USA
Anke Meyer-Baese	Florida State University, USA
Salvatore Micciche	UNIPA DiFC, Italy
Letizia Milli	University of Pisa, Italy
Marija Mitrovic	Institute of Physics Belgrade, Serbia
Andrzej Mizera	University of Warsaw, Poland
Chiara Mocenni	University of Siena, Italy
Roland Molontay	Budapest UTE, Hungary
Sifat Afroj Moon	University of Virginia, USA
Alfredo Morales	MIT, USA
Andres Moreira	UTFSM, Chile
Greg Morrison	University of Houston, USA
Igor Mozetic	Jozef Stefan Institute, Slovenia
Sarah Muldoon	State University of New York, Buffalo, USA
Tsuyoshi Murata	Tokyo Institute of Technology, Japan
Jose Nacher	Toho University, Japan
Nishit Narang	NIT Delhi, India
Filipi Nascimento Silva	Indiana University, USA
Muaz A. Niazi	National Univ. of Science & Technology, Pakistan
Peter Niemeyer	Leuphana University Lueneburg, Germany
Jordi Nin	ESADE, Universitat Ramon Llull, Spain
Rogier Noldus	Ericsson, Netherlands
Masaki Ogura	Osaka University, Japan
Andrea Omicini	Università di Bologna, Italy
Gergely Palla	Eötvös University, Hungary
Fragkiskos Papadopoulos	Cyprus University of Technology, Cyprus
Symeon Papadopoulos	Centre for Research & Technology, Greece
Alice Patania	University of Vermont, USA
Leto Peel	Maastricht University, Netherlands
Hernane B. B. Pereira	Senai Cimatec, Brazil
Josep Perelló	Universitat de Barcelona, Spain
Anthony Perez	Université d'Orléans, France
Juergen Pfeffer	Technical University of Munich, Germany
Carlo Piccardi	Politecnico di Milano, Italy
Pietro Hiram Guzzi	Univ. Magna Gracia of Catanzaro, Italy
Yoann Pigné	Université Le Havre Normandie, France
Bruno Pinaud	University of Bordeaux, France
Flavio L. Pinheiro	Universidade Nova de Lisboa, Portugal
Manuel Pita	Universidade Lusófona, Portugal
Clara Pizzuti	CNR-ICAR, Italy

Jan Platos	VSB - Technical University of Ostrava, Czech Republic
Pawel Pralat	Toronto Metropolitan University, Canada
Rafael Prieto-Curiel	Complexity Science Hub, Austria
Daniele Proverbio	University of Trento, Italy
Giulia Pullano	Georgetown University, USA
Rami Puzis	Ben-Gurion University of the Negev, Israel
Christian Quadri	Università degli Studi di Milano, Italy
Hamid R. Rabiee	Sharif University of Technology, Iran
Filippo Radicchi	Indiana University, USA
Giancarlo Ragozini	University of Naples Federico II, Italy
Juste Raimbault	IGN-ENSG, France
Sarah Rajtmajer	Penn State, USA
Gesine D. Reinert	University of Oxford, UK
Élisabeth Remy	Institut de Mathématiques de Marseille, France
Xiao-Long Ren	Univ. of Electronic Science and Tech., China
Laura Ricci	University of Pisa, Italy
Albano Rikani	INSERM, France
Luis M. Rocha	Binghamton University, USA
Luis E. C. Rocha	Ghent University, Belgium
Fernando E. Rosas	Imperial College London, UK
Giulio Rossetti	CNR-ISTI, Italy
Camille Roth	CNRS/CMB/EHESS, France
Celine Rozenblat	UNIL, Switzerland
Giancarlo Ruffo	Univ. degli Studi del Piemonte Orientale, Italy
Arnaud Sallaberry	University of Montpellier, France
Hillel Sanhedrai	Northeastern University, USA
Iraj Saniee	Bell Labs, Nokia, USA
Antonio Scala	CNR Institute for Complex Systems, Italy
Michael T. Schaub	RWTH Aachen University, Germany
Irene Sendiña-Nadal	Universidad Rey Juan Carlos, Spain
Mattia Sensi	Politecnico di Torino, Italy
Ke-ke Shang	Nanjing University, China
Julian Sienkiewicz	Warsaw University of Technology, Poland
Per Sebastian Skardal	Trinity College, Ireland
Fiona Skerman	Uppsala University, Sweden
Oskar Skibski	University of Warsaw, Poland
Keith M. Smith	University of Strathclyde, UK
Igor Smolyarenko	Brunel University, UK
Zbigniew Smoreda	Orange Innovation, France
Annalisa Socievole	ICAR-CNR, Italy
Igor M. Sokolov	Humboldt University Berlin, Germany

Albert Solé-Ribalta	Universitat Oberta de Catalunya, Spain
Sara Sottile	University of Trento, Italy
Sucheta Soundarajan	Syracuse University, USA
Jaya Sreevalsan-Nair	IIIT Bangalore, India
Christoph Stadtfeld	ETH Zürich, Switzerland
Clara Stegehuis	University of Twente, Netherlands
Lovro Šubelj	University of Ljubljana, Slovenia
Xiaoqian Sun	Beihang University, China
Michael Szell	IT University of Copenhagen, Denmark
Boleslaw Szymanski	Rensselaer Polytechnic Institute, USA
Andrea Tagarelli	University of Calabria, Italy
Kazuhiro Takemoto	Kyushu Institute of Technology, Japan
Frank W. Takes	Leiden University, Netherlands
Fabien Tarissan	CNRS & ENS Paris-Saclay, France
Laura Temime	Cnam, France
François Théberge	TIMC, France
Guy Theraulaz	Université Paul Sabatier and CNRS, France
I-Hsien Ting	National University of Kaohsiung, Taiwan
Michele Tizzani	ISI Foundation, Italy
Michele Tizzoni	University of Trento, Italy
Olivier Togni	University of Burgundy, France
Leo Torres	Northeastern University, USA
Sho Tsugawa	University of Tsukuba, Japan
Francesco Tudisco	The University of Edinburgh, UK
Melvyn S. Tyloo	Los Alamos National Lab, USA
Stephen M. Uzzo	National Museum of Mathematics, USA
Lucas D. Valdez	IFIMAR-UNMdP, Argentina
Pim Van der Hoorn	Eindhoven University of Technology, Netherlands
Piet Van Mieghem	Delft University of Technology, Netherlands
Fabio Vanni	University of Insubria, Italy
Christian L. Vestergaard	Institut Pasteur, France
Tiphaine Viard	Télécom Paris, France
Julian Vicens	Eurecat, Spain
Blai Vidiella	CSIC, Spain
Pablo Villegas	Enrico Fermi Research Center (CREF), Italy
Maria Prosperina Vitale	University of Salerno, Italy
Pierpaolo Vivo	King's College London, UK
Johannes Wachs	Corvinus University of Budapest, Hungary
Huijuan Wang	Delft University of Technology, Netherlands
Lei Wang	Beihang University, China
Guanghui Wen	Southeast University, Nanjing, China
Mateusz Wilinski	Los Alamos National Laboratory, USA

Dirk Witthaut	Forschungszentrum Jülich, Germany
Bin Wu	Beijing Univ. of Posts and Telecom., China
Mincheng Wu	Zhejiang University of Technology, China
Tao Wu	Chongqing Univ. of Posts and Telecom., China
Haoxiang Xia	Dalian University of Technology, China
Gaoxi Xiao	Nanyang Technological University, Singapore
Nenggang Xie	Anhui University of Technology, China
Takahiro Yabe	MIT, USA
Kaicheng Yang	Northeastern University, USA
Yian Yin	Cornell University, USA
Jean-Gabriel Young	University of Vermont, USA
Irfan Yousuf	Univ. of Engineering and Technology, Pakistan
Yongguang Yu	Beijing Jiaotong University, China
Paolo Zeppini	University Cote d'Azur, France
Shi Zhou	University College London (UCL), UK
Wei-Xing Zhou	East China Univ. of Science and Techno., China
Eugenio Zimeo	University of Sannio, Italy
Lorenzo Zino	Politecnico di Torino, Italy
Michal R. Zochowski	University of Michigan, USA
Claudia Zucca	Tilburg University, Netherlands

Contents

Graph Neural Networks

Link Analysis and Ranking

Machine Learning and Networks

Network Embedding

Graph Neural Networks

Network Design Through Graph Neural Networks: Identifying Challenges and Improving Performance

Donald Loveland[1(✉)] and Rajmonda Caceres[2]

[1] University of Michigan, Ann Arbor, MI, USA
dlovelan@umich.edu
[2] MIT Lincoln Laboratory, Lexington, MA, USA
Rajmonda.Caceres@ll.mit.edu

Abstract. Graph Neural Network (GNN) research has produced strategies to modify a graph's edges using gradients from a trained GNN, with the goal of network design. However, the factors which govern gradient-based editing are understudied, obscuring *why* edges are chosen and *if* edits are grounded in an edge's importance. Thus, we begin by analyzing the gradient computation in previous works, elucidating the factors that influence edits and highlighting the potential over-reliance on structural properties. Specifically, we find that edges can achieve high gradients due to structural biases, rather than importance, leading to erroneous edits when the factors are unrelated to the design task. To improve editing, we propose **ORE**, an iterative editing method that (a) edits the highest scoring edges and (b) re-embeds the edited graph to refresh gradients, leading to less biased edge choices. We empirically study ORE through a set of proposed design tasks, each with an external validation method, demonstrating that ORE improves upon previous methods by up to 50%.

Keywords: Graph Neural Network · Network Design · Graph Editing

1 Introduction

Learning over graphs has become paramount in machine learning applications where the data possesses a connective structure, such as social networks [7],

Distribution Statement A. Approved for public release. Distribution is unlimited. This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.
H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 3–15, 2024.
https://doi.org/10.1007/978-3-031-53468-3_1

chemistry [8], and finance [25]. Fortunately, the field of graph mining has provided methods to extract useful information from graphs, albeit often needing heavy domain guidance [18]. The advent of graph neural networks (GNNs), a neural network generalized to learn over graph structured data, has helped alleviate some of these requirements by learning representations that synthesize both node and structure information [8,9,13]. Complimentary to inference, recent work has proposed methods that edit and design network structures using gradients from a trained GNN [11,17,19], enabling the efficient optimization of downstream learning tasks [31] in cyber security [5,15], urban planning [4], drug discovery [12], and more [3,14,16]. However, as gradient-based editing is applied more broadly, scrutinizing the conditions that allow for successful editing is critical. For instance, discerning the factors which influence gradient computation is still unknown, making it unclear when proposed edits can be trusted. In addition, it is unknown if gradient quality is dependent on graph structure and GNN architecture, causing further concern for practical applications.

Focusing strictly on gradient-based edit quality, we analyze the common mask learning paradigm [11,19,20,29], where a continuous scoring mask is learned over the edges in a graph. Specifically, we elucidate how structural factors, such as degree, neighborhood label composition, and edge-to-node distance (i.e., how far an edge is from a node) can influence the mask through the gradient. When these factors are not beneficial to the learning task, e.g. edge-to-node distance for a de-noising task when noise is uniformly-distributed across the graph, the learned mask can lead to erroneous edits. We additionally highlight how editing methods that rely on thresholding are more susceptible to such structural biases due to smoothing of the ground truth signal at the extreme values of the distribution. To improve editing, we propose a more fine-tuned sequential editing process, **ORE**, with two steps: (1) We **O**rder the edge scores and edit the top-k edges to prioritize high quality edges, and (2) we **R**e-embed the modified graph after the top-k edges have been **E**dited. These properties help prevent choosing edges near the expected mask value, and thus more likely to be based on irrelevant structural properties, as well as encourage edits that consider the influence of other removed edges with higher scores. We highlight the practical benefit of ORE by designing a systematic study that probes editing quality across a variety of common GNN tasks, graph structures, and architectures, demonstrating up to a 50% performance improvement for ORE over previous editing methods.

2 Related Work

Early network design solutions choose edits based on fixed heuristics, such as centrality scores [16] or triangle closing properties [14]. However, fixed heuristics generally require significant domain guidance and may not generalize to broader classes of networks and tasks. Reinforcement learning (RL) has enabled the ability to learn more flexible heuristics, such as in chemistry [30] and social networks [23]; however, RL can be prohibitively expensive due to data and computation requirements. To fulfill the need for efficient and flexible editing methods, gradient-based optimization has subsequently been applied to edge editing,

facilitated through trained GNNs. Although computing gradients for edges can be infeasible given the discrete nature of the input network, previous methods have adopted a continuous relaxation of the edge set, operating on a soft edge scoring mask that can be binarized to recover the hard edge set [11,19,20,24,29]. In its simplest form, the gradient of an edge is approximated as the gradient of the score associated with that edge, with respect to a loss objective [29]. As this is dependent on the initialization of the scoring mask, GNNExplainer proposes to leverage multiple rounds of gradient descent over the mask to arrive at a final score, rather than use the gradient directly [29]. CF-GNNExplainer extends GNNExplainer by generating counterfactual instances and measuring the change in the downstream objective [19]. Both of these methods convert the soft mask to a hard mask through fixed thresholding, which, when incorrectly chosen, can introduce noisy edits. Moreover, as mask learning is usually used to support broader objectives, such as robustness or explainability, studies fail to consider what conditions can inhibit the mask learning sub-component, instead focusing simply on the downstream objective. *Our work provides a direct analysis of mask quality through a systematic study across a wide array of tasks, GNNs, and topologies. We highlight that current mask-based editing methods can become susceptible to bias within the mask scores, prompting the development of ORE as a means of improving gradient-based edge editing.*

3 Notation

Let $G = (V, E, \mathbf{X}, \mathbf{Y})$ be a simple graph with nodes V, edges E, feature matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ with d node features, and label matrix $\mathbf{Y}$. $\mathbf{Y} \in \{0,1\}^{|V| \times c}$ with c classes for node classification, $\mathbf{Y} \in \mathbb{R}^{|V|}$ for node regression, and $\mathbf{Y} \in \{0,1\}^{c}$ for graph classification. $\mathbf{A} \in \{0,1\}^{|V| \times |V|}$ is the adjacency matrix of G, where $\mathbf{A}_{i,j} = 1$ denotes an edge between nodes i and j in G, otherwise $\mathbf{A}_{i,j} = 0$. While E and $\mathbf{A}$ represent similar information, E is used when discussing edge sets and $\mathbf{A}$ is for matrix computations. Additionally, a k-hop neighborhood of a node $i \in V$, $N_k(i)$, denotes the nodes and edges that are reachable within k-steps of i. For simplicity, k is dropped when referring to the 1-hop neighborhood. Additionally, we denote $||\mathbf{B}||_1$ as the L^1-norm of a matrix $\mathbf{B}$, $G - e_i$ as the removal of an edge from G, and $G - i$ as the removal of a node from G. For a k-layer GNN, learning is facilitated through message passing over k-hop neighborhoods of a graph [8]. A node i's representations are updated by iteratively aggregating the features of nodes in i's 1-hop neighborhood, denoted AGGR, and embedding the aggregated features with i's features, usually through a non-linear transformation parameterized by a weight matrix $\mathbf{W}$, denoted ENC. The update for node i is expressed as $\mathbf{r}_i^{(l)} = \mathrm{ENC}(\mathbf{r}_i^{(l-1)}, \mathrm{AGGR}(\mathbf{r}_u^{(l-1)}, u \in N(i)))$ for $l \in \{1, 2, ..., k\}$, where $r_i^{(0)} = x_i$. The update function is applied k times, resulting in node representations that can be used to compute predictions. For graph-level tasks, a readout function aggregates the final representation of all nodes into a single graph-level representation.

4 Optimization for Network Editing

The network design objective is given in Eq. 1, where we want to find a new adjacency matrix, $\mathbf{A}^*$, that improves a function f, parameterized by a GNN,

$$\begin{aligned} \min_{\mathbf{A}^*} \quad & ||\mathbf{A} - \mathbf{A}^*||_1 \\ \text{s.t.} \quad & f(\mathbf{X}, \mathbf{A}^*) - f(\mathbf{X}, \mathbf{A}) \geq 0. \end{aligned} \tag{1}$$

As $\mathbf{A}$ is discrete and f introduces non-linear and non-convex constraints, it is difficult to find an exact solution. Thus, we soften the constraints and focus on increasing f while maintaining the size of A, as shown in Eq. 2,

$$\min_{\mathbf{A}^*} \quad -f(\mathbf{X}, \mathbf{A}^*) + \lambda||\mathbf{A} - \mathbf{A}^*||_1. \tag{2}$$

where λ trades off the objective and the size of the remaining edge set. The negative term incentivizes the optimizer to improve f. As the optimization is still over a discrete adjacency matrix, we re-parameterize $\mathbf{A}$, as done in [10,29], and introduce a continuous mask $\mathbf{M} \in \mathbb{R}^{n \times n}$. $\mathbf{M}$ is introduced into a GNN's aggregation function as $\text{AGGR}(m_{u,v} \cdot \mathbf{r}_u^{(i-1)}, u \in N(v)))$, where $m_{u,v}$ is the mask value on the edge that connects nodes u and v. By introducing $\mathbf{M}$ into AGGR, it is possible to directly compute partial derivatives over $\mathbf{M}$, enabling gradient-based optimization over the mask values. As the aggregation function is model-agnostic, we can easily inject the mask into any model that follows this paradigm.

4.1 Graph Properties that Influence Edge Scores

We aim to study the gradient of the scoring mask $\mathbf{M}$ for a graph G. We assume access to a trained, 2-layer GNN with structure $(\mathbf{A} + \mathbf{I})^2\mathbf{X}\mathbf{W}$, where $\mathbf{I}$ is the identity matrix. We analyze a node classification setting, where a node i's feature vector is $\mathbf{x}_i = \mathbf{y}_i + \mathcal{N}(\mu, \Sigma)$, and $\mathbf{y}_i$ is the one-hot encoding of class y_i. After two layers of propagation, the feature vector for node i becomes,

$$\mathbf{r}_i^{(2)} = \mathbf{x}_i + \sum_{j \in N(i)} \mathbf{M}_{i,j}\mathbf{x}_j + \sum_{j \in N(i)} \mathbf{M}_{i,j}(\mathbf{x}_j + \sum_{k \in N(j)} \mathbf{M}_{j,k}\mathbf{x}_k). \tag{3}$$

Then, the class prediction for i is $\underset{z_i}{\text{argmax}}$, where $\mathbf{z}_i = \mathbf{r}_i^{(2)}W$. As $\mathbf{M}$ is commonly learned through gradient ascent, and only $\mathbf{r}_i^{(2)}$ depends on $\mathbf{M}$, we focus on the partial derivative of $\mathbf{r}_i^{(2)}$ with respect to a mask value $\mathbf{M}_{u,v}$, where u, v are nodes in G. As the GNN has two layers, the edges must be within two-hops of i to have a non-zero partial derivative. The partial derivative for the one- and two-hop scenarios are the first and second cases of Eq. 4, respectively,

$$\frac{\partial \mathbf{r}_i^{(2)}}{\partial \mathbf{M}_{u,v}} = \begin{cases} 2(\mathbf{y}_j + \mathbf{M}_{i,j}\mathbf{y}_i + (\mathbf{M}_{i,j} + 1)\mathcal{N}(\mu, \Sigma)) \\ \quad + \sum_{k \in N(j) - i} \mathbf{M}_{j,k}(\mathbf{y}_k + \mathcal{N}(\mu, \Sigma)), & u = i, v = j \in N(i) \\ \mathbf{M}_{i,j}(\mathbf{y}_k + \mathcal{N}(\mu, \Sigma)), & u = j \in N(i), v = k \in N(j) \end{cases} \tag{4}$$

To understand the gradient ascent process, we consider when $y_i = 0$, without loss of generality, and simplify Eq. 4. This leads to four scenarios, $y_j \in \{0, 1\}$ where $j \in N(i)$ and $y_k \in \{0, 1\}$ where $k \in N_2(i)$; however, y_j only impacts case 1 and y_k only impacts case 2, thus we can analyze each in isolation. To elucidate possible biases, we show the difference in gradients by subtracting each possible scenario (for similarly initialized $\mathbf{M}_{i,j}$), denoted as $\Delta\partial\mathbf{r}_{i,0}^{(2)}$, in Eq. 5,

$$\Delta\partial\mathbf{r}_{i,0}^{(2)} = \begin{cases} (\mathbf{M}_{i,j}+2)\mathcal{N}(\mu+1,\Sigma), & y_j = 0, y_k = 0 \\ \mathbf{M}_{i,j} + (\mathbf{M}_{i,j}+2)\mathcal{N}(\mu,\Sigma), & y_j = 1, y_k = 0 \\ 2(\mathbf{M}_{i,j}+1) + (\mathbf{M}_{i,j}+2)\mathcal{N}(\mu,\Sigma), & y_j = 0, y_k = 1 \\ 2\mathbf{M}_{i,j} + (\mathbf{M}_{i,j}+2)\mathcal{N}(\mu,\Sigma), & y_j = 1, y_k = 1 \end{cases}$$
$$+ \sum_{k \in N(j)-i, y_k = y_j} M_{j,k}\mathcal{N}(\mu+1,\Sigma) + \sum_{k \in N(j)-i, y_k \neq y_j} M_{j,k}\mathcal{N}(\mu,\Sigma). \quad (5)$$

First, all cases in Eq. 5 tend to be greater than 0, leading to higher scores for edges closer to i. Additionally, if elements of $\mathbf{M} \sim U(-1, 1)$ as in [19,29], the last two summation terms in Eq. 5 scale as $h_j(d_j - 1)$ and $(1-h_j)(d_j-1)$, respectively, where h_j and d_j represent the homophily and degree properties of the node j. Thus, high degree and high homophily can additionally bias edge selection, similar to the heuristic designed by [26] where they use $h_j d_j$ to optimize network navigation. Each of the above structural factors can either coincide with the true edge importance, or negatively influence edits when such structural properties are uninformative to the network design task.

4.2 ORE: Improved Edge Editing

Previous mask learning methods [11,19,29] have focused on fixed thresholding to generate an edge set. As shown above, it is possible that the gradients are biased towards unrelated structural properties, and thus thresholding near the expected mask value can introduce incorrect edits. To improve the mask, we introduce **ORE**, which operates by sorting the learned mask values, editing only a fixed budget of the highest scoring edges, and then re-embedding the edited graph to obtain an updated mask. Ordering the mask values and only operating on the extreme ends of the mask value distribution allows ORE to choose edges that are likely to be governed by the mask learning procedure, rather than edges with high scores due to structural biases. Additionally, as seen in Eq. 5, the gradient for an edge is dependent on downstream edges aggregated during message passing, motivating our re-embedding step to account for interactions between edits. The total editing budget is denoted as b, where b/s edges are removed for s steps. If a task requires the solution to contain a single connected component, edges that would disconnect the graph are preserved, their gradients are deactivated, and their mask values are set to one.

5 Experimental Setup

5.1 Network Editing Process

We study four GNN architectures: GCN [13], GraphSage [9], GCN-II [22], and Hyperbolic GCN [2]. As attention weights have been shown to be unreliable for edge scoring [29], we leave them out of this study. After training, each model's weights are frozen and the edge mask variables are optimized to modify the output prediction. We train three independent models on different train-val-test (50-25-25) splits for each task and the validation set is used to choose the best hyperparameters over a grid search. Then, editing is performed over 50 random data points sampled from the test set. For regression tasks, we directly optimize the output of the GNN, and for classification tasks, we optimize the cross entropy loss between the prediction and class label. For ORE, $s = b$ so that one edge is edited per step. Additionally, b is set such that roughly 10% (or less) of the edges of a graph (or computational neighborhood) are edited. The exact budget is specified for each task. All hyperparameters and implementation details for both the GNN training and mask learning are outlined in an anonymous repo[1].

Algorithm 1. ORE Algorithm

Input: GNN model f, Features $\mathbf{X}$, Adj. Matrix $\mathbf{A}$, Steps s, Epochs e, Budget b, λ
Result: Edited Adjacency Matrix $\mathbf{A}$
Initialize mask matrix $\mathbf{M}$ over edges in G
for 1 *to* s **do**
  for 1 *to* e **do**
    $P = f(\mathbf{X}, \mathbf{A}, \mathbf{M})$; `// Forward pass for prediction`
    $L = -P - \lambda|\mathbf{M}|$; `// Loss on` P `(can modify objective)`
    $\mathbf{M} \leftarrow \mathbf{M} - \alpha \frac{dL}{d\mathbf{M}}$; `// Update mask`
  end
  $O = argsort(\mathbf{M})$
  $I = O[: (b/s)]$; `// Get top indices to edit`
  $\mathbf{A}[I] = 0$; `// Remove edges from` G
end
Return $\mathbf{A}$

Editing Baselines: We utilize two fixed heuristics for editing: iterative edge removal through random sampling and edge centrality scores [1]. We also study CF-GNNExplainer [19], though we extend the algorithm to allow for learning objectives outside of counterfactuals and variable thresholds that cause b edits to fairly compare across methods. These changes do not hurt performance and are simple generalizations. Note that while we focus on CF-GNNExplainer, as they are the only previous mask learning work to consider editing, their mask generation is highly similar to other previous non-editing methods, allowing us to indirectly compare to thresholding-based methods in general [20,24,29].

[1] https://anonymous.4open.science/r/ORE-93CC/GNN_details.md.

5.2 Learning Tasks

In this section we detail the proposed tasks. For each, the generation process, parameters, and resultant dataset stats are provided in an anonymous repo[2].

Improving Motif Detection: We begin with node classification tasks similar to [19,20,29] with a goal of differentiating nodes from two different generative models. *Tree-grid* and *tree-cycle* are generated by attaching either a 3×3 grid or a 6 node cycle motif to random nodes in a 8-level balanced binary tree. We train the GNNs using cross entropy, and then train the mask to maximize a node's class prediction. As the generation process is known, we extrinsically verify if an edit was correct by determining if it corresponds to an edge inside or outside of the motifs. The editing budget is set to the size of the motifs, i.e. $b = 6$ for tree-cycle and $b = 12$ for tree-grid. Each model is trained to an accuracy of 85%.

Increasing Shortest Paths (SP): The proposed task is to delete edges to increase the SP between two nodes in a graph. This task has roots in adversarial attacks [21] and network interdiction [27] with the goal of force specific traffic routes. The task is performed on three synthetic graphs: Barabási-Albert (BA), Stochastic Block Model (SBM), and Erdős-Rényi (ER). The parameters are set to enforce each graph has an average SP length of 8. The GNN is trained through MSE of SP lengths, where the SP is estimated by learning embedding for each node and then computing the L^2 distance between each node embedding for nodes in the training set. The GNN is then used to increase the SP for pairs of nodes in the test set, which is externally verified through NetworkX. The editing budget $b = 30$ given the larger graphs. Each model is trained to an RMSE of 2.

Decreasing the Number of Triangles: The proposed task is to delete edges to decrease the number of triangles in a graph. Since triangles are often associated with influence, this task can support applications that control the spread of a process in a network, such disease or misinformation [6]. We consider the same graphs as in the SP task, BA, SBM, and ER, but instead generate 100000 different graphs each with 100 nodes. Each generation method produces graphs that, on average, have between 20 and 25 triangles, as computed by NetworkX's triangle counter. The GNNs are trained using MSE and then used to minimize the number of triangles in the graph, which is externally verified through NetworkX. The editing budget $b = 20$. Each GNN is trained to an RMSE of 6.

Improving Graph-Level Predictions: MUTAG is a common dataset of molecular graphs used to evaluate graph classification algorithms. The proposed task is to turn mutagenic molecules into non-mutagenic molecules by deleting mutagenic functional groups [20,29]. We first train the GNN models to sufficiently predict whether a molecule is mutagenic, then edit the molecules to reduce the probability of mutagenicity. We only edit mutagenic molecules that possess mutagenic functional groups, as in [20]. The editing budget $b = 5$. Each GNN is trained to an accuracy above 75%. To focus on edit quality, we do

[2] https://anonymous.4open.science/r/ORE-93CC/Dataset_details_stats.md.

not include chemistry-based feasibility checks, however it is possible to incorporate constraints into ORE either through the mask learning objective, when the constraint is differentiable, or by rejecting edits when the constraint is not differentiable.

6 Results

We present the empirical results for each task, beginning with an in-depth analysis on motif detection. Then, we collectively analyze the shortest path, triangle counting, and mutag tasks, noting trends in editing method and GNN design.

6.1 Motif Detection

In Fig. 1 we show the percent change metrics for the tree-grid and tree-cycle datasets across the GNN models. Better performance is indicated by a higher percentage of edges removed outside the motif, and a lower percentage of edges removed from inside the motif. We include performance for ORE and CF-GNNExplainer with different GNN backbones. On both datasets, the Pareto front is comprised primarily by variants of ORE, highlighting that ORE is generally better at maintaining in motif edges while removing out of motif edges.

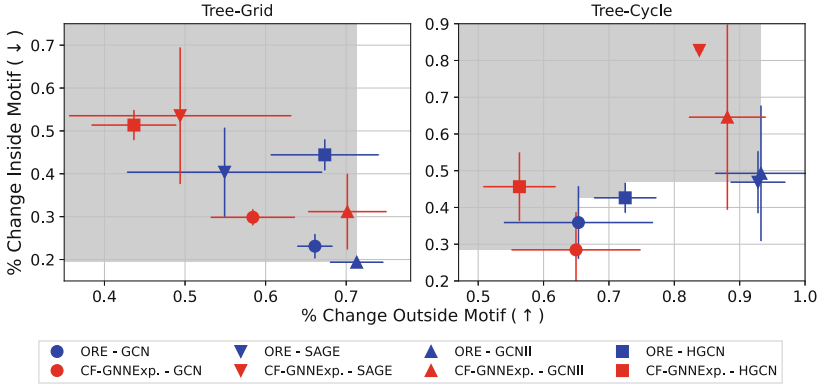


Fig. 1. Performance on tree-grid and tree-cycle across GNNs (shapes) and editing methods (colors). The axis show the percent change in edges outside and inside the motifs. Error bars indicate standard deviation in experiments. *Performance improves towards the bottom right*, as the goal is to remove edges outside the motif and retain edges inside the motif, as shown by the gray Pareto front.

How do Editing Methods Vary Across GNNs? In Fig. 1, ORE with GCNII yields the best performance; however, nearly every ORE and GNN combination outperforms the CF-GNNExplainer variant with the same GNN, demonstrating the intrinsic benefit of ORE, as well as the dependence on GNN model. To

probe how performance varies across GNNs, we stratify performance by structural factors, as motivated by our analysis in Eq. 5. In Fig. 2, we focus on the edge-to-node distance, showing that GCN is more susceptible than GCNII to this bias as the correlation between mask score and distance is higher. This result suggests that GCNII is able to suppress the use of factors unrelated to the editing task and better leverage the true importance of the edited edges. We hypothesize that GCNII's ability to retain distinct node representations by combatting oversmoothing can enable more salient gradients, however further theoretical analysis is required to confirm this behavior.

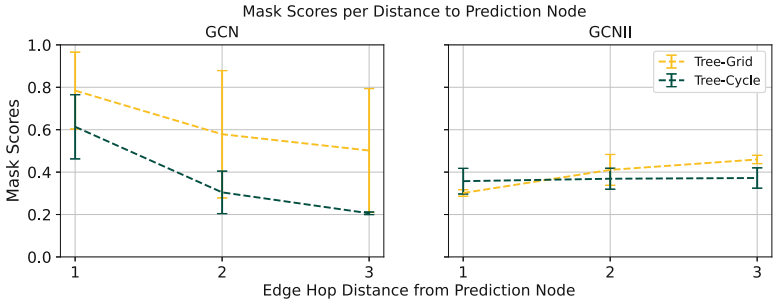


Fig. 2. Mask score distribution stratified by distance to ego-node for **GCN** and **GCNII**. Yellow denotes Tree-Grid, green denotes Tree-Cycle. For GCN, the closer an edge is to the ego-node, the higher the scores, leading to bias within the editing. GCNII minimizes bias for this unrelated property, improving editing.

How Does ORE Improve Performance? In Fig. 3a, granular performance metrics are presented in a 2D histogram for ORE and CF-GNNExplainer with GCNII, demonstrating the percent change of inside and outside motif edges for tree-grid. Result trends are similar for tree-cycle. ORE is shown to drop significantly less edges inside the motifs, denoted by the dark red boxes in the bottom right, indicating stronger performance. While both editing methods perform well at removing edges outside the motifs, CF-GNNExplainer tends to additionally remove inside edges, indicating a poorer trade-off between outside and inside motif edges. We further analyze how this arises in Fig. 3b, where the percent change metrics are presented across edit iterations (CF-GNNExplainer is not iterative and thus constant). For ORE, we see that the rates of change for inside and outside edges are significantly different – ORE more rapidly removes outside edges while maintaining inside edges, improving the final edit solution. In addition, ORE achieves similar outside edge removal to CF-GNNExplainer, while achieving a 10% increase in inside edges, supporting our hypothesis that knowledge of earlier edits allows ORE to adjust mask scores, improving editing.

6.2 Shortest Path, Triangle Counting, and Graph Classification

In Table 1, we outline the performance metrics for the SP, triangle counting, and mutag tasks. For each task, we measure the average percent change in their associated metric. In the SP experiments, all GNNs improve over the baselines, demonstrating the learned masked values extracted from the GNNs can outperform crafted heuristics, such as centrality, which leverages shortest path information in its computation. Given that ORE with GCN performs well on this task, it is possible that the structural biases identified previously, such as reliance on degree, could coincide with the SP task and improve mask scores. In the triangle counting task, edge centrality is a strong baseline for BA graphs, likely due to centrality directly editing the hub nodes that close a large number of triangles. Across the ER and SBM graphs, which do not possess a hub structure, we find that ORE with a GCNII backbone performs significantly better than both the baselines and other GNN models. Mutag reinforces these findings where GCNII removes nearly all of the mutagenic bonds for the mutagenic molecules. Notably, the Hyperbolic GCN performs poorly across experiments, possible explained by most tasks possessing Euclidean geometry, e.g. 82% of the molecules in the mutagenic dataset are roughly Euclidean as computed by the Gromov hyperbolicity metric [28]. Comparing editing methods, ORE with GCN and GCNII significantly outperforms CF-GNNExplainer with GCN across all three downstream tasks, highlighting the value of refined and iteratively optimized edge masks.

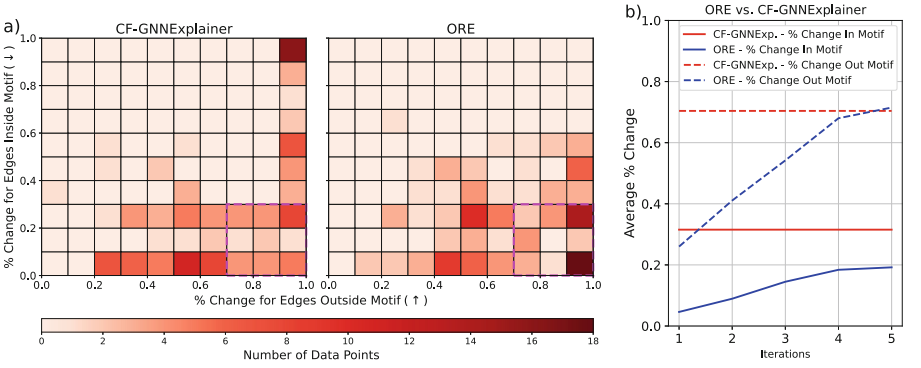


Fig. 3. Analysis on **GCNII** and **Tree-Grid**. (a) Histograms where the axes denote the percent change in edges inside and outside of the motif, boxes capture the counts. *ORE outperforms CF-GNNExplainer, as shown by the darker boxes in the bottom right.* (b) Performance across edit iterations. Blue denotes ORE, red denotes CF-GNNExplainer, dashed lines denote out motif change, and solid lines denote in motif change. *ORE rapidly removes edges outside the motifs while maintaining edges inside the motif, improving upon CF-GNNExplainer.*

Table 1. Results for SP, triangle counting, and mutag tasks. CF-GNNExplainer leverages a GCN, often one of the better performers in motif analysis. All metrics are average percent change, where higher is better. Error is the standard deviation across each model. The highlighted boxes indicate best performaners.

	Shortest Path			Triangle Counting			Mutag
	BA	ER	SBM	BA	ER	SBM	—
Random	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.36 ± 0.02	0.29 ± 0.01	0.38 ± 0.0	0.48 ± 0.04
Centrality	0.04 ± 0.01	0.08 ± 0.02	0.13 ± 0.0	0.59 ± 0.02	0.07 ± 0.0	0.09 ± 0.01	0.70 ± 0.07
CF-GNNEx	0.26 ± 0.05	0.31 ± 0.04	0.23 ± 0.03	0.42 ± 0.04	0.24 ± 0.09	0.32 ± 0.06	0.72 ± 0.04
ORE-GCN	0.52 ± 0.06	0.89 ± 0.05	0.33 ± 0.06	0.57 ± 0.05	0.33 ± 0.04	0.36 ± 0.06	0.66 ± 0.08
ORE-SAGE	0.36 ± 0.11	0.74 ± 0.07	0.17 ± 0.07	0.37 ± 0.05	0.29 ± 0.04	0.37 ± 0.06	0.42 ± 0.17
ORE-GCNII	0.24 ± 0.04	0.47 ± 0.11	0.29 ± 0.06	0.64 ± 0.05	0.41 ± 0.04	0.52 ± 0.05	0.89 ± 0.04
ORE-HGCN	0.38 ± 0.06	0.73 ± 0.06	0.18 ± 0.05	0.36 ± 0.08	0.40 ± 0.04	0.45 ± 0.07	0.47 ± 0.16

7 Conclusion

In this work, we focused on studying network design though gradient-based edge editing. We began by identifying structural factors that influence the common mask-based learning paradigm, and empirically demonstrated how these factors can impact performance across complex models and tasks. To improve editing, we introduced a sequential editing framework, ORE, that allowed for (a) the identification of higher quality edges near the extremes of the mask distribution and (b) mask scores to reflect updates from higher scoring edges. As network design evaluation has limited datasets, we proposed a set of editing tasks with external validation mechanisms, and studied both ORE and a strong editing baseline, CF-GNNExplainer, with different GNN backbones. We found that ORE outperformed CF-GNNExplainer across all experiments, while additionally demonstrated the impact of GNN architecture on the success of editing.

References

1. Brandes, U.: A faster algorithm for betweenness centrality. J. Math. Soc. **25**, 163–177 (2001)
2. Chami, I., Ying, Z., Ré, C., Leskovec, J.: Hyperbolic graph convolutional neural networks. In: NeurIPS, vol. 32 (2019)
3. Chan, H., Akoglu, L.: Optimizing network robustness by edge rewiring: a general framework. Data Min. Knowl. Discov. **30**(5), 1395–1425 (2016)
4. Domingo, M., Thibaud, R., Claramunt, C.: A graph-based approach for the structural analysis of road and building layouts. Geo-spatial Inf. Sci. **22**(1), 59–72 (2019)
5. Enoch, S., Mendonça, J., Hong, J., Ge, M., Kim, D.S.: An integrated security hardening optimization for dynamic networks using security and availability modeling with multi-objective algorithm. Comp. Netw. **208**, 108864 (2022)
6. Erd, F., Vignatti, A., da Silva, M.V.G.: The generalized influence blocking maximization problem. Soc. Netw. Anal. Mining (2021)
7. Fan, W., et al.: Graph neural networks for social recommendation. In: WWW, pp. 417–426 (2019)

8. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. CoRR (2017)
9. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. CoRR (2017)
10. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax (2017)
11. Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., Tang, J.: Graph structure learning for robust graph neural networks. In: SIGKDD (2020)
12. Jin, W., Barzilay, R., Jaakkola, T.: Junction tree variational autoencoder for molecular graph generation. In: ICML, PMLR (2018)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Kun, J., Caceres, R.S., Carter, K.M.: Locally boosted graph aggregation for community detection. arXiv preprint arXiv:1405.3210 (2014)
15. Laishram, R., Sariyüce, A., Eliassi-Rad, T., Pinar, A., Soundarajan, S.: Measuring and improving the core resilience of networks (2018)
16. Li, D., Eliassi-Rad, T., Zhang, H.R.: Optimal intervention on weighted networks via edge centrality. In: 5th International Workshop on Epidemiology Meets Data Mining and Knowledge Discovery at KDD (2022)
17. Li, G., Duda, M., Zhang, X., Koutra, D., Yan, Y.: Interpretable sparsification of brain graphs: better practices and effective designs for graph neural networks. arXiv preprint arXiv:2306.14375 (2023)
18. Liu, Y., Safavi, T., Dighe, A., Koutra, D.: Graph summarization methods and applications: a survey. ACM Comput. Surv. **51**(3), 1–34 (2018)
19. Lucic, A., ter Hoeve, M., Tolomei, G., de Rijke, M., Silvestri, F.: Counterfactual explanations for graph neural networks. CoRR, Cfgnnexplainer (2021)
20. Luo, D., et al.: Parameterized explainer for graph neural network. In: NeurIPS (2020)
21. Miller, B.A., Shafi, Z., Ruml, W., Vorobeychik, Y., Eliassi-Rad, T., Alfeld, S.: Pathattack: attacking shortest paths in complex networks. In: ECML-PKDD, pp. 532–547 (2021)
22. Wei, Z., Chen, M., Ding, B., Huang, Z., Li, Y.: Simple and deep graph convolutional networks. In: ICML (2020)
23. Morales, P., Caceres, R., Eliassi-Rad, T.: Selective network discovery via deep reinforcement learning on embedded spaces. Appl. Netw. Sci. (2021)
24. Schlichtkrull, M.S., Cao, N.D., Titov, I.: Interpreting graph neural networks for NLP with differentiable edge masking. In: ICLR (2021)
25. Sharma, S., Sharma, R.: Forecasting transactional amount in bitcoin network using temporal GNN approach. In: ASONAM (2020)
26. Şimşek, Ö., Jensen, D.: Navigating networks by using homophily and degree. Proc. Natl. Acad. Sci. **105**(35), 12758–12762 (2008)
27. Smith, J.C., Prince, M., Geunes, J.: Modern network interdiction problems and algorithms. In: Pardalos, P.M., Du, D.-Z., Graham, R.L. (eds.) Handbook of Combinatorial Optimization, pp. 1949–1987. Springer, New York (2013). https://doi.org/10.1007/978-1-4419-7997-1_61
28. Väisälä, J.: Gromov hyperbolic spaces. Exposition. Math. **23**(3), 187–231 (2005). https://doi.org/10.1016/j.exmath.2005.01.010
29. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Generating explanations for graph neural networks. In: NeurIPS, Gnnexplainer (2019)

30. Zhou, Z., Kearnes, S., Li, L., Zare, R.N., Riley, P.: Optimization of molecules via deep reinforcement learning. Sci. Rep. (2019)
31. Zhu, H., Gupta, V., Ahuja, S.S., Tian, Y., Zhang, Y., Jin, X.: Network planning with deep reinforcement learning. (2021)

Sparse Graph Neural Networks with Scikit-Network

Simon Delarue(✉) and Thomas Bonald

Institut Polytechnique de Paris, Paris, France
simon.delarue@telecom-paris.fr

Abstract. In recent years, Graph Neural Networks (GNNs) have undergone rapid development and have become an essential tool for building representations of complex relational data. Large real-world graphs, characterised by sparsity in relations and features, necessitate dedicated tools that existing dense tensor-centred approaches cannot easily provide. To address this need, we introduce a GNNs module in Scikit-network, a Python package for graph analysis, leveraging sparse matrices for both graph structures and features. Our contribution enhances GNNs efficiency without requiring access to significant computational resources, unifies graph analysis algorithms and GNNs in the same framework, and prioritises user-friendliness.

Keywords: Graph Neural Networks · Sparse Matrices · Python

1 Introduction

Graph Neural Networks (GNNs) are an extension of traditional deep learning (DL) methods for relational data structured as graphs [4]. In the past few years, GNN-based methods have gained increasing attention thanks to their impressive performance on a wide range of machine learning tasks, such as node classification, graph classification, or link prediction [16,19,25,28]. GNNs derive internal graph element representations using entity relationships and associated features, making them highly valuable for real-world data, where information frequently spans across multiple dimensions.

Real-world graphs, exemplified by large social networks or web collections with millions or billions of elements, each potentially having numerous attributes, pose substantial challenges for GNNs training [12,13]. Tremendous efforts have been made to tackle this challenge and scale up GNNs [6–8,16,29,30]. Several existing approaches rely on parallelisation or approximation techniques such as sampling or batch training, to reduce memory consumption. Few take the sparse nature of real-world graphs into account: in a graph with n nodes, the number of edges m is much lower than the number of possible edges, of order n^2. This sparsity can be exploited in data representation and algorithms to reduce the memory footprint and the computation times.

Numerous Python packages already provide GNN implementations, including PyTorch Geometric [11], Deep Graph Library [27], Spektral [14], Stellar

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 16–24, 2024.
https://doi.org/10.1007/978-3-031-53468-3_2

Graph [10] or Dive Into Graphs library [20]. But to allow natural integration with existing DL frameworks, such as PyTorch [23] or TensorFlow [1], and benefit from differentiable operators, these libraries rely upon a dense tensor-centred paradigm which does not align with graph sparsity. This dramatically hinders the use of such libraries on large real-world graphs, by requiring access to servers with large quantities of RAM.

To address this gap, we propose a GNNs module implementation relying on sparse matrices for both graph adjacency and features. Our implementation aligns seamlessly with Scikit-network[1] [3], a Python package inspired by Scikit-learn [5] for graph analysis. Scikit-network leverages the sparse formats provided by SciPy [26] for encoding graphs. It already provides various state-of-the-art graph algorithms, including ranking, clustering, and embedding algorithms, with a high computational efficiency, comparable to that of other tools like graph-tool [24] or IGraph [9], and generally much higher than that of the popular NetworkX library [15]. By only relying on NumPy [17] and SciPy [26], the development of a GNN module in Scikit-network stays true to the core principles of the package: performance and ease of use.

To summarise, our contributions encompass three key aspects. Firstly, we introduce an efficient GNN module within Scikit-network, harnessing the power of sparse matrices for both graph and features to address the characteristics of real-world graphs. Secondly, our package bridges the gap between traditional graph analysis methods and GNNs, offering a unified platform that operates within the same sparse graph representation. Lastly, we prioritise simplicity by designing our package to rely solely on foundational Python libraries, specifically NumPy and SciPy, sparing users the complexity associated with larger tensor-based DL frameworks.

The rest of the paper is organised as follows. We start by reviewing the existing related work in Sect. 2. Then, we formulate the computation of the GNNs message passing scheme using sparse matrices in Sect. 3. In Sect. 4 we briefly describe our GNNs module design and we show its performance compared to other GNNs libraries in Sect. 5.

2 Related Work

Several Python packages already exist to help with the development and usage of GNNs. Pytorch Geometric (PyG) [11] proposes a general message passing interface with all recent GNN-based aggregation schemes. In order to reduce the computation time when running on large complex networks, several processing tools such as sampling or batch training are proposed. Spektral [14] is built upon the same gather-scatter paradigm as PyG, but implements GNNs on top of the user-friendly API Keras. Stellar Graph [10] is also based on Keras but uses its own graph representation. Deep Graph Library [27] involves a combination of user-configurable message passing functions and sparse-dense matrix multiplications to provide the user with a GNNs framework. However, in all these

[1] https://github.com/sknetwork-team/scikit-network.

libraries, the implementation design is driven by the necessity to integrate with existing deep learning frameworks, namely PyTorch [23] or TensorFlow [1], that provide differentiable operators. This implies the use of dense tensors for the encoding of graph elements, which comes at the expense of the sparse nature of real-world graphs. Our work differs from these approaches in the sense that we leverage sparse format representations for both the structure of the graph and the features of the nodes, and we only rely on Python fundamental libraries.

Other libraries, such as Dive Into Graphs library [20] offer a high-level toolbox for deep learning on graphs. However, its goal is slightly different from the previous libraries; it goes beyond the implementation of elementary tasks and includes advanced research-oriented methods such as benchmarks, graph generation or 3D graphs. In Scikit-network, we mainly focus on efficiency and ease-of-use rather than extensive features for the users.

3 Message Passing with Sparse Matrices

3.1 Message Passing Overview

Traditional GNNs models rely on an architecture that propagates the signal (or features) information across the network, through a serie of iterations (or layers). At each layer l, this architecture involves two steps for computing the representation of a node u, h_u^l; (i) the aggregation of this node's neighbourhood information into a *message* and (ii) the update of the node's representation using this aggregated information and the previous embedding of the node. These two steps can be written as follows:

$$h_{\mathcal{N}(u)}^l = \phi(\{h_v^{l-1}, \forall v \in \mathcal{N}(u)\}) \tag{1}$$

$$h_u^l = \psi\left(h_u^{l-1}, h_{\mathcal{N}(u)}^l\right) \tag{2}$$

where $\mathcal{N}(u)$ denotes the neighbourhood of node u, ϕ is an aggregation function and ψ is an update function.

3.2 Using Sparse Matrices in Scikit-Network

In Scikit-network, we represent a node attributed graph $G = (V, E, X)$ by its adjacency matrix A and its node-feature matrix X, both encoded in SciPy's Compressed Sparse Row format (CSR). This format uses three arrays to represent a matrix: two arrays of size m, where m is the number of non-zero elements in the matrix, and one array of size $n + 1$, where n is the number of rows of the matrix. Among various SciPy sparse formats (such as COO, DOD, etc.), the CSR format was initially chosen for Scikit-network as it is the most memory-efficient for common algebraic operations like listing neighbours and performing matrix-vector product [21].

The concept of *message passing scheme* within a graph convolutional layer, as seen in GCN [19], involves the gathering and scattering of information from a node's neighbourhood to compute the node's updated representation. In scikit-network, these operations are executed using the following layer-wise propagation rule:

$$H^{l+1} = \sigma(\tilde{A}H^l W^l + b^l) \tag{3}$$

where $\tilde{A}$ is the (normalised) adjacency matrix of the graph (with or without self-loops), H^l denotes the matrix of node representations at layer l, with $H^0 = X$, W^l and b^l represent trainable weight matrix and bias vector, and σ is a non-linear activation function. In our Scikit-network implementation, we employ sparse matrix multiplication (SpMSpM) to compute the term $\tilde{A}H^l$ when $l = 0$. Subsequently, for $l \geq 1$ and thus when the dimension of the H^l matrix is significantly reduced to the hidden-layer dimension, we use sparse-dense matrix multiplications (SpMM). In the same manner, the backward propagation only requires SpMM multiplications between reduced-sized matrices.

4 Scikit-Network GNN Design

As illustrated in Fig. 1, initialising a GNN model in Scikit-network is straightforward. Like other graph analysis algorithms in the package, GNN training is built upon the `.fit_predict()` method. This method hides forward and backward propagation's complexity to the user, providing them directly with the final predictions. Additional training details, like loss, accuracy and temporary layer outputs, are easily accessible using the `history` parameter.

5 Performance

To evaluate Scikit-network's GNN implementation, we compare (i) the achieved accuracy level, which should be similar to that of other implementations, and (ii) the training time required for the model to learn representations. In both cases, we use a GCN [19] model for a node classification task and compare with the two most widely used libraries: PyG and DGL[2]. The computer used to run all the experiments is a Mac with OS 12.6.8, equipped with a M1 Pro processor and 16 GB of RAM. For fair comparison, all experiments are run on a CPU device.

[2] We rely on the number of *forks* associated with each package on GitHub as a metric to gauge library usage. Please note that this metric provides only a partial view of actual project usage.

Code Listing 1.1. Node classification using Graph Neural Network in Scikit-network.

```
# Graph Neural Network
hidden_dim = 16

gnn = GNNClassifier(
    dims=[hidden_dim, n_labels],
    layer_types='Conv',
    activations='ReLu',
    verbose=True)

labels_pred = gnn.fit_predict(
    adjacency, features, labels,
    n_epochs=100, history=True)
```

Code Listing 1.2. Graph analysis algorithms in Scikit-network.

```
# PageRank
pagerank = PageRank()
scores = pagerank.
             fit_predict(
             adjacency)

# Louvain clustering
louvain = Louvain()
labels = louvain.
             fit_predict(
             adjacency)
```

Fig. 1. Graph methods using Scikit-network. On the left, node classification using a GNN model. On the right, Louvain clustering [2] and PageRank scoring [22]. Traditional graph algorithms and deep learning based models use the same API.

5.1 Datasets

We use three real-world datasets of varying sizes considering their structure and attributes. The datasets include Wikipedia-based networks[3], `Wikivitals` and `Wikivitals+`. Nodes in these datasets represent Wikipedia articles, and there is a link between two nodes if the corresponding articles are referencing each other (in either direction) through hypertext links on Wikipedia. Additionally, each article comes with a feature vector, corresponding to the number of occurrences of each word (or token) in its summary. `OGBN-arxiv` [18] models a citation network among Computer Science papers from arXiv. For this dataset, we use node connections as the feature matrix. We detail the dataset characteristics in Table 1.

Table 1. Dataset statistics. Adjacency matrix A is summarised with its number of nodes $|V|$, edges $|E|$ and density δ_A. d refers to the number of unique attributes. The node-attribute matrix X is summarized using its number of non-zero elements m and its density δ_X. For dataset marked with $\star$, we use the adjacency matrix as features.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	δ_A	d	m	δ_X
`Wikivitals`	1.00×10^4	8.24×10^5	1.64×10^{-2}	3.78×10^4	1.363×10^6	3.59×10^{-3}
`Wikivitals+`	4.51×10^4	3.94×10^6	3.86×10^{-3}	8.55×10^4	4.78×10^6	1.24×10^{-3}
`OGBN-arxiv` $\star$	1.69×10^5	1.66×10^6	8.13×10^{-5}	1.69×10^5	1.66×10^6	8.13×10^{-5}

[3] https://netset.telecom-paris.fr/.

5.2 Performances and Running Time

Tables 2 and 3 respectively display the running time of the training process and the corresponding accuracy scores for the different GNNs implementations. For DGL and PyG, the dense feature matrix format hinders training models on large graphs without additional tricks, e.g., sampling or batch training (which we did not use for fair comparison). Therefore, these implementations trigger an out-of-memory (OOM) error when used on the `OGBN-arxiv` dataset. In contrast, Scikit-network does not require these extra steps to achieve good performance on this dataset within a reasonable computation time. Furthermore, we can observe the benefits of using Scikit-network regarding the characteristics of the graph: the sparser the graph, the more efficient the computation.

Table 2. Average computation times and standard deviations (3 runs) for 100 epochs model training on 3 real-world datasets.

Package	`Wikivitals`	`Wikivitals+`	`OGBN-arxiv`
`DGL`	76.6 ± 4.2	2396.1 ± 36.5	OOM
`PyG`	**28.7 ± 0.1**	1242.8 ± 28.6	OOM
`Sknetwork`	139.5 ± 0.6	**632.9 ± 7.9**	**60.1 ± 0.4**

Table 3. Average accuracy on test set and standard deviations (3 runs) for 100 epochs model training on 3 real-world datasets.

Package	`Wikivitals`	`Wikivitals+`	`OGBN-arxiv`
`DGL`	78.5 ± 1.7	80.1 ± 0.2	OOM
`PyG`	78.1 ± 0.3	79.7 ± 0.5	OOM
`Sknetwork`	78.4 ± 1.6	80.2 ± 0.2	83.3 ± 0.5

5.3 Impact of Graph Characteristics

Graph Density. We further validate this observation in Fig. 2, where we initialise a random graph (Erdos-Rényi) with a fixed number of nodes $|V| = 1 \times 10^4$ and varying density. In this setup, node connections are used as features. We can notice that for highly sparse graphs, i.e., low density, both DGL and PyG implementations face challenges due to their reliance on dense matrices. On the opposite, Scikit-network implementation shows great performance improvements compared to these methods once below a density threshold (approximately 1×10^{-3}).

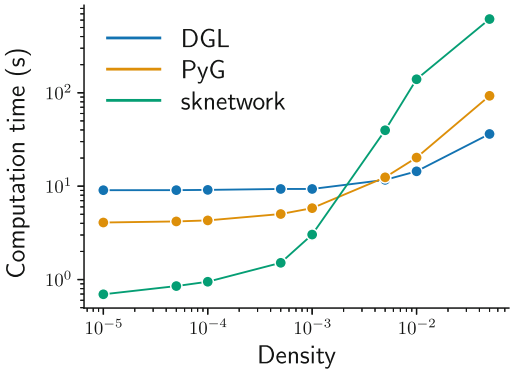


Fig. 2. Computation time for several GCN [19] implementations, according to adjacency and feature matrix densities. *Notice the log-scale on both axis.*

Number of Nodes. To evaluate the impact of the number of nodes on the performance of our implementation, we fix the densities of graph matrix δ_A and feature matrix δ_X, such as $\delta_A = \delta_X = 5 \times 10^{-4}$, and vary the number of nodes in the random graphs generated. In Fig. 3 we show Scikit-network's benefits in terms of training model computation time, compared to DGL and PyG implementations.

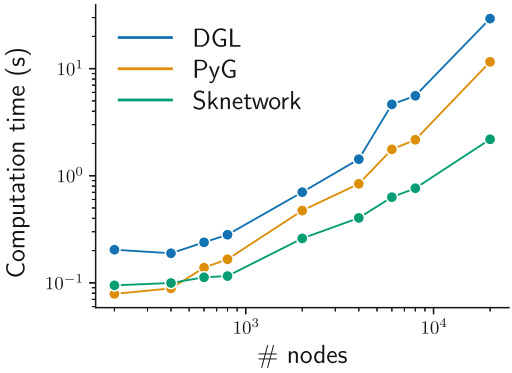


Fig. 3. Computation time for several GCN [19] implementations, according to the number of nodes in the graph. *Notice the log-scale on both axis.*

6 Conclusion

We presented the Scikit-network Graph Neural Network module. Our implementation achieves great performance on real-world graphs both in terms of accuracy

and running time, by making use of sparse encoding and operations in the learning process. Moreover, our design relies solely on foundational Python libraries, NumPy and SciPy, and does not require the use of tensor-centred traditional Deep Learning frameworks. With this module, Scikit-network offers a unified platform gathering traditional graph analysis algorithms and deep learning-based models. In the future, we plan to extend the current module by adding additional state-of-the-art GNNs models and layers, as well as optimizers.

Acknowledgements. The authors would like to thank Tiphaine Viard for the numerous discussions, as well as her insightful comments and suggestions about the paper.

References

1. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. In: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283 (2016)
2. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Statist. Mech. Theory Exp. **2008**(10), P10008 (2008). https://doi.org/10.1088/1742-5468/2008/10/p10008
3. Bonald, T., De Lara, N., Lutz, Q., Charpentier, B.: Scikit-network: graph analysis in python. J. Mach. Learn. Res. **21**(1), 7543–7548 (2020)
4. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond Euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017). https://doi.org/10.1109/MSP.2017.2693418
5. Buitinck, L., et al.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122 (2013)
6. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
7. Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.10568 (2017)
8. Chiang, W.L., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.J.: Cluster-gcn: an efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 257–266 (2019)
9. Csardi, G., Nepusz, T., et al.: The igraph software package for complex network research. Int. J. Complex Syst. **1695**(5), 1–9 (2006)
10. Data61, C.: Stellargraph machine learning library (2018). https://github.com/stellargraph/stellargraph
11. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
12. Fey, M., Lenssen, J.E., Weichert, F., Leskovec, J.: Gnnautoscale: scalable and expressive graph neural networks via historical embeddings. In: International Conference on Machine Learning, pp. 3294–3304. PMLR (2021)
13. Frasca, F., Rossi, E., Eynard, D., Chamberlain, B., Bronstein, M., Monti, F.: Sign: scalable inception graph neural networks. arXiv preprint arXiv:2004.11198 (2020)
14. Grattarola, D., Alippi, C.: Graph neural networks in tensorflow and keras with spektral [application notes]. IEEE Comput. Intell. Mag. **16**(1), 99–106 (2021)

15. Hagberg, A., Swart, P., Chult, D.S.: Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab. (LANL), Los Alamos (2008)
16. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Adv. Neural Inf. Pocess. Syst. **30** (2017)
17. Harris, C.R., et al.: Array programming with numpy. Nature **585**(7825), 357–362 (2020)
18. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687 (2020)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
20. Liu, M., et al.: Dig: a turnkey library for diving into graph deep learning research. J. Mach. Learn. Res. **22**(1), 10873–10881 (2021)
21. Lutz, Q.: Graph-based contributions to machine-learning. Theses, Institut Polytechnique de Paris (2022). https://theses.hal.science/tel-03634148
22. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bring order to the web. Tech. rep., Technical report, Stanford University (1998)
23. Paszke, A.,et al.: Pytorch: an imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32** (2019)
24. Peixoto, T.P.: The graph-tool python library. Figshare (2014). https://doi.org/10.6084/m9.figshare.1164194
25. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
26. Virtanen, P., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nat. Methods **17**(3), 261–272 (2020)
27. Wang, M., et al.: Deep graph library: a graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315 (2019)
28. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
29. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 974–983 (2018)
30. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: Graphsaint: graph sampling based inductive learning method. arXiv preprint arXiv:1907.04931 (2019)

Enhancing Time Series Analysis with GNN Graph Classification Models

Alex Romanova(✉)

Melenar, LLC, McLean, VA 22101, USA
sparkling.dataocean@gmail.com
http://sparklingdataocean.com

Abstract. The use of Graph Neural Networks (GNNs) in time series analysis is on the rise, yet the application of GNN Graph Classification in this field remains in its early stages. In our research, we repurpose GNN Graph Classification, traditionally rooted in disciplines like biology and chemistry, to delve into the intricacies of time series datasets. We demonstrate how graphs are constructed within individual time series and across multiple datasets, highlighting the versatility of GNN techniques beyond their standard applications. A key observation in our study was the sensitivity of the GNN Graph Classification model to graph topology. While initially seen as a potential concern for model robustness, this sensitivity turned out to be beneficial for pinpointing outliers. Our findings underscore the innovation of applying GNN Graph Classification to time series analysis, unlocking new dimensions in data interpretation. This research lays the groundwork for integrating these methodologies, indicating vast potential for their wider application and opening up promising avenues for future exploration.

Keywords: Graph Neural Network · Graph Classification · Graph Topology · Electroencephalography · EEG · Climate · Time Series · Deep Learning

1 Introduction

In 2012, a significant breakthrough occurred in the fields of deep learning and knowledge graphs. Convolutional Neural Network (CNN) image classification was introduced through AlexNet [1], showcasing its superiority over previous machine learning techniques in various domains [2]. Concurrently, Google introduced knowledge graphs, enabling machines to understand relationships between entities and revolutionizing data integration and management, enhancing products with intelligent and 'magical' capabilities [3].

The growth of deep learning and knowledge graphs occurred simultaneously for years, with CNN excelling at grid-structured data tasks but struggling with graph-structured ones. Conversely, graph techniques thrived on graph structured data but lacked deep learning's capability. In the late 2010s, Graph Neural Networks (GNN) emerged, combining deep learning and graph processing, and revolutionizing how we handle graph-structured data [4].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 25–36, 2024.
https://doi.org/10.1007/978-3-031-53468-3_3

GNN models apply deep learning to graph-structured data, capturing entity relationships and graph dynamics. They tackle tasks from Node Classification and Regression to Link Prediction, emphasizing diverse graph analysis aspects. Node Classification and Regression predict labels or values for individual nodes based on their graph context. Link Prediction infers potential edges by assessing the likelihood of connections between nodes. Distinctly, GNN Graph Classification operates at a holistic level, categorizing entire graphs based on their overarching structures, rather than individual components. Our study centers on this GNN Graph Classification, emphasizing its unique capability to analyze and categorize entire graph entities.

Analytical thinking steers us from identifying a problem to selecting the most fitting solution. In the spirit of repurposing tools beyond their conventional use, we've taken GNN Graph Classification models, traditionally employed in domains like chemistry and medicine, and applied them innovatively to extract patterns from time series data. We are also exploring the potential of Graph Classification in text analytics. Details of that study are reserved for now due to an ongoing conference paper submission.

To explore the application of GNN Graph Classification models to time series, we extend our previous research on climate data [5]. For this study we also examine application of Graph Classification models to electroencephalography (EEG) signal data. The combination of these two domains highlights the versatility of GNN Graph Classification models in addressing diverse real-world challenges. Our experiments utilize two Kaggle datasets: the 'EEG-Alcohol' set, which predicts alcoholism through brain-reaction graphs [6], and climate dataset with 40 years of daily temperature data for the 1000 most populous cities worldwide [7].

GNN Graph Classification models are notably sensitive to graph topology. This allows the model to detect changes and patterns swiftly, but also introduces a risk of overfitting or misinterpreting noise. It's essential to manage this aspect, especially when dealing with data that has high variability, like climate data with seasonal shifts. Nonetheless, this sensitivity can be an asset, helping detect outliers and anomalies. We'll discuss this further as we review results from the EEG and climate datasets, emphasizing how outlier detection is crucial.

2 Related Work

In 2012, the introduction of AlexNet models and Knowledge Graphs marked it as a breakthrough year for deep learning and knowledge graphs. AlexNet model[1], along with the success of Convolutional Neural Networks in image classification, outperformed previous state-of-the-art machine learning techniques across various domains [2]. Concurrently, Google's introduction of Knowledge Graphs [8] enabled machines to understand entity relationships, driving a new era in data integration and management that enhance product intelligence and user experience [3]. In the late 2010s, Graph Neural Networks emerged as a fusion of deep learning and knowledge graphs. GNN enable complex data analysis and predictions by effectively capturing relationships between graph nodes [9,10].

While GNNs have been employed across a myriad of disciplines, our study zeroes in on the niche area of GNN Graph Classification. This methodology proves invaluable in sectors like chemistry, medicine, and biology for delving into intricate relationships within molecular structures, proteins, and biomolecules. Utilizing graphs to represent these entities allows researchers to uncover novel insights, potentially revolutionizing therapeutic or treatment avenues [11–13].

In the vast landscape of GNNs, our focus is sharply tuned to the potential of GNN Graph Classification in time series data analysis. One salient feature of these models is their nuanced sensitivity to graph topology. While this can complicate handling noise and adapting to spatial-temporal variations, it emerges as a strength when detecting outliers and anomalies. Upcoming sections will further explore this intricate balance, showcasing moments where the model's discernment comes to the fore. Interestingly, our literature review suggests that the application of GNN Graph Classification to time series data remains an underexplored avenue. This observation underscores the novelty and value of our current research direction.

With the evolution of technology, methods to analyze time series data have also matured. While traditional approaches may falter at the intricacies and spatial-temporal challenges inherent to this data, a comprehensive review by [14] underscored GNNs' promise. Notably absent, however, was a focus on GNN graph classification within the time series context. Our study endeavors to fill this void, illuminating the potential of GNN Graph Classification for in-depth time series insights.

In this study, we examine the potential of GNN Graph Classification models in the context of time series data, with a specific focus on healthcare and environmental sectors. By transforming time series, prevalent across various domains, into graphs, we aim to tap into their inherent relationships. Our primary goal centers on classifying EEG signals and climate data, confronting the challenges of graph-centric time series classification [13].

Recent research has leveraged CNN deep learning methods for atmospheric imaging, particularly in the estimation of tropical cyclones [15]. However, surveys suggest that the application of deep learning to climate data mining is still in its early stages and continues to evolve [16,17]. In our study, we extend this exploration by introducing the application of GNN Graph Classification models to climate data, aiming to uncover nuanced patterns and insights from this complex dataset.

Deep learning has reshaped EEG signal interpretation, especially in neural engineering and biomedical fields [18–20]. In a prior study, we utilized CNN image classification combined with graph mining to differentiate EEG patterns between alcoholic and control subjects [21]. Here, we further explore the efficacy of GNN Graph Classification models on EEG datasets.

3 Methods

The input for GNN Graph Classification models consists of a collection of small labeled graphs representing objects in the dataset. These graphs are composed of

nodes and edges, with associated features that describe attributes of the entities and relationships between them.

The data flow diagram in Fig. 1 contrasts and compares the processes applied to climate data and EEG data. In both cases, graph edges are established based on pairs of vectors if their cosine similarity surpasses a set threshold. To ensure connectivity within our graphs, we integrated a virtual node, linking it with all existing nodes. This approach, as showcased in Fig. 2, effectively transforms isolated graph segments into unified single connected components.

For both scenarios, we employ a Graph Convolutional Network Convolution (GCNConv) model from the PyTorch Geometric Library (PyG) to perform GNN Graph Classification [22].

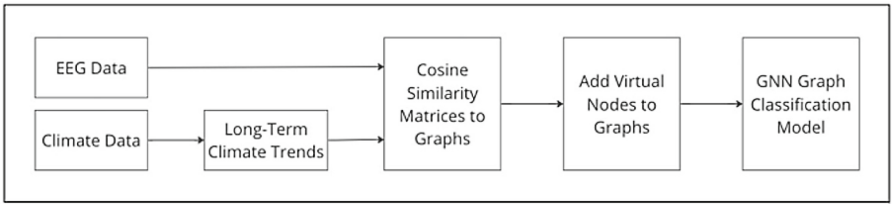


Fig. 1. Data flow diagrams for EEG data and Climate data scenarios: common steps and different steps.

3.1 Transform EEG Data to Graphs

For the EEG scenario, individual graphs are constructed for every combination of brain and trial, with alcohol or control group indicators serving as classification labels. Electrode locations function as nodes, and the EEG channel signals, which represent electrical activity at these positions, act as node attributes. This encapsulates the interplay between various electrode placements and their related brain electrical responses. Subsequently, the GNN Graph Classification model is trained on these graphs, aiming to distinguish between the alcohol and control categories.

3.2 Cosines for Consecutive Years as Indicators

To identify long-term climate patterns, we'll derive sequences of cosines from daily temperature vectors spanning consecutive years. A declining trend in these average cosine similarities might signal increasing day-to-day temperature variations, potentially alluding to climatic shifts. Conversely, a rising trend might suggest a more consistent, stable climate.

3.3 Graph Construction from Climate Data

For our climate analysis, we construct city-centric graphs. Their labels, either 'stable' or 'unstable', hinge on the average cosine similarities observed between

consecutive years. Within these graphs, each node corresponds to a specific combination of city and year, encapsulating daily temperature vectors of that year as its features. This graphical representation aids our model in distinguishing patterns characteristic of either stable or fluctuating climate conditions.

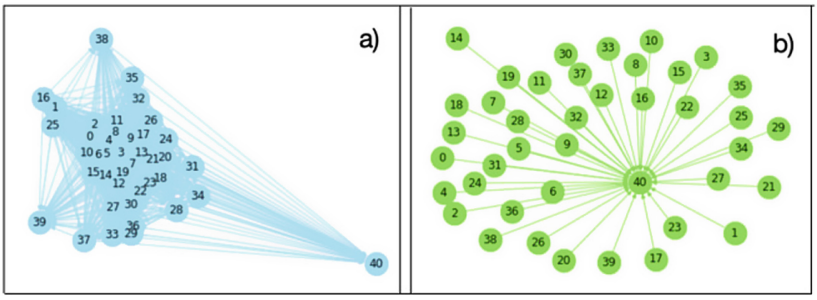


Fig. 2. (a) A highly connected graph with high degree of connectivity representing stable climate patterns in Malaga, Spain. (b) A sparsely connected graph with low degree of connectivity indicating unstable and unpredictable climate patterns in Orenburg, Russia.

3.4 From Cosine Similarity Matrices to Graph Construction

Initially, we calculated cosine similarity matrices and identified vector pairs with cosine values exceeding a set threshold to create graph adjacency matrices. By adding a virtual node to every graph, we ensured the consolidation of disconnected parts into a single connected component, readying them for GNN Graph Classification models.

3.5 Implementing the GNN Graph Classification Model

We employed the GCNConv model from the PyTorch Geometric Library (PyG) [22] for our GNN Graph Classification tasks. This model harnesses convolutional operations to extract graph features, leveraging edges, node attributes, and graph labels during the process. For those interested in the data conversion to the PyG format, details can be found in our technical blogs [23,24].

In our trials involving EEG and climate data, the model quickly demonstrated outstanding performance, achieving a remarkable 100% accuracy in just a few training epochs. Such early success prompted us to wonder about its feasibility. Delving deeper, we attributed this to the model's acute sensitivity to nuances in graph topology. The implications and details of this heightened sensitivity are further explored in the experiments section.

4 Experiments

4.1 EEG Data Graph Classification

EEG Data Source. We utilized the 'EEG-Alcohol' Kaggle dataset [6], which consists of EEG correlates from a study on genetic predisposition to alcoholism. It includes data from 8 subjects, each exposed to different stimuli while their brain electrical activity was recorded using 64 electrodes at a sampling rate of 256 Hz for 1 s. The total number of person-trial pairs was 61. Our data preparation process involved using some code from Ruslan Klymentiev's Kaggle notebook [25] and developing our own code to transform EEG channel data into time series.

Prepare Input Data for Graph Classification Model. For the EEG data, separate graphs were created for each person-trial, with labels indicating the alcohol or control group. Electrode positions were used as nodes and EEG channel signals as node features. Cosine similarity matrices were calculated for each graph to select node pairs with cosine similarities above a certain threshold, and virtual nodes were added to all graphs to transform them into single connected components. The challenge of a small training dataset (61 person-trial graphs) was addressed by randomly varying threshold values within the range (0.75, 0.95), augmenting the input dataset to 1037 graphs and enhancing the GNN Graph Classification model's performance.

Train the Model. The study employed the GCNConv model from the PyG library [22] for classifying EEG data into alcoholic and control groups. The input dataset was randomly split into 15% testing data (155 graphs) and 85% training data (882 graphs). The model achieved an accuracy of about 98.4% on the training data and 98.1% on the testing data. The slight fluctuations in accuracy can be attributed to the relatively small testing dataset.

Interpreting Model Results. In our model's assessment, an interesting pattern emerged. Out of the 17 discrepancies in predictions, every misclassified graph pinpointed a distinct individual from the control group, yet was categorized by the model as part of the alcohol group. The prediction probabilities for these errors hovered between 0.45 and 0.55, highlighting the model's degree of uncertainty and its propensity to make close calls. This uniform misclassification for one individual could hint at specific patterns in their EEG data resembling those of alcohol subjects, suggesting potential latent alcohol-influenced markers.

In our current study, we noticed that all outlier graphs featured 'single stimulus' trials, which the model frequently misinterpreted. This echoes findings from our previous research [21], where we integrated CNN image classification with graph mining for EEG analysis. A consistent theme across both studies is the striking similarity of the "single stimulus" patterns for both groups.

Figure 3, adapted from our earlier work, underscores the difficulty in distinguishing between the groups based solely on these trials, highlighting their limited utility in precise group categorization.

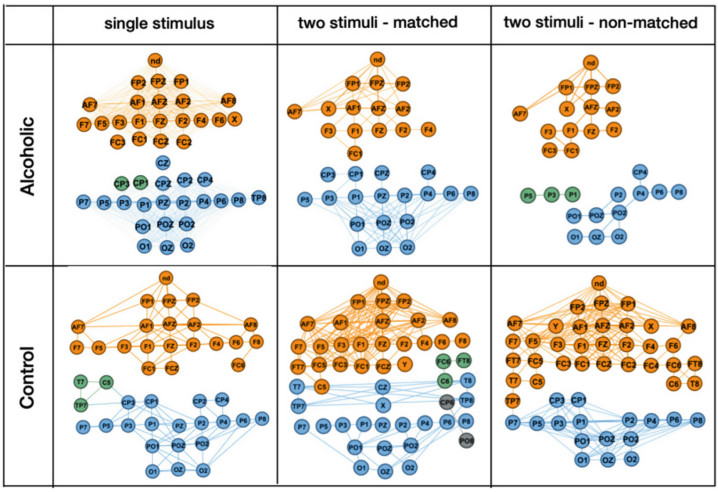


Fig. 3. Graph visualization from our previous study [21] illustrates that reactions on 'single stimulus' trials are similar for persons from Alcohol and Control groups.

4.2 Climate Data Graph Classification

Climate Data Source. We utilized average daily temperature data from year 1980 to year 2019 for 1000 most populous cities worldwide from a Kaggle dataset [7]. For the GNN Graph Classification model, we created separate graphs for all cities, using city-year combinations as nodes, daily temperature vectors of corresponding years as node features, and cosine similarities higher than certain thresholds as graph edges. Graph labels, indicating stable or unstable long-term climate trends, were created by calculating average cosines between daily temperature vectors of consecutive years.

Average Cosines Between Consecutive Years. We undertook the task of computing average cosines between consecutive years for each city to understand temporal temperature patterns. Our results, as presented in Table 1, show cities closer to the equator having consistently warm temperatures throughout the year, leading to higher average cosine similarities. In contrast, Table 2 lists cities situated in areas such as Canada and Northern Europe, which face distinct seasonal temperature changes and thus, have lower cosine similarities. These computed average cosines served as the basis for labeling graphs in our GNN graph classification as 'stable' or 'unstable'.

Table 1. Very high average cosine similarities indicate stable climate with less variance in daily temperature patterns.

City	Country	Consecutive Year Score
Malabo	Equatorial Guinea	0.999790
Hagta	Guam	0.999726
Oranjestad	Aruba	0.999724
Willemstad	Curaçao	0.999723
Monrovia	Liberia	0.999715

Table 2. A decrease in the average cosine similarity between consecutive years can indicate an increase in the variance or difference in daily temperature patterns, which could be a sign of climate change.

City	Country	Consecutive Year Score
Calgary	Canada	0.739156
Edmonton	Canada	0.778996
Reykjavik	Iceland	0.809576
Krasnoyarsk	Russia	0.815763
Yaroslavl'	Russia	0.827339

Graph Formulation for Climate Data in GNN Graph Classification. In preparing input data for the GNN Graph Classification model, we constructed unique graphs for each city based on their 'stable' or 'unstable' climate categorization. Edges within these graphs were drawn between pairs of nodes that exhibited strong cosine similarities. To ensure each graph maintained a unified structure, we integrated virtual nodes. The utility of these virtual nodes is illustrated in Fig. 2: a city with a stable climate, such as Malaga, Spain, presents a dense web of connections, whereas a city with an unstable climate, like Orenburg, Russia, has more isolated connections. This pronounced difference underscores the value of virtual nodes in enhancing the model's pattern recognition.

Model Training and Evaluation. In our study, we employed the GCNConv model from the PyG library [22] to identify abnormal climate trends using graph representations of daily temperature data. Guidance on data preparation and format conversion for PyTorch Geometric is detailed in our technical blog [23]. The model's performance, evaluated using PyG's accuracy metrics, achieved approximately 96% accuracy on the training data and 99% on the test data.

Table 3. Cities with unstable temperature located in lower latitudes.

City	Country	Latitude	Longitude
Nanning	China	22.820	108.320
Yulin	China	22.630	110.150
San Luis Potosi	Mexico	22.170	−101.000

Table 4. Cities with stable temperature located in high latitudes.

City	Country	Latitude	Longitude
Monaco	Monaco	43.740	7.407
Nice	France	43.715	7.265
Marseille	France	43.290	5.375

Interpreting Model Results. Our GNN Graph Classification model adeptly captured node interconnections within the graphs, offering predictions on temperature pattern consistency over timeframes. Most of the graphs corresponding to equatorial cities were identified as stable, while those from high-latitude cities were largely deemed unstable. These outcomes align with insights drawn from our evaluations using consecutive-year cosine values. However, there were discrepancies in the model's predictions for 36 out of the 1000 cities examined.

Applying the GNN Graph Classification model to climate data brought forth some anomalies. For instance, cities in lower latitudes like China and Mexico manifested unstable temperature patterns, diverging from typical climate expec-

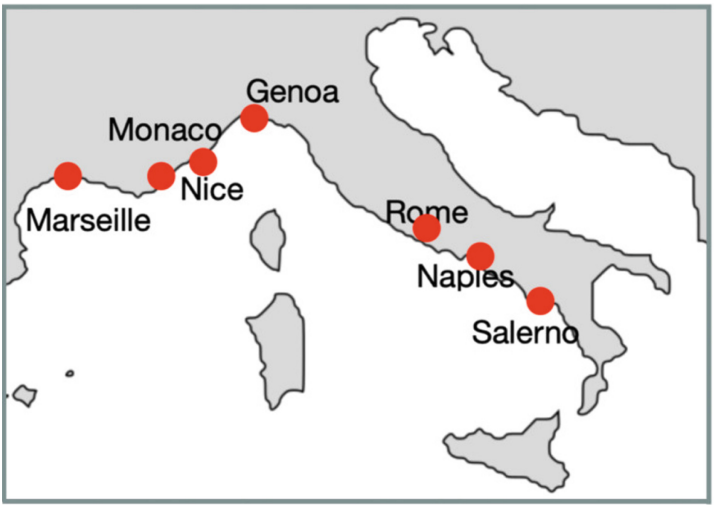


Fig. 4. Results of our previous studies [26,27]: cities located near the Mediterranean Sea have very stable and consistent temperature patterns.

tations, as detailed in Table 3. In contrast, as observed in Table 4, certain cities in higher European latitudes surprisingly showcased temperature stability.

In our previous research [26,27], where we utilized CNN models for time series analysis, we introduced a novel approach using symmetry metrics. This method hinged on transforming entity pair vectors into GAF images. Contrasting with the traditional cosine similarity metrics, the symmetry metrics-based model pinpointed cities on the Western Mediterranean Coast as notably stable, as illustrated in Fig. 4 from that research. This finding aligns seamlessly with the results presented in this paper.

Cities such as Monaco, Nice, and Marseille, highlighted in Table 4, emerged as anomalies. Despite geographical indicators suggesting potential climate variability, they consistently exhibited stable temperature trends. Furthermore, their corresponding graphs were categorized as unstable in the context of long-term climate metrics. This divergence accentuates the nuanced sensitivity of our GNN Graph Classification model. Such findings both corroborate our previous research and deepen our understanding of the intricate intricacies of climate patterns.

5 Conclusions

In our pioneering work, we repurposed GNN Graph Classification models, traditionally used in fields like biology and chemistry, for time series analysis on EEG and climate data. By introducing virtual nodes, we bridged fragmented input graphs, deepening our data representation. For climate data, we uniquely labeled graphs based on average cosine values between consecutive years, providing a fresh perspective on climate trends.

Our findings revealed a pronounced sensitivity in the model's interpretation of graph topology. While initially viewed as a possible shortcoming, this sensitivity proved to be a valuable strength. In the EEG data, anomalies were notably centered around one individual from the control group. Additionally, the "single stimulus" patterns consistently emerged as indistinguishable between the Alcoholic and Control groups, echoing observations from our prior research.

In our climate analysis, cities such as Monaco, Nice, and Marseille in the Mediterranean region defied expectations with their stable temperature patterns. These observations align with our prior research, where we introduced an innovative symmetry metric for time series analysis via CNN models, offering a distinct perspective compared to the conventional cosine similarity measures. The concordant findings from both the symmetry metrics and GNN Graph Classification models underscore their collective strength in detecting and emphasizing outliers within the dataset.

In closing, our study reinforces the versatility and promise of GNN Graph Classification models, emphasizing their ability to discern intricate patterns, anomalies, and relationships within data. The harmony between our current findings and those from previous research speaks to the consistent strength of our methodologies. We believe these insights pave the way for further exploration and broader applications of GNN Graph Classification models in forthcoming analytical pursuits.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.: ImageNet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. (2012)
2. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
3. Noy, N., Gao, Y., Jain, A., Narayanan, A., Patterson, A., Taylor, J.: Industry-scale knowledge graphs: lessons and challenges. acmqueue (2019)
4. Bronstein, M., Bruna, J., Cohen, T., Veličković, P.: Geometric deep learning: grids, groups, graphs, geodesics, and gauges (2021). https://doi.org/10.48550/arXiv.2104.13478
5. Romanova, A.: GNN graph classification method to discover climate change patterns. In: Artificial Neural Networks and Machine Learning (ICANN). Springer, Cham (2023)
6. kaggle.com. EEG-Alcohol Data Set (2017)
7. kaggle.com. Temperature History of 1000 Cities 1980 to 2020 (2020)
8. Bradley, A.: Semantics Conference, 2017 (2017)
9. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.: SA Comprehensive Survey on Graph Neural Networks (2019)
10. Wang, M., Qiu, L., Wang, X.: A Survey on Knowledge Graph Embeddings for Link Prediction. Symmetry (2021)
11. Adamczyk, J.: Application of Graph Neural Networks and graph descriptors for graph classification (2022)
12. Hu, W., et al.: Strategies for Pre-training Graph Neural Networks (2020)
13. He, H., Queen, O., Koker, T., Cuevas, C., Tsiligkaridis, T., Zitnik, M.: Domain Adaptation for Time Series Under Feature and Label Shifts (2023)
14. Jin, M., et al.: A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection (2023)
15. Ardabili, S., Mosavi, A., Dehghani, M., Varkonyi-Koczy, A.: Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets (2019)
16. Liu, Y., Racah, E.: Deep Learning and Machine Learning in Hydrological Processes, Climate Change and Earth (2019)
17. Liu, Y., et al.: Application of Deep Convolutional Neural Networks for Detecting Extreme Weather in Climate Datasets (2016)
18. Craik, A., He, Y., Contreras-Vidal, J.: Deep Learning for Electroencephalogram (EEG) Classification Tasks: A Review (2019)
19. Gemein, L.A.W.: A Machine-Learning-Based Diagnostics of EEG Pathology (2020)
20. Roy, Y., Banville, H., Albuquerque, I., Fauber, J.: Deep Learning-Based Electroencephalography Analysis: A Systematic Review (2019)
21. Romanova, A.: Time series pattern discovery by deep learning and graph mining. In: Database and Expert Systems Applications (DEXA) (2021)
22. PyG. Pytorch Geometric Library: Graph Classification with Graph Neural Networks (2023)
23. GNN Graph Classification for Climate Change Patterns. Graph Neural Network (GNN) Graph Classification - A Novel Method for Analyzing Time Series Data (2023). http://sparklingdataocean.com/2023/02/11/cityTempGNNgraphs/
24. GNN Graph Classification for EEG Pattern Analysis. Graph Neural Network for Time-Series Analysis (2023). http://sparklingdataocean.com/2023/05/08/classGraphEeg/

25. Klymentiev, R.: EEG Data Analysis (2019)
26. Romanova, A.: Unsupervised time series classification for climate data. In: Proceedings of the Northern Lights Deep Learning Conference (NLDL) (2022)
27. Romanova, A.: Symmetry metrics for pairwise entity similarities. In: Integration and Web Intelligence (iiWAS). LNCS, vol. 13635. Springer, Cham (2022)

When Do We Need Graph Neural Networks for Node Classification?

Sitao Luan[1,2(✉)], Chenqing Hua[1,2], Qincheng Lu[1], Jiaqi Zhu[1], Xiao-Wen Chang[1], and Doina Precup[1,2,3]

[1] McGill University, Montreal, Canada
{sitao.luan,chenqing.hua,qincheng.lu,jiaqi.zhu}@mail.mcgill.ca,
{chang,dprecup}@cs.mcgill.ca
[2] Mila, Montreal, Canada
[3] DeepMind, London, UK

Abstract. Graph Neural Networks (GNNs) extend basic Neural Networks (NNs) by additionally making use of graph structure based on the relational inductive bias (edge bias), rather than treating the nodes as collections of independent and identically distributed (*i.i.d.*) samples. Though GNNs are believed to outperform basic NNs in real-world tasks, it is found that in some cases, GNNs have little performance gain or even underperform graph-agnostic NNs. To identify these cases, based on graph signal processing and statistical hypothesis testing, we propose two measures which analyze the cases in which the edge bias in features and labels does not provide advantages. Based on the measures, a threshold value can be given to predict the potential performance advantages of graph-aware models over graph-agnostic models.

1 Introduction

In the past decade, deep Neural Networks (NNs) [11] have revolutionized many machine learning areas and one of their major strength is their capacity and effectiveness of learning latent representation from Euclidean data. Recently, the focus has been put on its applications on non-Euclidean data, *e.g.,* relational data or graphs. Combining with graph signal processing and convolutional neural networks [12], numerous Graph Neural Networks (GNNs) have been proposed [7,8,10,21,27] that empirically outperform traditional neural networks on graph-based machine learning tasks, *e.g.,* node classification, graph classification, link prediction, graph generation, *etc.*

Nevertheless, growing evidence shows that GNNs do not always gain advantages over traditional NNs on relational data [14,16,17,20,23,30]. In some cases, even a simple Multi-Layer Perceptron (MLP) can outperform GNNs by a large margin, *e.g.,* as shown in Table 1, MLP outperform baseline GNNs on *Cornell, Wisconsin, Texas* and *Film* and perform almost the same as baseline GNNs on *PubMed, Coauthor CS* and *Coauthor Phy.* This makes us wonder when it is appropriate to use GNNs. In this work, we explore an explanation and propose

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 37–48, 2024.
https://doi.org/10.1007/978-3-031-53468-3_4

two proper measures to determine when to use GNNs for a node classification task.

A common way to leverage graph structure is to apply graph filters in each hidden layer of NNs to help feature extraction. Most existing graph filters can be viewed as operators that aggregate node information from its direct neighbors. Different graph filters yield different spectral or spatial GNNs. Among them, the most commonly used is the *renormalized affinity matrix* [10], which corresponds to a low-pass (LP) filter [24] mainly capturing the low-frequency components of the input, *i.e.*the locally smooth features across the whole graph [28].

The use of LP graph filters relies on the assumption that nodes tend to share attributes with their neighbors, a tendency called homophily [9,25] that is widely exploited in node classification tasks. GNNs that are built on the homophily assumption learn to assign similar labels to nodes that are closely connected [29], which corresponds to an assumption of intrinsic smoothness on latent label distribution. We call this kind of relational inductive bias [2] the edge bias. We believe it is a key factor leading to GNNs' superior performance over NNs' in many tasks.

Table 1. Accuracy (%) Comparison of Baseline GNNs and MLP

Datasets\Models	MLP Acc	GCN Acc	GAT Acc	GraphSAGE Acc	Baseline Average	Diff(MLP, Baseline)	Edge Homophily
Cornell	85.14	60.81	59.19	82.97	67.66	**17.48**	0.3
Wisconsin	87.25	63.73	60.78	87.84	70.78	**16.47**	0.21
Texas	84.59	61.62	59.73	82.43	67.93	**16.66**	0.11
Film	36.08	30.98	29.71	35.28	31.99	**4.09**	0.22
Chameleon	46.21	61.34	61.95	47.32	56.87	−10.66	**0.23**
Squirrel	29.39	41.86	43.88	30.16	38.63	−9.24	**0.22**
Cora	74.81	87.32	88.07	85.98	87.12	−12.31	0.81
Citeseer	73.45	76.70	76.42	77.07	76.73	−3.28	0.74
Pubmed	87.86	88.24	87.81	88.59	88.21	**−0.35**	**0.80**
DBLP	77.39	85.87	85.89	81.19	84.32	−6.93	0.81
Coauthor CS	93.72	93.91	93.41	94.38	93.90	**−0.18**	**0.81**
Coauthor Phy	95.77	96.84	96.32	OOM	96.58	**−0.81**	**0.93**
AMZ Comp	83.89	87.03	89.74	83.70	86.82	−2.93	0.78
AMZ Photo	90.87	93.61	94.12	87.97	91.90	−1.03	**0.83**

However, the existing homophily metrics are not appropriate to display the edge bias, *e.g.,* as shown in Table 1, MLP does not necessarily outperform baseline GNNs on some low homophily datasets (*Chameleon and Squirrel*) and does not significantly underperform baseline GNNs on some high homophily datasets

(*PubMed,Coauthor CS,Coauthor Phy and AMZ Photo*). Thus, a metric that is able to indicate whether or not the graph-aware models can outperform graph-agnostic models is needed.

Contributions. In this paper, we discover that graph-agnostic NNs are able to outperform GNNs on a non-trivial set of graph datasets. To explain the performance inconsistency, we propose the Normalized Total Variation (NTV) and Normalized Smoothness Value (NSV) to measure the effect of edge bias on features and labels of an attribute graph. NSV leads us to conduct statistical hypothesis testings to examine how significant the effect of edge bias is. With the measures and analyses on 14 real-world datasets, we are able to predict and explain the expected performance of graph-agnostic MLPs and GNN models.

The rest of this paper is organized as follows: In Sect. 2, we introduce the notations and the background; In Sect. 3, we propose two measures of the effect of edge-bias and discuss their potential usage; In Sect. 4, we discuss the related works.

2 Preliminaries

After stating the motivations, in this section, we will introduce the used notations and formalize the idea. We use bold fonts for vectors (*e.g.,* $\boldsymbol{v}$). Suppose we have an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ without bipartite component, where $\mathcal{V}$ is the node set with $|\mathcal{V}| = N$; $\mathcal{E}$ is the edge set without self-loop; $A \in \mathbb{R}^{N\times N}$ is the symmetric adjacency matrix with $A_{ij} = 1$ if and only if $e_{ij} \in \mathcal{E}$, otherwise $A_{ij} = 0$; D is the diagonal degree matrix, *i.e.* $D_{ii} = \sum_j A_{ij}$ and $\mathcal{N}_i = \{j : e_{ij} \in \mathcal{E}\}$ is the neighborhood set of node i. A graph signal is a vector $\boldsymbol{x} \in \mathbb{R}^N$ defined on $\mathcal{V}$, where x_i is defined on the node i. We also have a feature matrix $X \in \mathbb{R}^{N\times F}$ whose columns are graph signals and each node i has a corresponding feature vector $X_{i:}$ with dimension F, which is the i-th row of X. We denote $Z \in \mathbb{R}^{N\times C}$ as label encoding matrix, where $Z_{i:}$ is the one hot encoding of the label of node i.

2.1 Graph Laplacian and Affinity Matrix

The (combinatorial) graph Laplacian is defined as $L = D - A$, which is a Symmetric Positive Semi-Definite (SPSD) matrix [4]. Its eigendecomposition gives $L = U\Lambda U^T$, where the columns of $U \in \mathbb{R}^{N\times N}$ are orthonormal eigenvectors, namely the *graph Fourier basis*, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ with $\lambda_1 \leq \cdots \leq \lambda_N$, and these eigenvalues are also called *frequencies*. The graph Fourier transform of the graph signal $\boldsymbol{x}$ is defined as $\boldsymbol{x}_{\mathcal{F}} = U^{-1}\boldsymbol{x} = U^T\boldsymbol{x} = [\boldsymbol{u}_1^T\boldsymbol{x}, \ldots, \boldsymbol{u}_N^T\boldsymbol{x}]^T$, where $\boldsymbol{u}_i^T\boldsymbol{x}$ is the component of $\boldsymbol{x}$ in the direction of $\boldsymbol{u_i}$.

Finding the eigenvalues and eigenvectors of a graph Laplacian is equivalent to solving a series of conditioned minimization problems relevant to function smoothness defined on $\mathcal{G}$. A smaller λ_i indicates that basis $\boldsymbol{u_i}$ is a smoother

function defined on $\mathcal{G}$ [6], which means any two elements of $\boldsymbol{u_i}$ corresponding to two connected nodes will be more similar. This property plays an important role in our paper.

Some graph Laplacian variants are commonly used, *e.g.,* the symmetric normalized Laplacian $L_{\text{sym}} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}AD^{-1/2}$ and the random walk normalized Laplacian $L_{\text{rw}} = D^{-1}L = I - D^{-1}A$. L_{rw} and L_{sym} share the same eigenvalues that are in $[0, 2)$, and their corresponding eigenvectors satisfy $\boldsymbol{u}^i_{\text{rw}} = D^{-1/2}\boldsymbol{u}^i_{\text{sym}}$.

The affinity (transition) matrices can be derived from the Laplacians, *e.g.,* $A_{\text{rw}} = I - L_{\text{rw}} = D^{-1}A$, $A_{\text{sym}} = I - L_{\text{sym}} = D^{-1/2}AD^{-1/2}$ and $\lambda_i(A_{\text{rw}}) = \lambda_i(A_{\text{sym}}) = 1 - \lambda_i(A_{\text{sym}}) = 1 - \lambda_i(A_{\text{rw}}) \in (-1, 1]$. [10] introduced the renormalized affinity and Laplacian matrices as $\hat{A}_{\text{sym}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, $\hat{L}_{\text{sym}} = I - \hat{A}_{\text{sym}}$, where $\tilde{A} \equiv A + I, \tilde{D} \equiv D + I$. It essentially adds a self-loop and is widely used in Graph Convolutional Network (GCN) as follows,

$$Y = \text{softmax}(\hat{A}_{\text{sym}}\ \text{ReLU}(\hat{A}_{\text{sym}}XW_0)\ W_1) \tag{1}$$

where $W_0 \in \mathbb{R}^{F \times F_1}$ and $W_1 \in \mathbb{R}^{F_1 \times O}$ are parameter matrices. GCN can learn by minimizing the following cross entropy loss

$$\mathcal{L} = -\text{trace}(Z^T \log Y). \tag{2}$$

The random walk renormalized matrices $\hat{A}_{\text{rw}} = \tilde{D}^{-1}\tilde{A}$ can also be applied to GCN and $\hat{A}_{\text{rw}}$ shares the same eigenvalues as $\hat{A}_{\text{sym}}$. The corresponding Laplacian is defined as $\hat{L}_{\text{rw}} = I - \hat{A}_{\text{rw}}$. Specifically, the nature of random walk matrix makes $\hat{A}_{\text{rw}}$ behaves as a mean aggregator $(\hat{A}_{\text{rw}}\boldsymbol{x})_i = \sum_{j \in \{\mathcal{N}_i \cup i\}} x_j/(D_{ii} + 1)$ which is applied in [8] and is important to bridge the gap between spatial- and spectral-based graph convolution methods.

3 Measuring the Effect of Edge Bias

In this section, we will derive two measures for the effect of edge bias and conduct hypothesis testing for the effect. We analyze the behaviors of these measures and apply them on 14 real world datasets. The measurement results are used to predict the potential performance differences between GNNs and MLPs.

3.1 Normalized Total Variation (NTV) and Normalized Smoothness Value (NSV) for Measuring Edge Bias

NTV. Graph Total Variation (GTV) is a quantity to characterize how much graph signal varies *w.r.t.* graph filters and is defined as follows [1,3],

$$GTV(\boldsymbol{x}) = \left\| \boldsymbol{x} - \hat{A}\boldsymbol{x} \right\|_p^p$$

where $\hat{A}$ generally represents normalized or renormalized filters, the ℓ_p-norm can be replaced by the Frobenius norm when we measure a matrix X. *GTV* generally measures the utility of the edge bias by gauging the distance between node features and its aggregated neighborhood features. To eliminate the influence of the magnitude of $\boldsymbol{x}$ or X and make it comparable, we define Normalized Total Variation (NTV) as follows,

$$\mathrm{NTV}(\boldsymbol{x}) = \frac{\left\|\boldsymbol{x} - \hat{A}\boldsymbol{x}\right\|_2^2}{2\left\|\boldsymbol{x}\right\|_2^2}, \ \mathrm{NTV}(X) = \frac{\left\|X - \hat{A}X\right\|_F^2}{2\left\|X\right\|_F^2} \tag{3}$$

the division of factor 2 guarantees that $0 \leq \mathrm{NTV} \leq 1$. A small NTV value implies $\boldsymbol{x} \approx \hat{A}\boldsymbol{x}$ or $X \approx \hat{A}X$.

NSV. Even when the features of the node resemble its aggregated neighborhood, it does not necessarily mean that the average pairwise attribute distance of connected nodes is smaller than that of unconnected nodes. Based on this argument, we define Normalized Smoothness Value (NSV) as a measure of the effect of the edge bias.

The total pairwise attribute distance of connected nodes is equivalent to the Dirichlet energy of X on $\mathcal{G}$ as follows,

$$E_D^{\mathcal{G}}(X) = \sum_{i \hookrightarrow j} \|X_{i:} - X_{j:}\|_2^2 = \sum_{i \hookrightarrow j} (\boldsymbol{e}_i - \boldsymbol{e}_j)^T XX^T (\boldsymbol{e}_i - \boldsymbol{e}_j) = tr\left(\sum_{i \hookrightarrow j} (\boldsymbol{e}_i - \boldsymbol{e}_j)^T XX^T (\boldsymbol{e}_i - \boldsymbol{e}_j)\right)$$
$$= tr\left(\sum_{i \hookrightarrow j} (\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T XX^T\right) = \mathrm{trace}(X^T L X).$$

The total pairwise distance of unconnected nodes can be derived from the Laplacian L^C of the complementary graph $\mathcal{G}^C$. To get L^C, we introduce the adjacency matrix of $\mathcal{G}^C$ as $A^C = (\mathbf{1}\mathbf{1}^T - I) - A$, its degree matrix $D^C = (N-1)I - D$, and $L^C = D^C - A^C = NI - \mathbf{1}\mathbf{1}^T - L$. Then, the total pairwise attribute distance of unconnected nodes (Dirichlet energy of X on $\mathcal{G}^{\mathcal{C}}$) is

$$E_D^{\mathcal{G}^C}(X) = \mathrm{trace}\left(X^T L^C X\right) = \mathrm{trace}\left(X^T (NI - \mathbf{1}\mathbf{1}^T) X\right) - E_D^{\mathcal{G}}(X)$$

$E_D^{\mathcal{G}}(X)$ and $E_D^{\mathcal{G}^C}(X)$, are non-negative and are closely related to sample covariance matrix (see Appendix A for details) as follows,

$$E_D^{\mathcal{G}}(X) + E_D^{\mathcal{G}^C}(X) = \mathrm{trace}\left(X^T (NI - \mathbf{1}\mathbf{1}^T) X\right) = N(N-1) \cdot \mathrm{trace}\left(\mathrm{Cov}(X)\right).$$

Since $\mathrm{trace}\left(\mathrm{Cov}(X)\right)$ is the total variation in X, we can say that the total sample variation can be decomposed in a certain way onto $\mathcal{G}$ and $\mathcal{G}^C$ as $E_D^{\mathcal{G}}(X)$ and $E_D^{\mathcal{G}^{\mathcal{C}}}(X)$. Then, the average pairwise distance (variation) of connected nodes and unconnected nodes can be calculated by normalizing $E_D^{\mathcal{G}}(X)$ and $E_D^{\mathcal{G}^{\mathcal{C}}}(X)$,

$$E_{\mathrm{N}}^{\mathcal{G}}(X) = \frac{E_D^{\mathcal{G}}(X)}{2\left|\mathcal{E}\right|}, \quad E_{\mathrm{N}}^{\mathcal{G}^{\mathcal{C}}}(X) = \frac{E_D^{\mathcal{G}^{\mathcal{C}}}(X)}{N(N-1) - 2\left|\mathcal{E}\right|} \tag{4}$$

and the Normalized Smoothness Value (NSV) is defined as

$$\mathrm{NSV}^{\mathcal{G}}(X) = \frac{E_{\mathrm{N}}^{\mathcal{G}}(X)}{E_{\mathrm{N}}^{\mathcal{G}}(X) + E_{\mathrm{N}}^{\mathcal{G}^c}(X)}. \tag{5}$$

We can see that $0 \leq \mathrm{NSV}^{\mathcal{G}}(X) \leq 1$ and it can be used to interpret the edge bias: (1) For labels Z, $\mathrm{NSV}^{\mathcal{G}}(Z) \geq 0.5$ means that the proportion of connected nodes that share different labels is larger than that of unconnected nodes, which implies that edge bias is harmful for Z and the homophily assumption is invalid; (2) For features X, $\mathrm{NSV}^{\mathcal{G}}(X) \geq 0.5$ means that the average pairwise feature distance of connected nodes is greater than that of unconnected nodes, which suggests that the feature is non-smooth. On the contrary, small $\mathrm{NSV}(Z)$ and $\mathrm{NSV}(X)$ indicates that the homophily assumption holds and the edge bias is potentially beneficial.

The above analysis raises another question: how much does NSV deviating from 0.5 or what is the exact NSV to indicate the edge bias is statistically beneficial or harmful. In the following section, we study the problem from statistical hypothesis testing perspective and provide thresholds by the p-values.

3.2 Hypothesis Testing for Edge Bias

Consider the following distributions of labels and features,

For labels Z:

- $P_1 = \mathbb{P}\left(Z_{i:} \neq Z_{j:} \middle| e_{ij} \in \mathcal{E}\right)$ = The proportion of connected nodes that share different labels;
- $P_2 = \mathbb{P}\left(Z_{i:} \neq Z_{j:} \middle| e_{ij} \notin \mathcal{E}\right)$ = The proportion of unconnected nodes that share different labels.

For features X:

- $D_1 = \|X_{i:} - X_{j:}\|_2^2 \ \middle| e_{ij} \in \mathcal{E}$ = Distribution of pairwise feature distance of connected nodes;
- $D_2 = \|X_{i:} - X_{j:}\|_2^2 \ \middle| e_{ij} \notin \mathcal{E}$ = Distribution of pairwise feature distance of unconnected nodes.

Suppose P_1, P_2, D_1, D_2 follow:

$$P_1 \sim \mathrm{Binom}(n_1, p_1),\ P_2 \sim \mathrm{Binom}(n_2, p_2);\ D_1 \sim N(d_1, \sigma_1^2),\ D_2 \sim N(d_2, \sigma_2^2).$$

Consider the hypotheses for labels

$$H_0^L : p_1 = p_2;\ H_1^L : p_1 \neq p_2;\ H_2^L : p_1 \geq p_2;\ H_3^L : p_1 \leq p_2$$

and hypotheses for features

$$H_0^F : d_1 = d_2;\ H_1^F : d_1 \neq d_2;\ H_2^F : d_1 \geq d_2;\ H_3^F : d_1 \leq d_2$$

To conduct the hypothesis tests, we use Welch's t-test for features and χ^2 test for labels. We can see $E_{\mathrm{N}}^{\mathcal{G}}(Z)$ and $E_{\mathrm{N}}^{\mathcal{G}^C}(Z)$ are sample estimation of the mean p_1 and p_2 for label Z; $E_{\mathrm{N}}^{\mathcal{G}}(X)$ and $E_{\mathrm{N}}^{\mathcal{G}^C}(X)$ are sample estimation of mean d_1 and d_2 for X. Thus, the p-values of hypothesis tests can suggest if NSV statistically deviates from 0.5. The smoothness of labels and features can be indicated as follows,

For feature X:

- p-value(H_0^F vs H_1^F): > 0.05, H_0^F holds, feature is non-smooth; ≤ 0.05, to be determined.
- p-value(H_0^F vs H_2^F): ≤ 0.05, feature is statistically significantly non-smooth.
- p-value(H_0^F vs H_3^F): ≤ 0.05, feature is statistically significantly smooth.

For label Z:

- p-value(H_0^L vs H_1^L): > 0.05, H_0^L holds, label is non-smooth; ≤ 0.05, to be determined.
- p-value(H_0^L vs H_2^L): ≤ 0.05, label is statistically significantly non-smooth.
- p-value(H_0^L vs H_3^L): ≤ 0.05, label is statistically significantly smooth.

Results of hypothesis testing are summarized in Table 2. We can see that for the datasets where baseline GNNs underperform MLP, *Cornell, Texas* and *Wisconsin* has statistically significantly non-smooth labels and *Film* has non-smooth labels. In these datasets, the edge bias will provide harmful information no matter the features are smooth or not. For other datasets, they have statistically significantly smooth labels, which means the edge bias can statistically provide benefits to the baseline GNNs and lead them to have superiority performance over MLP.

3.3 Why NTV and NSV Work

We explain why and how NTV and NSV can be used to explain the performance gain and loss of GNNs over graph-agnostic NNs. We simplify the explanation by removing the non-linearity as [28]. Let $\hat{A}$ denote a general filter with $\left\|\hat{A}\right\|_2 = 1$ in GNNs.

NTV. When the NTV of node features X and labels Z are are small, it implies

$$\hat{A}X \approx X,\ \hat{A}Z \approx Z. \tag{6}$$

The loss function of GNNs and MLP can be written as follows,

$$\text{GNNs: } \min_W \left\|\hat{A}XW - Z\right\|_F, \text{ MLP: } \min_W \|XW - Z\|_F, \tag{7}$$

Table 2. Statistics of Datasets and the Performance Differences

Datasets\Measures	Features					Labels					Baseline Average -
	NTV	NSV	H_0^F vs H_1^F	H_0^F vs H_2^F	H_0^F vs H_3^F	NTV	NSV	H_0^L vs H_1^L	H_0^L vs H_2^L	H_0^L vs H_3^L	MLP
Cornell	0.33	0.48	0.00	1.00	**0.00**	0.33	0.53	0.0003	**0.00**	1.00	−17.48
Texas	0.33	0.48	0.00	1.00	**0.00**	0.42	0.60	0.00	**0.00**	1.00	−16.66
Wisconsin	0.38	0.51	**0.72**	0.36	0.64	0.40	0.55	0.00	**0.00**	1.00	−16.47
Film	0.39	0.50	**0.19**	0.90	0.10	0.37	0.50	**0.05**	0.97	0.03	−4.09
Coauthor CS	0.36	0.36	0.00	1.00	**0.00**	0.19	0.18	0.00	1.00	**0.00**	0.18
Pubmed	0.33	0.44	0.00	1.00	**0.00**	0.25	0.24	0.00	1.00	**0.00**	0.35
Coauthor Phy	0.35	0.36	0.00	1.00	**0.00**	0.16	0.09	0.00	1.00	**0.00**	0.81
AMZ Photo	0.41	0.39	0.00	1.00	**0.00**	0.23	0.17	0.00	1.00	**0.00**	1.03
AMZ Comp	0.41	0.38	0.00	1.00	**0.00**	0.25	0.22	0.00	1.00	**0.00**	2.93
Citeseer	0.35	0.45	0.00	1.00	**0.00**	0.22	0.24	0.00	1.00	**0.00**	3.28
DBLP	0.37	0.46	0.00	1.00	**0.00**	0.21	0.20	0.00	1.00	**0.00**	6.93
Squirrel	0.47	0.54	0.00	**0.00**	1.00	0.44	0.49	0.00	1.00	**0.00**	9.24
Chameleon	0.45	0.45	0.00	1.00	**0.00**	0.45	0.49	0.00	1.00	**0.00**	10.66
Cora	0.38	0.47	0.00	1.00	**0.00**	0.20	0.19	0.00	1.00	**0.00**	12.31

where W is the learnable parameter matrix. When $\hat{A}Z \approx Z$,

$$\min_W \left\|\hat{A}XW - Z\right\|_F \approx \min_W \left\|\hat{A}XW - \hat{A}Z\right\|_F \leq \min_W \left\|\hat{A}\right\|_2 \|XW - Z\|_F = \min_W \|XW - Z\|_F . \quad (8)$$

This suggests that GNNs work more effectively than graph-agnostic methods when $\mathrm{NTV}^{\mathcal{G}}(Z)$ is small. However, when labels are non-smooth on $\mathcal{G}$, a projection onto the column space of $\hat{A}$ will hurt the expressive power of the model. In a nutshell, GNNs potentially have stronger expressive power than NNs when $\mathrm{NTV}^{\mathcal{G}}(Z)$ is small.

NSV. We first rewrite the softmax function as follows,

$$Y = \mathrm{softmax}(\hat{A}XW) = \left(\exp(Y')\mathbf{1}\mathbf{1}^T\right)^{-1} \odot \exp(Y') \quad (9)$$

where $Y' = \hat{A}XW$, $\mathbf{1} \in \mathbb{R}^{C\times 1}$ and C is the output dimension. The loss function (2) can be written as

$$\mathcal{L} = -\mathrm{trace}\left(Z^T\hat{A}XW\right) + \mathrm{trace}\left(\mathbf{1}^T \log\left(\exp(Y')\mathbf{1}\right)\right). \quad (10)$$

We denote $\tilde{X} = XW$ and consider $-\mathrm{trace}\left(Z^T\hat{A}XW\right)$, which plays the main role in the above optimization problem.

$$-\mathrm{trace}\left(Z^T\hat{A}XW\right) = -\mathrm{trace}\left(Z^T\hat{A}\tilde{X}\right) = -\sum_{i\hookrightarrow j}\hat{A}_{ij}Z_{i:}\tilde{X}^T_{j:}. \quad (11)$$

To minimize $\mathcal{L}$, if $\hat{A}_{ij} \neq 0$, then $\tilde{X}_{j:}$ will learn to get closer to $Z_{i:}$ and this means: (1) If $Z_{i:} = Z_{j:}$, $\tilde{X}_{j:}$ will learn to approach to the unseen ground truth label $Z_{j:}$ which is beneficial; (2) If $Z_{i:} \neq Z_{j:}$, $\tilde{X}_{j:}$ tends to learn a wrong label, in which case the edge bias becomes harmful. Conventional NNs can be treated as a special case with only $\hat{A}_{ii} = 1$, otherwise 0. So the edge bias has no effect on conventional NNs.

To evaluate the effectiveness of edge bias, NSV makes a comparison to see if the current edges in $\mathcal{E}^{\mathcal{G}}$ have significantly less probability of indicating different pairwise labels than the rest edges. If NSV together with the p-value suggests that the edge bias is statistically beneficial, we are able to say that GNNs will obtain performance gain from edge bias; otherwise, the edge bias will have a negative effect on GNNs. NTV, NSV, p-values and the performance comparison of baseline models on 14 real-world datasets shown in Table 2 are consistent with our analysis.

4 Related Works

Smoothness (Homophily). The idea of node homophily and its measures are mentioned in [26] and defined as follows,

$$H_{\text{node}}(\mathcal{G}) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{\left|\{u \mid u \in \mathcal{N}_v, Z_{u,:} = Z_{v,:}\}\right|}{d_v}$$

Or in [30], the edge homophily is defined as follows,

$$H_{\text{edge}}(\mathcal{G}) = \frac{\left|\{e_{uv} \mid e_{uv} \in \mathcal{E}, Z_{u,:} = Z_{v,:}\}\right|}{|\mathcal{E}|}$$

To avoid sensitivity to imbalanced classes, the class homophily [15] is defined as

$$H_{\text{class}}(\mathcal{G}) = \frac{1}{C-1} \sum_{k=1}^{C} \left[h_k - \frac{\left|\{v \mid Z_{v,k} = 1\}\right|}{N}\right]_+, \quad h_k = \frac{\sum_{v \in \mathcal{V}} \left|\{u \mid Z_{v,k} = 1, u \in \mathcal{N}_v, Z_{u,:} = Z_{v,:}\}\right|}{\sum_{v \in \{v \mid Z_{v,k}=1\}} d_v}$$

where $[a]_+ = \max(a, 0)$; h_k is the class-wise homophily metric. The above measures only consider the label consistency of connected nodes but ignore the unconnected nodes. Stronger label consistency can potentially happen in unconnected nodes, in which case the edge bias is not necessarily beneficial for GNNs. Aggregation homophily [18,19] tries to capture the post-aggregation node similarity and is proved to be better than the above homophily measures. But, it is not able to give a clear threshold value to determine when GNNs can outperform graph-agnostic NNs.

Connections and Differences among Terminologies. We draw the connections and differences among edge bias, homophily/heterophily and smoothness/non-smoothness, which are frequently used in the literature that might cause confusion. Edge bias or homophily/smoothness assumption is a major and strong condition that is taken for granted when designing GNN models. When the homophily/smooth assumption holds, edge bias will have positive effects for training GNNs; On the contrary, when heterophily/non-smoothness assumption holds, edge bias will cause negative effects. The fact that, the current measures of homophily/heterophily do not consider unconnected nodes, poses chanllenges to fully examine the effect of edge bias or if homophily/heterophily assumption holds. The edge bias might cause some other problems, *e.g.,* over-smoothing [13], loss of rank [21] and training difficulty [5,22], but we mainly discuss homophily/heterophily problem in this paper.

5 Conclusion

In this paper, we developed two measures, Normalized Total Variation (NTV) and Normalized Smoothness Value (NSV), which can predict and explain the expected performance of graph-agnostic MLPs and GNN models on graphs. These measures analyze the impact of edge bias on the features and labels of an attribute graph, helping to determine when graph-aware models will outperform graph-agnostic models. By conducting statistical hypothesis testing based on these measures, we are able to determine the threshold value for predicting the potential performance advantages of GNNs over NNs. Overall, our work contributes to a better understanding of the situations in which GNNs should be used, providing insights into the performance of GNNs compared to NNs on various real-world benchmark graph datasets.

A Details of NSV and Sample Covariance Matrix

The sample covariance matrix S is computed as follows

$$\begin{aligned} X &= \begin{bmatrix} \boldsymbol{x}_{1:} \\ \vdots \\ \boldsymbol{x}_{N:} \end{bmatrix}, \quad \bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_{i:} = \frac{1}{N}\mathbf{1}^T X, \\ S &= \frac{1}{N-1}\left(X - \mathbf{1}\bar{\boldsymbol{x}}\right)^\top \left(X - \mathbf{1}\bar{\boldsymbol{x}}\right) \end{aligned} \tag{12}$$

It is easy to verify that

$$\begin{aligned} S &= \frac{1}{N-1}\left(X - \frac{1}{N}\mathbf{1}\mathbf{1}^T X\right)^\top \left(X - \frac{1}{N}\mathbf{1}\mathbf{1}^T X\right) \\ &= \frac{1}{N-1}\left(X^T X - \frac{1}{N}X^T \mathbf{1}\mathbf{1}^T X\right) \\ &= \frac{1}{N(N-1)}\text{trace}\left(X^T(NI - \mathbf{1}\mathbf{1}^T)X\right) \\ &= \frac{1}{N(N-1)}\left(E_D^{\mathcal{G}}(X) + E_D^{\mathcal{G}^C}(X)\right) \end{aligned} \tag{13}$$

References

1. Ahmed, H.B., Dare, D., Boudraa, A.-O.: Graph signals classification using total variation and graph energy informations. In: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp. 667–671. IEEE (2017)
2. Battaglia, P.W., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)
3. Chen, S., Sandryhaila, A., Moura, J.M., Kovacevic, J.: Signal recovery on graphs: variation minimization. IEEE Trans. Signal Process. **63**(17), 4609–4624 (2015)
4. Chung, F.R.: Spectral Graph Theory, vol. 92. American Mathematical Soc. (1997)
5. Cong, W., Ramezani, M., Mahdavi, M.: On provable benefits of depth in training graph convolutional networks. Adv. Neural. Inf. Process. Syst. **34**, 9936–9949 (2021)
6. Daković, M., Stanković, L., Sejdić, E.: Local smoothness of graph signals. Math. Probl. Eng. **2019**, 1–14 (2019)
7. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Adv. Neural Inf. Process. Syst. **29** (2016)
8. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Adv. Neural Inf. Process. Syst. **30** (2017)
9. Hamilton, W.L.: Graph representation learning. Synth. Lect. Artif. Intell. Mach. Learn. **14**(3), 1–159 (2020)
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2016)
11. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
13. Li, Q., Han, Z., Wu, X.-M.: Deeper insights into graph convolutional networks for semi-supervised learning. Proc. AAAI Conf. Artif. Intell. **32** (2018)
14. Lim, D., et al.: Large scale learning on non-homophilous graphs: new benchmarks and strong simple methods. Adv. Neural. Inf. Process. Syst. **34**, 20887–20902 (2021)
15. Lim, D., Li, X., Hohne, F., Lim, S.-N.: New benchmarks for learning on non-homophilous graphs. arXiv preprint arXiv:2104.01404 (2021)
16. Liu, M., Wang, Z., Ji, S.: Non-local graph neural networks. arXiv preprint arXiv:2005.14612 (2020)

17. Luan, S.: On addressing the limitations of graph neural networks. arXiv preprint arXiv:2306.12640 (2023)
18. Luan, S., et al.: Is heterophily a real nightmare for graph neural networks to do node classification? arXiv preprint arXiv:2109.05641 (2021)
19. Luan, S., et al.: Revisiting heterophily for graph neural networks. Adv. Neural. Inf. Process. Syst. **35**, 1362–1375 (2022)
20. Luan, S., et al.: When do graph neural networks help with node classification: investigating the homophily principle on node distinguishability. Adv. Neural Inf. Process. Syst. **36** (2023)
21. Luan, S., Zhao, M., Chang, X.-W., Precup, D.: Break the ceiling: stronger multi-scale deep graph convolutional networks. Adv. Neural Inf. Process. Syst. **32** (2019)
22. Luan, S., Zhao, M., Chang, X.-W., Precup, D.: Training matters: unlocking potentials of deeper graph convolutional neural networks. arXiv preprint arXiv:2008.08838 (2020)
23. Luan, S., Zhao, M., Hua, C., Chang, X.-W., Precup, D.: Complete the missing half: augmenting aggregation filtering with diversification for graph convolutional networks. In: NeurIPS 2022 Workshop: New Frontiers in Graph Learning (2022)
24. Maehara, T.: Revisiting graph neural networks: all we have is low-pass filters. arXiv preprint arXiv:1905.09550 (2019)
25. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: homophily in social networks. Ann. Rev. Sociol. **27**(1), 415–444 (2001)
26. Pei, H, Wei, B., Chang, K.C.-C., Lei, Y., Yang, B.: Geom-gcn: geometric graph convolutional networks. In: International Conference on Learning Representations (2020)
27. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
28. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., Weinberger, K.: Simplifying graph convolutional networks. In: International Conference on Machine Learning, pp. 6861–6871. PMLR (2019)
29. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems, pp. 321–328 (2004)
30. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Generalizing graph neural networks beyond homophily. arXiv preprint arXiv:2006.11468 (2020)

Training Matters: Unlocking Potentials of Deeper Graph Convolutional Neural Networks

Sitao Luan[1,2(✉)], Mingde Zhao[1,2], Xiao-Wen Chang[1], and Doina Precup[1,2,3]

[1] McGill University, Montreal, Canada
{sitao.luan,mingde.zhao}@mail.mcgill.ca, {chang,dprecup}@cs.mcgill.ca
[2] Mila, Montreal, Canada
[3] DeepMind, London, UK

Abstract. The performance limit of deep Graph Convolutional Networks (GCNs) are pervasively thought to be caused by the inherent limitations of the GCN layers, such as their *insufficient expressive power*. However, if this were true, modifying only the training procedure for a given architecture would not likely to enhance performance. Contrary to this belief, our paper demonstrates several ways to achieve such improvements. We begin by highlighting the training challenges of GCNs from the perspective of graph signal energy loss. More specifically, we find that the loss of energy in the backward pass during training hinders the learning of the layers closer to the input. To address this, we propose several strategies to mitigate the training problem by slightly modifying the GCN operator, from the energy perspective. After empirical validation, we confirm that these changes of operator lead to significant decrease in the training difficulties and notable performance boost, without changing the composition of parameters. With these, we conclude that the root cause of the problem is more likely the *training difficulty* than the others.

1 Introduction

As a structure that is capable of modeling relational information [7,10,14,16,28], graph has inspired the emerge of Graph Neural Networks (GNNs), a machine learning paradigm that achieve state-of-the-art performance on complex tasks [2–4,7,16,19,20,22–25,27,32].

GCN [16], being arguably the most popular method of all GNNs, is applied pervasively for being lightweight and having relatively capable performance. However, the development of GCNs on more complicated tasks is hindered by the fact that their performance is still relatively limited and cannot be easily boosted: the capacity of GCN seems not scalable with the depth of the architectures, while the performance of typical deep learning architectures mostly becomes better with the increment of the depth. Several investigations about the possible cause of the problem have been carried out, including

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 49–60, 2024.
https://doi.org/10.1007/978-3-031-53468-3_5

- Oversmoothing Problem [18]: stacking aggregation operations in GNNs is shown to make the representation of connected nodes to be more indistinguishable and therefore causes information loss;
- Loss of rank [26]: the numerical ranks of the outputs in hidden layers will decrease with the increment of network depth.
- Inevitable convergence to some subspace [29]: the layer outputs get closer to a fixed subspace with the increment of the network depth;

These analyses show that despite the increment of trainable parameters, simply deepening GCNs is not helpful, therefore it is more promising to just switch to alternate solutions. Following these, efforts have been made to propose alternate GCN architectures to increase the expressive power with additional computational expenses, *e.g.,* augmenting architectures with layer concatenation operations [15,26]. However, the computational costs introduced often outweigh the performance boost, therefore no alternative is yet popular enough to replace GCN.

The intractability of deep GCNs naturally leads to the belief that deeper GCNs *cannot be trained well* and *cannot have better performance* without the change of architectures. However, in this paper, we question such idea and argue that the crucial factor limiting the performance of GCN architectures is more likely to be the *difficulty in training* instead of *insufficient expressive power*. First, from graph signal energy perspective, we prove that, during training, the energy loss during backward pass makes the training of layers that are closer to the input difficult. Then, we show both in theory and in experiments, it is actually possible, in several ways, to significantly lower training difficulty and gain notable performance boost by only changing slightly the training process of deep GCN, without changing the expressive power. These observations lead us to the discovery that the performance limit of GCN is more likely to be caused by inappropriate training rather than GCNs being inherently incapable.

The methodologies we propose in this paper includes Topology Rescaling (TR) for the graph operator (*e.g.,* graph Laplacian), weight normalization, energy normalization and weight initialization for enhancing the training of the parameters in the layers, as well as skip (residual) connections that do not use concatenation of layer outputs, *i.e.* no additional parameters.

The paper is organized as follows. In Sect. 2, we introduce backgrounds of graph Laplacian, graph partition and graph signal energy. In Sect. 3, we analyze from the perspective of energy and gradient, arguing that *the energy loss of the backward pass during training* leads to training difficulty, which we will in the end verify as the core factor limiting the performance of GNNs. In Sect. 4, we propose 4 methodologies that addresses the training difficulty problem from different perspectives. In Sect. 5, we validate the effectiveness of the methods in lowering the training difficulty.

2 Preliminaries

We use bold fonts for vectors (*e.g.,* $\boldsymbol{v}$), block vectors (*e.g.,* $\boldsymbol{V}$) and matrix blocks (*e.g.,* $\boldsymbol{V_i}$). Suppose we have an undirected connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ without a

bipartite component, where $\mathcal{V}$ is the node set with $|\mathcal{V}| = N$, $\mathcal{E}$ is the edge set with $|\mathcal{E}| = E$. Let $A \in \mathbb{R}^{N\times N}$ be the adjacency matrix of $\mathcal{G}$, *i.e.* $A_{ij} = 1$ for $e_{ij} \in \mathcal{E}$ and $A_{ij} = 0$ otherwise. The graph Laplacian is defined as $L = D - A$, where D is a diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$. The symmetric normalized Laplacian is defined as $L_{\text{sym}} = I - D^{-1/2}AD^{-1/2}$ with eigenvalues $\lambda(L_{\text{sym}}) \in [0, 2)$ and its renormalized version is defined as

$$\tilde{L}_{\text{sym}} = I - \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}, \quad \tilde{A} = A + I, \quad \tilde{D} = \text{diag}(\tilde{D}_{ii}), \quad \tilde{D}_{ii} = \textstyle\sum_j \tilde{A}_{ij} \tag{1}$$

and its eigenvalues $\lambda(\tilde{L}_{\text{sym}}) \in [0, 2)$ [5].

The eigendecomposition of L gives us $L = U\Lambda U^{-1}$, where $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N] \in \mathbb{R}^{N\times N}$ is formed by the orthonormal eigenvectors, referred to as the *graph Fourier basis*, and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is formed by the eigenvalues, which are nonnegative and are referred to as *frequencies*. Traditionally, graph Fourier basis is defined specifically by eigenvectors of L, but in this paper, graph Fourier basis is formed by eigenvectors of the Laplacian we use. The smaller eigenvalue λ_i indicates larger global smoothness of $\boldsymbol{u}_i$ [6], which means any two elements of $\boldsymbol{u_i}$ corresponding to two directly connected nodes will have similar values. Thus, $\boldsymbol{u_i}$ with small λ_i tends to partition the graph into large communities. This property is crucial for later analysis.

A graph signal is a vector $\boldsymbol{x} \in \mathbb{R}^N$ defined on $\mathcal{V}$, where x_i is defined on the node i. We also have a feature matrix (graph signals) $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ whose columns are graph signals and each node i has a feature vector $\boldsymbol{X_{i,:}}$, which is the i-th row of $\boldsymbol{X}$. The graph Fourier transform of the graph signal $\boldsymbol{x}$ is defined as $\boldsymbol{x}_{\mathcal{F}} = U^{-1}\boldsymbol{x} = U^T\boldsymbol{x} = [\boldsymbol{u}_1^T\boldsymbol{x}, \ldots, \boldsymbol{u}_N^T\boldsymbol{x}]^T$, where $\boldsymbol{u}_i^T\boldsymbol{x}$ is the component of $\boldsymbol{x}$ in the direction of $\boldsymbol{u_i}$.

In addition to various graph Laplacians, various affinity matrices derived from graph Laplacians have been adopted in GNNs. The most widely used one is the renormalized affinity matrix

$$\hat{A} \equiv I - \tilde{L}_{sym} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$$

with $\lambda(\hat{A}) = 1 - \lambda(\tilde{L}_{sym}) \in (-1, 1]$, and it is used in GCN [16] as follows

$$\boldsymbol{Y} = \text{softmax}(\hat{A}\ \text{ReLU}(\hat{A}\boldsymbol{X}W_0)\ W_1) \tag{2}$$

where $W_0 \in \mathbb{R}^{F\times F_1}$ and $W_1 \in \mathbb{R}^{F_1\times O}$ are parameter matrices.

Definition 1. *(Energy of signal on graph [9, 33]) For a signal $\boldsymbol{x}$ defined on graph $\mathcal{G}$, its energy is defined as $\|\boldsymbol{x}_{\mathcal{F}}\|_2^2$, where $x_{\mathcal{F}}$ is the graph Fourier transform of $\boldsymbol{x}$.*

The energy represents the intensity of a graph signal projected onto the frequency domain. However, considering undirected graph $\mathcal{G}$, the graph Laplacian is symmetric and the graph Fourier basis matrix is orthogonal, leading to $\|\boldsymbol{x}_{\mathcal{F}}\|_2^2 = \|\boldsymbol{x}\|_2^2 = \sum_i {x_i}^2$, which depends on only the signal itself.

Definition 2. *(Energy-preserving operator [9] or isometric operator[11]) An operator Φ defined on graph signal is energy-preserving if for any graph signal $\boldsymbol{x}$, it satisfies $\|(\Phi\boldsymbol{x})_{\mathcal{F}}\|_2^2 = \|\boldsymbol{x}_{\mathcal{F}}\|_2^2$.*

The energy-preserving property means the operator does not change the energy intensity in the frequency domain after being applied on graph signals.

3 Energy Loss During Back Propagation

In this section, we first show that ReLU($\hat{A}\cdot$) is an *energy-losing* operator. This property is the natural explanation for the over-smoothing [18], loss of rank[26] and loss of expressive power [29] phenomena, from which deep GCN will suffer during feed-forward process. According to the above analysis, the top layers will lose signal energy more serious than bottom layers. However, we will show that, contrary to our empirical intuition, deep GCNs lose energy in bottom layers instead of in top layers. Rather than investigating from feed-forward perspective, we will explain this contradiction from backward view by analyzing the gradient propagation in the following section.

3.1 Forward Pass Analyses: Difficult and Complicated

Theorem 1. *1 For any undirected connected graph $\mathcal{G}$, ReLU($\hat{A}\cdot$) is an energy-losing graph operator, i.e., for any graph signal $\boldsymbol{x}$*

$$\left\| \left(ReLU(\hat{A}\boldsymbol{x}) \right)_{\mathcal{F}} \right\|_2^2 \leq \|\boldsymbol{x}_{\mathcal{F}}\|_2^2$$

The strict inequality holds for any $\boldsymbol{x}$ which is independent of $[\tilde{D}_{11}^{1/2}, \ldots, \tilde{D}_{NN}^{1/2}]^T$, where $\tilde{D}_{ii}$ for $i = 1, \ldots, N$ are defined in (1)

Through the forward analysis of energy flow in deep GCN, we can see that the energy of column features should reduce in top layers. But from Fig. 1(a)–(c) yielded by a numerical test with 10-layer GCN, we can see that the energy of column features in top layers (Fig. 1(c)) do not have significant changes, while in bottom layers (Fig. 1(a), (b)) the energy of features shrinks during training. The cause of this contradiction is that we either have neglected [18] or have put too strong assumptions [26,29] on parameter matrices in forward analysis while ignore how parameter matrices changes in backpropagation. In the following, we will try to do gradient analysis from backward view and explain the energy loss in bottom layers in deep GCN.

3.2 Backward Pass Analyses: Identifying the Core Problem

We first decompose the deep GCN architecture as follows

$$\begin{aligned}
&\boldsymbol{Y_0} = \boldsymbol{X}, \quad \boldsymbol{Y_1'} = \hat{A}\boldsymbol{X}W_0, \quad \boldsymbol{Y_1} = \mathrm{ReLU}(\hat{A}\boldsymbol{X}W_0) = \mathrm{ReLU}(\boldsymbol{Y_1'}) = \mathbf{1}_{\mathbb{R}^+}(\boldsymbol{Y_1'}) \odot \boldsymbol{Y_1'} \\
&\boldsymbol{Y_{i+1}'} = \hat{A}\boldsymbol{Y_i}W_i, \quad \boldsymbol{Y_{i+1}} = \mathrm{ReLU}(\boldsymbol{Y_{i+1}'}) = \mathbf{1}_{\mathbb{R}^+}(\boldsymbol{Y_{i+1}'}) \odot \boldsymbol{Y_{i+1}'}, \quad i = 1, \ldots, n \\
&\boldsymbol{Y} = \mathrm{softmax}(\hat{A}\boldsymbol{Y_n}W_n) \equiv \mathrm{softmax}(\boldsymbol{Y'}), \quad l = -\mathrm{trace}(\boldsymbol{Z}^T \log \boldsymbol{Y})
\end{aligned} \tag{3}$$

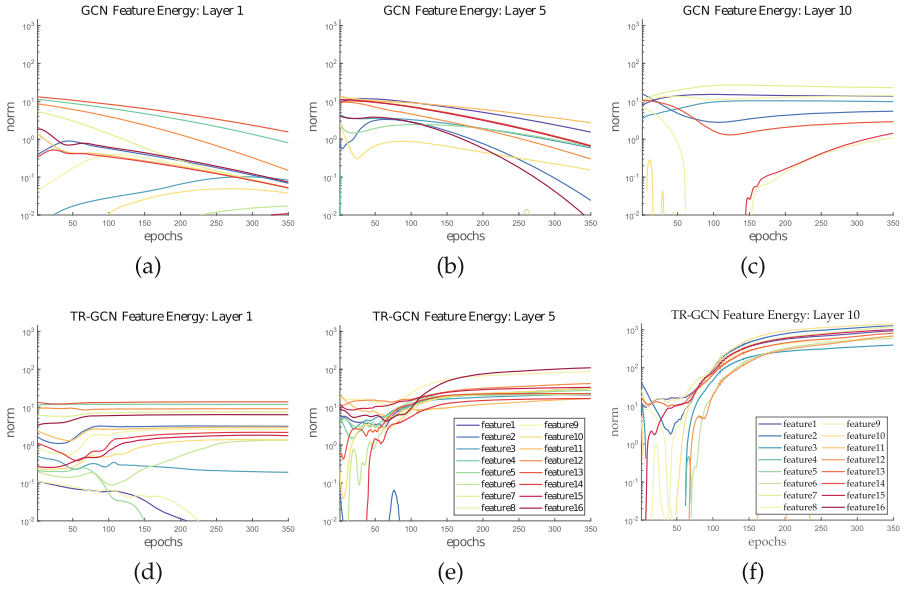


Fig. 1. Comparison of energy changes in hidden layers of GCN and TR-GCN ($r = 1$) during training

where $\mathbf{1}_{\mathbb{R}^+}(\cdot)$ and $\log(\cdot)$ are pointwise indicator and log functions; $\odot$ is the Hadamard product; $\boldsymbol{Z} \in \mathbb{R}^{N \times C}$ is the ground truth matrix with one-hot label vector $\boldsymbol{Z}_{i,:}$ in each row, C is number of classes; l is the scalar loss. Then the gradient propagates in the following way,

$$\begin{aligned} &\text{Output Layer} \quad \frac{\partial l}{\partial \boldsymbol{Y'}} = \text{softmax}(\boldsymbol{Y'}) - \boldsymbol{Z}, \quad \frac{\partial l}{\partial W_n} = \boldsymbol{Y_n}^T \hat{A} \frac{\partial l}{\partial \boldsymbol{Y'}}, \quad \frac{\partial l}{\partial \boldsymbol{Y_n}} = \hat{A} \frac{\partial l}{\partial \boldsymbol{Y'}} W_n^T \\ &\text{Hidden Layers} \quad \frac{\partial l}{\partial \boldsymbol{Y_i'}} = \frac{\partial l}{\partial \boldsymbol{Y_i}} \odot \mathbf{1}_{\mathbb{R}^+}(\boldsymbol{Y_i'}), \quad \frac{\partial l}{\partial W_{i-1}} = \boldsymbol{Y_{i-1}}^T \hat{A} \frac{\partial l}{\partial \boldsymbol{Y_i'}}, \quad \frac{\partial l}{\partial \boldsymbol{Y_{i-1}}} = \hat{A} \frac{\partial l}{\partial \boldsymbol{Y_i'}} W_{i-1}^T \end{aligned} \tag{4}$$

The gradient propagation of GCN differs from that of multi-layer perceptron (MLP) by an extra multiplication of $\hat{A}$ when the gradient signal flows through $\boldsymbol{Y_i}$. Since $\left|\lambda_i(\hat{A})\right| \leq 1$, this multiplication will cause energy loss of gradient signal (see Fig. 2(c)). In addition, oversmoothing does not only happen in feed-forward process, but also exists in backpropagation when we see $\frac{\partial l}{\partial \boldsymbol{Y_{i-1}}} = \hat{A} \frac{\partial l}{\partial \boldsymbol{Y_i'}} W_{i-1}^T$ as a backward view of hidden layers as (3). In forward view, parameter matrix W_i is fixed and we update $\boldsymbol{Y_i}$; in backward view, $\boldsymbol{Y_i}$ is fixed and we update W_i. And the difference is in forward view, the input $\boldsymbol{X}$ is a fixed feature matrix, but in backward view, the scale of the input $\frac{\partial l}{\partial \boldsymbol{Y'}} = \text{softmax}(\boldsymbol{Y'}) - \boldsymbol{Z}$ (the prediction error) is getting smaller during training. Thus, the energy loss is more significant in W_i (see Fig. 2(a)) from backward view and is more serious in bottom layers instead of in top layers.

This energy losing phenomenon is not an expressive power problem but a training issue. But this does not mean the training issue is the root cause of the

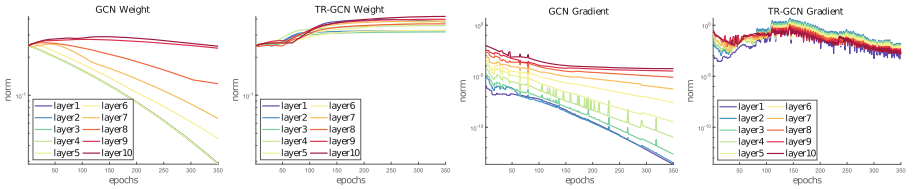


Fig. 2. Comparison of weight and gradient norm in hidden layers of GCN and TR-GCN ($r = 1$): the pairs have the same x- and y-ranges.

performance limit problem, which we will draw conclusion later. In the following section, we propose method to alleviate the energy loss.

4 Methods to Alleviate BP Energy Loss

In this section, we propose 4 methodologies to handle the problem of BP energy loss: *spectra shift*, *weight initialization*, *normalization* and *skip (residual) connection*.

4.1 Spectra Shift and Topology Rescaling (TR)

From the analysis in Sect. 3 and theorem 1, we can that $\left|\lambda_i(\hat{A})\right| \leq 1$ is one of the main reasons of energy losing. To adjust $\left|\lambda_i(\hat{A})\right|$ while maintaining certain topological properties of the original graph associated with $\hat{A}$ (*e.g.,* the graph Fourier basis, the gap between the frequencies), we shift the spectra of $\hat{A}$ by changing $\hat{A}$ to $\hat{A}_r = rI + \hat{A}$, where r is a real scalar.

Physical Meaning. Spectra shift is also a commonly used method in community detection [1] to address the so-called "resolution limit" challenge [8,13,36], *i.e.* it can only produce the modules at a certain characteristic scale [34] while is unable to extract densely connected substructures with small sizes. Spectra shift rescales the graph topology with a proper self-loop assignment through which we can adjust the strength (degree) of each node [34] and r is named resolution parameter. The translation of strengths has no impact on the original connection of nodes, which are the building blocks of the topology. The shift only balance the property of each node individually and in the same way for all of them.

Spectra shift essentially allows the graph operator to adjusts the scale of the components of graph signal in graph frequency domain. To see this, suppose that $\hat{A}$ has the eigendecomposition $\hat{A} = \hat{U}\hat{\Lambda}\hat{U}^T$, where $\hat{U}$ is orthogonal. Then

$$\boldsymbol{x} = \sum_i \hat{\boldsymbol{u}}_i(\hat{\boldsymbol{u}}_i^T\boldsymbol{x}), \quad \hat{A}\boldsymbol{x} = \sum_i \hat{\lambda}_i\hat{\boldsymbol{u}}_i(\hat{\boldsymbol{u}}_i^T\boldsymbol{x}), \quad \hat{A}_r\boldsymbol{x} = \sum_i (\hat{\lambda}_i + r)\hat{\boldsymbol{u}}_i(\hat{\boldsymbol{u}}_i^T\boldsymbol{x}) \tag{5}$$

Note that the components of $\boldsymbol{x}$, $\hat{A}\boldsymbol{x}$ and $\hat{A}_r\boldsymbol{x}$ in the direction $\hat{\boldsymbol{u}}_i$ are $\hat{\boldsymbol{u}}_i^T\boldsymbol{x}$, $\hat{\lambda}_i\hat{\boldsymbol{u}}_i^T\boldsymbol{x}$, and $(\hat{\lambda}_i + r)\tilde{\boldsymbol{u}}_i^T\boldsymbol{x}$, respectively. Thus applying the operator $\hat{A}$ to $\boldsymbol{x}$ just scales the component of $\boldsymbol{x}$ in the direction of $\boldsymbol{u}_i$ by $\hat{\lambda}_i$ for each i.

Tuning the resolution parameter r actually rescales those components in the way that global information (high smoothness) will be increased with positive r, and local information (low smoothness) will be enhanced with negative r. The GCN with $\hat{A}_r$ is called topology rescaling GCN (TR-GCN).

Note that $\lambda_i(\hat{A}) \in (-1, 1]$. A shift which makes $\max_i \left|\lambda_i(\hat{A}) + r\right| \geq 1$ is considered risky because it will cause gradient exploding and numerical instability during training as stated in [16]. However, through our analysis, TR-GCN will not only overcome the difficulty when training in deep architecture (see Fig. 1(d)–(f) and Fig. 2(b), (d)), but also will not lose expressive power (see Table 1) by setting a proper r (depends on the task and size of the network).

4.2 Weight Initialization

The gradient propagation does not only depends on $\hat{A}$ but also depends on the scale of W_i. An initialization with proper scale would make W_i get undiminished gradient from the start of training and move to the correct direction with a clearer signal [21]. Thus, we adjust the scale of each element in W_i initialized by [12] with a tunable constant λ_{init} as follows,

$$\lambda_{\text{init}} \times U(-\frac{1}{\sqrt{F_{i+1}}}, \frac{1}{\sqrt{F_{i+1}}}) \quad \text{or} \quad \lambda_{\text{init}} \times N(0, \sqrt{\frac{2}{F_i + F_{i+1}}}) \tag{6}$$

4.3 Normalization

Normalization is a natural method to control the energy flow. A direct method would be to normalize the output matrix in each hidden layer with a constant λ_E; Or, an indirect method can also be used to normalize the weight matrix [30] by a constant λ_W, which shares the same spirit of normalizing the largest singular value of W_i [29].

$$\begin{aligned} &\text{Energy Normalization: } \boldsymbol{Y_i} = \lambda_E \cdot \boldsymbol{Y_i} / \left\|\boldsymbol{Y_i}\right\|_2 \\ &\text{Weight Normalization: } W_i = \lambda_W \cdot W_i / \left\|W_i\right\|_2 \end{aligned} \tag{7}$$

4.4 Skip Connection

Skip (residual) connections [15] is a widely used technique in training deep neural networks and has achieved success in feature extraction. It helps with gradient propagation without introducing additional parameters. Skip connections, if adapted in GCNs, will have the general form as follows:

$$\boldsymbol{Y}_{i+1} = \boldsymbol{Y}_i + \sigma(\hat{A}\boldsymbol{Y}_i W_i) \tag{8}$$

where σ is the activation function, $\boldsymbol{Y}_i$ is the input of the i-th layer and $\boldsymbol{Y}_{i+1}$ is the output of the i-th layer as well as the input of the $(i+1)$-th layer.

It is shown that existing GCN models are difficult to train when they are scaled with more than 7-layer-deep. This is possible due to the increase of the effective context size of each node and overfitting issue as stated in [16]. There exists one method ResGCN [17] that seeks also to address such problem via residual connections, but it actually uses concatenation of the intermediate outputs of hidden layers, introducing excessive parameters. The effectiveness shown in experiments are actually not only the result of the skip-connections but also the expressive power of additional parameters. However, in experiments, we will show that residual connections alone could accomplish the task.

5 Experiments

This section is crucial to the paper's main hypothesis: can we boost the performance of GNNs by just training them better? For this purpose, we patch the most-popular baseline GCN with the ideas in previous section to form a set of detailed comparative tests and fix the architecture to be 10-layers deep throughout the entire section[1]. Particularly, we have selected the node classification tasks on Cora, CiteSeer and PubMed, the three most popular datasets. We use the most classic setting on training, which is identical to the one suggested in [35]. The section features two sets of experiments, the first of which validates the effectiveness of the proposed methods lowering the training difficulty while the second demonstrates the potential performance boost when the patched methods are fine-tuned. For all experiments, we used Adam optimizer and ReLU as the activation function (PyTorch implementation).

5.1 Training Difficulty and Generalization

Instead of demonstrating how good the performance of the patched method could possibly be, the first set of experiments focuses on validating the effectiveness of the proposed ideas aiming to lower the training difficulty with a detailed ablation study. Also, we investigate the potential loss of generalization abilities, *i.e.* whether these ideas lead to overfitting.

For fair comparison, we use the same base architecture for the baseline and all the patched methods: 10 GCN layers each with width 16. Also, we utilize the same set of basic hyperparameters: a learning rate of 0.001, weight decay of 5×10^{-4}, 0 dropout. We train all methods to the same extent by using the same training procedures for all the methods: each method in each run is trained until the validation loss is not improved for 200 epochs.

With these, we run each method on Cora dataset with public split (20 training data for each class) for 20 independent runs and obtain the final reported

[1] The source code will be submitted within the supplementary materials for blind review and open-source afterwards.

classification accuracy together with the standard deviation of the accuracy. The results also include the errors (losses) computed on the training set and the test set. The results are reported in Table 1, together with the hyperparameters included additionally by the patched methods. Note that these hyperparameters are not fine-tuned. Also, since all methods are trained with the same base loss (negative log-likelihood) and additional losses introduced by the patched methods only increase the total loss, the comparison of loss among the methods can fairly tell that the patched methods' ability of lowering the training difficulty if their training losses are lower than the baseline.

Table 1. Ablation Tests for Training Difficulties on Cora

Train Loss		*Train Acc*		*Test Loss*		*Test Acc*		Change L	Change All	Change W, b				
Mean	Std	Mean	Std	Mean	Std	Mean	Std	resolution	skip	weight norm	energy norm	weight init	weight const	
1.946	0.000	14.29%	0.00%	1.960	0.035	23.11%	8.80%		N			uniform		
0.004	0.008	99.93%	0.21%	4.608	3.244	57.10%	7.85%	1.00	N			uniform		
0.106	0.101	98.07%	2.14%	1.806	0.485	67.45%	4.37%		N			normal	1.8	
0.005	0.010	100.00%	0.00%	2.908	1.838	65.13%	4.53%	1.00	N			normal	0.8	
0.811	0.795	71.93%	28.03%	1.184	0.306	64.68%	9.94%		N	7		uniform		
0.011	0.011	100.00%	0.00%	1.088	0.105	69.72%	2.55%		N		800	uniform		
0.359	0.361	89.79%	16.61%	1.562	0.328	64.35%	4.65%	1.00	N	5		uniform		
0.002	0.003	100.00%	0.00%	1.912	0.627	62.08%	2.95%	1.00	N		550	uniform		
0.008	0.009	99.93%	0.21%	1.723	1.045	68.15%	5.29%	1.00	Y			uniform		
0.034	0.024	99.79%	0.46%	1.318	0.530	72.30%	2.59%		Y			normal	0.9	
0.378	0.194	95.43%	2.77%	1.009	0.146	73.98%	2.68%		Y	7		uniform		
0.003	0.003	100.00%	0.00%	1.543	0.488	71.28%	2.95%		Y		2900	uniform		
0.001	0.001	100.00%	0.00%	1.969	0.811	67.58%	4.93%	1.00	Y			uniform		
0.005	0.003	100.00%	0.00%	1.447	0.498	69.32%	2.83%	1.00	Y			normal	0.5	
0.193	0.115	98.50%	1.08%	1.247	0.233	68.48%	3.72%	1.00	Y	3		uniform		
0.080	0.041	100.00%	0.00%	2.074	0.218	70.52%	2.39%	1.00	Y		325	uniform		

Each row represents a method. The first four columns are featured with color indicators: the greener the better result, the redder the worse. The changes applied unto the baseline are highlighted in the later columns. The first row has no colored changes and is therefore the baseline. We use different highlight colors to indicate the change on the operators: blue for the changes on graph operator L, red for the changes on W and b and purple (blue + red) for the changes applied on all L, W and b.

From the results on the training set, we can observe significantly smaller training loss (more than 50%) and significantly higher training accuracy (more than 6 times), comparing those of the patched methods and the original baseline. Considering that all of the compared methods have exactly the same parameter composition, we can safely say that the proposed methods are indeed effective lowering the training difficulties. However, we cannot conclude from the results which single idea contributes the most to the training difficulty alleviation.

Comparing the results on the test set, we can see that the error and accuracy on the test set ruled out the argument of overfitting: generally all the losses and accuracy on the test set are improved significantly. With all the observations in this set of experiments, the validation of the hypothesis is finished: we can make GNNs perform better by training them better.

5.2 Finetuned Performance Boost

In this second set of experiments, we fine tune each method (including the baseline) and compare their best reported performance. This shows how much potential could be unlocked by better training procedures. The fine-tuning is

conducted with Bayesian optimization [31] to the same extent[2]. Each result reported in Table 2 is averaged from 20 independent runs together with the standard deviation[3].

Table 2. Fine-tuned Performance on Node Classification Tasks

Cora		*CiteSeer*		*PubMed*		Change L	Change All	Change W, b		
Mean	Std	Mean	Std	Mean	Std	resolution	skip	weight norm	energy norm	weight init
74.66%	1.37%	60.39%	2.67%	74.01%	1.40%					uniform
82.06%	*0.56%*	*71.54%*	*2.54%*	*78.48%*	*1.52%*					uniform
83.52%	***0.91%***	*73.64%*	*0.75%*	***79.20%***	***1.16%***					uniform
83.10%	*0.84%*	*73.74%*	*1.00%*	*78.92%*	*0.77%*					uniform
82.96%	*1.21%*	***73.84%***	***0.82%***	*78.76%*	*0.91%*					
83.52%	*0.51%*	*73.30%*	*1.33%*	*79.00%*	*0.67%*					uniform
82.92%	*0.71%*	*72.98%*	*1.34%*	*78.78%*	*0.59%*					uniform
82.16%	*0.88%*	*71.40%*	*1.35%*	*79.00%*	*1.05%*					

All the architectures are fixed with depth 10.

From the results in the table, we observe that the patched methods obtain statistically significant performance boost. Therefore, together with the observations from the previous set of experiments, we conclude that the proposed methods could indeed alleviate the performance limit problem by lowering the training difficulty.

6 Conclusion

In this paper, we verify the hypothesis that the cause of the performance limit problem of deep GCNs are more likely the training difficulty rather than insufficient capabilities. Out of the analyses on signal energy, we address the problem by proposing several methodologies that seek to mitigate the training process. The contribution enables lightweight GCN architectures to gain better performance when stacked deeper.

Though the proposed methods show effectiveness in lowering the training loss and improving the performance in practice, the methods introduce additional hyperparameters that require tuning. In future works, we would investigate the possibilities of a learnable resolution (self-loop) in the graph operator that is optimized end-to-end together with the system, essentially turning meta-learning

[2] All methods are fixed 10-layer deep. Methods share the same search range for the base hyperparameters (learning rate in $[10^{-6}, 10^{-1}]$, weight decay in $[10^{-5}, 10^{-1}]$, width in $\{100, 200, \ldots, 5000\}$, dropout in $(0, 1)$). The hyperparameters unique to the patched methods are also fixed for each patched method (resolution in $[-1, 5]$, weight constant in $[0.1, 5]$, weight normalization coefficient in $[1, 15]$, energy normalization coefficient in $[25, 2500]$). The search stops if the performance is not improved for 64 candidates.

[3] GCN is reproduced and performed fine-tuning upon.

the self-loop that guides the representation learning on graphs. Also, we would like to seek other possible theoretically-inspired approaches to alleviate training difficulties.

References

1. Arenas, A., Fernandez, A., Gomez, S.: Analysis of the structure of complex networks at different resolution levels. New J. Phys. **10**(5), 053039 (2008)
2. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. arXiv preprint arXiv:1611.08097 (2016)
3. Chen, J., Ma, T., Xiao, C.: Fastgcn: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
4. Chen, J, Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. arXiv preprint arXiv:1710.10568 (2017)
5. Chung, F.R., Graham, F.C.: Spectral Graph Theory. vol. 92. American Mathematical Society (1997)
6. Daković, M., Stanković, L., Sejdić, E.: Local smoothness of graph signals. Math. Probl. Eng. **2019** (2019)
7. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. arXiv preprint arXiv:1606.09375 (2016)
8. Fortunato, S., Barthelemy, M.: Resolution limit in community detection. Proc. Natl. Acad. Sci. **104**(1), 36–41 (2007)
9. Gavili, A., Zhang, X.-P.: On the shift operator, graph frequency, and optimal filtering in graph signal processing. IEEE Trans. Signal Process. **65**(23), 6303–6318 (2017)
10. Gilmer, J, Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 1263–1272 (2017). JMLR. org
11. Girault, B., Gonçalves, P., Fleury, É.: Translation on graphs: an isometric shift operator. IEEE Signal Process. Lett. **22**(12), 2416–2420 (2015)
12. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256 (2010)
13. Good, B.H., De Montjoye, Y.-A., Clauset, A.: Performance of modularity maximization in practical contexts. Phys. Rev. E **81**(4), 046106 (2010)
14. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. arXiv preprint arXiv:1706.02216 (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
17. Li, G, Müller, M., Thabet, A., Ghanem, B.: Can GCNs go as deep as CNNs? arXiv preprint arXiv:1904.03751 (2019)
18. Li, Q., Han, Z., Wu, X.: Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. arXiv preprint arXiv:1801.07606 (2018)
19. Liao, R., Zhao, Z., Urtasun, R., Zemel, R.S.: Lanczosnet: multi-scale deep graph convolutional networks. arXiv preprint arXiv:1901.01484 (2019)

20. Lim, D., et al.: Large scale learning on non-homophilous graphs: new benchmarks and strong simple methods. Adv. Neural. Inf. Process. Syst. **34**, 20887–20902 (2021)
21. Luan, S.: On addressing the limitations of graph neural networks. arXiv preprint arXiv:2306.12640 (2023)
22. Luan, S., Hua, C., Lu, Q., Zhu, J., Chang, X.-W., Precup, D.: When do we need GNN for node classification? arXiv preprint arXiv:2210.16979 (2022)
23. Luan, S., et al.: Is heterophily a real nightmare for graph neural networks to do node classification? arXiv preprint arXiv:2109.05641 (2021)
24. Luan, S., et al.: Revisiting heterophily for graph neural networks. Adv. Neural. Inf. Process. Syst. **35**, 1362–1375 (2022)
25. Luan, S., et al.: When do graph neural networks help with node classification: Investigating the homophily principle on node distinguishability. Adv. Neural Inf. Process. Syst. **36** (2023)
26. Luan, S., Zhao, M., Chang, X.-W., Precup, D.: Break the ceiling: stronger multi-scale deep graph convolutional networks. Adv. Neural Inf. Process. Syst. **32** (2019)
27. Luan, S., Zhao, M., Hua, C., Chang, X.-W., Precup, D.: Complete the missing half: augmenting aggregation filtering with diversification for graph convolutional networks. In: NeurIPS 2022 Workshop: New Frontiers in Graph Learning (2022)
28. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5115–5124 (2017)
29. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. arXiv preprint arXiv:1905.10947 (2019)
30. Salimans, T., Kingma, D.P.: Weight normalization: a simple reparameterization to accelerate training of deep neural networks. Adv. Neural Inf. Process. Syst. 901–909 (2016)
31. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., de Freitas, N.: Taking the human out of the loop: a review of bayesian optimization. Proc. IEEE **104**(1), 148–175 (2016)
32. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. arXiv preprint arXiv:1211.0053 (2012)
33. Stanković, L., Sejdić, E., Daković, M.: Reduced interference vertex-frequency distributions. IEEE Signal Process. Lett. **25**(9), 1393–1397 (2018)
34. Xiang, J., Tet al.: Multi-resolution community detection based on generalized self-loop rescaling strategy. Physica A **432**, 127–139 (2015)
35. Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. arXiv preprint arXiv:1603.08861 (2016)
36. Zhang, X.-S., et al.: Modularity optimization in community detection of complex networks. EPL (Europhys. Lett.) **87**(3), 38002 (2009)

E-MIGAN: Tackling Cold-Start Challenges in Recommender Systems

Ahlem Drif[1(✉)] and Hocine Cherifi[2]

[1] Computer Sciences Department, Faculty of Sciences, Setif 1 University, Setif, Algeria
adrif@univ-setif.dz

[2] Laboratoire Interdisciplinaire Carnot de Bourgogne (ICB), UMR 6303, CNRS, University of Burgundy, Dijon, France

Abstract. A recommender system based on Graph Neural Networks can effectively capture user-item interactions through the graph structure, leading to highly personalized and relevant recommendations. However, existing works adapting Graph Neural Networks (GNN) to recommendations struggle with the cold-start problem. Indeed, it is difficult to make accurate recommendations for new users or items with little or no interaction data. Building on previous work, we introduce an Enhanced Mutual Interaction Graph Attention Network (E-MIGAN) for this purpose. It is based on self-supervised representation learning on a large-scale bipartite graph. It is composed of three components: i) The attention network module that learns attention weights for each node and its neighbors, ii) The mutual interaction module computes a mutual interaction matrix for each node and its neighbors on each item, which encodes the pairwise interactions, and iii) A Content-Based Embedding model, which overcomes the cold start issue. The empirical study on real-world datasets proves that E-MIGAN achieves state-of-the-art performance, demonstrating its effectiveness in capturing complex interactions in graph-structured data.

Keywords: Hybrid recommender systems · mutual influence · Graph Attention Network · Collaborative Filtering · Content Based Filtering · Graph Neural Networks (GNN)

1 Introduction

Hybrid models that combine multiple recommendation techniques have gained popularity in recent years [1–7]. For instance, a hybrid model can combine collaborative and content-based filtering to address the cold-start problem. Additionally, it can be extended to incorporate graph-based techniques such as GNNs. Graph neural network (GNN) has become a new state-of-art approach for recommender systems. The central concept behind GNN is an information propagation mechanism, i.e., to iteratively aggregate feature information from neighbors in graphs. The neighborhood aggregation mechanism enables GNN to model

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 61–73, 2024.
https://doi.org/10.1007/978-3-031-53468-3_6

the correlation among users, items, and related features [8]. Graph Convolutional Matrix Completion (GCMC) [9] and Neural Graph Collaborative Filtering (NGCF) [10] already address several issues in the world of recommendation. However, they struggle with modeling higher-order feature interactions, relying heavily on past user interactions for predictions. To address this limitation, we developed a Mutual Interaction Graph Attention Network recommender model [11] that allows mapping the original data to higher-order feature interactions. It models the mutual influence relationship between aspect users and items. Furthermore, MIGAN is designed to handle large-scale datasets efficiently. Despite this, it also has limitations, which motivates the current proposed approach.

This work proposes an Enhancing Mutual Interaction Graph Attention Network recommender model for the Item-wise cold-start problem. The main contributions of the Enhancing-MIGAN recommended model can be described as compared to the related works as follows:

- Like other collaborative filtering recommenders, MIGAN struggles with the *item-cold-start* problem. Making accurate recommendations for new users or items with little or no interaction data is difficult. The proposed hybrid model combines the forces of content-based and collaborative filtering recommenders to achieve precise and relevant recommendations. The content-based approach we suggest implementing here is a stacked recommender comprising a set of regression models and a meta-learner.
- The stacked content-based recommender comprises a stack of machine learning (ML) models. This recommender creates a profile model for each user and extracts valuable features from the item-based side information. Its constituents allow for gaining predictive accuracy.
- The interactive graph neural attention network recommender exploits the encoding ability of the interactive attention between users and items. It learns the mutual influence generated by the contributions of items that carry collaborative signals on user decisions, helping to account for complex user-item interactions. Therefore, it boosts the accuracy of recommender systems by indicating which higher-order feature interactions are informative for the prediction.

The remainder of this paper is organized as follows. Section 2 presents a literature review of relevant neural graph recommender approaches. Section 3 discusses the proposed methodology. Section 4 reports the comparative analysis of the proposed *Content Enhanced-MIGAN* with some recent state-of-the-art approaches on real-world datasets. Finally, Sect. 5 presents the conclusion.

2 Related Work

Otunba et al. [12] stack a Generalized Matrix Factorization (GMF) and MLP ensemble to propagate the prediction from constituent models to the final output. Bao et al. [13] combine component recommendation engines, which are

considered wrappers for a set of smaller, concrete pre-trained models. The wrapper then follows a well-defined strategy to aggregate its predictions into a final one. Da et al. [14] proposed three ensemble approaches based on multimodal interactions. Unlike previous works with stacked recommenders, the stacking content-based recommender that we develop in this work creates a profile model for each user. Its main advantage is the ability of the embedding representation to integrate the side information in the hybrid architecture.

Many works on GNN-based recommendation systems have been proposed in the last few years. The most obvious explanation is that GNN techniques are effective at learning representations for graph data in various domains [15,16], and most of the data in recommendation has a graph structure. Graph Convolutional Networks (GCN) is one of the popular GNN models. They operate through a series of message-passing steps between nodes in the graph. At each stage, each node aggregates information from its neighbors applies a neural network layer to the aggregated information, and then sends the transformed data to its neighbors. This process is repeated for a fixed number of steps or until convergence is achieved [17]. *Graph Attention Network (GAT)* [18] uses a function called attention to selectively aggregate information from neighboring nodes in the graph. Unlike GCN, GAT can learn different weights for each neighboring node, allowing them to capture complex patterns in the graph structure. It makes GAT particularly useful for tasks where the relationships between nodes are highly non-linear and require a more fine-grained approach to modeling.

Knowledge Graph Attention Network (KGAT) [19] is based on GAT. It creates a heterogeneous graph with nodes comprising users, items, and attributes. It then aggregates and updates each node's embedding by iteratively propagating the embeddings from its neighbors.

Neural Graph Collaborative Filtering (NGCF) [10] is a representative and state-of-the-art GCN model for recommendation. It is used for Collaborative Filtering by propagating the user and item embeddings over the user-item graph, capturing connectivity between users and their neighbors. The authors in [20] report studies of the embedding quality refined by GCN and implemented three simplified variants of NGCF to get a better result. *Mutual-Interaction Graph Attention Network for collaborative filtering recommendation (MIGAN)* [11] model efficiently the interaction characteristic. The key idea is that MIGAN determines which weights best represent the users' mutual effect on the item. Building on MIGAN, the proposed model incorporates item content embeddings to graph-based MIGAN while tweaking its recommendation mechanism to address the item-wise cold start problem.

3 The Proposed Framework Enhanced-MIGAN

In general, the recommender system problem is centered around suggesting items relevant to the user from a big data pool. Therefore, the problem relies on three

principal components: a set of items I, a set of users U, and their interactions. We define Bipartite Graphs: Let $G = (U_1, I_1, E)$ be a bipartite graph. A bipartite graph comprises two independent sets of vertices, U_1 and I_1. The edges connect a vertex from one set U_1 to one in I_1.

As illustrated in Fig. 1, the proposed framework is an enhancement of MIGAN framework, by incorporating content-based embeddings from the metadata available about the items and generating user profiles. This novel variant of MIGAN takes another benefit from the potential of the content-based recommender system. Hence, its strengths are:

1. **Capturing complex relationships:** This framework learns the most relevant weights representing mutual influence on the item which can capture more complex relationships between user and item embeddings. This can be beneficial as the user-item interaction data exhibits more complex patterns.
2. **Incorporating additional features:** The content-based model orients the hybrid to learn each user's preferences concerning resources' traits, including but not limited to their type, genre, and content. This can help improve the recommendation quality by leveraging more information about users and items.

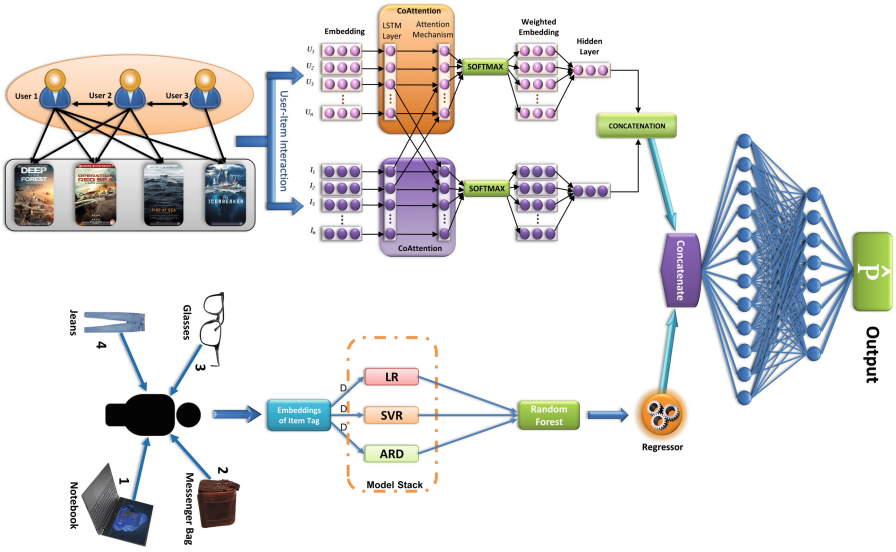


Fig. 1. The Architecture of the Content-Enhanced MIGAN. The content stacked recommender captures the side information to create a profile model for each user while optimizing the stack's learners' objective function. The collaborative filtering MIGAN recommender presents higher-order feature interactions.

3.1 Mutual-Interaction Graph Attention Network Recommender

The proposed framework's first component is a graph neural network design for modeling complex interactions in graph-structured data. MIGAN representation is based on the Bipartite Graph Neural Networks (BGNN) [21] to model the dependencies between the nodes on a large scale. This representation is fed up to a co-attention neural network recommender. Here, the developed co-attention layer puts more emphasis not only on learning the complex relationship between the target users (or items) and their neighbors but also it learning the most relevant weights that represent the users' mutual influence on the item. This idea is illustrated in Fig. 2.

The first recommender consists of two main operations: an attention network module and a mutual interaction module. The attention network module learns attention weights for each node and its neighbors. The mutual interaction module computes a mutual interaction matrix for each node and its neighbors on each item, which encodes the pairwise interactions between them. Then, MIGAN uses the mutual interaction matrix to compute a weighted sum of the node features.

Applying a co-attention mechanism in the context of collaborative filtering recommendation allows for discriminating the items that are interesting for users even those with no previous interaction, through deducing higher attention weights. The first embedding layers e_u and e_i captures latent features of users p_u and items q_i. Long-Short-Term-Memory (LSTM) layers follow them to enable long-range learning. Each LSTM state includes two inputs: the current feature vector and the output vector h_t_1 from the previous state. Its output vector is h_t. Each node embedding layer is chained with an LSTM layer which allows propagating without modification, updating, or resetting states using simple learned gating functions. The LSTM representation is expressed as follows:

$$h_{tu} = g_1(p) \tag{1}$$

$$h_{ti} = g_2(q) \tag{2}$$

The learned representation is H_p and H_q, respectively, with $d \times n$ dimensions for H_p and $d \times m$ for H_q. Users' and items' embedded inputs are projected into a vector representation space using the attention technique. For the interactive attention mechanism, we build an attention map to predict the distribution of the items. For this purpose, we compute a matrix $L = tanh(H_p^\top W_{pq} H_q)$, where $L \in \mathbb{R}^{n \times m}$, and W_{pq} is a $d \times d$ a learnable parameters matrix. The features of co-attention maps are defined as:

$$\begin{aligned} \alpha_p^* &= tanh(W_p H_p + (W_q H_q) L^\top) \\ \alpha_q^* &= tanh(W_q H_q + (W_p H_p) L) \end{aligned} \tag{3}$$

The interactive attention model uses a tangent function to model the mutual interactions between users and items. Afterward, we compute the probability

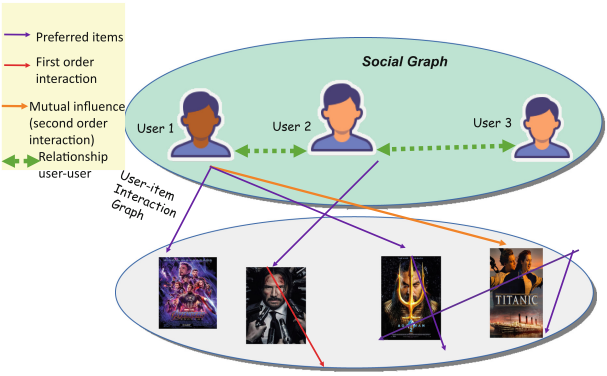


Fig. 2. Interaction between users and items with particular characteristics can reveal the possibility that an item is interesting for similar users. In this example, we can see that both user 1 and user 2 have watched the same movies, Avengers and Aquaman, which means that they have similar tastes. So if user 2 watched another movie, for example, John Wick, then there is a high probability that user 1 will like the same movie, so the MIGAN recommender recommends it to him. It is a first-order interaction. Moreover, one can deduce a mutual influence based on interactive attention weights at more than a first-order interaction level. For example, user 3 influences user 1, as user 1 shares similar preferences with user 2, generating a recommendation of the Titanic movie based on user 3 preferences.

distribution over the embedding space. The softmax function is used to generate the attention weights:

$$\alpha_u = Softmax(f(\alpha_p^*)) \tag{4}$$

$$\alpha_i = Softmax(f(\alpha_q^*)) \tag{5}$$

where f: is a multi-layer neural network.

Then, the high-order interaction latent space of users and items is given by:

$$f_1 = [\beta\iota_u \oplus \beta\iota_i] \tag{6}$$

where β_p and β_q: are the derived attention weights.

As a result, the predicted matrix $\hat{R_{ui}}$ is defined as:

$$\hat{R_{ui}} = f(f_1) \tag{7}$$

where f: is a dense layer using a sigmoid activation function. Finally, we train the model to minimize the loss function which is the Mean Absolute Error (MAE):

$$L(R_{ui}, \hat{R}_{ui}) = \frac{1}{|C|} \sum_{(u,i)\in C} |(R_{ui} - \hat{R_{ui}})| \tag{8}$$

3.2 Content-Based Embeddings

The content-based recommender system injects items' attributes as a deciding factor for recommendations. First, the recommender extracts "tags" from items' descriptions or reviews and uses them to calculate each item's embedding vector with the help of Stanford's pre-trained GloVe vectors [22]. A trained model is a user profile in a stacking ensemble learning technique for a content-based recommender system. Thus, it generates a user profile for each user u in the form of a learned model. The training procedure of this stacked content-based recommender is summarized in Algorithm 1. Let $S = \{S_1, S_2, ..., S_l\}$ be different regression models, and x_{train} be the training dataset. U and emb are independent variables, R is the dependent variable. The base regression models' hyperparameters are $\forall s \in S : \theta_s$.

Algorithm 1: Stacked content-based recommender

Input : U: list of user ids : size n
I: list of item ids: size m
D_i: Description of item i
I_U: All items a specific user has interacted with
θ_s: the model hyperparameters.
Output: $\hat{R}_{BL}$

```
begin
    // Calculating embeddings for each item
    foreach i ∈ I do
        Tags(i) ← ExtractKeywords(i) ;
        Embedding(i) ← GloVe(Tags(i))
    end
    // Generating user profiles
    profiles = ∅ ;
    foreach u ∈ U do
        regressors = InitRegressors(θ_reg) ;
        blender = InitBlender(θ_blender)
        // Training the stack
        features= Train(U, emb(i), R)
        profile= Build.model(regressors, blender,features) ;
    end
    // Constructing predicted utility matrix
    R̂_BL = [] ;
    foreach profile ∈ profiles do R̂_BL ← profile.Predict() ;
    return R̂_BL
end
```

Each $s \in S$ is trained separately with the same training dataset. Each model provides predictions for the outcomes (R), which are then cast into a meta-learner (blender). In other words, the S predictions of each regressor become

features for the blender. The latter can be any model such as linear regression, SVR, Decision Tree,...etc.

$$f_{blender}(x) = f_{STK}(S_1(x), S_2(x), ..., S_s(x)) \tag{9}$$

where a meta-learner learns the weight vector w. A blender model can then be defined and tuned with its hyperparameters $\theta_{blender}$. It is then trained on the outputs of the stack S. It learns the mapping between the outcome of the stacked predictors and the final ground-truth ratings. The expression of the final prediction is as follows:

$$\hat{R_{BL}} = \phi(f_{blender}(x), f_{STK}(S_1(x), S_2(x), ..., S_s(x))) \tag{10}$$

Once the two recommenders needed for the task at hand are trained, we apply an aggregation function to merge their outputs into a single utility matrix. The Enhanced-MIGAN framework is summarized in Algorithm 2. It uses the simple unweighted average aggregation function followed by a fully connected layer. The final predicted utility matrix $\hat{P}_{ui}$ is as follows.

$$\hat{P}_{ui} = f_{agg}(\hat{R}_{ui}, \hat{R}_{BL}) \tag{11}$$

Algorithm 2: Content-Enhanced Mutual-Interction Graph Attention Neural Network recommender system.

Input
  $X_u : User\ features\ list$
  $X_i : Item\ features\ list$
  $U : List\ of\ user\ : size = n$
  $I : List\ of\ item\ : size = m$
  $\varphi : interactive\ attention\ neural\ network\ hyperparameters$
  $R : Rating\ matrix$
Output $\hat{P}_{ui} : prediction\ matrix$
Begin
  Phase 1 Preparing data to be passed to the BGNN
    foreach u $\in U, i \in V$ **do** $Rj = minmax(R)$
      $emb_u, emb_i = BGNN(X_u, X_i, R_j)$
  Phase 2 Build Mutual Interaction Graph Attention Neural Network model
    $att_u = Attention(LSTM(emb_u))$
    $att_i = Attention(LSTM(emb_i))$
    $att_{ui} = CoAttention(LSTM(emb_u), LSTM(emb_i))$
    MutualInteractiveAttention = BuildModel(concatenate(att_u, att_i, att_{ui})) ;
    $MutualInteractiveAttention.trainModel(D)$;
    $\hat{R}_{ui} = MutualInteractiveAttention.predict(\varphi)$
  Phase 3 Aggregating the utility matrix of sub-recommenders
    $\hat{P}_{ui} = aggregation(\hat{R}_{ui}, \hat{R}_{BL})$
return $\hat{P}_{ui}$

4 Experiments and Results

Our experiments are conducted on the *MovieLens Dataset* [23]. The MovieLens dataset presents real, timestamped 5-star ratings, as given by users of the MovieLens website on different films (number of Users = 6040, number of Movies = 3883, Sparsity = 95.5%). The machine used throughout the development and evaluation phases is MacBook Pro, 2014. It has 2.2 GHz Intel Core i7. It has 16 GB of DDR3 RAM.

The dataset will be divided into 90% training data, 10% testing data in a *stratified* manner, where the proportion of appearance of each user would be the same both in training and test data. Furthermore, the training data is further divided into 90% pure training, and 10% validation set, in a shuffling manner. Enhanced-MIGAN will be evaluated using two widely used metrics for recommender systems evaluation. They are Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) [24]. Here, we depict the hyperparameters analysis step, applied to the regression models and a meta-learner. A grid search is performed over five algorithms (polynomial-kernel support vector machines (SVM poly), RBF-kernel support vector machines (SVM rbf), decision trees (DT), automatic relevance detection regression (ARD) and linear regression (LR)). Results will be evaluated using $MAP@k$ and $NDCG@k$ for $k \in 10, 30, 50$ on the MovieLens dataset. Table 1 shows the changes in NDCG and MAP scores following a combination of regression models that compose the best stack. The structure of each stack is as follows:

- **Stack 1:** SVR_{poly} - LR - SVM_{rbf} — Random Forest
- **Stack 2:** LR - SVR_{poly} - ARD — Random Forest
- **Stack 3:** LR - Random Forest - ARD — SVM_{poly}
- **Stack 4:** DT-LR-SVR_{poly} — Random Forest

Table 1. The best scoring of the stacked content-based recommender

	Mean Average Precision			Normalized DCG		
Stack	MAP@10	MAP@30	MAP@50	NDCG@10	NDCG@30	NDCG@50
Stack 1	0.74	0.66	0.69	0.65	0.70	0.68
Stack 2	**0.80**	**0.78**	**0.79**	**0.69**	**0.76**	**0.76**
Stack 3	0.72	0.68	0.70	0.64	0.69	0.71
Stack 4	0.78	0.76	0.74	0.67	0.70	0.72

The experiment enhances weak learners with strong learners. Therefore, **Stack 2** has the best score and it will be picked for further framework. After that, a grid search algorithm over Random Forest is performed to identify its hyperparameters. The best meta-learner hyperparameters are: $n_{estimators} = 300, max_{depth} = 40, max_{features} = 2, min_{samplesLeaf} = 4, min_{samplesSplit} = 2$.

Table 2. The best hyperparameter for MIGAN recommender system. Results are evaluated based on MAP and NDCG metrics.

Hyperparameter	Range	Best settings
Dimensions of the embedding	$\alpha \in [30, 100]$	$\alpha = 50$
Number of dense layers after the Interactive attention block	$\theta \in [2, 20]$	$\theta = 3$
Number of neurons per dense layer	$\tau \in [30, 150]$	$\tau = 100$
Activation function used in the dense layers	$\sigma \in selu, elu, relu$	$\sigma = elu$
Optimizer	$\lambda \in sgd, adam, adagrad$	$\lambda = Adam$

Getting the best performance from MIGAN recommender system leads to tweaking the hyperparameters reported in Table 2. We compare the proposed recommender framework Content Enhanced-MIGAN with the following baselines:

- **Mutual Interaction Graph Attention Network "MIGAN"** [11] a state-of-the-art algorithm based on self-supervised representation learning on a large-scale bipartite graph (BGNN).
- **Neural Graph Collaborative Filtering (NGCF)** [10]: a representative and state-of-the-art GCN model for recommendation and is a Graph Neural Network used for Collaborative Filtering by propagating the user and item embeddings over the user- item graph, capturing connectivities between users and their neighbors.
- **Neural Collaborative Filtering (NCF)** [25]: This recommender system applies the multi-layer perceptron to learn the user-item interaction function.
- **Light Graph Convolutional Network (LightGCN)** [20] simplifies the aggregation process by using a linear transformation of the user-item interactions.

Figure 3 presents the MAP@k performance versus k-top items. It appears that the content-enhanced MIGAN can recall the relevant items for the user better than the other models with a significant margin, for example, it achieves MAP@10 = 0.93 and MAP@30 = 0.92 which is higher than the baselines. Figure 4 shows that the proposed framework exhibits a high NDCG score on MovieLens ($NDGC@10 = 0.84$ and $NDCG@30 = 0.87$). The more interested the users are in an item, the more likely users with similar preferences will recommend it. The content-enhanced MIGAN can deal with users with few user-item interactions and items with no interactions, all with the help of a content-stacked recommender.

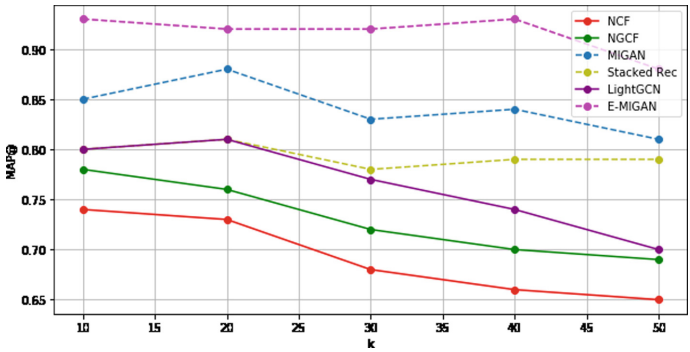


Fig. 3. Results of the comparison on MovieLens dataset. Evaluation of the performance of Top-K recommended lists, in terms of MAP. The ranking position K ranges from 10 to 50.

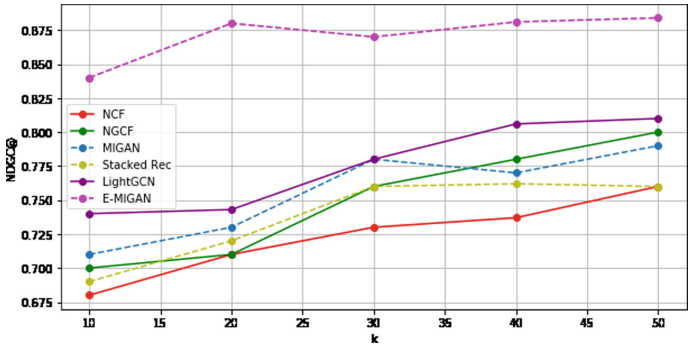


Fig. 4. Results of the comparison on MovieLens dataset. Evaluation of the performance of Top-K recommended lists, in terms of NDGC. The ranking position K ranges from 10 to 50.

5 Conclusion

We introduced a Content Enhanced MIGAN framework, a novel hybrid recommender model that integrates content-based embeddings to adjust the recommendation generation process. The stacked recommender effectively obtains the user's overall interest built by the stack's learners. It captures the side information to create a profile model for each user while optimizing the stack's learners' objective function. Furthermore, the mutual interaction graph attention neural network recommender carries relevant collaborative signals bringing up the impact of the complex user-item interactions on user decisions. By incorporating content-based embeddings into this collaborative filtering recommender, the proposed framework results in more accurate and personalized recommendations.

The empirical study on real-world datasets proves that the content-enhanced-MIGAN framework significantly outperforms the state-of-the-art methods in

recommendation performance. Future research avenues can include testing the efficiency of our framework for the evolution of users' preferences and item popularity over time.

References

1. Thorat, P.B., Goudar, R.M., Barve, S.: Survey on collaborative filtering, content-based filtering and hybrid recommendation system. Int. J. Comput. Appl. **110**(4), 31–36 (2015)
2. Lucas, J.P., Luz, N., Moreno, M.N., Anacleto, R., Almeida Figueiredo, A., Martins, C.: A hybrid recommendation approach for a tourism system. Expert Syst. Appl. **40**(9), 3532–3550 (2013)
3. Nguyen, L.V., Nguyen, T.-H., Jung, J.J., Camacho, D.: Extending collaborative filtering recommendation using word embedding: a hybrid approach. Concurr. Comput. Pract. Exp. **35**(16), e6232 (2023)
4. Drif, A., Guembour, S., Cherifi, H.: A sentiment enhanced deep collaborative filtering recommender system. In: Benito, R.M., Cherifi, C., Cherifi, H., Moro, E., Rocha, L.M., Sales-Pardo, M. (eds.) COMPLEX NETWORKS 2020 2020. SCI, vol. 944, pp. 66–78. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-65351-4_6
5. Lasfar, A., Mouline, S., Aboutajdine, D., Cherifi, H.: Content-based retrieval in fractal coded image databases. In: Proceedings 15th International Conference on Pattern Recognition, ICPR-2000, vol. 1, pp. 1031–1034. IEEE (2000)
6. Drif, A., Eddine Zerrad, H., Cherifi, H.: Context-awareness in ensemble recommender system framework. In: 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), pp. 1–6. IEEE (2021)
7. Ahlem, D.R.I.F., Saadeddine, S., Hocine, C.: An interactive attention network with stacked ensemble machine learning models for recommendations. In: Optimization and Machine Learning: Optimization for Machine Learning and Machine Learning for Optimization, pp. 119–150 (2022)
8. Mai, P., Pang, Y.: Vertical federated graph neural network for recommender system. arXiv preprint arXiv:2303.05786 (2023)
9. Wu, Y., Liu, H., Yang, Y.: Graph convolutional matrix completion for bipartite edge prediction. In: KDIR, pp. 49–58 (2018)
10. Wang, X., He, X., Wang, M., Feng, F., Chua, T.-S.:. Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 165–174 (2019)
11. Drif, A., Cherifi, H.: MIGAN: mutual-interaction graph attention network for collaborative filtering. Entropy **24**(8), 1084 (2022)
12. Otunba, R., Rufai, R.A., Lin, J.: Deep stacked ensemble recommender. In: Proceedings of the 31st International Conference on Scientific and Statistical Database Management, pp. 197–201 (2019)
13. Bao, X., Bergman, L., Thompson, R.: Stacking recommendation engines with additional meta-features. In: Proceedings of the Third ACM Conference on Recommender Systems, pp. 109–116 (2009)
14. Da Costa, A.F., Manzato, M.G.: Exploiting multimodal interactions in recommender systems with ensemble algorithms. Inf. Syst. **56**, 120–132 (2016)
15. Guo, Q., Qiu, X., Xue, X., Zhang, Z.: Syntax-guided text generation via graph neural network. Sci. China Inf. Sci. **64**, 1–10 (2021)

16. Zhou, J., et al.: Graph neural networks: a review of methods and applications. AI Open **1**, 57–81 (2020)
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
18. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
19. Wang, X., He, X., Cao, Y., Liu, M., Chua, T.-S.: KGAT: knowledge graph attention network for recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 950–958 (2019)
20. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: LightGCN: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 639–648 (2020)
21. He, C., et al.: Cascade-BGNN: toward efficient self-supervised representation learning on large-scale bipartite graphs. arXiv preprint arXiv:1906.11994 (2019)
22. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
23. Harper, F.M., Konstan, J.A.: The MovieLens datasets: history and context. ACM Trans. Interact. Intel. Syst. (TIIS) **5**(4), 1–19 (2015)
24. Drif, A., Zerrad, H.E., Cherifi, H.: EnsVAE: ensemble variational autoencoders for recommendations. IEEE Access **8**, 188335–188351 (2020)
25. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)

Heterophily-Based Graph Neural Network for Imbalanced Classification

Zirui Liang[1], Yuntao Li[1], Tianjin Huang[1], Akrati Saxena[2], Yulong Pei[1(✉)], and Mykola Pechenizkiy[1]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
{y.pei.1,m.pechenizkiy}@tue.nl
[2] Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
a.saxena@liacs.leidenuniv.nl

Abstract. Graph neural networks (GNNs) have shown promise in addressing graph-related problems, including node classification. However, in real-world scenarios, data often exhibits an imbalanced, sometimes highly-skewed, distribution with dominant classes representing the majority, where certain classes are severely underrepresented. This leads to a suboptimal performance of standard GNNs on imbalanced graphs. In this paper, we introduce a unique approach that tackles imbalanced classification on graphs by considering graph heterophily. We investigate the intricate relationship between class imbalance and graph heterophily, revealing that minority classes not only exhibit a scarcity of samples but also manifest lower levels of homophily, facilitating the propagation of erroneous information among neighboring nodes. Drawing upon this insight, we propose an efficient method, called `Fast Im-GBK`, which integrates an imbalance classification strategy with heterophily-aware GNNs to effectively address the class imbalance problem while significantly reducing training time. Our experiments on real-world graphs demonstrate our model's superiority in classification performance and efficiency for node classification tasks compared to existing baselines.

Keywords: Graph neural networks · Imbalanced classification · Heterophily

1 Introduction

GNNs have gained popularity for their accuracy in handling graph data. However, their accuracy, like other deep learning models, is highly dependent on data quality. One major challenge is class imbalance, where some classes have far fewer examples than others. This can lead to biased classification results, favoring the majority class while neglecting the minority classes [5]. The issue of imbalanced datasets commonly arises in classification and recognition tasks, where accurate classification of minority classes is critical. Graph imbalance classification has real-world applications, like identifying spammers in social networks [18] and

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 74–86, 2024.
https://doi.org/10.1007/978-3-031-53468-3_7

detecting fraud in financial networks [8]. In these cases, abnormal nodes are rare, making graph imbalance classification very challenging. Finding effective solutions to this problem is valuable for both research and practical applications.

The class-imbalanced problem has been extensively studied in machine learning and deep learning, as evident by prior research [6]. However, these methods may not effectively handle imbalanced graph data due to the interconnected nature of nodes within graphs. Graph nodes are characterized not only by their own properties but also by the properties of their neighboring nodes, introducing non-i.i.d. (independent and identically distributed) characteristics. Recent studies on graph imbalance classification have focused on data augmentation techniques, such as GraphSMOTE [19] and GraphENS [12]. However, our observations indicate that class imbalance in graphs is often accompanied by heterophilic connections of minority nodes, where minority nodes have more connections with nodes of diverse labels than the majority class nodes. This finding suggests that traditional techniques may be insufficient in the presence of heterophily.

To address this challenge, we propose incorporating a graph heterophily handling strategy into graph imbalanced classification. Our approach builds upon the bi-kernel design of GBK-GNN [2] to capture both homophily and heterophily within the graph. Additionally, we introduce a class imbalance-aware loss function, such as logit adjusted loss, to appropriately reweight minority and majority nodes. The complexity of GBK-GNN makes training computationally challenging. To overcome this, we propose an efficient version of the GBK-GNN that achieves both efficacy and efficiency in training.

Our main contributions are as follows: (1) We provide comprehensive insights into the imbalance classification problem in graphs from the perspective of graph heterophily and investigate the relationship between class imbalance and heterophily. (2) We present a novel framework that integrates graph heterophily and class-imbalance handling based on the insights and its fast implementation that significantly reduces training time. (3) We conduct extensive experiments on various real-world graphs to validate the effectiveness and efficiency of our proposed framework in addressing imbalanced classification on graphs.

2 Related Work

Imbalanced Classification. Efforts to counter class imbalance in classification entail developing unbiased classifiers that account for label distribution in training data. Existing strategies fall into three categories: loss modification, post-hoc correction, and re-sampling techniques. Loss modification adjusts the objective function by assigning greater weights [5] to minority classes. Post-hoc correction methods [11] adapt logits during inference to rectify underrepresented minority class predictions. Re-sampling employs techniques, such as sampling strategies [13] or data generation [1], to augment minority class data. The widely utilized Synthetic Minority Over-sampling Technique (SMOTE) [1] generates new instances by merging minority class data with nearest neighbors.

To tackle class imbalance in graph-based classification, diverse approaches harness graph structural information to mitigate the challenge. GraphSMOTE

[19] synthesizes minor nodes by interpolating existing minority nodes, with connectivity guided by a pretrained edge predictor. The Topology-Aware Margin (TAM) loss [16] considers each node's local topology by comparing its connectivity pattern to the class-averaged counterpart. When nearby nodes in the target class are denser, the margin for that class decreases. This change enhances learning adaptability and effectiveness through comparison. GraphENS [12] is another technique that generates an entire ego network for the minor class by amalgamating distinct ego networks based on similarity. These methods effectively combat class imbalance in graph-based classification, leveraging graph structures and introducing inventive augmentation techniques.

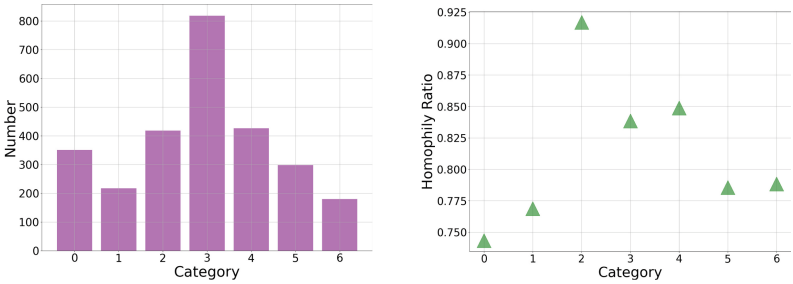
Heterophily Problem. In graphs, homophily [10] suggests that nodes with similar features tend to be connected, and heterophily suggests that nodes with diverse features and class labels tend to be connected. Recent investigations have analyzed the impact of heterophily on various tasks, emphasizing the significance of accounting for attribute information and devising methods attuned to heterophily [20]. Newly proposed models addressing heterophily can be classified into two categories: non-local neighbor extension and GNN architecture refinement [20]. Methods grounded in non-local neighbor extension seek to alleviate this challenge through neighborhood exploration. The H2GCN method [22], for instance, integrates insights from higher-order neighbors, revealing that two-hop neighbors frequently encompass more nodes of the same class as the central ego node. NLGNN [9] follows a pathway of employing attention mechanisms or pointer networks to prioritize prospective neighbor nodes based on attention scores or their relevance to the ego node. Approaches enhancing GNN architectures aspire to harness both local and non-local neighbor insights, boosting model capacity by fostering distinct and discerning node representations. A representative work is GBK-GNN (Gated Bi-Kernel Graph Neural Network) [2], which employs dual kernels to encapsulate homophily and heterophily details and introduces a gate to ascertain the kernel suitable for a given node pair.

3 Motivation

Node classification on graphs, such as one performed by the Graph Convolutional Network (GCN), differs fundamentally from non-graph tasks due to the interconnectivity of nodes. In imbalanced class distributions, minority nodes may have a higher proportion of heterophilic edges in their local neighborhoods, which can negatively impact classification performance.

To investigate the relationship between homophily and different classes, especially minorities, we conducted a small analysis on four datasets: Cora, CiteSeer, Wiki, and Coauthor CS (details about datasets can be found in Sect. 5.1). Our analysis involves computing the average homophily ratios and calculating node numbers across different categories. In particular, the average homophily ratio for nodes with label y is defined as:

$$h\left(y, \mathcal{G}_{\mathcal{V}}\right) = \frac{1}{|\mathcal{V}_y|} \sum_{i \in \mathcal{V}_y} \frac{|\mathcal{N}_{i^s}|}{|\mathcal{N}_i|} \tag{1}$$



(a) Distribution of nodes in Cora. (b) Homophily Ratio in Cora.

Fig. 1. Category distributions (left) and average homophily ratios (right) of Cora.

where $\mathcal{V}y$ represents a set of nodes with label y, $\mathcal{N}i$ is the set of neighbors of node v_i (excluding v_i) in graph $\mathcal{G}$, and $\mathcal{N}_{i^s}$ is the set of neighbors (excluding v_i) whose class is the same as v_i.

Observations. The results for Cora datasets are shown in Fig. 1. It is evident that the average homophily ratios of minority classes are relatively smallersuggesting higher proportions of heterophilic edges in their local neighborhoods. This could negatively affect classification performance when using existing imbalance strategies like data augmentation and loss reweight. This finding highlights the importance of considering the homophily and heterophily properties of nodes when designing graph classification models. We propose a novel approach that addresses imbalance issue effectively, considering node heterophily.

Problem Formulation. An attributed graph is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{X})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ represents the set of n nodes in $\mathcal{G}$, and $\mathcal{E}$ is the set of edges with edge e_{ij} connecting nodes v_i and v_j. For simplicity, we consider undirected graphs, but the argument could be generalized for directed graphs. The node attribute matrix is represented by $\mathrm{X} = \left[\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top\right] \in \mathcal{R}^{n \times d}$, where $\mathbf{x}_i$ indicates the feature of node v_i. $\mathcal{Y}$ denotes the label information for nodes in $\mathcal{G}$ and each node v_i is associated with a label $y_i \in \mathcal{R}^n$. During the training, only a subset of the classes, denoted by $\mathcal{Y}_\mathcal{L}$, is available containing the labels for node subset $\mathcal{V}_\mathcal{L}$. Moreover, $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ represents the set of m classes assigned to each node, with $|C_i|$ denoting the size of the i-th class, which indicates the number of nodes with class c_i. To quantify the degree of imbalance in the graph, we use the imbalance ratio, defined as $\mathbf{r} = \frac{\max_i(|C_i|)}{\min_i(|C_i|)}$. For imbalanced graphs, the imbalance ratio of $\mathcal{Y}_\mathcal{L}$ is high (Table 1 of Sect. 5 shows that the first two datasets have relatively balanced classes and the last two have imbalanced classes).

Imbalance Node Classification on Graphs. In the context of an imbalanced graph $\mathcal{G}$ with a high imbalance ratio $\mathbf{r}$, the aim is to develop a node classifier $f : f(\mathcal{V}, \mathcal{E}, \mathrm{X}) \rightarrow \mathcal{Y}$ that can work well for classifying nodes belonging to both the majority and minority classes.

4 Methodology

In this section, we present our solution to address class imbalance that incorporates heterophily handling and imbalance handling components (Sect. 4.1). We also propose a fast version that effectively reduces training time (Sect. 4.2). The main objective of our model is to minimize the loss of minority classes while ensuring accurate information exchange during the message-passing process.

4.1 Im-GBK

Heterophily Handling. We build our model based on the GBK-GNN [2] model, which is a good model for graph classification, though not able to handle class imbalance. GBK-GNN is designed to address the lack of distinguishability in GNN, which stems primarily from the incapability to adjust weights adaptively for various node types based on their distinct homophily properties. As a consequence, a bi-kernel feature transformation method is employed to capture either homophily or heterophily information. In this work, we, therefore, introduce a learnable kernel-based selection gate that aims to distinguish if a pair of nodes are similar or not and then selectively choose appropriate kernels, i.e., homophily or heterophily kernel. The formal expression for the input transformation is presented below.

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W_f}\mathbf{h}_i^{(l-1)} + \frac{1}{|\mathcal{N}(v_i)|}\sum_{v_j \in \mathcal{N}(v_i)} \alpha_{ij}\mathbf{W_s}\mathbf{h}_j^{(l-1)} + (1-\alpha_{ij})\,\mathbf{W_d}\mathbf{h}_j^{(l-1)}\right) \tag{2}$$

$$\alpha_{ij} = Sigmoid\left(\mathbf{W_g}\left[\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}\right]\right) \tag{3}$$

$$\mathcal{L} = \mathcal{L}_0 + \lambda\sum_{l}^{L}\mathcal{L}_g^{(l)} \tag{4}$$

where $\mathbf{W_s}$ and $\mathbf{W_d}$ are the kernels for homophilic and heterophilic edges, respectively. The value of α_{ij} is determined by $\mathbf{W_g}$ and the embedding layer of nodes i and j. The loss function consists of two parts: $\mathcal{L}_0$, a cross-entropy loss for node classification, and $\mathcal{L}_g^{(l)}$, a label consistency-based cross-entropy, i.e., to discriminate if labels of a pair of nodes are consistent for each layer l to guide the training of the selection gate or not. A hyper-parameter λ is introduced to balance the two losses. The original GBK-GNN method does not explicitly address the class imbalance issue, which leads to the model being biased toward the majority class. Our method employs a class-imbalance awareness for the GBK-GNN design, and therefore it mitigates the bias.

Class-Imbalance Handling with Logit Adjusted Loss. When traditional softmax is used on such imbalanced data, it can result in biases toward the majority class. This is because the loss function used to train the model typically treats all classes equally, regardless of their frequency. As a result, the

model tends to optimize for overall accuracy by prioritizing the majority classes while performing poorly on the minority classes. This issue can be addressed by adjusting the logits (i.e., the inputs to the softmax function) for each class to be inversely proportional to their frequencies in the training data, which effectively reduces the weight of the majority classes and increases the weight of the minority classes [11]. In this study, we calculate logit-adjusted loss as follows:

$$\mathcal{L}_{logit-adjusted} = -\log \frac{e^{f_y(x)+\tau\cdot\log \pi_y}}{\sum_{y'\in[L]} e^{f_{y'}(x)+\tau\cdot\log \pi_{y'}}} \tag{5}$$

where π_y is the estimate of the class prior. In this approach, a label-dependent offset is added to each logit, which differs from the standard softmax cross-entropy approach. Additionally, the class prior offset is enforced during the learning of the logits rather than being applied post-hoc, as in other methods.

Class-Imbalance Handling with Balanced Softmax. Another approach called balanced softmax [13] focuses on assigning larger weights to the minority classes and smaller weights to the majority class, which encourages the model to focus more on the underrepresented classes and improve their performance. The traditional softmax function treats all classes equally, which can result in a bias towards the majority class in imbalanced datasets. Balanced softmax, on the other hand, adjusts the temperature parameter of the softmax function to balance the importance of each class, effectively reducing the impact of the majority class and increasing the impact of the minority classes. Formally, given N_k, l_v, y_v, l_{v,y_v} represent *k-th* of class N, logit and the label of node v, logit associated with the true label y_v for the input node v, respectively. In this study, the balanced softmax is calculated as:

$$\mathcal{L}_{balanced-softmax} = -\log \frac{e^{l_{v,y_v}+\log N_{y_v}}}{\sum_{k\in\mathcal{Y}} e^{l_{v,k}+\log |N_k|}}. \tag{6}$$

Loss Function. We design our loss function to combine these two components that handle heterophily and class imbalance in the proposed Im-GBK model. The learning objective of our model consists of (i) reducing the weight of the majority classes and increasing the weight of the minority classes in the training data, and (ii) improving the model's ability to select the ideal gate. To achieve this objective, we incorporate two loss components into the loss function:

$$\mathcal{L} = \mathcal{L}_{im} + \lambda \sum_{l}^{L} \mathcal{L}_g^{(l)}. \tag{7}$$

The first component, from the class-imbalance handler, denoted as $\mathcal{L}_{im}$, applies either the *logit adjusted loss* $\mathcal{L}_{logit-adjusted}$ or the *Balanced Softmax* $\mathcal{L}_{balanced-softmax}$ approach to reduce the impact of the majority class and focus on underrepresented classes. The second component, denoted as $\mathcal{L}g^{(l)}$, applies cross-entropy loss to each layer l of the heterophily handler to improve the model's discriminative power and adaptively select gate to emphasize homophily

or heterophily. The hyper-parameter λ balances the two losses in the overall loss function.

4.2 Fast Im-GBK

The major limitation of Im-GBK lies in efficiency, as the additional time is required to compute the gate result from the heterophily handling component. The challenge arises from the need to label edges connecting unknown nodes, which is typically addressed by learning an edge classifier. Specifically, in the message-passing process, the gate acts as an edge classifier that predicts the homophily level of a given edge. As a result, this process requires additional time proportional to the number of edges in the graph, expressed as $T_{\mathbf{extra}} = |\mathbf{E}| \times time(e_{ij})$, where $time(e_{ij})$ represents the time to compute one edge. Therefore, we propose to use a graph-level homophily ratio [21] instead of the pair-wise edge classifier. This removes the kernel selection process before training and significantly reduces training time. We use Eq. 2 to aggregate and update the embedding layers similarly. The formal definition of the gate generator is:

$$H(\mathbf{G}) = \frac{\sum_{(v_i,v_j)\in\mathbf{E}} \mathbb{I}\left(y_i = y_j\right)}{|\mathbf{E}|} \tag{8}$$

$$\alpha_{ij} = \begin{cases} \max((1-\epsilon)\mathbb{I}(y_i = y_j), \epsilon) & y_i, y_j \in V_{train} \\ H(\mathbf{G}) & \text{otherwise} \end{cases} \tag{9}$$

$$\mathcal{L}_{Fast\ Im-GBK} = \mathcal{L}_{im} \tag{10}$$

where the hyper-parameter ϵ will serve as the minimum similarity threshold, and $\mathcal{L}_{im}$ is aforementioned Class-Imbalance Handling loss.

5 Experiments

We address four questions to enhance our understanding of the model:

RQ1: How do Im-GBK and fast-Im-GBK models perform in comparison to baselines in node classification for an imbalanced scenario? **RQ2:** How is the performance efficiency of fast-Im-GBK compared to other baseline models? **RQ3:** What is the role of each component in our Im-GBK model, and do they positively impact the classification performance?

5.1 Experiment Settings

Datasets. We conduct experiments on four datasets from PyTorch Geometric [3], which are commonly used in graph neural network literature, to evaluate the efficiency of our proposed models for the node classification task when classes

Table 1. Statistics of the node classification datasets.

	Hom.Ratio	Imbalance Ratio	Nodes	Edges	Features	Classes
CiteSeer [14]	0.736	2.655	3327	9104	3703	6
Cora [14]	0.810	4.544	2708	10556	1433	7
Wiki [14]	0.712	45.111	2405	17981	4973	17
Coauthor CS [15]	0.808	35.051	18333	163788	6805	15

are imbalanced. Table 1 presents a summary of the datasets. In addition, using CiteSeer and Cora datasets, we generate two extreme instances in which each minority class with random selection has only five training examples.

Experiment Environment. For each dataset, we randomly select 60% of total samples for training, 20% for validation, and the remaining 20% for testing. We use Adam optimizer and set learning rate $lr, weight\ decay$ as 0.001 and $5e^{-4}$, respectively. All baselines follow the same setting except for the layer number of 128. We run each experiment on the same random seeds to ensure reproducibility. Model training is done on NVIDIA GeForce RTX 3090 (24 GB) GPU with 90 GB memory. The code depends on PyTorch 1.7.0 and PyG 2.0.4.

Evaluation Metric. The quality of classification is assessed by average accuracy, AUC-ROC, and F1 scores. Each experiment is repeated five times to avoid randomness and compute the final value.

Baselines. We evaluate our approach with representative and state-of-the-art approaches, including three classic GNN models (GCN [7], GAT [17], GraphSage [4]) using three traditional imbalance techniques (Over-sampling, Re-weight, SMOTE [1]), original GBK-GNN [2], GraphSMOTE [19], and TAM [16] (we choose the combination of GCN+TAM+Balanced Softmax and GCN+TAM+ENS, referred to as GCN-TAM-BS and GCN-TAM-ENS, respectively).

5.2 Comparisons with Baselines (RQ1)

In this section, we analyze the performance of classic graph neural networks (GNNs) and traditional imbalance learning approaches. As shown in the learning objective (Sect. 4.1), λ plays an important role in the tradeoff between classification error and consistency. Thus, we first explore the impact of hyperparameter λ on the performance. We set λ between 0 and 5 with an interval of 0.5 and the results are shown in Fig. 2. According to our analysis, the impact of λ on the results is insignificant if it is not 0. Therefore, in the following experiments, we always set $\lambda = 1$, considering the overall trends for both methods on all datasets. We examine the performance of different models. The results are reported in Table 2 and Table 3 on original datasets and extreme datasets, respectively. From the results, it can be observed that:

– The experiment demonstrates that Im-GBK models, which use logit adjust loss and balanced softmax (denoted as Im-GBK (LogitAdj) and Im-GBK

Table 2. Comparison of different methods on original datasets.

	Cora			CiteSeer			Wiki			Coauthor-CS		
	ACC	AUC	F-1	ACC	AUC	F-1	ACC	AUC	F-1	ACC	AUC	F-1
GCN	0.853	0.981	0.847	0.719	0.905	0.687	0.664	0.875	0.592	0.935	0.995	0.914
GCN+SMOTE	0.855	0.980	0.848	0.709	0.904	0.673	0.649	0.865	0.605	0.939	0.996	0.921
GCN+Re-weight	0.848	0.981	0.840	0.712	0.905	0.683	0.672	0.873	0.631	0.935	0.996	0.915
GAT	0.853	0.969	0.846	0.730	0.896	0.702	0.243	0.643	0.191	0.889	0.975	0.822
GAT+SMOTE	0.839	0.970	0.822	0.716	0.876	0.687	0.281	0.691	0.250	0.370	0.721	0.185
GAT+Re-weight	0.827	0.965	0.821	0.703	0.888	0.677	0.125	0.580	0.111	0.913	0.987	0.891
GraphSAGE	0.805	0.968	0.780	0.697	0.895	0.676	0.691	0.877	0.578	0.905	0.988	0.829
GraphSAGE+SMOTE	0.798	0.971	0.776	0.724	0.902	0.702	0.693	0.884	0.688	0.633	0.947	0.397
GraphSAGE+Re-weight	0.794	0.966	0.769	0.703	0.888	0.677	0.668	0.884	0.563	0.898	0.986	0.816
GraphSMOTE	0.872	0.984	0.864	**0.769**	**0.929**	**0.741**	0.569	0.859	0.440	0.940	0.996	0.926
GCN-TAM	**0.878**	0.931	**0.868**	0.752	0.840	0.727	0.651	0.823	0.563	0.929	0.956	0.914
GBK-GNN-BS	0.876	0.974	0.866	0.730	0.915	0.707	0.681	0.881	0.611	0.936	**0.997**	0.918
Im-GBK (LogitAdj)	0.866	0.979	0.853	0.728	0.912	0.699	0.674	0.887	0.622	0.932	0.996	0.912
Im-GBK (BLSM)	0.861	0.979	0.846	0.721	0.909	0.700	0.677	0.888	0.615	0.933	0.996	0.914
Fast Im-GBK	0.876	**0.988**	0.863	0.766	0.926	0.738	**0.723**	**0.911**	**0.655**	**0.951**	**0.997**	**0.939**

(BLSM), respectively), achieve comparable or better results to state-of-the-art methods in most original datasets. Specifically, in Cora and CiteSeer, Im-GBK (LogitAdj) and Im-GBK (BLSM) are comparable to GraphSMOTE and GCN-TAM, while they outperform it on Wiki. Furthermore, Fast Im-GBK demonstrates superiority on Wiki and Coauthor-CS, and all proposed methods performed better than the baselines in extremely imbalanced cases.

– The models designed specifically for imbalance problems perform better in the extreme cases (Cora Extreme and CiteSeer Extreme); refer to Table 3. Our models show superior performance in extreme circumstances, as indicated by achieving the best performance consistently w.r.t the most of metrics, whereas GCN-TAM and GraphSMOTE, as specifically designed models for the imbalanced classification task, also exhibit their capabilities in differentiating minority classes.
– Models designed to account for graph heterophily generally outperform classic GNNs and modified classic GNNs. This also empirically validates the rationality of using heterophilic neighborhoods for imbalanced classification to some extent.

Overall, the experiment demonstrates that the proposed methods are effective in imbalanced node classification and offer better or comparable performance to state-of-the-art. In extreme cases, our approaches outperform and show a better capability in differentiating minority classes.

5.3 Comparison in Efficiency (RQ2)

Section 3 showed that the proposed Im-GBK model could be time-consuming due to its second loss function component. To address this, we replace the kernel selection process by using a graph-level homophily ratio. Table 5 presents

Table 3. Comparison of different methods on extreme datasets.

	Cora Extreme			CiteSeer Extreme		
	ACC	AUC	F-1	ACC	AUC	F-1
GCN	0.746	0.929	0.647	0.697	0.877	0.607
GCN+SMOTE	0.745	0.930	0.648	0.699	0.877	0.610
GCN+Re-weight	0.756	0.934	0.667	0.700	0.878	0.612
GAT	0.679	0.878	0.532	0.694	0.880	0.605
GAT+SMOTE	0.653	0.861	0.427	0.677	0.847	0.593
GAT+Re-weight	0.689	0.869	0.516	0.681	0.873	0.596
GraphSAGE	0.657	0.873	0.498	0.682	0.866	0.596
GraphSAGE+SMOTE	0.671	0.884	0.494	0.680	0.871	0.593
GraphSAGE+Re-weight	0.674	0.878	0.509	0.687	0.873	0.598
GraphSMOTE	0.770	0.928	0.674	0.703	0.893	0.612
GCN-TAM-BS	**0.829**	0.888	**0.797**	0.718	0.801	0.649
GCN-TAM-ENS	0.790	0.884	0.769	0.680	0.797	**0.659**
GBK-GNN	0.692	0.905	0.521	0.696	0.892	0.607
Im-GBK (LogitAdj)	0.717	0.914	0.599	0.703	0.896	0.615
Im-GBK (BLSM)	0.800	0.931	0.761	0.705	0.897	0.655
Fast Im-GBK	0.727	**0.941**	0.570	**0.737**	**0.899**	0.641

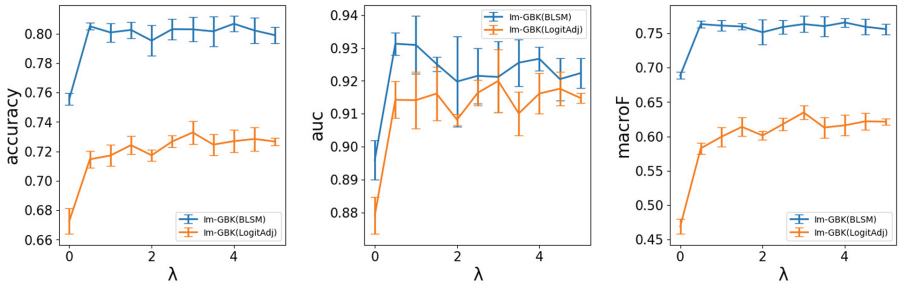


Fig. 2. Experimental results on Cora (extreme)

training time comparisons, revealing that models using gate-selection mechanisms, like GBK-GNN and Im-GBK, require significantly more training time. For example, while most models could complete one epoch within 1 s, GBK-GNN and Im-GBK took over 10 s and around 160 s to train one epoch on the CS dataset. GraphSMOTE also requires more time than Fast Im-GBK because it has to generate several synthetic nodes for each minority class. However, the results demonstrate that our proposed Fast Im-GBK model shows a significant reduction in training time compared to GBK-GNN and GraphSMOTE.

Table 4. Ablation Experiment Results

Class-Imbalance Handling Loss		Heterophily Handling	ACC	AUC	F1
Logit adjusted loss	Balanced Softmax				
×	×	×	0.697	0.877	0.607
×	×	√	0.696	0.892	0.607
×	√	×	**0.710**	0.871	0.618
×	√	√	0.705	**0.897**	**0.655**
√	×	×	0.667	0.872	0.566
√	×	√	0.703	0.896	0.615

5.4 Ablation Analysis (RQ3)

Subsection 5.3 showed that the proposed method exhibits clear advantages in performance compared to other baselines in differentiating minority classes for extremely imbalanced graphs. To further investigate the fundamental factors underlying the performance improvements of our proposed method in Im-GBK, we conduct ablation analyses using one of the extreme cases, CiteSeer Extreme. We show the effectiveness of the model handling class imbalance classification by ablating the model Class-Imbalance Handler and Heterophily Handler, respectively. In Table 4, 'Class-Imbalance Handling Loss' represents two Class-Imbalance Handling losses introduced in Sect. 4.1. 'Heterophily handling' refers to the method introduced in Sect. 4.1 to capture graph heterophily, and '×' means this part is ablated. Considering all strategies, it can be observed from Table 4 that either dropping 'Class-Imbalance Handling Loss' or 'Heterophily handling' components will result in a decrease in performance.

6 Conclusion

In this paper, we studied the problem of imbalanced classification on graphs from the perspective of graph heterophily. We observed that if a model cannot handle heterophilic neighborhoods in graphs, its ability to address imbalanced classification will be impaired. To address the graph imbalance problem effectively, we proposed a novel framework, Im-GBK, and its faster version, Im-GBK, that simultaneously tackles heterophily and class imbalance. Our framework overcomes the limitations of previous techniques by achieving higher efficiency while maintaining comparable performance. Extensive experiments are conducted on various real-world datasets,

Table 5. Average Execution Time (s) per epoch on CS.

	Time
GCN	0.0166
GAT	0.1419
GraphSage	0.0135
GraphSMOTE	5.309
Fast Im-GBK	0.5897
GBK-GNN	12.320
Im-GBK (LogitAdj)	11.594
Im-GBK (BLSM)	11.271

demonstrating that our model outperforms most baselines. Furthermore, a comprehensive parameter analysis is performed to validate the efficacy of our approach. In future research, we aim to explore alternative methods for modeling graph heterophily and extend our approach to real-world applications, such as fraud and spammer detection.

References

1. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
2. Du, L., et al.: GBK-GNN: gated bi-kernel graph neural networks for modeling both homophily and heterophily. In: Proceedings of the ACM Web Conference 2022, pp. 1550–1558 (2022)
3. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch geometric. arXiv preprint arXiv:1903.02428 (2019)
4. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs (2018)
5. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. Intell. Data Anal. **6**(5), 429–449 (2002)
6. Johnson, J.M., Khoshgoftaar, T.M.: Survey on deep learning with class imbalance. J. Big Data **6**(1), 1–54 (2019)
7. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2017)
8. Liu, Y., et al.: Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In: Proceedings of the Web Conference 2021, pp. 3168–3177 (2021)
9. Liu, Y., Zheng, Y., Zhang, D., Chen, H., Peng, H., Pan, S.: Towards unsupervised deep graph structure learning. In: Proceedings of the ACM Web Conference 2022, pp. 1392–1403 (2022)
10. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: homophily in social networks. Ann. Rev. Sociol. **27**(1), 415–444 (2001)
11. Menon, A.K., Jayasumana, S., Rawat, A.S., Jain, H., Veit, A., Kumar, S.: Long-tail learning via logit adjustment (2021)
12. Park, J., Song, J., Yang, E.: GraphENS: neighbor-aware ego network synthesis for class-imbalanced node classification. In: International Conference on Learning Representations (2021)
13. Ren, J., et al.: Balanced meta-softmax for long-tailed visual recognition (2020)
14. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
15. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
16. Song, J., Park, J., Yang, E.: TAM: topology-aware margin loss for class-imbalanced node classification. In: International Conference on Machine Learning, pp. 20,369–20,383. PMLR (2022)
17. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks (2018)
18. Wu, Y., Lian, D., Xu, Y., Wu, L., Chen, E.: Graph convolutional networks with Markov random field reasoning for social spammer detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 1054–1061 (2020)

19. Zhao, T., Zhang, X., Wang, S.: GraphSMOTE: imbalanced node classification on graphs with graph neural networks. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 833–841 (2021)
20. Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., Yu, P.S.: Graph neural networks for graphs with heterophily: a survey. arXiv preprint arXiv:2202.07082 (2022)
21. Zhu, J., et al.: Graph neural networks with heterophily. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11,168–11,176 (2021)
22. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., Koutra, D.: Beyond homophily in graph neural networks: current limitations and effective designs. Adv. Neural. Inf. Process. Syst. **33**, 7793–7804 (2020)

TimeGNN: Temporal Dynamic Graph Learning for Time Series Forecasting

Nancy Xu[1(✉)], Chrysoula Kosma[2], and Michalis Vazirgiannis[1,2]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
nancyx@kth.se
[2] École Polytechnique IPP, Palaiseau, France
{kosma,mvazirg}@lix.polytechnique.fr

Abstract. Time series forecasting lies at the core of important real-world applications in many fields of science and engineering. The abundance of large time series datasets that consist of complex patterns and long-term dependencies has led to the development of various neural network architectures. Graph neural network approaches, which jointly learn a graph structure based on the correlation of raw values of multivariate time series while forecasting, have recently seen great success. However, such solutions are often costly to train and difficult to scale. In this paper, we propose TimeGNN, a method that learns dynamic temporal graph representations that can capture the evolution of inter-series patterns along with the correlations of multiple series. TimeGNN achieves inference times 4 to 80 times faster than other state-of-the-art graph-based methods while achieving comparable forecasting performance.

Keywords: Time Series Forecasting · Graph Structure Learning · GNNs

1 Introduction

From financial investment and market analysis [6] to traffic [21], electricity management, healthcare [4], and climate science, accurately predicting the future real values of series based on available historical records forms a coveted task over time in various scientific and industrial fields. There are a wide variety of methods employed for time series forecasting, ranging from statistical [2] to recent deep learning approaches [22]. However, there are several major challenges present. Real-world time series data are often subject to noisy and irregular observations, missing values, repeated patterns of variable periodicities and very long-term dependencies. While the time series are supposed to represent continuous phenomena, the data is usually collected using sensors. Thus, observations are determined by a sampling rate with potential information loss. On the other hand, standard sequential neural networks, such as recurrent (RNNs) [27] and convolutional networks (CNNs) [20], are discrete and assume regular spacing between observations. Several continuous analogues of such architectures that implicitly handle the time information have been proposed to address irregularly sampled missing data [26].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 87–99, 2024.
https://doi.org/10.1007/978-3-031-53468-3_8

The variable periodicities and long-term dependencies present in the data make models prone to shape and temporal distortions, overfitting and poor local minima while training with standard loss functions (e. g., MSE). Variants of DTW and MSE have been proposed to mitigate these phenomena and can increase the forecasting quality of deep neural networks [16,19].

A novel perspective for boosting the robustness of neural networks for complex time series is to extract representative embeddings for patterns after transforming them to another representation domain, such as the spectral one. Spectral approaches have seen much use in the text domain. Graph-based text mining (i. e., Graph-of-Words) [25] can be used for capturing the relationships between the terms and building document-level representations. It is natural, then, that such approaches might be suitable for more general sequence modeling. Capitalizing on the recent success of graph neural networks (GNNs) on graph structured data, a new family of algorithms jointly learns a correlation graph between interrelated time series while simultaneously performing forecasting [3,29,32]. The nodes in the learnable graph structure represent each individual time series and the links between them express their temporal similarities. However, since such methods rely on series-to-series correlations, they do not explicitly represent the inter-series temporal dynamics evolution. Some preliminary studies have proposed simple computational methods for mapping time series to temporal graphs where each node corresponds to a time step, such as the visibility graph [17] and the recurrence network [7].

In this paper, we propose a novel neural network, *TimeGNN*, that extends these previous approaches by jointly learning dynamic temporal graphs for time series forecasting on raw data. TimeGNN (i) extracts temporal embeddings from sliding windows of the input series using dilated convolutions of different receptive sizes, (ii) constructs a learnable graph structure, which is forward and directed, based on the similarity of the embedding vectors in each window in a differentiable way, (iii) applies standard GNN architectures to learn embeddings for each node and produces forecasts based on the representation vector of the last time step. We evaluate the proposed architecture on various real-world datasets and compare it against several deep learning benchmarks, including graph-based approaches. Our results indicate that TimeGNN is significantly less costly in both inference and training while achieving comparable forecasting performance. The code implementation for this paper is available at https://github.com/xun468/Time-GNN.

2 Related Work

Time Series Forecasting Models. Time series forecasting has been a long-studied challenge in several application domains. In terms of statistical methods, linear models including the autoregressive integrated moving average (ARIMA) [2] and its multivariate extension, the vector autoregressive model (VAR) [10] constitute the most dominant approaches. The need for capturing non-linear patterns and overcoming the strong assumptions for statistical methods, e. g., the stationarity assumption, has led to the application of deep neural networks,

initially introduced in sequential modeling, to the time series forecasting setting. Those models include recurrent neural networks (RNNs) [27] and their improved variants for alleviating the vanishing gradient problem, namely the LSTM [12] and the GRU [5]. An alternative method for extracting long-term dependencies via large receptive fields can be achieved by leveraging stacked dilated convolutions, as proposed along with the Temporal Convolution Network (TCN) [1]. Bridging CNNs and LSTMs to capture both short-term local dependency patterns among variables and long-term patterns, the Long- and Short-term Time-series network (LSTNet) [18] has been proposed. For univariate point forecasting, the recently proposed N-BEATS model [24] introduces a deep neural architecture based on a deep stack of fully-connected layers with basis expansion. Attention-based approaches have also been employed for time-series forecasting, including Transformer [30] and Informer [35]. Finally, for efficient long-term modeling, the most recent Autoformer architecture [31] introduces an auto-correlation mechanism in place of self-attention, which extracts and aggregates similar sub-series based on the series periodicity.

Graph Neural Networks. Over the past few years, graph neural networks (GNNs) have been applied with great success to machine learning problems on graphs in various fields, including chemistry for drug screening [14] and biology for predicting the functions of proteins modeled as graphs [9]. The field of GNNs has been largely dominated by the so-called message passing neural networks (MPNNs) [8], where each node updates its feature vector by aggregating the feature vectors of its neighbors. In the case of time series data on arbitrary known graphs, e. g., in traffic forecasting, several architectures that combine sequential models with GNNs have been proposed [21,28,33,34].

Joint Graph Structure Learning and Forecasting. However, since spatial-temporal forecasting requires an apriori topology which does not apply in the case of most real-world time series datasets, graph structure learning has arisen as a viable solution. Recent models perform joint graph learning and forecasting for multivariate time series data using GNNs, intending to capture temporal patterns and exploit the interdependency among time series while predicting the series' future values. The most dominant algorithms include NRI [15], MTGNN [32] and GTS [29], in which the graph nodes represent the individual time series and their edges represent their temporal evolution. MTGNN obtains the graph adjacency from the as a degree-k structure from the pairwise scores of embeddings of each series in the multivariate collection, which might pose challenges to end-to-end learning. On the other hand, NRI and GTS employ the Gumbel softmax trick [13] to differentiably sample a discrete adjacency matrix from the edge probabilities. Both models compute fixed-size representations of each node based on the time series, with the former dynamically producing the representations per individual window and the latter extracting global representations from the whole training series. MTGNN combines temporal convolution with graph convolution layers, and GTS uses a Diffusion Convolutional Recurrent Neural Network (DCRNN) [21], where the hidden representations of nodes are diffused using graph convolutions at each step.

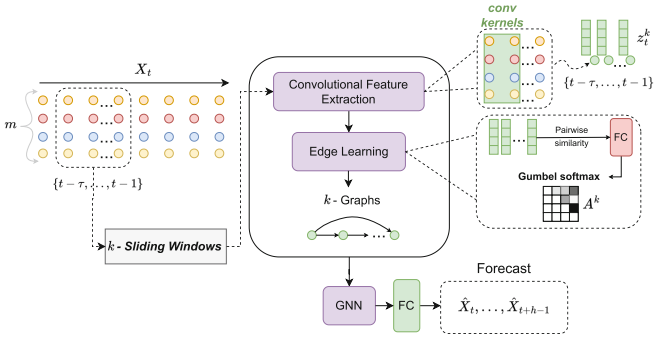


Fig. 1. The proposed TimeGNN framework time series for graph learning from raw time series and forecasting based on embeddings learned on the parameterized graph structures.

3 Method

Let $\{\mathbf{X}_{i,1:T}\}_{i=1}^{m}$ be a multivariate time series that consists of m channels and has a length equal to T. Then, $\mathbf{X}_t \in \mathbb{R}^m$ represents the observed values at time step t. Let also $\mathcal{G}$ denote the set of temporal dynamic graph structures that we want to infer.

Given the observed values of τ previous time steps of the time series, i. e., $\mathbf{X}_{t-\tau},\ldots,\mathbf{X}_{t-1}$, the goal is to forecast the next h time steps (e. g., $h=1$ for 1-step forecasting), i. e., $\hat{\mathbf{X}}_t, \hat{\mathbf{X}}_{t+1},\ldots,\hat{\mathbf{X}}_{t+h-1}$. These values can be obtained by the forecasting model $\mathcal{F}$ with parameters Φ and the graphs $\mathcal{G}$ as follows:

$$\hat{\mathbf{X}}_t, \hat{\mathbf{X}}_{t+1},\ldots,\hat{\mathbf{X}}_{t+h-1} = \mathcal{F}(\mathbf{X}_{t-\tau},\ldots,\mathbf{X}_{t-1};\mathcal{G};\Phi) \tag{1}$$

3.1 Time Series Feature Extraction

Unlike previous methods which extract one feature vector per variable in the multivariate input, our method extracts one feature vector per time step in each window k of length τ. Temporal sub-patterns are learned using stacked dilated convolutions, similar to the main blocks of the inception architecture [23].

Given the sliding windows $\mathbf{S} = \{\mathbf{X}_{t-\tau+k-K},\ldots,\mathbf{X}_{t+k-K-1}\}_{k=1}^{K}$, we perform the following convolutional operations to extract three feature maps $\mathbf{f}_0^k$, $\mathbf{f}_1^k$, $\mathbf{f}_2^k$, per window $\mathbf{S}^k$. Let $\mathbf{f}_i^k \in \mathbb{R}^{\tau\times d}$ for hidden dimension d of the convolutional kernels, such that:

$$\begin{aligned}
\mathbf{f}_0^k &= \mathbf{S}^k * \mathbf{C}_0^{1,1} + \mathbf{b}_{01} \\
\mathbf{f}_1^k &= (\mathbf{S}^k * \mathbf{C}_1^{1,1} + \mathbf{b}_{11}) * \mathbf{C}_2^{3,3} + \mathbf{b}_{23} \\
\mathbf{f}_2^k &= (\mathbf{S}^k * \mathbf{C}_2^{1,1} + \mathbf{b}_{21}) * \mathbf{C}_2^{5,5} + \mathbf{b}_{25}
\end{aligned} \tag{2}$$

where $*$ the convolutional operator, $\mathbf{C}_0^{1,1}$, $\mathbf{C}_1^{1,1}$, $\mathbf{C}_2^{1,1}$ convolutional kernels of size 1 and dilation rate 1, $\mathbf{C}_2^{3,3}$ a convolutional kernel of size 3 and dilation rate 3, $\mathbf{C}_2^{5,5}$

a convolutional kernel of size 5 and dilation rate 5, and $\mathbf{b}_{01}, \mathbf{b}_{11}, \mathbf{b}_{21}, \mathbf{b}_{23}, \mathbf{b}_{25}$ the corresponding bias terms.

The final representations per window k are obtained using a fully connected layer on the concatenated features $\mathbf{f}_0^k, \mathbf{f}_1^k, \mathbf{f}_2^k$, i. e., $\mathbf{z}^k = \mathrm{FC}(\mathbf{f}_0^k \| \mathbf{f}_1^k \| \mathbf{f}_2^k)$, such that $\mathbf{z}^k \in \mathbb{R}^{\tau \times d}$. In the next sections, we refer to each time step of the hidden representation of the feature extraction module in each window k as $\mathbf{z}_i^k, \forall\ i \in \{1, \ldots \tau\}$.

3.2 Graph Structure Learning

The set $\mathcal{G} = \{\mathcal{G}^k\}, k \in \mathbb{N}^*$ describes the collection of graph structures that are parameterized for all individual sliding window of length τ of the series, where K defines the total number of windows. The goal of the graph learning module is to learn each adjacency matrix $\mathbf{A}^k \in \{0, 1\}^{\tau \times \tau}$ for a temporal window of observations $\mathbf{S}^k$. Following the works of [15,29], we use the Gumbel softmax trick to sample a discrete adjacency matrix as described below.

For the Gumbel softmax trick, let $\mathbf{A}^k$ refer to a random variable of the matrix Bernoulli distribution parameterized by $\boldsymbol{\theta}^k \in [0, 1]^{\tau \times \tau}$, so that $A_{ij}^k \sim Ber(\theta_{ij}^k)$ is independent for pairs (i, j). By applying the Gumbel reparameterization trick [13] for enabling differentiability in sampling, we can obtain the following:

$$\begin{aligned} A_{ij}^k = \sigma((\log(\theta_{ij}^k/(1-\theta_{ij}^k)) + (\mathbf{g}_{i,j}^1 - \mathbf{g}_{i,j}^2))/s), \\ \mathbf{g}_{i,j}^1, \mathbf{g}_{i,j}^2 \sim \mathrm{Gumbel}(0, 1), \forall\ i, j \end{aligned} \tag{3}$$

where $\mathbf{g}_{i,j}^1, \mathbf{g}_{i,j}^2$ are vectors of i.i.d samples drawn from Gumbel distribution, σ is the sigmoid activation and s is a parameter that controls the smoothness of samples, so that the distribution converges to categorical values as $s \to 0$.

The link predictor takes each pair of extracted features $(\mathbf{z}_i^k, \mathbf{z}_j^k)$ of window k and maps their similarity to a $\theta_{ij}^k \in [0, 1]$ by applying fully connected layers. Then the Gumbel reparameterization trick is used to approximate a sigmoid activation function while retaining differentiability:

$$\theta_{ij}^k = \sigma\Big(\mathrm{FC}\big(\mathrm{FC}(\mathbf{z}_i^k \| \mathbf{z}_j^k)\big)\Big) \tag{4}$$

In order to obtain directed and forward (i. e., no look-back in previous time steps in the history) graph structures $\mathcal{G}$ we only learn the upper triangular part of the adjacency matrices.

3.3 Graph Neural Network for Forecasting

Once the collection $\mathcal{G}$ of learnable graph structures per sliding window k are sampled, standard GNN architectures can be applied for capturing the node-to-node relations, i. e., the temporal graph dynamics. GraphSAGE [11] was chosen as the basic building GNN block of the node embedding learning architecture as it can effectively generalize across different graphs with the same attributes. GraphSAGE is an inductive framework that exploits node feature information

and generates node embeddings (i. e., $\mathbf{h}_u$ for node u) via a learnable function, by sampling and aggregating features from a node's local neighborhood (i. e., $\mathcal{N}(u)$).

Let $(\mathcal{V}^k, \mathcal{E}^k)$ correspond to the set of nodes and edges of the learnable graph structure for each $\mathcal{G}^k$. The node embedding update process for each $p \in \{1, \ldots, P\}$ aggregation steps, employs the mean-based aggregator, namely convolutional, by calculating the element-wise mean of the vectors in $\{\mathbf{h}_u^{p-1}, \forall u \in \mathcal{N}(u)\}$, such that:

$$\mathbf{h}_u^p \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_u^{p-1}\} \cup \{\mathbf{h}_u^{p-1} \forall u \in \mathcal{N}(u)\})) \tag{5}$$

where $\mathbf{W}$ trainable weights. The final normalized (i. e., $\tilde{\mathbf{h}}_u^p$) representation of the last node (i. e., time step) in each forward and directed graph denoted as $\mathbf{z}_{u_T} = \tilde{\mathbf{h}}_{u_T}^p$ is passed to the output module. The output module consists of two fully connected layers which reduce the vector into the final output dimension, so as to correspond to the forecasts $\hat{\mathbf{X}}_t, \hat{\mathbf{X}}_{t+1}, \ldots, \hat{\mathbf{X}}_{t+h-1}$. Figure 1 demonstrates the feature extraction, graph learning, GNN and output modules of the proposed TimeGNN architecture.

3.4 Training and Inference

To train the parameters of Eq. (1) for the time series point forecasting task, we use the mean absolute error loss (MAE). Let $\hat{\mathbf{X}}^{(i)}, i \in \{1, ..., K\}$ denote the predicted vector values for K samples, then the MAE loss is defined as:

$$\mathcal{L} = \frac{1}{K} \sum_{i=1}^{K} \|\hat{\mathbf{X}}^{(i)} - \mathbf{X}^{(i)}\|$$

The optimized weights for the feature extraction, graph structure learning, GNN and output modules are selected based on the minimum validation loss during training, which is evaluated as described in the experimental setup (Sect. 4.3)

4 Experimental Evaluation

We next describe the experimental setup, including the datasets and baselines used for comparisons. We also demonstrate and analyze the results obtained by the proposed TimeGNN architecture and the baseline models.

4.1 Datasets

This work was evaluated on the following multivariate time series datasets:

Exchange-Rate which consists of the daily exchange rates of 8 countries from 1990 to 2016, following the preprocessing of [18].

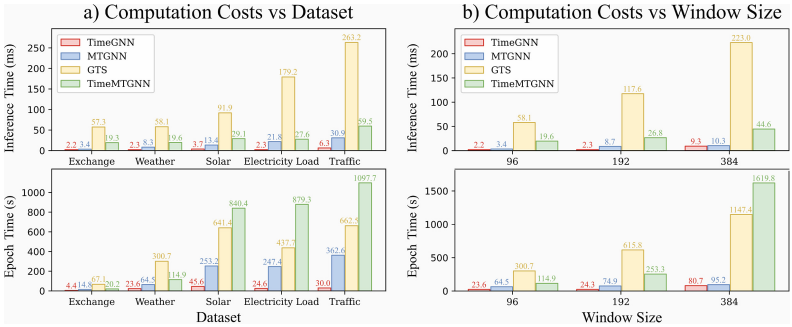


Fig. 2. Computation costs of TimeGNN, TimeMTGNN and baseline models. (a) The inference and epoch training time per epoch between datasets. (b) The inference and epoch times with varying window sizes on the weather dataset

Weather that contains hourly observations of 12 climatological features over a period of four years[1], preprocessed as in [35].

Electricity-Load is based on the UCI Electricity Consuming Load dataset[2] that records the electricity consumption of 370 Portuguese clients from 2011 to 2014. As in [35], the recordings are binned into hourly intervals over the period of 2012 to 2014 and incomplete clients are removed.

Solar-Energy contains the solar power production records in 2006, sampled every 10 minutes from 137 PV plants in Alabama State[3].

Traffic is a collection of 48 months, between 2015 and 2016, of hourly data from the California Department of Transportation[4]. The data describes the road occupancy rates (between 0 and 1) measured by different sensors.

4.2 Baselines

We consider five baseline models for comparison with our TimeGNN proposed architecture. We chose two graph-based methods, MTGNN [32] and GTS [29], and three non graph-based methods, LSTNet [18], LSTM [12], and TCN [1]. Also, we evaluate the performance of TimeMTGNN, a variant of MTGNN that includes our proposed graph learning module. LSTM and TCN follow the size of the hidden dimension and number of layers of TimeGNN. Those were fixed to three layers with hidden dimensions of 32, 64 for the Exchange-Rate and

[1] https://www.ncei.noaa.gov/data/local-climatological-data/.
[2] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014.
[3] http://www.nrel.gov/grid/solar-power-data.html.
[4] http://pems.dot.ca.gov.

Weather datasets and 128 for Electricity, Solar-Energy and Traffic. In the case of MTGNN, GTS, and LSTNet, parameters were kept as close as possible to the ones mentioned in their experimental setups.

4.3 Experimental Setup

Each model is trained for two runs for 50 epochs and the average mean squared error (MSE) and mean absolute error (MAE) score on the test set are recorded. The model chosen for evaluation is the one that performs the best on the validation set during training. The same dataloader is used for all models where the train, validation, and test splits are 0.7, 0.1, and 0.2 respectively. The data is split first and each split is scaled using the standard scalar. The dataloader uses windows of length 96 and batch size 16. The forecasting horizons tested are 1, 3, 6, and 9 time steps into the future, where the exact value of the time step is dependent on the dataset (e. g., 3 time steps would correspond to 3 h into the future for the weather dataset and 3 days into the future for the Exchange dataset). In this paper, we use single-step forecasting for ease of comparison with other baseline methods. For training, we use the Adam optimizer with a learning rate of 0.001. Experiments for the Weather and Exchange datasets were conducted on an NVIDIA T4 and Electricity-Load, Solar, and Traffic on an NVIDIA A40.

4.4 Results

Scalability. We compare the inference and training times of the graph-based models TimeGNN, MTGNN, GTS in Fig. 2. These figures also include recordings from the ablation study of the TimeMTGNN variant, which is described in the relevant paragraph below. Figure 2(a) shows the computational costs on each dataset. Among the baseline models, GTS is the most costly in both inference and training time due to the use of the entire training dataset for graph construction. In contrast, MTGNN learns static node features and is subsequently more efficient. In inference time, as the number of variables increases there is a noticeable increase in inference time for MTGNN and GTS as their graph sizes also increase. TimeGNN's graph does not increase in size with the number of variables and consequently, the inference time scales well across datasets. The training epoch times follow the observations in inference time.

Since the size of the graphs used by TimeGNN is based on window size, the cost of increasing the window size on the weather dataset is shown in Fig. 2(b). As the window size increases, so does the cost of inference and training for all models. As the graph learning modules for MTGNN and GTS do not interact with the window size, the increase in cost can primarily be attributed to their forecasting modules. MTGNN's inference times do not increase as dramatically as GTS's, implying a more robust forecasting module. As the window size increases, TimeGNN's inference and training cost growth is slower than the other methods and remains the fastest of the GNN methods. The time-based graph learning module does not become overly cumbersome as window sizes increase.

Table 1. Forecasting performance for all multivariate datasets and baselines for different horizons h - best in bold, second best underlined.

Exchange-Rate								
Metric		LSTM	TCN	LSTN	GTS	MTGNN	TimeGNN	TimeMTGNN
h = 1	mse	0.328 ± 0.007	0.094 ± 0.118	**0.004 ± 0.000**	0.005 ± 0.001	0.006 ± 0.002	0.129 ± 0.012	**0.004 ± 0.001**
	mae	0.475 ± 0.033	0.191 ± 0.163	**0.033 ± 0.000**	0.041 ± 0.004	0.048 ± 0.011	0.294 ± 0.029	0.034 ± 0.005
h = 3	mse	0.611 ± 0.001	0.063 ± 0.035	0.013 ± 0.003	0.009 ± 0.000	0.012 ± 0.000	0.368 ± 0.059	**0.008 ± 0.001**
	mae	0.631 ± 0.031	0.190 ± 0.041	0.078 ± 0.012	0.063 ± 0.000	0.078 ± 0.000	0.501 ± 0.045	**0.061 ± 0.003**
h = 6	mse	0.877 ± 0.105	0.189 ± 0.221	0.033 ± 0.005	**0.014 ± 0.001**	0.024 ± 0.001	0.354 ± 0.031	0.019 ± 0.004
	mae	0.775 ± 0.032	0.290 ± 0.214	0.139 ± 0.008	**0.081 ± 0.005**	0.111 ± 0.000	0.453 ± 0.052	0.099 ± 0.016
h = 9	mse	0.823 ± 0.118	0.123 ± 0.030	0.030 ± 0.006	**0.020 ± 0.001**	0.035 ± 0.003	0.453 ± 0.149	0.034 ± 0.002
	mae	0.743 ± 0.080	0.277 ± 0.037	0.124 ± 0.011	**0.096 ± 0.001**	0.140 ± 0.008	0.543 ± 0.084	0.139 ± 0.010
Weather								
Metric		LSTM	TCN	LSTN	GTS	MTGNN	TimeGNN	TimeMTGNN
h = 1	mse	**0.162 ± 0.001**	0.176 ± 0.006	0.193 ± 0.001	0.209 ± 0.003	0.232 ± 0.008	0.178 ± 0.001	0.182 ± 0.003
	mae	0.202 ± 0.003	0.220 ± 0.011	0.236 ± 0.002	0.213 ± 0.004	0.230 ± 0.002	**0.185 ± 0.000**	0.186 ± 0.000
h = 3	mse	**0.221 ± 0.000**	0.232 ± 0.003	0.233 ± 0.001	0.320 ± 0.005	0.263 ± 0.003	0.234 ± 0.001	0.234 ± 0.002
	mae	0.265 ± 0.000	0.275 ± 0.000	0.285 ± 0.000	0.320 ± 0.001	0.273 ± 0.000	**0.249 ± 0.001**	0.251 ± 0.001
h = 6	mse	0.268 ± 0.004	0.274 ± 0.002	**0.266 ± 0.001**	0.374 ± 0.003	0.301 ± 0.003	0.287 ± 0.002	0.282 ± 0.007
	mae	0.320 ± 0.004	0.323 ± 0.001	0.321 ± 0.000	0.388 ± 0.002	0.311 ± 0.002	**0.297 ± 0.001**	0.300 ± 0.003
h = 9	mse	0.292 ± 0.007	0.307 ± 0.009	**0.288 ± 0.000**	0.399 ± 0.002	0.329 ± 0.001	0.316 ± 0.001	0.311 ± 0.002
	mae	0.342 ± 0.003	0.350 ± 0.005	0.345 ± 0.003	0.420 ± 0.004	0.339 ± 0.004	**0.331 ± 0.001**	**0.331 ± 0.001**
Electricity-Load								
Metric		LSTM	TCN	LSTN	GTS	MTGNN	TimeGNN	TimeMTGNN
h = 1	mse	0.226 ± 0.002	0.267 ± 0.001	0.064 ± 0.001	0.135 ± 0.002	**0.046 ± 0.000**	0.211 ± 0.003	0.047 ± 0.000
	mae	0.323 ± 0.000	0.375 ± 0.002	0.167 ± 0.001	0.246 ± 0.001	**0.131 ± 0.000**	0.309 ± 0.001	0.135 ± 0.000
h = 3	mse	0.255 ± 0.001	0.329 ± 0.015	**0.065 ± 0.001**	0.303 ± 0.019	0.079 ± 0.001	0.179 ± 0.003	0.077 ± 0.000
	mae	0.339 ± 0.000	0.406 ± 0.013	**0.163 ± 0.002**	0.388 ± 0.019	0.171 ± 0.000	0.320 ± 0.002	0.173 ± 0.000
h = 6	mse	0.253 ± 0.005	0.331 ± 0.010	0.125 ± 0.006	0.334 ± 0.000	**0.097 ± 0.000**	0.246 ± 0.004	0.104 ± 0.015
	mae	0.340 ± 0.006	0.408 ± 0.009	0.238 ± 0.005	0.413 ± 0.000	**0.189 ± 0.001**	0.332 ± 0.004	0.200 ± 0.016
h = 9	mse	0.271 ± 0.009	0.349 ± 0.022	0.144 ± 0.013	0.289 ± 0.021	0.108 ± 0.002	0.258 ± 0.010	**0.104 ± 0.001**
	mae	0.351 ± 0.003	0.410 ± 0.019	0.251 ± 0.013	0.368 ± 0.020	0.198 ± 0.002	0.344 ± 0.007	**0.196 ± 0.001**
Solar-Energy								
Metric		LSTM	TCN	LSTN	GTS	MTGNN	TimeGNN	TimeMTGNN
h = 1	mse	0.019 ± 0.000	0.012 ± 0.000	0.007 ± 0.000	0.012 ± 0.001	**0.006 ± 0.000**	0.022 ± 0.000	**0.006 ± 0.000**
	mae	0.064 ± 0.000	0.055 ± 0.001	0.035 ± 0.000	0.046 ± 0.003	**0.026 ± 0.000**	0.059 ± 0.000	**0.026 ± 0.000**
h = 3	mse	0.031 ± 0.000	0.030 ± 0.001	0.026 ± 0.000	0.044 ± 0.001	**0.022 ± 0.002**	0.030 ± 0.000	**0.022 ± 0.000**
	mae	0.086 ± 0.002	0.087 ± 0.004	0.080 ± 0.000	0.098 ± 0.003	**0.058 ± 0.002**	0.071 ± 0.000	**0.058 ± 0.000**
h = 6	mse	0.046 ± 0.001	0.050 ± 0.000	0.049 ± 0.004	0.103 ± 0.001	**0.042 ± 0.000**	0.044 ± 0.000	0.043 ± 0.002
	mae	0.108 ± 0.005	0.121 ± 0.005	0.125 ± 0.013	0.163 ± 0.001	**0.086 ± 0.001**	0.090 ± 0.000	0.088 ± 0.004
h = 9	mse	0.067 ± 0.003	0.073 ± 0.001	0.068 ± 0.000	0.167 ± 0.003	**0.055 ± 0.001**	0.060 ± 0.002	0.060 ± 0.000
	mae	0.138 ± 0.009	0.150 ± 0.005	0.154 ± 0.004	0.218 ± 0.006	**0.101 ± 0.001**	0.109 ± 0.001	0.110 ± 0.000
Traffic								
Metric		LSTM	TCN	LSTN	GTS	MTGNN	TimeGNN	TimeMTGNN
h = 1	mse	0.558 ± 0.007	0.594 ± 0.091	0.246 ± 0.002	0.520 ± 0.010	**0.233 ± 0.003**	0.567 ± 0.002	0.293 ± 0.026
	mae	0.296 ± 0.005	0.352 ± 0.025	0.203 ± 0.002	0.319 ± 0.013	**0.157 ± 0.002**	0.281 ± 0.000	0.162 ± 0.001
h = 3	mse	0.595 ± 0.014	0.615 ± 0.002	0.447 ± 0.010	0.970 ± 0.027	**0.438 ± 0.001**	0.622 ± 0.006	0.465 ± 0.012
	mae	0.318 ± 0.007	0.363 ± 0.003	0.286 ± 0.009	0.456 ± 0.010	**0.205 ± 0.000**	0.306 ± 0.002	0.218 ± 0.007
h = 6	mse	0.603 ± 0.001	0.680 ± 0.021	0.465 ± 0.005	0.938 ± 0.048	**0.450 ± 0.009**	0.623 ± 0.004	0.495 ± 0.012
	mae	0.321 ± 0.003	0.403 ± 0.013	0.288 ± 0.002	0.461 ± 0.023	**0.213 ± 0.003**	0.311 ± 0.007	0.239 ± 0.001
h = 9	mse	0.614 ± 0.011	0.655 ± 0.017	**0.467 ± 0.010**	0.909 ± 0.024	0.471 ± 0.000	0.622 ± 0.002	0.494 ± 0.000
	mae	0.329 ± 0.010	0.382 ± 0.014	0.290 ± 0.006	0.453 ± 0.016	**0.220 ± 0.002**	0.313 ± 0.002	0.236 ± 0.005

Forecasting Quality. Table 1 summarizes the forecasting performance of the baseline models and TimeGNN for different horizons $h \in \{1, 3, 6, 9\}$.

In general, GTS has the best forecasting performance on the smaller Exchange-Rate dataset. The use of the training data during graph construction

may give GTS an advantage over the other methods on this dataset. TimeGNN however shows signs of overfitting during training and is unable to match the other two GNNs. On the Weather dataset, the purely recurrent methods perform the best in MSE score across all horizons. TimeGNN is competitive with the recurrent methods on these metrics and surpasses the recurrent models on MAE. This suggests TimeGNN is producing more significant outlier predictions than the recurrent methods and TimeGNN is the best performing GNN method.

On the larger Electricity-Load, Solar-Energy, and Traffic datasets, in general, MTGNN is the top performer with LSTNet close behind. However, for larger horizons, TimeGNN performs better than GTS and competitively with LSTNet and the other recurrent models. This shows that time-domain graphs can successfully capture long-term dependencies within a dataset although TimeGNN struggles more with short-term predictions. This could also be attributed to the simplicity of TimeGNN's forecasting module compared to the other graph-based approaches.

Ablation Study. To empirically examine the effects of the forecasting module and the representation power of the proposed graph construction module in TimeGNN, we conducted an ablation study where we replaced MTGNN's graph construction module with our own, so-called TimeMTGNN baseline. The remaining modules and the hyperparameters in TimeMTGNN are kept as similar as possible to MTGNN. TimeMTGNN shows comparable forecasting performance to MTGNN on the larger Electricity-Load, Solar-Energy, and Traffic datasets and higher performance on the smaller Exchange-Rate and Weather datasets. This shows the TimeGNN graph construction module is capable of learning meaningful graph representations that do not impede and in some cases improve forecasting quality. As seen in Fig. 2, the computational performance of TimeMTGNN suffers in comparison to MTGNN. A major contributing factor is the number of graphs produced. MTGNN learns a single graph for a dataset while TimeGNN produces one graph per window, accordingly, the number of GNN operations is greatly increased. However, the focus of this experiment was to confirm that the proposed temporal graph-learning module preserves or improves accuracy over static ones rather than to optimize efficiency.

5 Conclusion

We have presented a novel method of representing and dynamically generating graphs from raw time series. While conventional methods construct graphs based on the variables, we instead construct graphs such that each time step is a node. We use this method in TimeGNN, a model consisting of a graph construction module and a simple GNN-based forecasting module, and examine its performance against state-of-the-art neural networks. While TimeGNN's relative performance differs between datasets, this representation is clearly able to

capture and learn the underlying properties of time series. Additionally, it is far faster and more scalable than existing graph methods as both the number of variables and the window size increase.

Acknowledgements. This work is supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and the IP Paris "PhD theses in Artificial Intelligence (AI)" funding programme by ANR. Computational resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at Alvis partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

1. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 (2018)
2. Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. Wiley, New York (2015)
3. Cao, D., et al.: Spectral temporal graph neural network for multivariate time-series forecasting. Adv. Neural. Inf. Process. Syst. **33**, 17766–17778 (2020)
4. Chauhan, S., Vig, L.: Anomaly detection in ECG time signals via deep long short-term memory networks. In: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–7. IEEE (2015)
5. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
6. Ding, X., Zhang, Y., Liu, T., Duan, J.: Deep learning for event-driven stock prediction. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
7. Donner, R.V., Zou, Y., Donges, J.F., Marwan, N., Kurths, J.: Recurrence networks-a novel paradigm for nonlinear time series analysis. New J. Phys. **12**(3), 033025 (2010)
8. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning, pp. 1263–1272. PMLR (2017)
9. Gligorijević, V., et al.: Structure-based protein function prediction using graph convolutional networks. Nat. Commun. **12**(1), 3168 (2021)
10. Hamilton, J.D.: Time Series Analysis. Princeton University Press, Princeton (2020)
11. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
13. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with Gumbel-Softmax. arXiv preprint arXiv:1611.01144 (2016)
14. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. J. Comput. Aided Mol. Des. **30**, 595–608 (2016)
15. Kipf, T., Fetaya, E., Wang, K.C., Welling, M., Zemel, R.: Neural relational inference for interacting systems. In: International Conference on Machine Learning, pp. 2688–2697. PMLR (2018)

16. Kosma, C., Nikolentzos, G., Xu, N., Vazirgiannis, M.: Time series forecasting models copy the past: How to mitigate. In: Pimenidis, E., Angelov, P., Jayne, C., Papaleonidas, A., Aydin, M. (eds.) Artificial Neural Networks and Machine Learning–ICANN 2022: 31st International Conference on Artificial Neural Networks, Bristol, UK, 6–9 September 2022, Proceedings, Part I, vol. 13529, pp. 366–378. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15919-0_31
17. Lacasa, L., Luque, B., Ballesteros, F., Luque, J., Nuno, J.C.: From time series to complex networks: the visibility graph. Proc. Natl. Acad. Sci. **105**(13), 4972–4975 (2008)
18. Lai, G., Chang, W.C., Yang, Y., Liu, H.: Modeling long-and short-term temporal patterns with deep neural networks. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 95–104 (2018)
19. Le Guen, V., Thome, N.: Deep time series forecasting with shape and temporal criteria. IEEE Trans. Pattern Anal. Mach. Intell. **45**(1), 342–355 (2022)
20. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
21. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting. arXiv preprint arXiv:1707.01926 (2017)
22. Lim, B., Zohren, S.: Time-series forecasting with deep learning: a survey. Phil. Trans. R. Soc. A **379**(2194), 20200209 (2021)
23. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
24. Oreshkin, B.N., Carpov, D., Chapados, N., Bengio, Y.: N-beats: neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437 (2019)
25. Rousseau, F., Vazirgiannis, M.: Graph-of-word and TW-IDF: new approach to Ad Hoc IR. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 59–68 (2013)
26. Rubanova, Y., Chen, R.T., Duvenaud, D.K.: Latent ordinary differential equations for irregularly-sampled time series. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
27. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
28. Seo, Y., Defferrard, M., Vandergheynst, P., Bresson, X.: Structured sequence modeling with graph convolutional recurrent networks. In: Cheng, L., Leung, A., Ozawa, S. (eds.) Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, 13–16 December 2018, Proceedings, Part I 25, pp. 362–373. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04167-0_33
29. Shang, C., Chen, J., Bi, J.: Discrete graph structure learning for forecasting multiple time series. arXiv preprint arXiv:2101.06861 (2021)
30. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
31. Wu, H., Xu, J., Wang, J., Long, M.: Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. Adv. Neural. Inf. Process. Syst. **34**, 22419–22430 (2021)
32. Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 753–763 (2020)

33. Yu, B., Yin, H., Zhu, Z.: Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875 (2017)
34. Zhao, L., et al.: T-GCN: a temporal graph convolutional network for traffic prediction. IEEE Trans. Intell. Transp. Syst. **21**(9), 3848–3858 (2019)
35. Zhou, H., et al.: Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 11106–11115 (2021)

UnboundAttack: Generating Unbounded Adversarial Attacks to Graph Neural Networks

Sofiane Ennadir[1(✉)], Amr Alkhatib[1], Giannis Nikolentzos[2], Michalis Vazirgiannis[1,2], and Henrik Boström[1]

[1] EECS, KTH Royal Institute of Technology, Stockholm, Sweden
ennadir@kth.se
[2] LIX, Ecole Polytechnique, Paris, France

Abstract. Graph Neural Networks (GNNs) have demonstrated state-of-the-art performance in various graph representation learning tasks. Recently, studies revealed their vulnerability to adversarial attacks. While the available attack strategies are based on applying perturbations on existing graphs within a specific budget, proposed defense mechanisms successfully guard against this type of attack. This paper proposes a new perspective founded on unrestricted adversarial examples. We propose to produce adversarial attacks by generating completely new data points instead of perturbing existing ones. We introduce a framework, so-called UnboundAttack, leveraging the advancements in graph generation to produce graphs preserving the semantics of the available training data while misleading the targeted classifier. Importantly, our method does not assume any knowledge about the underlying architecture. Finally, we validate the effectiveness of our proposed method in a realistic setting related to molecular graphs.

Keywords: Adversarial Attacks · Graph Neural Networks

1 Introduction

In recent years, Graph Neural Networks (GNNs) emerged as an effective approach to learning powerful graph representations. These neural network-based models, for instance Graph Convolution Networks (GCNs) [11], have shown to be highly effective in a number of graph-based applications such as drug design [10]. However, recent literature has shown that these architectures can be attacked by injecting small perturbations into the input [2,22]. These attacks, referred to as adversarial attacks in the literature, are highly critical, and this vulnerability has raised tremendous concerns about applying them in safety-critical applications such as financial and healthcare applications. For example, a malicious user could exploit this limitations by adding some inaccurate information to social networks. As a result, several studies focus on developing methods to mitigate the possible perturbation effects in parallel to these attacks. The proposed methods include adversarial training [5], enhancing the robustness of an input GNN through edge pruning [25], and recently proposing robustness certificates [15].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 100–111, 2024.
https://doi.org/10.1007/978-3-031-53468-3_9

The currently available attacks are mainly based to applying small perturbations on either the structure or the node features of the graph [23,26]. Given that most of the proposed defense strategies enhance the robustness of the classifiers to small perturbations [9], they have shown some success in detecting these attacks and in limiting their effect. Moreover, most existing approaches formulate the problem of generating adversarial attacks as a search or constrained optimization problem. While the available constrained optimization tools are easily applicable in continuous input domains (i. e., images), adapting them to discrete domains such as graphs represents a significant challenge. Furthermore, in contrast to images, changing the graph structure by adding/deleting an edge may be infeasible and easily detectable in many settings. For instance, given a molecular graph where the edges represent chemical bonds, by deleting/adding an edge, the emerging graph may not represent a realistic molecule anymore.

To tackle the aforementioned limitations, in this paper, we introduce UnboundAttack, a more general and realistic attack mechanism which creates new adversarial examples from scratch instead of just applying perturbations to an input graph. The approach capitalizes on recent advancements in the field of Generative Adversarial Networks (GANs) to generate a set of legitimate graphs that share similar properties with the input graphs. These properties include degree distribution, diameter and subgraph structures among others. This approach of producing artificially generated graphs that do not emerge directly from input samples and which can mislead a targeted victim model is known as unbounded adversarial attacks. The term "unbounded" in this setting refers to the idea that these attacks are not directly linked to a specific existing graph but rather to a more general view of the dataset to be attacked. We validate in an experimental setting that these attacks can actually mislead the victim classifier but not some oracle function, thus presenting a major threat for real-world applications. The proposed framework is general and can operate on top of any GNN. Our main contributions are summarized as follows:

- We propose UnboundAttack, a generative framework for crafting from scratch adversarial attacks to pretrained GNNs. The proposed framework assumes no knowledge about the underlying architecture of the attacked model and may be applied to an ensemble of available models.
- We designed a realistic experimental setting using molecular data in which our model is evaluated and we show its effectiveness and ability to generate realistic and relevant attacks.

2 Related Work

Given the discrete nature of graphs, applying attack methods from other domains is very challenging. Similarly to the image domain attacks, most available methods formulate the task as a search problem. The objective of the task is to find the closest adversarial perturbation to a given input data point. This approach has led to several proposed attack strategies. For example, Nettack [26] introduced a targeted attack on both the graph structure and nodes features based

on a greedy optimization algorithm of an attack loss to a surrogate model. In addition, [27] formulate the problem as a bi-level optimization task and leverages meta-gradients to solve it. [23] expanded this work by proposing a black-box gradient attack algorithm to overcome several limitations of the original work. From another perspective, [3] propose to use Reinforcement Learning to solve the search problem and hence generate adversarial attacks. In the same context, the work [17] proposed to inject fake nodes into the graph and leveraged Reinforcement Learning to manipulate the labels and links of the injected nodes without changing the connectivity and other metrics between existing node. While the majority of the work is focusing on node classification, very few methods were proposed for the graph classification task. For instance, [18] proposed a new optimization-based approach to tackle the adversarial attack in a black-box setting. Moreover, [13] formulated the adversarial attack problem as an optimization problem and proposed an efficient solution based on a searching algorithm and query-efficient gradient. Finally, [24] designed an attack strategy based on learning a scoring function to rank nodes based on the attacker's expectation.

Formulating adversarial attacks as an optimal search problem has important limitations. For example, the search/optimization process should be performed for each attack input, which generally leads to a limited set of non-diverse perturbations. The attacks are therefore easily detectable by defense mechanisms such edge pruning [25]. In this perspective, unconstrained optimization approaches have lately been proposed to tackle these limitations. While the unconstrained approach has been investigated in other domains, such as images, to our knowledge, it has not been studied for graphs. For instance, [16] proposed synthesizing unrestricted adversarial images entirely from scratch using a conditional generative model and [1] proposed to semantically manipulate images to attack models.

3 Preliminaries

Before continuing with our contribution, we begin by introducing the graph classification problem and some key notation.

3.1 Graph Neural Networks

Let $G = (V, E)$ be a graph where V is its set of vertices and E its set of edges. We will denote by $n = |V|$ and $m = |E|$ the number of vertices and number of edges, respectively. Let $\mathcal{N}(v)$ denote the set of neighbors of a node $v \in V$, i. e., $\mathcal{N}(v) = \{u \colon (v, u) \in E\}$. The degree of a node is equal to its number of neighbors, i. e., equal to $|\mathcal{N}(v)|$ for a node $v \in V$. A graph is commonly represented by its adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ where the (i, j)-th element of this matrix is equal to the weight of the edge between the i-th and j-th node of the graph and a weight of 0 in case the edge does not exist. In some settings, the nodes of a graph might be annotated with feature vectors. We use $\mathbf{X} \in \mathbb{R}^{n \times K}$ to denote the node features where K is the feature dimensionality.

A GNN model consists of a series of neighborhood aggregation layers which use the graph structure and the nodes' feature vectors from the previous layer

to generate new representations for the nodes. Specifically, GNNs update nodes' feature vectors by aggregating local neighborhood information. Suppose we have a GNN model that contains T neighborhood aggregation layers. Let also $\mathbf{h}_v^{(0)}$ denote the initial feature vector of node v, i. e., the row of matrix $\mathbf{X}$ that corresponds to node v. At each iteration ($t > 0$), the hidden state $\mathbf{h}_v^{(t)}$ of a node v is updated as follows:

$$\mathbf{a}_v^{(t)} = \text{AGGREGATE}^{(t)}\Big(\big\{\mathbf{h}_u^{(t-1)} : u \in \mathcal{N}(v)\big\}\Big); \mathbf{h}_v^{(t)} = \text{COMBINE}^{(t)}\Big(\mathbf{h}_v^{(t-1)}, \mathbf{a}_v^{(t)}\Big) \tag{1}$$

where AGGREGATE is a permutation invariant function that maps the feature vectors of the neighbors of a node v to an aggregated vector. This aggregated vector is passed along with the previous representation of v (i. e., $\mathbf{h}_v^{(t-1)}$) to the COMBINE function which combines those two vectors and produces the new representation of v. After T iterations of neighborhood aggregation, to produce a graph-level representation, GNNs apply a permutation invariant readout function, e. g., the sum or mean operator, to nodes feature vectors as follows:

$$\mathbf{h}_G = \text{READOUT}\Big(\big\{\mathbf{h}_v^{(T)} : v \in V\big\}\Big) \tag{2}$$

3.2 Adversarial Attacks

Given a classifier f, an input data point $G \in \mathcal{G}$ and its corresponding label $y \in \mathcal{Y}$ where $f(G) = y$, the goal of an adversarial attack is to produce a perturbed graph $\tilde{G}$ slightly different from the original graph with its predicted class being different from the predicted class of G. This could be formulated as finding a $\tilde{G}$ with $f(\tilde{G}) = \tilde{y} \neq y$ subject to $d(G, \tilde{G}) < \epsilon$ with d being some distance function between the original and perturbed graphs. For instance, d can be a matrix norm of the difference of the aligned adjacency matrices $|| \mathbf{A} - \tilde{\mathbf{A}} ||$ (e. g., l_∞, l_2).

4 Proposed Method: UnboundAttack

4.1 Unbounded Adversarial Attacks

We begin by formally defining what in what follows will be referred to as unbounded adversarial attacks. Given the set of graphs under consideration $\mathcal{G}$, let $o : \mathcal{O} \subset \mathcal{G} \rightarrow \{1, 2, .., K\}$ be the oracle from which the true labels of the graphs $\mathcal{Y}$ have been extracted. For instance, this oracle may be a human expert that uses domain knowledge to determine the category/class of a graph. As previously mentioned, a classifier denoted as $f : \mathcal{G} \rightarrow \mathcal{Y} = \{1, 2, .., Y\}$ (e. g., a GNN in our setting) is trained by minimizing a loss function (e. g., cross-entropy loss) in order to approach and estimate this oracle function. The model f is an estimator of the oracle o and therefore we assume that $f \neq o$. Given a graph $G \in \mathcal{G}$, an adversarial attack consists of finding a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ slightly different from the original graph with its predicted class being different from the predicted class of G. In practice, the main objective is to generate a graph $\tilde{G}$ that shares

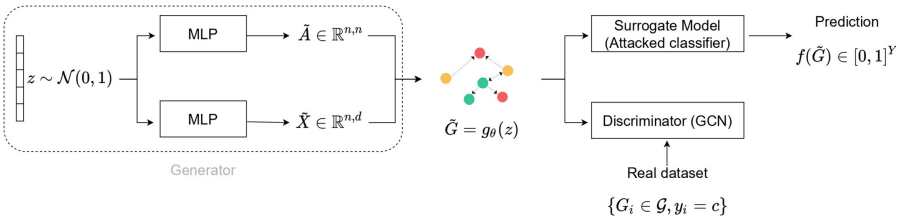


Fig. 1. Illustration of the proposed framework UnboundAttack for generating unbounded adversarial attacks. The framework consists of three main components: (1) A targeted GNN model; (2) A generator consisting of two MLPs taking a sampled vector as input. (3) A classifier distinguishing between generated and real graphs.

similar properties with the graphs of the training set (e. g., similar degree distribution, same motifs, etc.) as the graphs in the training set, and which can fool the classifier but not the oracle. We next define unbounded adversarial attacks.

Definition. (Unbounded adversarial attacks) *Given a small constant* $\epsilon > 0$ *and a graph comparison metric* d*, we define an unbounded adversarial example as a generated graph* $\tilde{G}$ *such as* $f(\tilde{G}) \neq o(\tilde{G})$ *and* $\forall G \in \mathcal{G}, d(G, \tilde{G}) < \epsilon$.

Notice that this definition can be seen as a generalization of the previous work on perturbation-based approaches since the generated graph needs to be similar to all graphs of the dataset. Note that d can be any function that measures distance of graphs, such as norm of the difference of their aligned adjacency matrices $\min_{\mathbf{P} \in \Pi} || \mathbf{A} - \mathbf{P}\tilde{\mathbf{A}}\mathbf{P}^\top ||$ where Π is the set of permutation matrices.

4.2 Architecture Overview

As discussed, we seek to generate a graph with same characteristics as the graphs in the training set, for which the model's prediction differs from the class provided by the oracle. Our objective can be divided into two parts. The first part concerns generating realistic graphs with the same proprieties and semantics as our training set while the second part ensures reaching our adversarial aim.

Recurrent neural networks (RNNs) [21] or other likelihood-based generative models such as variational auto-encoders (VAEs) [12] may be suitable candidates to tackle this task. However, in practice, we choose to use a likelihood-free implicit generative approach; the generative adversarial network (GAN) [6], with a similar approach to prior work of [4]. We highlight that the primary contribution of our work is to provide a new adversarial perspective based on unbounded attacks. Consequently, while we have based our architecture on the GAN framework, other generative approaches could be used according to the targeted downstream task. The flow of our adversarial framework is summarized in Fig. 1, we describe the different components in more details in what follows:

(1) ***Classifier.*** The victim classifier $f : \mathcal{G} \rightarrow [0,1]^Y$ is an instance of a GNN model following the general framework presented in Subsect. 3.1. We treat the

model as a gray-box. Thus, the model is supposed to be trained and fixed, and no assumptions are made about its internal architecture during the attack phase. Depending on the attack strategy, the attacker can chose to operate our framework as a white-box setting by directly using the victim classifier or as a gray-box setting by training a surrogate model.

(2) ***Generator.*** The generator g_θ learns to map a sampled D-dimensional vector from a normal distribution $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ into an adjacency matrix $\tilde{\mathbf{A}}$ and a feature matrix $\tilde{\mathbf{X}}$ representing a graph $\tilde{G}$. Given a previously chosen fixed graph size, we use two multi-layer perceptrons to process the input sampled vector. The output from each MLP is post-processed using a discretization strategy.

(3) ***Discriminator.*** The discriminator d_ϕ learns to distinguish between the generated and the real graphs. The model receives a synthetic graph and has to classify whether it is sampled from the true data distribution or generated by the generator g_θ. It should be noted that the discriminator needs to be invariant to the ordering of the nodes, and thus we employ a GCN [11].

4.3 Training and Loss Function

Generation Loss. The main objective of the training phase is to generate realistic graphs with the same semantics as those of the training set. The training consists, therefore, of learning a relevant discriminator and generator. At each training step, the discriminator enhances its ability to distinguish between real graphs and generated ones, while the generator improves its capacity to generate graphs that mislead the discriminator. The process can be seen as a game between two active players (generator/discriminator) and a third static player (victim classifier) that should converge into an equilibrium. This equilibrium game is regulated by the classical GAN min-max equation:

$$\min_{g_\theta} \max_{d_\phi} \quad \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log d_\phi(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[\log(1 - d_\phi(g_\theta(\mathbf{z})))] \tag{3}$$

To ensure stability during training, we employ the Wasserstein GAN + GP (WGAN-GP) [7] that uses gradient norm penalty to achieve Lipschitz continuity.

Adversarial Loss. Given our objective of generating unbounded adversarial attacks, by choosing a target attack class c, our model is trained to generate graphs that would be classified to the true class from the oracle (i. e., $o(\tilde{G}) =$ c) and to some other class from the classifier (i. e., $f(\tilde{G}) \neq c$). This is mainly reflected in the generator modeling the conditional data distribution $P(\cdot \mid y = c)$. In practice, during training, the discriminator is only given the set of real graphs whose training labels are equal to c, which is defined as: $\mathcal{G}_c = \{G_i \mid G_i \in \mathcal{G}, y_i = c\}$. Furthermore, the output of the generator is evaluated at each iteration by querying the attacked classifier. We strengthen the adversarial ability of the model by including an additional term in the loss function of the generator:

$$\mathcal{L}(\theta) = \mathcal{L}_{WGAN} + \beta \mathcal{L}_{Adv} \tag{4}$$

where $\beta \in [0, 1]$ is a trade-off parameter reflecting our desire to produce valid graphs and mislead the classifier. The second term of the loss function leads the output of the generator to be different from the target class. Specifically, $\mathcal{L}_{Adv}$ may be a reward function taking the generated graph as input and attributing a value based on the prediction probability formulated as $\mathcal{L}_{Adv} : \tilde{\mathcal{G}}_\theta \to \mathbb{R}$. For a single generated graph, we apply the following penalization term:

$$\mathcal{L}_{Adv}(\theta; \mathbf{z}) = \text{ReLU}\big(0.5 - \max_{i \neq c}(f(g_\theta(\mathbf{z}))_i\big) \tag{5}$$

where $\mathbf{z}$ refers to the i-th vector sampled from the normal distribution and given to the generator. Additionally, the $\max_{i \neq c}(f(g_\theta(\mathbf{z}))_i$ refers the maximum component (different from the c-th one) of the predicted probabilities vector of predicted probabilities. At each training step, we evaluate all the graphs produced by the generator given the sampled vectors from the normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$. By minimizing this quantity, we maximize the other classes' (different from c) probabilities, therefore reaching our adversarial target.

Attacks Generation. After several training epochs, we expect the min-max game between the three players (i. e., generator, discriminator, and victim classifier) to converge to an equilibrium. Furthermore, we consider that by achieving this equilibrium, the generator can produce realistic graphs that are both adversarial and preserve the training set's main properties. Therefore, we directly leverage this trained generator during the testing phase to produce a set of adversarial graphs without querying the victim classifier or the discriminator.

5 Experimental Evaluation

In this section, we investigate the ability of the UnboundAttack framework to produce adversarial examples in a realistic experimental setting. We first describe the experimental setup, and then report the results and provide examples of generated graphs. More specifically, we address two main points: **(Q1)** Validity of attacks and **(Q2)** Adversarial aspect of these attacks.

5.1 Experimental Setting

While experimenting with classical adversarial mechanisms is straightforward, evaluating our proposed approach is related to finding an accessible oracle capable of providing the labels of the generated graphs. In this experiment, we focus on chemical compounds using metrics available in the open-source chemoinformatics Python package RDKit. These metrics serve as our oracle from which we can derive the labels. We used the QM9 dataset [14] containing small organic molecules. Each of these available molecules is represented by an undirected graph G where nodes represent atoms and two atoms are connected by an edge if an atomic bond exists between them. Each node and edge in the graph is annotated with an one-hot vector indicating the type of the atom and atomic bond. We choose the following chemical-related metrics to be used as an oracle:

Table 1. Classification accuracy (± standard deviation) of the victim models on the Qm9 dataset on clean and attacked models. The lower the accuracy the better.

Attack strategy	Metric 1 - LogP		Metric 2 - SaS	
	GCN	**GIN**	**GCN**	**GIN**
Clean	97.8 % ± 0.7 %	97.1 % ± 0.2 %	91.4 % ± 0.3 %	89.8 % ± 0.2 %
Random	67.3 % ± 5.7 %	64.7 % ± 4.7 %	62.3 % ± 6.3 %	65.8 % ± 4.2 %
Gradient-based (PGD)	53.2 % ± 1.6 %	54.8 % ± 3.2 %	47.6 % ± 1.7 %	53.1 % ± 1.9 %
GradArgmax	48.5 % ± 2.7 %	51.7 % ± 2.3 %	45.3 % ± 2.4 %	54.9 % ± 3.6 %
Projective Ranking	47.8 % ± 2.3 %	58.3% ± 1.4 %	49.0 % ± 3.1 %	54.7 % ± 1.0 %
UnboundAttack	**45.9 %** ± 2.1 %	**47.3 %** ± 2.9 %	**27.1 %** ± 5.4 %	**31.2 %** ± 4.3 %
Attack strategy	Metric 3 - Density		Metric 4 - Weight	
	GCN	**GIN**	**GCN**	**GIN**
Clean	83.9 % ± 0.5 %	80.2 % ± 0.9 %	97.5 % ± 0.2 %	95.6 % ± 0.1 %
Random	58.3 % ± 4.9 %	67.3 % ± 5.2 %	59.4 % ± 4.2 %	63.7 % ± 5.7 %
Gradient-based (PGD)	49.5 % ± 1.7 %	54.2 % ± 2.7 %	54.0 % ± 2.9 %	51.5 % ± 2.2 %
GradArgmax	46.7 % ± 3.2 %	52.9 % ± 4.6 %	52.5 % ± 6.1 %	53.7 % ± 1.4 %
Projective Ranking	47.8 % ± 2.5 %	55.2 % ± 2.4 %	45.0 % ± 3.3 %	58.8 % ± 1.3 %
UnboundAttack	**43.2 %** ± 8.3 %	**49.7 %** ± 7.1 %	**30.3 %** ± 6.8 %	**40.7 %** ± 9.8 %

- The logP or Octanol-water partition coefficient is the partition coefficient representing the magnitude of the ratio of the concentration in Octanol.
- The Synthetic Accessibility score is a metric reflecting molecules' ease of synthesis (synthetic accessibility).
- The average molecular weight of the molecule and the molecule's density

Using RDKit, a score for each of the available graphs is calculated for the above metrics. By using a threshold (mean value), we convert each problem into a binary classification task. We used two 3-layers Multilayer Perceptron (MLP) models for the generator and a GCN as our discriminator. We arranged the number of message passing layers and hidden dimensions to be different from the victim model (especially in the GCN case) to assume no knowledge about the underlying architecture of the victim classifier. For our experimental evaluation, we used a discretization strategy based on the Gumbel-Softmax [8]. We demonstrate the UnboundAttack framework on two popular GNNs: (1) GCN [11]; and (2) GIN [20]. We used the sum operator as the readout function for both our victim model and the discriminator to produce graph-level representations. Furthermore, we used the cross-entropy loss with the Adam optimizer. We compare the proposed approach to other available methods, which are mainly based on constrained optimization, with respect to the accuracy of the attacker using a separate test set. We performed each experiment 3 times in a 3-fold cross-validation setting to estimate each attack's generalization performance (Table 2).

5.2 Performance Analysis

We first compare the method against the model's initial accuracy on test set (clean) before the attack. We additionally compare the proposed method against

Table 2. Results from the MMD and other metrics (± standard deviation) of the generated samples on the QM9 dataset for LogP metric.

Metric	MMD Metric		Other metrics	
	Deg.	**Clus.**	**Novelty**	**Uniqueness**
Gradient-based (PGD)	0.06 ± 0.01	0.03 ± 0.01	74.6 ± 1.2	66.3 ± 0.9
GradArgmax	0.08 ± 0.02	0.05 ± 0.02	68.6 ± 0.8	62.4 ± 1.1
Projective ranking	0.11 ± 0.01	0.03 ± 0.01	83.5 ± 1.8	82.7 ± 2.3
UnboundedAttack	0.14 ± 0.02	0.05 ± 0.04	89.7 ± 0.4	88.4 ± 0.6

different adversarial attack methods. The first comparison is against a random search method based on randomly adding/deleting edges. The second attack, adapted from [19], is a white-box gradient-based method that either adds/deletes edges by approaching the adversarial attack as an optimization method. The approach aims to find a set of perturbations that minimizes an attack loss given a finite budget of edge perturbations. We consider a specific budget Δ representing the maximum possible magnitude of the perturbations for both approaches and we used the Proximal Gradient descent as an optimization tool. Since this method is highly dependent on the chosen step-size ϵ, we tried different parameters and we reported the best result for each metric. We additionally compared our method to GradArgmax [2] which is based on a gradient-greedy method to select the optimal edge. After identifying the edges, removing or adding the edge depends on the sign of the gradient. We finally evaluate our method against ProjectiveRanking [24]. We note that the attack assumes access to the embedding representations of all nodes from the targeted classifier. Similar to their implementation, we used a 2-layers MLP to serve as the scoring module. We should mention that once the training phase of our method is completed, we can generate an unlimited number of adversarial attacks. In order to make the assessment fair, we set the number of generated points to be similar to the cardinality of the test set of other methods. Furthermore, we validate the quality of our generated attacks using two key perspectives. The first perspective, related to evaluating the quality of the generated graphs, is based on different graph metrics. We use the Maximum Mean Discrepancy (MMD) measures, as presented in [21], using the RBF kernel and for both the degree and clustering coefficient distributions. The second perspective is related to the biochemical validity of the generated samples and we used a Novelty and Uniqueness scores (similar to [4]) computed through RDKit. The classification performance of the GCN and GIN target models for all the metrics using different methods is reported in Table 1 and real examples are provided in Fig. 2.

The results demonstrate the effectiveness of the proposed UnboundAttack strategy. The approach achieves the best attack performance on all datasets and the difference in performance between the proposed approach and the other baselines is significant. In addition, the comparison metrics shows that our method is capable of generating both valid and unique graphs. The obtained results thus

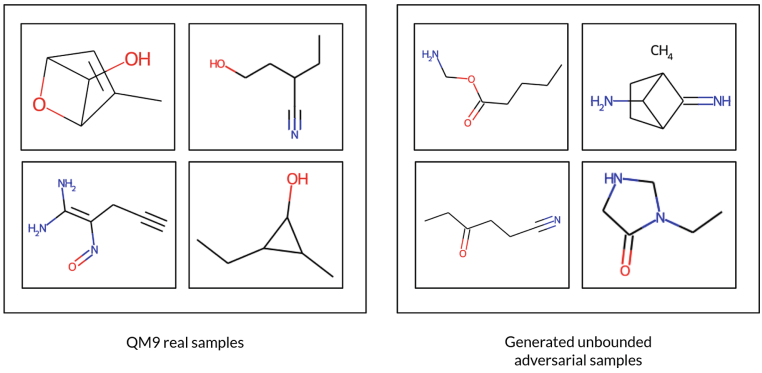


Fig. 2. Examples of graphs from the QM9 dataset *(left)*. Examples of generated attacks *(right)*. These examples have succeeded in misleading the classifier (i. e., $o(G) \neq f(G)$).

answer **Q1** and demonstrate our generator's ability to provide pertinent adversarial attacks from scratch. While it may be argued that the computational cost of our method is higher, we should note that the validity of the output in terms of adversarial attacks is much more reliable. Moreover, in contrast with other methods, such as gradient-based methods, where a specific process is performed for each example, our training is only performed once. We would also like to mention our proposed method's ability to generate diverse valid adversarial attacks. Finally, contrary to the perturbation-based methods, our approach is not limited to the test set to be attacked but can generate a wide range of examples.

6 Conclusion

This work explores a new perspective on adversarial attacks on GNNs. Instead of performing perturbations on a graph by adding/removing edges or editing the nodes' feature vectors, we propose to learn a new graph from scratch using graph generative models. The produced graph has similar semantics to those of the graphs of the training set, and hence may be an effective tool to mislead a victim model. The proposed approach does not assume any knowledge about the architecture of the targeted model. Experiments show that the method performs better or comparable to other methods in degrading the performance of the victim model. This work can be extended to other graph setting such as node classification and edge classification. Furthermore, we anticipate that the proposed architecture may support the development of new defense strategies that could limit the potential negative impact of adversarial attacks, enhancing the ability to deploy GNNs in real practical settings.

Acknowledgements. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

1. Bhattad, A., Chong, M.J., Liang, K., Li, B., Forsyth, D.A.: Unrestricted adversarial examples via semantic manipulation (2019). https://doi.org/10.48550/ARXIV.1904.06347, https://arxiv.org/abs/1904.06347
2. Dai, H., et al.: Adversarial attack on graph structured data (2018). https://doi.org/10.48550/ARXIV.1806.02371, https://arxiv.org/abs/1806.02371
3. Dai, H., et al.: Adversarial attack on graph structured data. In: Proceedings of the 35th International Conference on Machine Learning, pp. 1115–1124 (2018)
4. De Cao, N., Kipf, T.: MolGAN: an implicit generative model for small molecular graphs (2018). https://doi.org/10.48550/ARXIV.1805.11973, https://arxiv.org/abs/1805.11973
5. Feng, F., He, X., Tang, J., Chua, T.S.: Graph adversarial training: dynamically regularizing based on graph structure (2019). https://doi.org/10.48550/ARXIV.1902.08226, https://arxiv.org/abs/1902.08226
6. Goodfellow, I.J., et al.: Generative adversarial networks (2014). https://doi.org/10.48550/ARXIV.1406.2661, https://arxiv.org/abs/1406.2661
7. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of Wasserstein GANs (2017). https://doi.org/10.48550/ARXIV.1704.00028, https://arxiv.org/abs/1704.00028
8. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with Gumbel-Softmax (2016). https://doi.org/10.48550/ARXIV.1611.01144, https://arxiv.org/abs/1611.01144
9. Jin, W., et al.: Adversarial attacks and defenses on graphs: a review, a tool and empirical studies (2020). https://doi.org/10.48550/ARXIV.2003.00653, https://arxiv.org/abs/2003.00653
10. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. J. Comput. Aided Mol. Des. **30**(8), 595–608 (2016)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). https://doi.org/10.48550/ARXIV.1609.02907, https://arxiv.org/abs/1609.02907
12. Kipf, T.N., Welling, M.: Variational graph auto-encoders (2016). https://doi.org/10.48550/ARXIV.1611.07308, https://arxiv.org/abs/1611.07308
13. Mu, J., Wang, B., Li, Q., Sun, K., Xu, M., Liu, Z.: A hard label black-box adversarial attack against graph neural networks (2021). https://doi.org/10.48550/ARXIV.2108.09513, https://arxiv.org/abs/2108.09513
14. Ramakrishnan, R., Dral, P.O., Rupp, M., von Lilienfeld, O.A.: Quantum chemistry structures and properties of 134 kilo molecules. Sci. Data **1**, 140,022 (2014)
15. Schuchardt, J., Bojchevski, A., Gasteiger, J., Günnemann, S.: Collective robustness certificates: exploiting interdependence in graph neural networks. In: International Conference on Learning Representations (2021). https://openreview.net/forum?id=ULQdiUTHe3y
16. Song, Y., Shu, R., Kushman, N., Ermon, S.: Constructing unrestricted adversarial examples with generative models (2018). https://doi.org/10.48550/ARXIV.1805.07894, https://arxiv.org/abs/1805.07894
17. Sun, Y., Wang, S., Tang, X., Hsieh, T.Y., Honavar, V.: Adversarial attacks on graph neural networks via node injections: a hierarchical reinforcement learning approach. In: Proceedings of The Web Conference 2020, WWW 2020, pp. 673–683. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3366423.3380149

18. Wan, X., Kenlay, H., Ru, B., Blaas, A., Osborne, M.A., Dong, X.: Adversarial attacks on graph classification via Bayesian optimisation (2021). https://doi.org/10.48550/ARXIV.2111.02842, https://arxiv.org/abs/2111.02842
19. Xu, K., et al.: Topology attack and defense for graph neural networks: an optimization perspective (2019). https://doi.org/10.48550/ARXIV.1906.04214, https://arxiv.org/abs/1906.04214
20. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: 7th International Conference on Learning Representations (2019)
21. You, J., Ying, R., Ren, X., Hamilton, W.L., Leskovec, J.: GraphRNN: generating realistic graphs with deep auto-regressive models (2018). https://doi.org/10.48550/ARXIV.1802.08773, https://arxiv.org/abs/1802.08773
22. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (2018)
23. Zhan, H., Pei, X.: Black-box gradient attack on graph neural networks: deeper insights in graph-based attack and defense. arXiv preprint arXiv:2104.15061 (2021)
24. Zhang, H., et al.: Projective ranking: a transferable evasion attack method on graph neural networks. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM 2021, pp. 3617–3621. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3459637.3482161
25. Zhang, X., Zitnik, M.: GNNGuard: defending graph neural networks against adversarial attacks (2020). https://doi.org/10.48550/ARXIV.2006.08149, https://arxiv.org/abs/2006.08149
26. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856 (2018)
27. Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. In: 7th International Conference on Learning Representations (2019)

Uncertainty in GNN Learning Evaluations: The Importance of a Consistent Benchmark for Community Detection

William Leeney(✉) and Ryan McConville

University of Bristol, Bristol, UK
will.leeney@bristol.ac.uk

Abstract. Graph Neural Networks (GNNs) have improved unsupervised community detection of clustered nodes due to their ability to encode the dual dimensionality of the connectivity and feature information spaces of graphs. Identifying the latent communities has many practical applications from social networks to genomics. Current benchmarks of real world performance are confusing due to the variety of decisions influencing the evaluation of GNNs at this task. To address this, we propose a framework to establish a common evaluation protocol. We motivate and justify it by demonstrating the differences with and without the protocol. The W Randomness Coefficient is a metric proposed for assessing the consistency of algorithm rankings to quantify the reliability of results under the presence of randomness. We find that by ensuring the same evaluation criteria is followed, there may be significant differences from the reported performance of methods at this task, but a more complete evaluation and comparison of methods is possible.

Keywords: Graph Neural Networks · Community Detection · Hyperparameter Optimisation · Node Clustering · Representation Learning

1 Introduction

GNNs are a popular neural network based approach for processing graph-structured data due to their ability to combine two sources of information by propagating and aggregating node feature encodings along the network's connectivity [14]. Nodes in a network can be grouped into communities based on similarity in associated features and/or edge density [28]. Analysing the structure to find clusters, or communities, of nodes provides useful information for real world problems such as misinformation detection [21], genomic feature discovery [3], social network or research recommendation [38]. As an unsupervised task, clusters of nodes are identified based on the latent patterns within the dataset, rather than "ground-truth" labels. Assessing performance at the discovery of unknown information is useful to applications where label access is prohibited. Some applications of graphs deal with millions of nodes and there is

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 112–123, 2024.
https://doi.org/10.1007/978-3-031-53468-3_10

a low labelling rate with datasets that mimic realistic scenarios [11]. In addition, clustering is particularity relevant for new applications where there is not yet associated ground truth.

However, there is no widely accepted or followed way of evaluating algorithms that is done consistently across the field, despite benchmarks being widely considered as important. Biased benchmarks, or evaluation procedures, can mislead the narrative of research which distorts understanding of the research field. Inconclusive results that may be valuable for understanding or building upon go unpublished, which wastes resources, money, time and energy that is spent on training models. In fields where research findings inform policy decisions or medical practices, publication bias can lead to decisions based on incomplete or biased evidence, potentially causing harm or inefficiency. To accurately reflect the real-world capabilities of research, it would beneficial to use a common framework for evaluating proposed methods.

The framework detailed herein is a motivation and justification for this position. To demonstrate the need for this, we measure the difference between using the default parameters given by the original implementations to those optimised for under this framework. A metric is proposed for evaluating consistency of algorithm rankings over different random seeds which quantifies the robustness of results. This work will help guide practitioners to better model selection and evaluation within the field of GNN community detection.

Contributions. In this paper, we make the following key contributions: 1) We demonstrate that despite benchmarks being accepted as important, many experiments in the field follow different, often unclear, benchmarking procedures to report performance 2) We propose a framework for improving and producing fairer comparisons of GNNs at the task of clustering. For enablement, we open source the code[1] and encourage model developers to submit their methods for inclusion. 3) We quantify the extent to which a hyperparameter optimisation and ranking procedure affects performance using the proposed W Randomness Coefficient, demonstrating that model selection requires use of the framework for accurate comparisons.

2 Related Work

There is awareness of need for rigour in frameworks evaluating machine learning algorithms [25]. Several frameworks for the evaluation of supervised GNNs performing node classification, link prediction and graph classification exist [5,6,22,24]. We are interested in unsupervised community detection which is harder to train and evaluate. Current reviews of community detection provide an overview but do not evaluate and the lack of a consistent evaluation at this task has been discussed for non-neural methods [12,16] but this doesn't currently include GNNs.

[1] https://github.com/willleeney/ugle.

There are various frameworks for assessing performance and the procedure used for evaluation changes the performance of all algorithms [42]. Under consistent conditions, it has been show that simple models can perform better with a thorough exploration of the hyperparameter space [29]. This can be because performance is subject to random initialisations [6]. Lifting results from papers without carrying out the same hyperparameter optimisation over all models is not consistent and is a misleading benchmark. Biased selection of random seeds that skew performance is not fair. Not training over the same number of epochs or not implementing model selection based on the validation set results in unfair comparisons with inaccurate conclusions about model effectiveness. Hence, there is no sufficient empirical evaluation of GNN methods for community detection as presented in this work.

3 Methodology

This section details the procedure for evaluation; the problem that is aimed to solve; the hyperparameter optimisation and the resources allocated to this investigation; the algorithms that are being tested; the metrics of performance and datasets used. At the highest level, the framework coefficient calculation is summarised by Algorithm 1.

Algorithm 1 Overview of the evaluation framework. *train()* is the function to train a model; *optimise()* encompasses the hyperparameter optimisation and uses the model, training function, resources allocated and the current evaluation test; *evaluate()* returns the performance of a model and *ranking-coefficient()* calculates the rankings and coefficient of agreement across the randomness over all tests.

```
Require: tests, models, resources, train(), optimise() evaluate(), ranking-coefficient()
  for test in tests do:
     for model in models do:
        optimise(model, resources, test, train())
        results += evaluate(model, resources, test)
     end for
  end for
  W = ranking-coefficient(results)
```

The current approach to evaluation of algorithms suffer from several shortcomings that hinder the fairness and reliability of model comparisons. A consistent framework establishes a clear and objective basis for comparing models. A common benchmark practice promotes transparency by comprehensively documenting what affects performance, encouraging researchers to compare fairly. Only if the exact same evaluation procedure is followed can results be lifted from previous research, which saves the time and energy of all practitioners. A consistent practise must be established as there is current no reason for confidence in claims of performance. Trustworthy results builds understanding and allows progression of the field.

To evaluate the consistency and quality of the results obtained by this framework, two metrics are proposed. No algorithm evaluated is deterministic as all

require randomness to initialise the network. Therefore, the consistency of a framework can be considered as the average amount that the performance rankings change when different randomness is present over all tests in the framework. The tests of a framework are the performance of a metric on a specific dataset, the ranking is where an algorithm places relative to the other algorithms. Here, the different randomness means that each random seed that the algorithms are evaluated on.

Kendall's W coefficient of concordance [7] is used as the basis for the consistency metric. This is a statistical measure used to assess the level of agreement among multiple judges or raters who rank or rate a set of items. It quantifies the degree of agreement among the judges' rankings. The coefficient ranges from 0 to 1, where higher values indicate greater concordance or agreement among the rankings. The formula for calculating Kendall's W involves comparing pairs of items for each judge and computing the proportion of concordant pairs. We adapt the coefficient to quantify how consistent the ranking of the algorithms under this framework and refer to it as the W Randomness Coefficient. This can be calculated with a number of algorithms a, and random seeds n, along with tests of performance that create rankings of algorithms. For each test of performance, Kendall's W coefficient of concordance is calculated for the consistency of rankings across the random seed. The sum of squared deviations S is calculated for all algorithms, and calculated using the deviation from mean rank due for each random seed. This is averaged over all metrics and datasets that make up the suite of tests $\mathcal{T}$ to give the W Randomness Coefficient defined as Eq. 1. Using the one minus means that if the W is high then randomness has affected the rankings, whereas a consistent ranking procedure results in a lower number. By detailing the consistency of a framework across the randomness evaluated, the robustness of the framework can be maintained, allowing researchers to trust results and maintain credibility of their publications.

$$W = 1 - \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{12S}{n^2(a^3 - a)} \tag{1}$$

The second metric is used to compare whether the performance is better under the hyperparameters or default reported by original implementations. The different parameter sets are given a rank by comparing the performance on every test. This is then averaged across every test, to give the framework comparison rank. Demonstrating that failing to optimize hyperparameters properly can result in sub-optimal performance means that models that could have performed better with proper tuning may appear inferior. This affects decision-making and potentially leading to the adoption of sub-optimal solutions. In the real world, this can have costly and damaging consequences, and is especially critical in domains where model predictions impact decisions, such as healthcare, finance, and autonomous vehicles.

3.1 Problem Definition

The problem definition of community detection on attributed graphs is defined as follows. The graph, where N is the number of nodes in the graph, is represented

as $G = (A, X)$, with the relational information of nodes modelled by the adjacency matrix $A \in \mathbb{R}^{N \times N}$. Given a set of nodes V and a set of edges E, let $e_{i,j} = (v_i, v_j) \in E$ denote the edge that points from v_j to v_i. The graph is considered weighted so, the adjacency matrix $0 < A_{i,j} \leq 1$ if $e_{i,j} \in E$ and $A_{i,j} = 0$ if $e_{i,j} \notin E$. Also given is a set of node features $X \in \mathbb{R}^{N \times d}$, where d represents the number of different node attributes (or feature dimensions). The objective is to partition the graph G into k clusters such that nodes in each partition, or cluster, generally have similar structure and feature values. The only information typically given to the algorithms at training time is the number of clusters k to partition the graph into. Hard clustering is assumed, where each community detection algorithm must assign each node a single community to which it belongs, such that $P \in \mathbb{R}^N$ and we evaluate the clusters associated with each node using the labels given with each dataset such that $L \in \mathbb{R}^N$.

3.2 Hyperparameter Optimisation Procedure

There are sweet spots of architecture combinations that are best for each dataset [1] and the effects of not selecting hyperparameters (HPs) have been well documented. Choosing too wide of a HP interval or including uninformative HPs in the search space can have an adverse effect on tuning outcomes in the given budget [39]. Thus, a HPO is performed under feasible constraints in order to validate the hypothesis that HPO affects the comparison of methods. It has been shown that grid search is not suited for searching for HPs on a new dataset and that Bayesian approaches perform better than random [1]. There are a variety of Bayesian methods that can be used for hyperparameter selection. One such is the Tree Parzen-Estimator (TPE) [2] that can retain the conditionality of variables [39] and has been shown to be a good estimator given limited resources [40]. The multi-objective version of the TPE [23] is used to explore the multiple

Table 1. Resources are allocated an investigation, those detailed are shared across all investigations. Algorithms that are designed to benefit from a small number of HPs should perform better as they can search more of the space within the given budget. All models are trained with 1x 2080 Ti GPU on a server with 12GB of RAM, and a 16core Xeon CPU.

resource	associated allocation
optimiser	Adam
learning rate	$\{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$
weight decay	$\{0.05, 0.005, 0.0005, 0.0\}$
max epochs	5000
patience	$\{25, 100, 500, 1000\}$
max hyperparameter trials	300
seeds	$\{42, 24, 976, 12345, 98765, 7, 856, 90, 672, 785\}$
training/validation split	0.8
train/testing split	0.8

metrics of performance investigated. Given a limited budget, the TPE is optimal as it allows the efficient exploration of parameters (Table 1).

In this framework, a modification to the nested cross validation procedure is used to match reasonable computational budgets, which is to optimise hyperparameters on the first seed tested on and use those hyperparameters on the other seeds. Additionally, it beneficial to establish a common resource allocation such as number of epochs allowed for in training or number of hyperparameter trials. Ensuring the same resources are used in all investigations means that the relatively underfunded researchers can participate in the field, democratising the access to contribution. Conversely, this also means that highly funded researchers cannot bias their results by exploiting the resources they have available.

3.3 Suite of Tests

A test of an algorithm in this framework is the performance under a metric on a dataset. Algorithms are ranked on each test on every random seed used. For evaluation purposes, some metrics require the ground truth, and others do not, although regardless, this knowledge is not used during the training itself. Macro F1-score (F1) is calculated to ensure that evaluations are not biased by the label distribution, as communities sizes are often imbalanced. Normalised mutual information (NMI) is also used, which is the amount of information that can extract from one distribution with respect to a second.

For unsupervised metrics, modularity and conductance are selected. Modularity quantifies the deviation of the clustering from what would be observed in expectation under a random graph. Conductance is the proportion of total edge volume that points outside the cluster. These two metrics are unsupervised as they are calculated using the predicted cluster label and the adjacency matrix, without using any ground truth. Many metrics are used in the framework as they align with specific objectives and ensures that evaluations reflect a clear and understandable assessment of performance.

Generalisation of performance on one dataset can often not be statistically valid and lead to overfitting on a particular benchmark [27], hence multiple are used in this investigation. To fairly compare different GNN architectures, a range of graph topologies are used to fairly represent potential applications. Each dataset can be summarised by commonly used graph statistics: the average clustering coefficient [37] and closeness centrality [36]. The former is the proportion of all the connections that exist in a nodes neighbourhood compared to a fully connected neighbourhood, averaged across all nodes. The latter is the reciprocal of the mean shortest path distance from all other nodes in the graph. All datasets are publicly available[2], and have been used previously in GNN research [17].

Using many datasets for evaluation means that dataset bias is mitigated, which means that the framework is better at assessing the generalisation capability of models to different datasets. These datasets are detailed in Table 2,

[2] https://github.com/yueliu1999/Awesome-Deep-Graph-Clustering.

the following is a brief summary: Cora [19], CiteSeer [8], DBLP [30] are graphs of academic publications from various sources with the features coming from words in publications and connectivity from citations. AMAC and AMAP are extracted from the Amazon co-purchase graph [10]. Texas, Wisc and Cornell are extracted from web pages from computer science departments of various universities [4]. UAT, EAT and BAT contain airport activity data collected from the National Civil Aviation Agency, Statistical Office of the European Union and Bureau of Transportation Statistics [17].

Table 2. The datasets and associated statistics.

Datasets	Nodes	Edges	Features	Classes	Average Clustering Coefficient	Mean Closeness Centrality
amac [10]	13752	13752	80062	10	0.157	0.264
amap [10]	7650	7650	119081	8	0.404	0.242
bat [17]	131	131	1038	4	0.636	0.469
citeseer [8]	3327	3327	4552	6	0.141	0.045
cora [19]	2708	2708	5278	7	0.241	0.137
dblp [30]	4057	4057	3528	4	0.177	0.026
eat [17]	399	399	5994	4	0.539	0.441
uat [17]	1190	1190	13599	4	0.501	0.332
texas [4]	183	183	162	5	0.198	0.344
wisc [4]	251	251	257	5	0.208	0.32
cornell [4]	183	183	149	5	0.167	0.326

3.4 Models

We consider a representative suite of GNNs, selected based on factors such as code availability and re-implementation time. In addition to explicit community detection algorithms, we also consider those that can learn an unsupervised representation of data as there is previous research that applies vector-based clustering algorithms to the representations [20].

Deep Attentional Embedded Graph Clustering (DAEGC) uses a k-means target to self-supervise the clustering module to iteratively refine the clustering of node embeddings [34]. Deep Modularity Networks (DMoN) uses GCNs to maximises a modularity based clustering objective to optimise cluster assignments by a spectral relaxation of the problem [32]. Neighborhood Contrast Framework for Attributed Graph Clustering (CAGC) [35] is designed for attributed graph clustering with contrastive self-expression loss that selects positive/negative pairs from the data and contrasts representations with its k-nearest neighbours. Deep Graph Infomax (DGI) maximises mutual information between patch representations of sub-graphs and the corresponding high-level summaries [33]. GRAph Contrastive rEpresentation learning (GRACE) generates a corrupted view of the graph by removing edges and learns node representations by maximising agreement across two views [41]. Contrastive Multi-View Representation Learning on Graphs (MVGRL) argues that the best employment of contrastive methods for graphs is achieved by contrasting encodings' from first-order neighbours and

a general graph diffusion [9]. Bootstrapped Graph Latents (BGRL) [31] uses a self-supervised bootstrap procedure by maintaining two graph encoders; the online one learns to predict the representations of the target encoder, which in itself is updated by an exponential moving average of the online encoder. Self-GNN [13] also uses this principal but uses augmentations of the feature space to train the network. Towards Unsupervised Deep Graph Structure Learning (SUBLIME) [18] an encoder with the bootstrapping principle applied over the feature space as well as a contrastive scheme between the nearest neighbours. Variational Graph AutoEncoder Reconstruction (VGAER) [26] reconstructs a modularity distribution using a cross entropy based decoder from the encoding of a VGAE [15].

4 Evaluation and Discussion

The Framework Comparison Rank is the average rank when comparing performance of the parameters found through hyperparameter optimisation versus the default values. From Table 3 it can be seen that Framework Comparison Rank indicates that the hyperparameters that are optimised on average perform better. This validates the hypothesis that the hyperparameter optimisation significantly impacts the evaluation of GNN based approaches to community detection. The W Randomness Coefficient quantifies the consistency of rankings over the different random seeds tested on, averaged over the suite of tests in the framework. With less deviation of prediction under the presence of randomness, an evaluation finds a more confident assessment of the best algorithm. A higher W value using the optimised hyperparameters indicates that the default parameters are marginally more consistent over randomness however this does deviate more across all tests. By using and optimising for the W Randomness Coefficient with future extensions to this framework, we can reduce the impact of biased evaluation procedures. With this coefficient, researchers can quantify how trustworthy their results are, and therefore the usability in real-world applications. It is likely that there is little difference in the coefficients in this scenario as the default parameters have been evaluated with a consistent approach to model selection and constant resource allocation to training time. This sets the baseline for consistency in evaluation procedure and allows better understanding of relative method performance.

Table 3. Here is the quantification of intra framework consistency using the W Randomness Coefficient and inter framework disparity using the Framework Comparison Rank. Low values for the Framework Comparison Rank and W Randomness Coefficient are preferred.

Results	Default	HPO
W Randomness Coefficient	0.476 ± 0.200	0.489 ± 0.144
Framework Comparison Rank	1.772 ± 0.249	1.228 ± 0.249

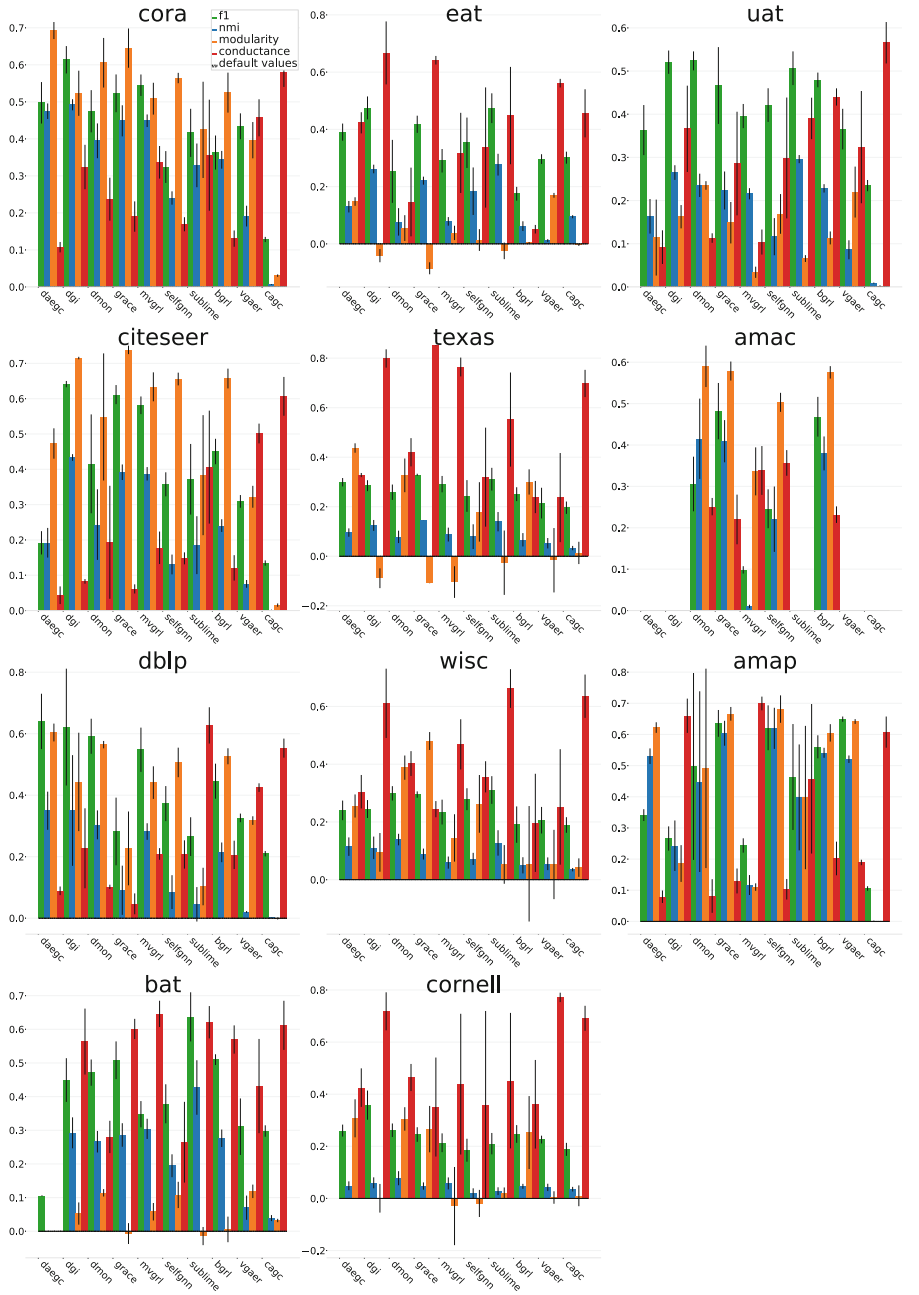


Fig. 1. The average performance and standard deviation of each metric averaged over every seed tested on for all methods on all datasets. The hyperparameter investigation under our framework is shown in colour compared with the default hyperparameters in dashed boxes. The lower the value for Conductance is better. Out Of Memory (OOM) occurrences on happened on the amac dataset with the following algorithms during HPO: daegc, sublime, dgi, cagc, vgaer and cagc under the default HPs.

The results of the hyperparameter optimisation and the default parameters is visualised in Fig. 1. From this, we can see the difference in performance under both sets of hyperparameters. On some datasets the default parameters work better than those optimised, which means that under the reasonable budget assumed in this framework they are not reproducible. This adds strength to the claim that using the default parameters is not credible. Without validation that these parameters can be recovered, results cannot be trusted and mistakes are harder to identify. On the other side of this, often the hyperparameter optimisation performs better than the default values. Algorithms were published knowing these parameters can be tuned to better performance. Using a reasonable resources, the performance can be increased, which means that without the optimisation procedure, inaccurate or misleading comparisons are propagated. Reproducible findings are the solid foundation that allows us to build on previous work and is a necessity for scientific validity.

Given different computational resources performance rankings will vary. Future iterations of the framework should experiment with the number of trials and impact of over-reliance on specific seeds or extending the hyperparameter options to a continuous distribution. Additionally, finding the best general algorithm will have to include a wide range of different topologies or sizes of graphs that are not looked at, neither do we explore other feature space landscapes or class distributions.

5 Conclusion

In this work we demonstrate flaws with how GNN based community detection methods are currently evaluated, leading to potentially misleading and confusing conclusions. To address this, an evaluation framework is detailed for evaluating GNNs at community detection that provides a more consistent and fair evaluation, and can be easily extended. We provide further insight that consistent HPO is key at this task by quantifying the difference in performance from HPO to reported values. Finally, a metric is proposed for the assessing the consistency of rankings that empirically states the trust researchers can have in the robustness of results.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2), 281–305 (2012)
2. Bergstra, J., Yamins, D., Cox, D.: Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: International Conference on Machine Learning, pp. 115–123. PMLR (2013)
3. Cabreros, I., Abbe, E., Tsirigos, A.: Detecting community structures in Hi-C genomic data. In: 2016 Annual Conference on Information Science and Systems (CISS), pp. 584–589. IEEE (2016)

4. Craven, M., McCallum, A., PiPasquo, D., Mitchell, T., Freitag, D.: Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-Mellon University Pittsburgh, PA, School of Computer Science (1998)
5. Dwivedi, V.P., Joshi, C.K., Laurent, T., Bengio, Y., Bresson, X.: Benchmarking graph neural networks. arXiv preprint arXiv:2003.00982 (2020)
6. Errica, F., Podda, M., Bacciu, D., Micheli, A.: A fair comparison of graph neural networks for graph classification. arXiv preprint arXiv:1912.09893 (2019)
7. Field, A.P.: Kendall's coefficient of concordance. Encycl. Stat. Behav. Sci. **2**, 1010–1011 (2005)
8. Giles, C.L., Bollacker, K.D., Lawrence, S.: CiteSeer: an automatic citation indexing system. In: Proceedings of the Third ACM Conference on Digital Libraries, pp. 89–98 (1998)
9. Hassani, K., Khasahmadi, A.H.: Contrastive multi-view representation learning on graphs. In: International Conference on Machine Learning, pp. 4116–4126. PMLR (2020)
10. He, R., McAuley, J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, pp. 507–517 (2016)
11. Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., Leskovec, J.: OGB-LSC: a large-scale challenge for machine learning on graphs. arXiv preprint arXiv:2103.09430 (2021)
12. Jin, D., et al.: A survey of community detection approaches: from statistical modeling to deep learning. IEEE Trans. Knowl. Data Eng. **35**(2), 1149–1170 (2021)
13. Kefato, Z.T., Girdzijauskas, S.: Self-supervised graph neural networks without explicit negative sampling. arXiv preprint arXiv:2103.14958 (2021)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
15. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
16. Liu, F., et al.: Deep learning for community detection: progress, challenges and opportunities. arXiv preprint arXiv:2005.08225 (2020)
17. Liu, Y., et al.: A survey of deep graph clustering: taxonomy, challenge, and application. arXiv preprint arXiv:2211.12875 (2022)
18. Liu, Y., Zheng, Y., Zhang, D., Chen, H., Peng, H., Pan, S.: Towards unsupervised deep graph structure learning. In: Proceedings of the ACM Web Conference 2022, pp. 1392–1403 (2022)
19. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Inf. Retrieval **3**(2), 127–163 (2000)
20. McConville, R., Santos-Rodriguez, R., Piechocki, R.J., Craddock, I.: N2D: (Not Too) deep clustering via clustering the local manifold of an autoencoded embedding. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 5145–5152. IEEE (2021)
21. Monti, F., Frasca, F., Eynard, D., Mannion, D., Bronstein, M.M.: Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673 (2019)
22. Morris, C., Kriege, N.M., Bause, F., Kersting, K., Mutzel, P., Neumann, M.: TUDataset: a collection of benchmark datasets for learning with graphs. arXiv preprint arXiv:2007.08663 (2020)

23. Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M.: Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pp. 533–541 (2020)
24. Palowitch, J., Tsitsulin, A., Mayer, B., Perozzi, B.: GraphWorld: fake graphs bring real insights for GNNs. arXiv preprint arXiv:2203.00112 (2022)
25. Pineau, J.: Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). J. Mach. Learn. Res. **22**(1), 7459–7478 (2021)
26. Qiu, C., Huang, Z., Xu, W., Li, H.: VGAER: graph neural network reconstruction based community detection. In: AAAI: DLG-AAAI 2022 (2022)
27. Salzberg, S.L.: On comparing classifiers: pitfalls to avoid and a recommended approach. Data Min. Knowl. Disc. **1**, 317–328 (1997)
28. Schaeffer, S.E.: Graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007)
29. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
30. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: ArnetMiner: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 990–998 (2008)
31. Thakoor, S., Tallec, C., Azar, M.G., Munos, R., Veličković, P., Valko, M.: Bootstrapped representation learning on graphs. In: ICLR 2021 Workshop on Geometrical and Topological Representation Learning (2021)
32. Tsitsulin, A., Palowitch, J., Perozzi, B., Müller, E.: Graph clustering with graph neural networks. arXiv preprint arXiv:2006.16904 (2020)
33. Velickovic, P., Fedus, W., Hamilton, W.L., Liò, P., Bengio, Y., Hjelm, R.D.: Deep graph infomax. ICLR (Poster) **2**(3), 4 (2019)
34. Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., Zhang, C.: Attributed graph clustering: a deep attentional embedding approach. arXiv preprint arXiv:1906.06532 (2019)
35. Wang, T., Yang, G., He, Q., Zhang, Z., Wu, J.: NCAGC: a neighborhood contrast framework for attributed graph clustering (2022). https://doi.org/10.48550/ARXIV.2206.07897, https://arxiv.org/abs/2206.07897
36. Wasserman, S., Faust, K., et al.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge (1994)
37. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)
38. Yang, J., McAuley, J., Leskovec, J.: Community detection in networks with node attributes. In: 2013 IEEE 13th International Conference on Data Mining, pp. 1151–1156. IEEE (2013)
39. Yang, L., Shami, A.: On hyperparameter optimization of machine learning algorithms: theory and practice. Neurocomputing **415**, 295–316 (2020)
40. Yuan, Y., Wang, W., Pang, W.: A systematic comparison study on hyperparameter optimisation of graph neural networks for molecular property prediction. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 386–394 (2021)
41. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Deep graph contrastive representation learning. arXiv preprint arXiv:2006.04131 (2020)
42. Zöller, M.A., Huber, M.F.: Benchmark and survey of automated machine learning frameworks. J. Artif. Intell. Res. **70**, 409–472 (2021)

Link Analysis and Ranking

Stochastic Degree Sequence Model with Edge Constraints (SDSM-EC) for Backbone Extraction

Zachary P. Neal(✉) and Jennifer Watling Neal

Michigan State University, East Lansing, MI 48824, USA
zpneal@msu.edu
https://www.zacharyneal.com

Abstract. It is common to use the projection of a bipartite network to measure a unipartite network of interest. For example, scientific collaboration networks are often measured using a co-authorship network, which is the projection of a bipartite author-paper network. Caution is required when interpreting the edge weights that appear in such projections. However, backbone models offer a solution by providing a formal statistical method for evaluating when an edge in a projection is statistically significantly strong. In this paper, we propose an extension to the existing Stochastic Degree Sequence Model (SDSM) that allows the null model to include edge constraints (EC) such as prohibited edges. We demonstrate the new SDSM-EC in toy data and empirical data on young children's' play interactions, illustrating how it correctly omits noisy edges from the backbone.

Keywords: backbone · bipartite · null model · projection · social network

1 Introduction

It is common to use the projection of a bipartite network to measure a unipartite network of interest. For example, scientific collaboration networks are often measured using a co-authorship network, which is the projection of a bipartite author-paper network [12]. Similarly, corporate networks are often measured using a board co-membership or 'interlocking directorate' network, which is the projection of a bipartite executive-board network [1]. The edges in a bipartite projection are weighted (e.g., number of co-authored papers, number of shared boards), but these weights do not provide an unbiased indicator the strength of the connection between vertices [5,9]. To overcome this bias, backbone extraction identifies the edges that are stronger than expected under a relevant null model, retaining only these edges to yield a simpler unweighted network (i.e., the backbone) that is more suitable for visualization and analysis.

Many null models exist for extracting the backbone of bipartite networks, with each model specifying different constraints on the random networks against

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 127–136, 2024.
https://doi.org/10.1007/978-3-031-53468-3_11

which an observed network is compared. However, none of the existing models permit constraints on specific edges. In this paper, we extend the fastest and most robust existing backbone model – the stochastic degree sequence model (SDSM) [11] – to accommodate one type of edge constraint: prohibited edges. Prohibited edges are edges that in principle cannot occur in the network, and can arise in many contexts. For example, in a bipartite author-paper network, an author cannot write a paper before their birth, and in a bipartite executive-board network, anti-trust laws prevent executives from serving on the boards of competitors. We illustrate the new stochastic degree sequence model with edge constraints (SDSM-EC) first in toy data, then in empirical data recording young childrens' membership in play groups.

1.1 Preliminaries

A bipartite network's vertices can be partitioned into two sets such that edges exist between, but not within, sets. In this work, we focus on a special case of a bipartite network – a two-mode network – where the two sets of vertices represent distinctly different entities that we call 'agents' and 'artifacts' (e.g. authors and papers, or executives and corporate boards).

To facilitate notation, we represent networks as matrices. First, we represent a bipartite network containing r 'agents' and c 'artifacts' as an $r \times c$ binary incidence matrix $\mathbf{B}$, where $B_{ik} = 1$ if agent i is connected to artifact k (e.g., author i wrote paper k), and otherwise is 0. The row sums $R = r_1 ... r_c$ of $\mathbf{B}$ contain the degree sequence of the agents (e.g., the number of papers written by each author), while the column sums $C = c_1 ... c_r$ of $\mathbf{B}$ contain the degree sequence of the artifacts (e.g., the number of authors on each paper). A prohibited edge in a bipartite network is represented by constraining a cell to equal zero, and therefore is sometimes called a 'structural zero' [13]. Second, we represent the projection of a bipartite network as an $r \times r$ weighted adjacency matrix $\mathbf{P} = \mathbf{B}\mathbf{B}^T$, where $\mathbf{B}^T$ represents the transpose of $\mathbf{B}$. In $\mathbf{P}$, P_{ij} equals the number of artifacts k that are adjacent to both agent i and agent j (e.g., the number of papers co-authored by authors i and j). Finally, we represent the backbone of a projection $\mathbf{P'}$ as an $r \times r$ binary adjacency matrix, where $P'_{ij} = 1$ if agent i is connected to agent j in the backbone, and otherwise is 0.

Let $\mathcal{B}$ be an ensemble of $r \times c$ binary incidence matrices, which can be constrained to have certain features present in $\mathbf{B}$. Let P^*_{ij} be a random variable equal to $(\mathbf{B}^*\mathbf{B}^{*T})_{ij}$ for $\mathbf{B}^* \in \mathcal{B}$. Decisions about which edges appear in a backbone extracted at the statistical significance level α are made by comparing P_{ij} to P^*_{ij}:

$$P'_{ij} = \begin{cases} 1 & \text{if } \Pr(P^*_{ij} \geq P_{ij}) < \frac{\alpha}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

This test includes edge P'_{ij} in the backbone if its weight in the observed projection P_{ij} is uncommonly large compared to its weight in projections of members of the ensemble P^*_{ij}.

2 Backbone Models

2.1 The Stochastic Degree Sequence Model (SDSM)

Models for extracting the backbone of bipartite projections differ in the constraints they impose on $\mathcal{B}$. The most stringent model – the Fixed Degree Sequence Model (FDSM) [17] – relies on a microcanonical ensemble that constrains each member of $\mathcal{B}$ to have exactly the same row and column sums as $\mathcal{B}$. Computing P^*_{ij} under the FDSM is slow because it requires approximation via computationally intensive Monte Carlo simulation. Despite recent advances in the efficiency of these simulations [2], it is often more practical to use the less stringent Stochastic Degree Sequence Model (SDSM) [9]. The SDSM relies on a canonical ensemble that constrains each member of $\mathcal{B}$ to have the same row and column sums as $\mathbf{B}$ *on average*. SDSM is fast and exact, and comparisons with FDSM reveal that it yields similar backbones [11].

Under the SDSM, P^*_{ij} follows a Poisson-binomial distribution whose parameters can be computed from the entries of probability matrix $\mathbf{Q}$, where $Q_{ik} = \Pr(B^*_{ik} = 1)$ for $\mathbf{B}^* \in$ a microcanonical $\mathcal{B}$. That is, Q_{ik} is the probability that B^*_{ik} contains a 1 in the space of all matrices with given row and column sums. Most implementations of SDSM approximate $\mathbf{Q}$ using the fast and precise Bipartite Configuration Model (BiCM) [14,15]. However, it can also be computed with minimal loss of speed and precision [11] using a logistic regression [9], which offers more flexibility. This method estimates the β coefficients in

$$B_{ik} = \beta_0 + \beta_1 r_i + \beta_2 c_k + \epsilon$$

using maximum likelihood, then defines Q_{ik} as the predicted probability that $B_{ik} = 1$.

2.2 The Stochastic Degree Sequence Model with Edge Constraints (SDSM-EC)

The constraints that SDSM imposes on $\mathcal{B}$ are determined by the way that $\mathbf{Q}$ is defined. In the conventional SDSM, $\mathbf{Q}$ is defined such that Q_{ik} is the probability that B^*_{ik} contains a 1 in the space of all matrices with given row and column sums, which only imposes constraints on the row and column sums of members of $\mathcal{B}$. To accommodate edge constraints, we define $\mathbf{Q}'$ such that Q'_{ik} is the probability that B^*_{ik} contains a 1 in the space of all matrices with given row and column sums *and no 1s in prohibited cells*.

The BiCM method cannot be used to approximate $\mathbf{Q}'$, however the logistic regression method can be adapted to approximate it. If B_{ik} is a prohibited edge, then $Q_{ik} = 0$ by definition. If B_{ik} is not a prohibited edge, then Q_{ik} is the predicted probability that $B_{ik} = 1$ based on a fitted logistic regression. Importantly, however, whereas the logistic regression used to estimate $\mathbf{Q}$ is fitted over all B_{ik}, the logistic regression used to estimate $\mathbf{Q}'$ is fitted only over B_{ik} that are not prohibited edges.

3 Results

3.1 Estimating Q′

In general the true values of Q_{ik} are unknown. However, for small matrices they can be computed from a complete enumeration of the space. To evaluate the precision of Q_{ik} estimated using the SDSM-EC method described above, we first enumerated all 4×4 incidence matrices with row sums {1, 1, 2, 2} and column sums {1, 1, 2, 2}; there are 211. Next, we constrained this space to matrices in which a randomly selected one or two cells always contain a zero (i.e. bipartite networks with one or two prohibited edges). Finally, we computed the true value of each Q_{ik} for all cells and all spaces, estimated each Q_{ik} using the logistic regression method, and computed the absolute deviation between the two.

Figure 1A illustrates that, compared to the cardinality of the unconstrained space ($|\mathcal{B}| = 211$), the cardinalities of the spaces constrained by one or two prohibited edges are much lower ($|\mathcal{B}| = 2 - 29$, gray bars). That is, while the SDSM evaluates whether a given edge's weight is significant by comparing its value to a large number of possible worlds, the SDSM-EC compares its value to a much smaller number of possible worlds. Figure 1B illustrates the deviations between the true value of Q_{ik} and the value estimated using the logistic regression method. It demonstrates that although SDSM-EC requires approximating Q_{ik}, these approximations tend to be close to the true values.

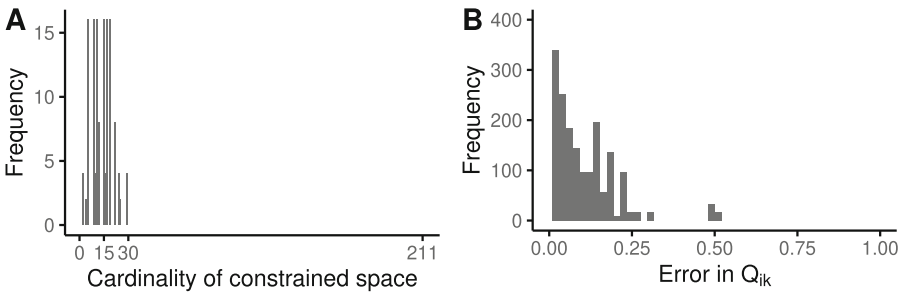


Fig. 1. (A) The cardinality of the space of matrices with row sums {1, 1, 2, 2} and column sums {1, 1, 2, 2} and one or two cells constrained to zero is small compared to the cardinality of the space without constrained cells. (B) The deviation between the true and estimated Q_{ik} for all such constrained spaces tends to be small.

3.2 Toy Illustration

We offer a toy example to illustrate the impact of imposing edge constraints in backbone extraction. Figure 2A illustrates a bipartite network that contains two types of agents (open and filled circles) and two types of artifacts (open and

filled squares), such that agents are only connected to artifacts of the same type. Such a network might arise in the context of university students joining clubs. For example, suppose Harvard students (open circles) only join Harvard clubs (open squares), while Yale students (filled circles) only join Yale clubs (filled squares).

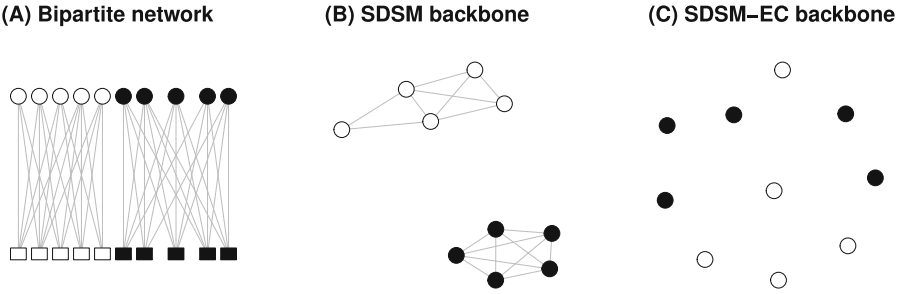


Fig. 2. (A) A bipartite network containing two groups of agents and two groups of artifacts, such that agents are connected only to their own group's artifacts. (B) The SDSM backbone of a projection of this bipartite graph, which assumes that an agent *could* be connected to another group's artifact, suggests within-group cohesion among agents. (C) The SDSM-EC projection, which assumes that an agent *could not* be connected to another group's artifact, suggests none of the edges in the projection are significant.

Figure 2B illustrates the backbone extracted from a projection of this bipartite network using the SDSM. Using the SDSM implies that there are no constraints on edges in the null model. In the context of student clubs, this means that in the null model it is possible for a Harvard student to join a Yale club, and vice versa, and that the pattern of segregation that appears in the bipartite network is chosen (i.e. homophily). The SDSM backbone displays a high level of within-group cohesion (i.e. homophily). This occurs for two reasons. First, agents from the same group share many artifacts (e.g., two Harvard students belong to many of the same clubs). Second, if agents were connected to artifacts randomly (e.g., Harvard students joined both Harvard and Yale clubs), as the SDSM null model assumes, then agents from the same group would have shared fewer artifacts. The presence of within-group connections in the SDSM backbone reflects the fact that it is noteworthy that pairs of Harvard students, or pairs of Yale students, are members of many of the same clubs because they could have chosen otherwise.

Figure 2C illustrates the backbone extracted using the SDSM-EC, where we specify that edges are prohibited between an agent and artifact of a different type. In the context of student clubs, this means that in the null model it is *not* possible for a Harvard student to join a Yale club, and vice versa, and that the pattern of segregation is enforced by university regulations. The SDSM-EC backbone is empty. This occurs because although agents from the same

group share many artifacts, they also share many artifacts under the null model. The absence of connections in the SDSM-EC backbone reflects the fact that it is uninteresting that pairs of Harvard students, or pairs of Yale students, are members of many of the same clubs because they could not have chosen otherwise.

3.3 Empirical Illustration

We offer an empirical example of the application of SDSM-EC to illustrate its practicality and impact. It can be difficult to directly measure social networks among very young children. One alternative is to infer these networks from observations of their play groups using bipartite backbones [8]. However, considering edge constraints can be important because the organization of the school can mean that it may be impossible to observe certain children playing together.

These data were collected in Spring 2013 by observing the behaviors of 53 children in a preschool in the Midwestern United States [3,6–8]. A scan observation method was employed whereby a randomly selected child was observed for a period of 10 s. After the 10 s period had elapsed, the trained observer coded the child's predominant behavior and, if applicable, the peers with whom they were interacting [4]. Here, we focus only on social play behaviors because they were the most common form of social behavior, and the most likely to involve direct interaction with peers. A total of 1829 social play events were observed during data collection. These data are organized as a bipartite network $\mathbf{B}$ where $B_{ik} = 1$ if child i was observed participating in a play group during observation k. A projection of $\mathbf{P} = \mathbf{B}\mathbf{B}^T$, where P_{ij} indicates the number of times children i and j were observed playing together provides an indirect indicator of the children's' social network, particularly when refined using backbone extraction [8].

In this context, two types of prohibited edges exist in the bipartite network. First, the school was organized into two age-based classrooms, a classroom of 3-year-olds and a classroom of 4-year-olds. Because these classrooms used different spaces, it was not possible to observe a 3-year old and a 4-year-old together. Therefore, edges from 3-year-olds to observations of 4-year-olds are prohibited, and likewise edges from 4-year-olds to observations of 3-year-olds are prohibited. Second, the children varied in their attendance status: some attended for the full day, some attended only in the morning, and some attended only in the afternoon. Because attendance status determines which children were present and able to play together, it was not possible to observe an AM child and a PM child together. Therefore, edges from AM children to observations conducted in the afternoon are prohibited, and likewise edges from PM children to observations conducted in the morning are prohibited.

Figure 3 illustrates two backbones extracted from these data, using shape to represent classroom (circles = 3-year-olds, squares = 4-year-olds) and color to represent attendance status (black = full day, gray = AM only, white = PM only). Figure 3A was extracted using the SDSM and therefore does not consider these edge constraints, while Fig. 3B was extracted using the SDSM-EC and

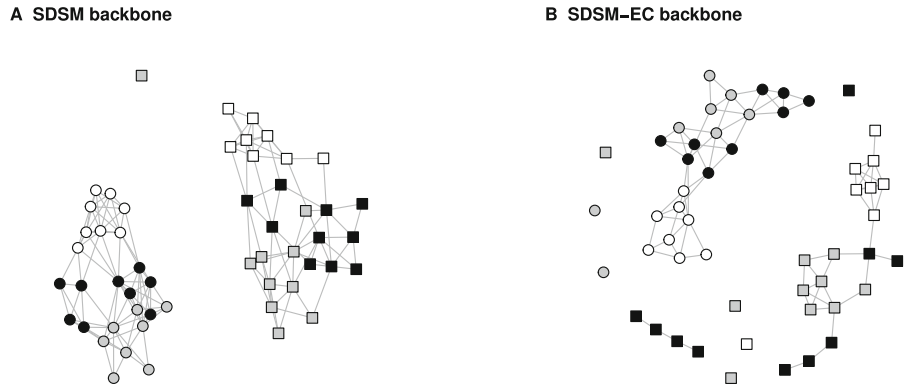


Fig. 3. (A) Backbone extracted using SDSM and (B) SDSM-EC from 1829 observations of 53 preschool childrens' play groups. Vertex shape represents age-based classrooms: circles = 3 year old classroom, squares = 4 year old classroom. Vertex color represents attendance status: black = full day, gray = AM only, white = PM only.

does consider these edge constraints. There are some similarities between the SDSM and SDSM-EC backbones that reflect characteristics of the setting: 3-year-olds (circles) are never connected to 4-year-olds (squares), and AM children (gray) are never connected to PM children (white), because it was not possible to observe such children together. However, there are also differences that highlight the impact of incorporating edge constraints using SDSM-EC. The SDSM-EC backbone contains many fewer edges ($E = 85$) than the SDSM backbone ($E = 153$). This occurs for similar reasons to the loss of edges in the toy example above, although is less extreme.

A hypothetical example serves to illustrate why the SDSM-EC backbone contains fewer edges in this context. Consider the case of an AM child and a Full Day child in the 3-year-old classroom who were observed to play together a few times. The SDSM compares this observed co-occurrence to the expected number of co-occurrences if these two children had played with other AM or Full Day children and with others in the 3-year-old classroom (which is possible), but also if they had played with PM children and children in the 4-year-old classroom (which is not possible). Under such a broad null model that includes some impossible play configurations, observing these two children playing together even just a few times seems noteworthy, and therefore an edge between them is included in the backbone. In contrast, the SDSM-EC compares this observed co-occurrence to the expected number of co-occurrences if these two children had played with other AM or Full Day children and with others in the 3-year-old classroom only, recognizing that it was not possible for the AM child to play with PM children or for either to play with children in the 4-year-old classroom. Under this more constrained null model that excludes impossible play configurations, observing these two children playing together just a few times is not particularly noteworthy, and therefore an edge between them is omitted from the backbone.

As this example illustrates, the SDSM-EC contains fewer edges because it correctly omits edges that might seem significantly strong when evaluated against a null model that includes impossible configuration, but that are not significantly strong when evaluated against a properly constrained null model that excludes impossible configurations.

4 Conclusion

Although bipartite projections offer a promising way to indirectly measure unipartite networks of interest, caution is required when interpreting the edge weights that appear in such projections. Backbone models offer a solution by providing a formal statistical method for evaluating when an edge in a projection is statistically significantly strong by comparison to a bipartite null model. However, extracting an accurate backbone using these methods requires that the null model is properly constrained. In many cases the FDSM (slower) and SDSM (faster) are appropriate and yield similar results [11], however these null models only constrain the degree sequences, but cannot impose edge constraints such as prohibited edges.

In this work, we have introduced the SDSM-EC, an extension of SDSM that allows the user to specify edge constraints in the form of prohibited edges. Prohibited edges arise in bipartite networks when a given agent cannot be connected to a given artifact, for example, because the agent is not present or because such a connection is legally prohibited. We have demonstrated in both a toy example and an empirical example that the SDSM-EC performs as expected, correctly omitting weaker edges in the backbone that are not significant when these constraints are considered, but that might have erroneously appeared significant under the SDSM. Therefore, we recommend that SDSM-EC be used to extract the backbones of bipartite projections when the bipartite network contains prohibited edges. The SDSM-EC is implemented in the `sdsm()` function of the `backbone` package for R [10].

We have focused on one common type of edge constraint: prohibited edges. However, a second type of edge constraint is also possible: required edges. Required edges arise in bipartite networks when a given agent must always be connected to a given artifact, for example, because the agent is the initiator of the artifact (e.g. a paper's lead author, a club's founder). It is trivial to extend the SDSM-EC to also accommodate such constraints. When $\mathbf{Q}$ is estimated, $Q_{ik} = 0$ for prohibited edges and $Q_{ik} = 1$ for required edges, then the remaining Q_{ik} values are computed using the same logistic regression method described above.

This work highlights the importance of using a properly constrained null model when extracting the backbone of bipartite projections, and identifies several avenues for future research. First, while $\mathbf{Q}$ under the SDSM can be estimated quickly and precisely using the BiCM [14,15], $\mathbf{Q}$ under the SDSM-EC must be estimated using logistic regression, which is slower and less precise [11]. Future work should investigate improved methods for estimating $\mathbf{Q}$, which has

the potential to benefit not only the SDSM-EC, but all variants of the SDSM. Second, while a broad class of bipartite null models exist [16] and now include edge constraints, future work should investigate the importance and feasibility of incorporating other types of constraints.

Acknowledgements. We thank Emily Durbin for her assistance collecting the empirical data.

Data Availability Statement. The data and code necessary to reproduce the results reported above are available at https://osf.io/7z4gu.

References

1. Burris, V.: Interlocking directorates and political cohesion among corporate elites. Am. J. Sociol. **111**(1), 249–283 (2005). https://doi.org/10.1086/428817
2. Godard, K., Neal, Z.P.: fastball: a fast algorithm to randomly sample bipartite graphs with fixed degree sequences. J. Complex Netw. **10**(6), cnac049 (2022). https://doi.org/10.1093/comnet/cnac049
3. Gornik, A.E., Neal, J.W., Lo, S.L., Durbin, C.E.: Connections between preschoolers' temperament traits and social behaviors as observed in a preschool setting. Soc. Dev. **27**(2), 335–350 (2018). https://doi.org/10.1111/sode.12271
4. Hanish, L.D., Martin, C.L., Fabes, R.A., Leonard, S., Herzog, M.: Exposure to externalizing peers in early childhood: homophily and peer contagion processes. J. Abnorm. Child Psychol. **33**, 267–281 (2005). https://doi.org/10.1007/s10802-005-3564-6
5. Latapy, M., Magnien, C., Del Vecchio, N.: Basic notions for the analysis of large two-mode networks. Soc. Netw. **30**(1), 31–48 (2008). https://doi.org/10.1016/j.socnet.2007.04.006
6. Neal, J.W., Brutzman, B., Durbin, C.E.: The role of full-and half-day preschool attendance in the formation of children's social networks. Early Childhood Res. Q. **60**, 394–402 (2022). https://doi.org/10.1016/j.ecresq.2022.04.003
7. Neal, J.W., Durbin, C.E., Gornik, A.E., Lo, S.L.: Codevelopment of preschoolers' temperament traits and social play networks over an entire school year. J. Pers. Soc. Psychol. **113**(4), 627 (2017). https://doi.org/10.1037/pspp0000135
8. Neal, J.W., Neal, Z.P., Durbin, C.E.: Inferring signed networks from preschoolers' observed parallel and social play. Soc. Netw. **71**, 80–86 (2022). https://doi.org/10.1016/j.socnet.2022.07.002
9. Neal, Z.P.: The backbone of bipartite projections: inferring relationships from co-authorship, co-sponsorship, co-attendance and other co-behaviors. Soc. Netw. **39**, 84–97 (2014). https://doi.org/10.1016/j.socnet.2014.06.001
10. Neal, Z.P.: backbone: an R package to extract network backbones. PLOS ONE **17**(5), e0269,137 (2022). https://doi.org/10.1371/journal.pone.0269137
11. Neal, Z.P., Domagalski, R., Sagan, B.: Comparing alternatives to the fixed degree sequence model for extracting the backbone of bipartite projections. Sci. Rep. **11**(1), 1–13 (2021). https://doi.org/10.1038/s41598-021-03238-3
12. Newman, M.E.: Scientific collaboration networks. I. Network construction and fundamental results. Phys. Rev. E **64**(1), 016,131 (2001). https://doi.org/10.1103/PhysRevE.64.016131

13. Ripley, R.M., Snijders, T.A.B., Boda, Z., Voros, A., Preciado, P.: Manual for Siena version 4.0. Technical report. Department of Statistics, Nuffield College, University of Oxford, Oxford (2023). R package version 1.3.14.4. https://www.cran.r-project.org/web/packages/RSiena/
14. Saracco, F., Di Clemente, R., Gabrielli, A., Squartini, T.: Randomizing bipartite networks: the case of the world trade web. Sci. Rep. **5**(1), 1–18 (2015). https://doi.org/10.1038/srep10595
15. Saracco, F., Straka, M.J., Di Clemente, R., Gabrielli, A., Caldarelli, G., Squartini, T.: Inferring monopartite projections of bipartite networks: an entropy-based approach. New J. Phys. **19**(5), 053,022 (2017). https://doi.org/10.1088/1367-2630/aa6b38
16. Strona, G., Ulrich, W., Gotelli, N.J.: Bi-dimensional null model analysis of presence-absence binary matrices. Ecology **99**(1), 103–115 (2018). https://doi.org/10.1002/ecy.2043
17. Zweig, K.A., Kaufmann, M.: A systematic approach to the one-mode projection of bipartite graphs. Soc. Netw. Anal. Min. **1**(3), 187–218 (2011). https://doi.org/10.1007/s13278-011-0021-0

Minority Representation and Relative Ranking in Sampling Attributed Networks

Nelson Antunes[1(✉)], Sayan Banerjee[2], Shankar Bhamidi[2], and Vladas Pipiras[2]

[1] Center for Computational and Stochastic Mathematics, University of Lisbon, Avenida Rovisco Pais, 1049-001 Lisbon, Portugal
nantunes@ualg.pt

[2] Department of Statistics and Operations Research, University of North Carolina, CB 3260, Chapel Hill, NC 27599, USA
{sayan,bhamidi,pipiras}@email.unc.edu

Abstract. We explore two questions related to sampling and minorities in attributed networks with homophily. The first question is to investigate sampling schemes which favor minority attribute nodes and which give preference to "more popular" nodes having higher centrality measures in the network. A data study shows the efficiency of Page-rank and walk-based network sampling schemes on a directed network model and a real-world network with small minorities. The second question concerns the effect of homophily and out-degrees of nodes on the relative ranking of minorities compared to majorities in degree-based sampling. Several synthetic network configurations are considered and the conditions for minority nodes to have a higher relative rank are investigated numerically. The results are also assessed with real-world networks.

Keywords: Random networks · attributes · homophily · sampling · minorities · ranking

1 Introduction

An attributed network can be defined as a graph in which nodes (and/or edges) have features. In a social network, node attributes can refer to gender, age, ethnicity, political ideologies. The attributes of nodes often co-vary and affect the graph structure. One standard phenomenon in many real-world systems is homophily [6], i.e., node pairs with similar attributes being more likely to be connected than node pairs with discordant attributes. For instance, many social networks show this property, which is the tendency of individuals to associate with others who are similar to them; e.g., with respect to an attribute. Additionally, the distribution of user attributes over the network is usually uneven, with coexisting groups of different sizes, e.g., one ethnic group (majority) may dominate other (minority).

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 137–149, 2024.
https://doi.org/10.1007/978-3-031-53468-3_12

Given that most real networks can only be observed indirectly, network sampling, and its impact on the representation/learning of the true network, is an activate area of research across multiple communities (see e.g. [2,3] and the references therein). In this context, there has been significant interest in attributed network sampling where there is a particular *small minority* of certain attribute nodes. Here, we explore two related questions in this area, namely, (a) settings where Page-rank and other exploration based sampling schemes favor sampling small minorities, (b) effects of homophily and out-degrees on the relative ranking of minorities compared to majorities in degree-based sampling. To this end, we shall use an attributed network model that incorporates homophily [1]. We employ the asymptotic theory developed in [1] to gain insight through data studies of the various network sampling schemes and attribute representation in concrete applications. The findings will also be assessed with real-world networks. More concretely, we investigate the following research problems:

(a) We consider the case where there is a particular small minority which has higher propensity to connect within itself as opposed to majority nodes; for substantial recent applications and impact of such questions, see [10,11,13]. In such setting, devising schemes where one gets a non-trivial representation of minorities is challenging if the sample size is much smaller than the network size. In this case, uniform sampling will clearly not be *fair* as the sampled nodes will tend to be more often from the majority attribute. Additionally, uniform sampling does not give preference to "more popular" minority nodes, i.e., higher degree/Page-rank nodes. Therefore, it is desirable to *explore* the network locally around the initial (uniformly sampled) random node and try to travel towards the "centre", thereby traversing edges along their natural direction. However, to avoid high sampling costs, the explored set of nodes should not be too large. We compare through a data study several sampling schemes derived from centrality measures like degree and Page-rank and show that they increase the probability of sampling a minority node and its "popularity". This is investigated in several network model configurations and in a real network dataset.

(b) We consider two degree-based sampling schemes and explore the effects of homophily and out-degrees of the model parameters on the relative ranking of minority compared to majority (in terms of proportion) in the samples. As in (a), we again study minority representation, but focus on degree-based sampling and are interested in dependence on structural network properties. The conditions in a asymptotic regime (when the number of nodes goes to infinity) are known for the minority nodes to rank higher (i.e. have larger proportions) than the majority nodes (based on the tail distribution and sum of the degrees) [1]. For three scenarios - heterophily, homogeneous homophily (homogenous mixing) and asymmetric homophily - the results are numerically investigated for the minority nodes to rank higher. The last two scenarios were briefly considered heuristically in [7,9] using fluid limits. We show that the results for two real networks with degree power-law distributions agree with those for the synthetic model.

The paper is organized as follows. A synthetic model with homophily is given in Sec. 2. Network sampling in the presence of a small minority is studied in Sec.

3. Relative ranking of minorities is investigated in Sec. 4. Sec. 5 concludes and indicates future work.

2 Network Model with Attributes and Homophily

Fix an attribute space $S = \{1, 2\}$. The nodes with attribute 1 will be referred as *minority* and attribute 2 as *majority*. While this paper only deals with these two types, the setting below can be extended to more general attribute spaces. Fix a probability mass function (π_1, π_2) on S and a possible asymmetric function $\kappa : S \times S \to \mathbb{R}$; this function measures propensities of pairs of nodes to connect, based on their attributes. Fix a preferential attachment parameter $\alpha \in [0, 1]$ and an out-degree function $m : S \to \mathbb{N}$ which modulates the number of edges that a node entering the system connects to, depending on its attribute type.

Nodes enter the system sequentially at discrete times $n \geq 1$ starting with a base connected graph G_0 with n_0 nodes at time $n = 0$ where every node has an attribute in S. Write v_n for the node that enters at time n and $a(v_n)$ for the corresponding attribute; every node v_n has attribute 1 with probability π_1 and attribute 2 with probability π_2. The dynamics of construction are recursively defined as: for $n \geq 0$ and $v \in G_n$, v_{n+1} attaches to the network via $m(a(v_{n+1})) = m_{a(v_{n+1})}$ outgoing edges. Each edge independently chooses an existing node in G_n to attach to, with probabilities (conditionally on G_n and $a(v_{n+1})$) given by

$$\mathbb{P}\left(v_{n+1} \rightsquigarrow v \,|\, G_n, a(v_{n+1}) = a^\star\right) := \frac{\kappa(a(v), a^\star)[\deg(v, n)]^\alpha}{\sum_{v' \in G_n} \kappa(a(v'), a)[\deg(v', n)]^\alpha}, \quad (1)$$

where $\deg(v, n)$ denote the degree of v at time n (if $G_0 = v_0$, initialize $\deg(v_0, 0) = 1$). A *tree* network is obtained if $m_1 = m_2 = 1$. The case $\kappa(.,.) = 1$ and $\alpha = 1$ corresponds to the well known *linear* preferential attachment model while $0 < \alpha < 1$ to the *sublinear* case. When referring to a synthetic network below, we shall always mean the model (1).

In measuring homophily, we extend the definition given for signed networks [12] to directed networks. Let V (resp. E) denote the set of nodes (resp. edges) of a network; for $a = 1, 2$, let V_a be the set of nodes with attribute a, and for $a \neq a'$, let $E_{aa'}$ be the set of edges between nodes of types a and a'. Let $p = |E|/(|V|(|V| - 1))$ be the edge density. For $a = 1, 2$, dyadicity $D_a = |E_{aa}|/(|V_a|(|V_a| - 1)p)$ measures the contrast in edges within the cluster of nodes a as compared to a setting where all edges are randomly distributed; thus $D_a > 1$ signals homophilic characteristics of type a nodes. Similarly, for $a \neq a'$, heterophilicity $H_{aa'} = |E_{aa'}|/(2|V_a||V_{a'}|p)$ denotes propensity of type a nodes to connect to type a' nodes as contrasted with random placement of edges with probability equal to the global edge density. If $H_{aa'} < 1$, nodes of type a do not tend to be connected to nodes of type a'.

3 Network Sampling and Minority Representation

In this section, we compare attribute representation of minorities under sampling schemes on synthetic and empirical networks.

3.1 Sampling Methods

We consider several sampling schemes derived from various centrality measures such as (in-)degree and Page-rank. All the methods have in common the following exploration idea of the network. A node is picked uniformly at random followed by a walk on the network with a fixed or random number of steps. The sampled node is the last node visited by the walk.

Uniform sampling (U). We choose a node at random and the number of walk steps is zero. This is equivalent to the classic uniform sampling method.

Sampling proportional to degree (D). We pick a node uniformly at random, and one of its neighbors is chosen at random (one walk step).

Sampling proportional to in-degree (ID). For directed network, a node is selected at random and one step is taken through an out-going edge chosen at random. If the out-degree of the node is zero, the selected node is sampled.

Sampling proportional to Page-rank (PR_c). After choosing an initial node at random, the number of steps to traverse the directed network is a geometric random variable (starting at zero) with parameter $(1-c)$. If the walk gets stuck in a node before the number of steps is reached, it returns this node as the sampled node. The equivalence of this algorithm and sampling proportional to Page-rank with damping factor c in the context of tree network models follows from [5].

Fixed length walk sampling (FL_M). We pick a node uniformly at random and walk a fixed number M of steps through the out-going edges chosen at random of the visited nodes. The same rule above applies if a node with zero out-degree is reached.

3.2 Asymptotic Analysis: Sampling in Tree Networks

We consider an asymmetric homophily scenario. Majority nodes (type 2) have equal propensity to connect to minority (type 1) or majority nodes. Minorities have relatively higher propensity to connect to other minority nodes compared to majority nodes.

Let $\kappa_{11} = \kappa_{22} = \kappa_{12} = 1$, $\kappa_{21} = a$ and $\pi_1 = \theta/(1+\theta)$. We analyze the sampling schemes when a and θ go to zero in a dependent way by letting $\theta = D\sqrt{a}$, where D is a positive constant. Let v be a node sampled from the network G_n and $a(v)$ its attribute, under one of the sampling methods above. From the analysis of the linear, tree model [1], as $a \to 0^+$, we have that $\mathbb{P}(a(v) = 1|G_n)$ behaves as $D\sqrt{a} + O(a)$ under uniform sampling; $2D\sqrt{a} - (4D^2 + \frac{1}{2})a + O(a^{3/2})$ under sampling proportional to degree; $3D\sqrt{a} + O(a)$ under sampling proportional to in-degree; and as $c \to 1^-$ and $n \to \infty$, $(2D^2 - \frac{1}{2} + \Delta)/(2D^2 + \frac{1}{2} + \Delta)$, where $\Delta = \sqrt{(2D^2 - 1/2)^2 + 4D^2}$ under sampling proportional to Page-rank and fixed length walk sampling. The next sections investigate how these results hold in a non-asymptotic regime in (sub-)linear, (non-)tree networks, as well as in a real network.

Table 1. Synthetic networks: structural properties.

	$\lvert V\rvert$	$\lvert E\rvert$	D_1	D_2	H_{12}	H_{21}	$\frac{\lvert E_{11}\rvert}{\lvert E\rvert}$	$\frac{\lvert E_{22}\rvert}{\lvert E\rvert}$	$\frac{\lvert E_{12}\rvert}{\lvert E\rvert}$	$\frac{\lvert E_{21}\rvert}{\lvert E\rvert}$	$\frac{\lvert V_1\rvert}{\lvert V\rvert}$	$\frac{\lvert V_2\rvert}{\lvert V\rvert}$
Syn. 1	10^5	99999	18.74	0.961	0.015	0.861	0.050	0.864	0.002	0.085	0.052	0.948
Syn. 2	10^5	99999	7.155	0.988	0.068	0.541	0.108	0.760	0.015	0.117	0.123	0.877
Syn. 3	25,000	46907	3.722	1.078	0.042	0.488	0.057	0.828	0.009	0.106	0.124	0.876

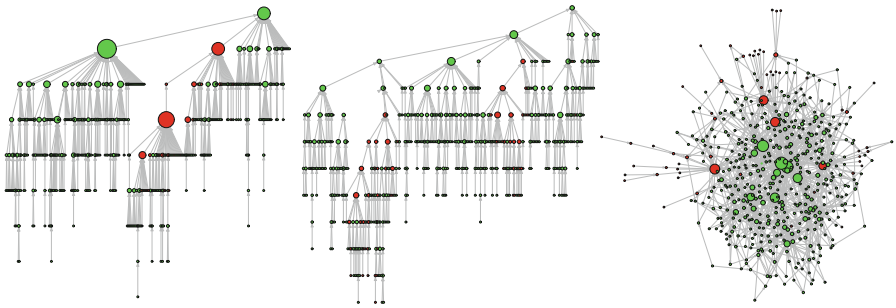


Fig. 1. Synthetic networks with 500 nodes: (l.h.s.) linear, tree network ($a = 0.003$, $D = 1$), (m.h.s.) sub-linear, tree network ($\alpha = 0.25$, $a = 0.02$, $D = 1$), (r.h.s.) linear, non-tree network ($m_1 = 1, m_2 = 2$, $a = 0.02$, $D = 1$). The red (green) circles represent the minority (majority) nodes with sizes proportional to the degrees.

3.3 Synthetic Networks

We generate a linear ($\alpha = 1$), tree network with $|V| = 10^5$ nodes, $a = 0.003$ ($D = 1$) where the probability that a node entering the network has attribute 1 (minority) is very small, $\pi_1 \approx 0.052$. The homophily and structural characteristics of the network are given in Table 1 (Syn. 1). Note that D_1 is large while D_2 is close to 1, and H_{12} is smaller than H_{21} (< 1) corresponding to an asymmetric homophily. A picture of a small network generated in this setting is shown in Fig. 1 (l.h.s.). In the linear case, there are minority nodes with a large degree. For each sampling method, we estimate the probability of sampling a minority node through the proportion of minority nodes sampled over 10^4 runs. Additionally, we also compute the average of the degree-ranks and Page-ranks of the minority sampled nodes. The results are given in Table 2. Rank is expressed as percent in Table 2, where higher rank corresponds to smaller top percent. The probability of sampling a minority node under uniform sampling is close to the asymptotic value $\sqrt{a} \approx 0.055$ and does not give preference to "more popular" nodes (with higher degree or Page-rank). Sampling proportional to degree approximately doubles the chance to pick a minority node approaching $2\sqrt{a} - \frac{9}{2}a \approx 0.096$ and leads to a higher rank. The results improve with sampling proportional to in-degree which agrees with the asymptotic analysis. For sampling proportion to Page-rank (PR_c) with $c = m/(m+1)$, $m \in \mathbb{N}$, the mean number of walk steps is m. The number of steps being random does not improve the results. If the value of c is close to 0, PR_c is akin to uniform sampling. On the other hand, when c is large, the walk can hit the root. This can be explained by the diameter

Table 2. Linear, tree network (Syn. 1): minority nodes representation

Sampling	U	D	ID	$PR_{1/2}$	$PR_{2/3}$	$PR_{3/4}$	$PR_{4/5}$	FL_2	FL_3	FL_4
prob.	0.052	0.110	0.133	0.077	0.090	0.093	0.089	0.189	0.191	0.150
degree-rank (%)	46.147	6.883	3.628	23.702	16.220	12.199	10.805	1.032	0.330	0.155
Page-rank (%)	46.215	7.323	3.912	23.759	16.199	12.195	10.781	0.825	0.212	0.080

Table 3. Sub-linear, tree network (Syn. 2): minority nodes representation

Sampling	U	D	ID	$PR_{2/3}$	$PR_{3/4}$	$PR_{4/5}$	$PR_{5/6}$	FL_4	FL_5	FL_6
prob.	0.125	0.176	0.226	0.199	0.220	0.226	0.223	0.387	0.401	0.381
degree-rank (%)	43.345	17.736	9.848	17.515	13.053	10.737	9.586	1.059	0.529	0.395
Page-rank (%)	43.037	18.954	10.417	17.284	12.783	10.553	9.358	0.617	0.384	0.143

Table 4. Linear, non-tree network (Syn. 3): minority nodes representation

Sampling	U	D	ID	$PR_{1/2}$	$PR_{2/3}$	$PR_{3/4}$	FL_2	FL_3	FL_4
prob.	0.1212	0.164	0.207	0.142	0.146	0.139	0.234	0.211	0.158
degree-rank (%)	69.045	17.184	9.443	46.636	35.758	29.978	3.211	1.283	0.609
Page-rank (%)	49.437	11.626	5.846	33.362	25.660	21.028	1.536	0.527	0.233

of the network which is 18 (in the tree case, it is $O(\log |V|)$). These drawbacks explain partly the good performance of fixed length walk sampling which also has the higher rank of the minority sampled nodes. This sampling scheme gives preference to nodes with a higher Page-rank as well.

We next consider the sub-linear, tree network with $\alpha = 0.25$ and $a = 0.02$ ($D = 1$) which gives $\pi_1 \approx 0.124$. The characteristics of the generated network are given in Table 1 (Syn. 2). An illustration of a small network with these characteristics is shown in Fig. 1 (m.h.s.). We estimate the probability of sampling a minority and its importance for each sampling scheme using 10^4 runs – see Table 3. The qualitative comparison of the performance of the sampling schemes is the same as in the linear case. However, the number of steps for sampling proportional to Page-rank and fixed length walk sampling is larger. The diameter of the generated network is 25.

Finally, we consider a linear, non-tree network with $m_1 = 1$ and $m_2 = 2$ and $a = 0.02$ ($D = 1$). The number of nodes is 25,000 which resulted in a network diameter of 16. The network properties are shown in Table 2 (Syn. 3) – see also Fig. 1 (r.h.s.) for a network generated with a smaller number of nodes. As seen from the results (averaged over 10^4 runs) in Table 4, the probability of sampling a minority node with fixed length walk sampling decreases compared to the sub-linear case due to the non-tree network structure (however, it is still approximately the double compared to uniform sampling).

3.4 Real Network

We inspect a social real-world network with a weak asymmetric homophily scenario to assess the probability of sampling a minority node. Hate is a retweet

Table 5. Empirical network: characteristics

	$\lvert V\rvert$	$\lvert E\rvert$	D_1	D_2	H_{12}	H_{21}	$\frac{\lvert E_{11}\rvert}{\lvert E\rvert}$	$\frac{\lvert E_{22}\rvert}{\lvert E\rvert}$	$\frac{\lvert E_{12}\rvert}{\lvert E\rvert}$	$\frac{\lvert E_{21}\rvert}{\lvert E\rvert}$	$\frac{\lvert V_1\rvert}{\lvert V\rvert}$	$\frac{\lvert V_2\rvert}{\lvert V\rvert}$
Hate	2700	9709	9.976	0.579	0.408	0.529	0.333	0.386	0.122	0.158	0.183	0.817

Table 6. Hate network: minority nodes representation

Sampling	U	D	ID	$PR_{1/2}$	$PR_{2/3}$	$PR_{3/4}$	FL_2	FL_3	FL_4
prob.	0.179	0.199	0.205	0.194	0.204	0.205	0.214	0.224	0.222
degree-rank (%)	25.349	12.662	18.073	23.837	22.465	20.737	18.377	17.883	18.326
Page-Rank (%)	31.150	26.0812	15.363	27.259	23.328	20.579	13.911	13.272	14.227

network where nodes denote users, and edges represent retweets among them. Users in the dataset are classified as either "hateful" (attribute 1) or "normal" (attribute 2) depending on the sentiment of their tweets [7]. "Hateful" users represent the minority. We consider the largest connected component of the network and remove loops and multiple edges for a comparison with the synthetic networks. Table 5 shows the key characteristics of interest of the directed network (with diameter 24). The results (averaged over 10^4 runs) in Table 6 are in line with the synthetic model, where fixed length walk sampling shows the higher probability of sampling a minority node in addition to a higher rank compared to uniform sampling. The smaller differences are due to the characteristics of the network, where the proportions of edges from "normal" to "hateful" users is only slightly higher than in the opposite direction. This can also be seen from the homophily measures H_{21} and H_{12}.

4 Relative Ranking of Minorities Under Sampling

The aim of this section is to quantify the *relative ranking* of nodes of type 1 compared to type 2 by observing the attribute type counts in a pre-specified fraction $\gamma \in (0, 1)$ of nodes selected under one of the following two sampling schemes:

A: select γ fraction of nodes with the highest degrees (strictly speaking, this sampling does not involve randomness at the sampling level);
B: sample without (or with, but not used in the scenarios below) replacement γ fraction of nodes with probability proportional to degrees.

If an attribute type predominates the other attribute type in a given sampling scheme, we call it the *higher ranked attribute* for that scheme. As in Sect. 3, we thus consider the proportion of minority nodes in samples, but now focus on degree-based sampling schemes A and B, dependence on γ (for small sample sizes), and also on network structural properties such as homophily and out-degrees.

4.1 Synthetic Networks

For the synthetic network (1), we explore the questions above in terms of its model parameters κ (the propensity matrix determining homophily) and $\boldsymbol{m} =$

Table 7. Heterophilic synthetic networks: proportion of minority nodes.

γ		0.01	0.02	0.03	0.04	0.05	0.1	0.15	0.20	0.3	0.4	0.5
Scheme A:	$m_1 = 1, m_2 = 1$	0.868	0.810	0.733	0.673	0.626	0.514	0.467	0.417	0.407	0.355	0.325
	$m_1 = 5, m_2 = 1$	0.372	0.458	0.515	0.560	0.576	0.660	0.679	0.701	0.732	0.754	0.513
Scheme B:	$m_1 = 1, m_2 = 1$	0.442	0.432	0.433	0.422	0.422	0.397	0.384	0.371	0.355	0.343	0.334
	$m_1 = 5, m_2 = 1$	0.535	0.5424	0.546	0.545	0.550	0.550	0.550	0.544	0.533	0.513	0.488

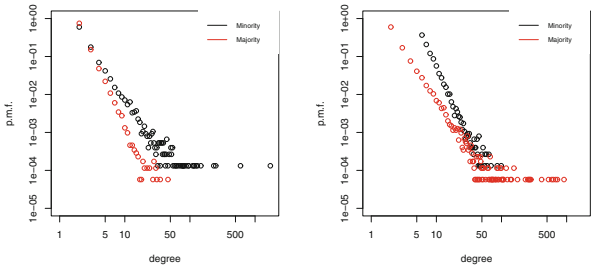


Fig. 2. Heterophilic networks (l.h.s.) $m_1 = m_2 = 1$ (r.h.s.) $m_1 = 5$, $m_2 = 1$: degree distribution.

(m_1, m_2) (the out-degree vector). We shall gain insight through the following results and the quantities involved. From the analysis of the linear model [1], we have: as $n, k \to \infty$,

$$\widehat{\eta}_a^{\boldsymbol{m}} := \frac{\sum_{v \in V: a(v)=a} \deg(v, n)}{2(n + n_0)} \to \eta_a^{\boldsymbol{m}}, \quad \mathbf{p}_n^{\boldsymbol{m},a}(k) \sim k^{-(1+2/\phi_a^{\boldsymbol{m}})}, \; a = 1, 2, \tag{2}$$

where $\eta_a^{\boldsymbol{m}}$ represents the limit of the normalized sum $\widehat{\eta}_a^{\boldsymbol{m}}$ of degrees of attribute type a and $\mathbf{p}_n^{\boldsymbol{m},a}(k)$ represents the proportion of nodes of type a with degree k which follows a power law with exponent $\Phi_a^{\boldsymbol{m}} := 2/\phi_a^{\boldsymbol{m}}$ in the limit. The quantities $\eta_a^{\boldsymbol{m}}$ and $\phi_a^{\boldsymbol{m}}$ are related to the relative ranking of minorities under the two sampling schemes A and B above and can be precisely computed. $(\eta_1^{\boldsymbol{m}}, \eta_2^{\boldsymbol{m}})$ is the minimizer of a suitable function [1, Eq. (4.1)], and

$$\phi_a^{\boldsymbol{m}} = 2 - m_a \pi_a / \eta_a^{\boldsymbol{m}}. \tag{3}$$

If $\phi_1^{\boldsymbol{m}} > \phi_2^{\boldsymbol{m}}$, the tail of the minority degree distribution is heavier (see Eq. (2)) and hence minorities are higher ranked in scheme A. On the other hand, if $\eta_1^{\boldsymbol{m}} > \eta_2^{\boldsymbol{m}}$, the probability of sampling a minority node is higher in each draw and hence the same conclusion holds in scheme B. We consider three different network configurations as follows (for the proofs of the results (4)–(9) below, see [1]).

Heterophilic Network. We first consider the scenario of a strongly heterophilic network, such that $\kappa_{11} = \kappa_{22} = 1$ and $\kappa_{12} = \kappa_{21} = K$ is large. In this case, node pairs with different attributes are more likely to be connected than node pairs with concordant attributes. As K increases, $\phi_1^{\boldsymbol{m}}$ and $\phi_2^{\boldsymbol{m}}$ behave as

$$\phi_1^{\boldsymbol{m}} \approx 2\left(1 - \frac{m_1\pi_1}{m_1\pi_1 + m_2\pi_2}\right), \qquad \phi_2^{\boldsymbol{m}} \approx 2\left(1 - \frac{m_2\pi_2}{m_1\pi_1 + m_2\pi_2}\right). \tag{4}$$

Table 8. Homogenous homophily networks ($m_1 = 5, m_2 = 1$; $m_1 = 5$, $m_1 = 2$) and homogenous mixing network ($m_1 = 2$, $m_2 = 1$): proportion of minority nodes.

γ		0.01	0.02	0.03	0.04	0.05	0.1	0.15	0.20	0.3	0.4	0.5
scheme A:	$m_1 = 5, m_2 = 1$	0.888	0.886	0.887	0.881	0.890	0.868	0.859	0.857	0.86	0.751	0.601
	$m_1 = 5$, $m_2 = 2$	0.700	0.718	0.697	0.693	0.694	0.692	0.698	0.658	0.673	0.6556	0.614
	$m_1 = 2$, $m_2 = 1$	0.552	0.598	0.601	0.594	0.593	0.589	0.589	0.576	0.586	0.547	0.580
Scheme B:	$m_1 = 5, m_2 = 1$	0.684	0.690	0.682	0.676 8	0.677	0.659	0.6441	0.629	0.596	0.558	0.516
	$m_1 = 5$, $m_2 = 2$	0.527	0.522	0.518	0.522	0.516	0.505	0.497	0.493	0.473	0.455	0.436
	$m_1 = 2$, $m_2 = 1$	0.5056	0.505	0.503	0.507	0.502	0.499	0.493	0.487	0.477	0.465	0.451

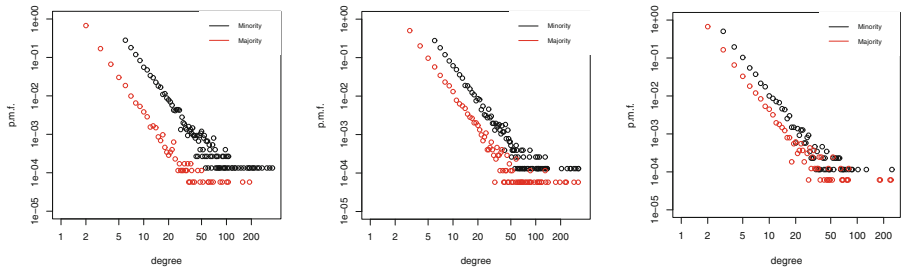


Fig. 3. Homogenous homophily (l.h.s.) $m_1 = 5$, $m_2 = 1$ (m.h.s.) $m_1 = 5$, $m_2 = 2$ and homogenous mixing: (r.h.s.) $m_1 = 2$, $m_2 = 1$: degree distribution.

Thus, the rank of minority nodes under scheme A depends on the relation between $m_1\pi_1$ and $m_2\pi_2$. Table 7 shows the results for two linear networks with 25,000 nodes, $K = 10$ and $\pi_1 = 0.3$. The out-degree vectors $\boldsymbol{m}$ are $(1, 1)$ and $(5, 1)$. For $m_1 = 1$, we have $\phi_1^{\boldsymbol{m}} \approx 1.373$ and $\phi_2^{\boldsymbol{m}} \approx 0.659$ (using (3)) which are close, respectively, to 1.4 and 0.6 given by the approximations in (4). In this case $m_1\pi_1 < m_2\pi_2$, and the minority nodes rank higher under scheme A due to the fact that majority nodes tend to connect to minority nodes, increasing their ranks. This holds for any tree network. For $m_1 = 5$, we have $\phi_1^{\boldsymbol{m}} \approx 0.688$ and $\phi_2^{\boldsymbol{m}} \approx 1.377$ which are close, respectively, to 0.636 and 1.364 given by (4). In this setting $m_1\pi > m_2\pi_2$, the minority nodes increase the ranks of majority nodes for small values of γ, by connecting to the majority with more output edges. (Note that when $\gamma = 1$ the relative ranking is given by the proportion of minority nodes in the network.) Figure 2 shows the degree distribution for each attribute, where in (l.h.s.) the minority has a heavier tail ($\phi_1^{\boldsymbol{m}} > \phi_2^{\boldsymbol{m}}$) and in (r.h.s.) it is the majority ($\phi_1^{\boldsymbol{m}} < \phi_2^{\boldsymbol{m}}$).

As K gets larger, $\eta_1^{\boldsymbol{m}}$ and $\eta_2^{\boldsymbol{m}}$ approach the same limit value

$$\eta_1^{\boldsymbol{m}} \approx \eta_2^{\boldsymbol{m}} \approx \frac{m_1\pi_1 + m_2\pi_2}{2}, \tag{5}$$

which implies that the differences between the relative rankings are smaller between the two attributes for scheme B. Table 7 shows the relative ranking of the minority for the networks described above under this scheme (the results were averaged over a large number of runs). For $m_1 = 1$, we have $\eta_1^{\boldsymbol{m}} \approx 0.478$ and $\eta_2^{\boldsymbol{m}} \approx 0.522$ which are close to 0.5 given by the approximation in (5). For

Table 9. Asymmetric homophily: proportion of minority nodes.

γ		0.01	0.02	0.03	0.04	0.05	0.1	0.15	0.20	0.3	0.4	0.5
Scheme A:	$m_1 = 1, m_2 = 1$	0.712	0.632	0.589	0.570	0.526	0.468	0.4565	0.397	0.384	0.345	0.323
	$m_1 = 2, m_2 = 1$	0.980	0.938	0.911	0.890	0.866	0.769	0.725	0.702	0.597	0.601	0.596
Scheme B:	$m_1 = 1, m_2 = 1$	0.423	0.411	0.396	0.392	0.390	0.370	0.363	0.357	0.342	0.334	0.327
	$m_1 = 2, m_2 = 1$	0.591	0.585	0.568	0.567	0.559	0.539	0.520	0.501	0.4766	0.449	0.425

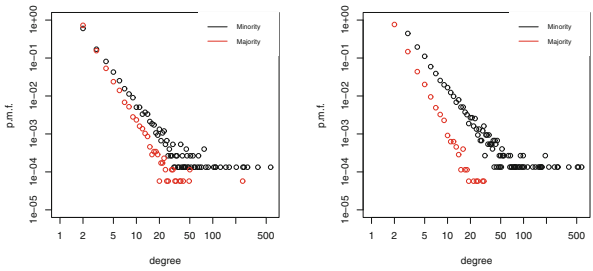


Fig. 4. Asymmetric Homophily: (l.h.s.) $m_1 = m_2 = 1$ (r.h.s.) $m_1 = 2$, $m_2 = 1$: degree distribution.

$m_1 = 5$, we have $\eta_1^{\boldsymbol{m}} \approx 1.144$, $\eta_2^{\boldsymbol{m}} \approx 1.056$ which approach 1.1 in (5). However, the higher value of $\eta_1^{\boldsymbol{m}}$ makes the minority slightly more dominant for scheme B.

Homogenous Homophily and Homogenous Mixing. We consider the cases of a strong homogeneous homophily with $\kappa_{21} = \kappa_{21} = 1$ and $\kappa_{11} = \kappa_{22} = K$ large and homogenous mixing with all the elements of the matrix κ equal to 1. As K goes to infinity, the exponents of the tail degree distribution per attribute are equal and behave as

$$\phi_1^{\boldsymbol{m}} = \phi_2^{\boldsymbol{m}} \approx 1, \tag{6}$$

which also holds in the case of homogenous mixing. However, we will see that the relative ranking of the minority under scheme A will depend on the ratio m_1/m_2. Table 8 depicts two homogenous homophily networks with 25,000 nodes, $K = 10$, $\pi_1 = 0.3$, and $\boldsymbol{m}$ vectors $(5, 1)$ and $(5, 2)$ which result in $\phi_1^{\boldsymbol{m}} \approx 1.022$, $\phi_2^{\boldsymbol{m}} \approx 0.948$ and $\phi_1^{\boldsymbol{m}} \approx 1.003$, $\phi_2^{\boldsymbol{m}} \approx 0.997$, respectively. An homogenous mixing network with 25,000 nodes, $\pi_1 = 0.35$ and $\boldsymbol{m} = (2, 1)$ is also considered. Figure 3 shows the degree distributions per attribute. Despite the degree tail exponents being similar from the plots, if m_1 is larger than m_2, the degrees of minority nodes get a high initial boost. Additionally, from the works [4,8], for multi-attributes, there is a "persistence phenomenon", i.e., the maximal degree nodes from any attribute type emerge from, with high probability, the oldest nodes of that type added to the network. Therefore, the results in Table 8 show that minority nodes have a higher ranking under scheme A.

On the other hand, as K goes to infinity (homogeneous homophily) and also for homogenous mixing,

$$\eta_1^{\boldsymbol{m}} \approx m_1\pi_1, \qquad \eta_2^{\boldsymbol{m}} \approx m_2\pi_2. \tag{7}$$

Table 10. Empirical networks: proportion of minority nodes.

γ		0.01	0.02	0.03	0.04	0.05	0.1	0.15	0.20	0.3	0.4	0.5
Scheme A:	Hate	0.778	0.778	0.716	0.704	0.689	0.637	0.588	0.533	0.430	0.356	0.304
	APS	0.154	0.115	0.132	0.157	0.203	0.211	0.250	0.266	0.297	0.289	0.300
Scheme B:	Hate	0.472	0.460	0.460	0.460	0.451	0.427	0.413	0.404	0.386	0.311	0.282
	APS	0.269	0.294	0.277	0.278	0.302	0.291	0.284	0.283	0.300	0.298	0.305

For the networks considered with $\boldsymbol{m} = (5, 1)$ and $\boldsymbol{m} = (5, 2)$, the exact values (resp. approximations in (7)) are $\eta_1^{\boldsymbol{m}} \approx 1.534$, $\eta_2^{\boldsymbol{m}} \approx 0.666$ (resp. 1.5 and 0.7), and $\eta_1^{\boldsymbol{m}} \approx 1.504$, $\eta_2^{\boldsymbol{m}} \approx 1.396$ (resp. 1.5 and 1.4). For $\boldsymbol{m} = (2, 1)$, the true value and approximation match with $\eta_1^{\boldsymbol{m}} \approx 0.7$, $\eta_2^{\boldsymbol{m}} \approx 0.65$. Thus, if $m_1\pi_1 > m_2\pi_2$, the minority nodes rank higher under scheme B – see Table 8.

In both types of networks, minority nodes can increase their popularity via schemes A and B through a higher ratio m_1/m_2. In the context of social networks, it means minorities increasing their social interaction.

Asymmetric Homophily. The last scenario is the case of a strong asymmetric homophily network (slightly different from Sec. 3), where $\kappa_{11} = K$ is large, and $\kappa_{22} = \kappa_{12} = \kappa_{21} = 1$. As K tends to infinity,

$$\phi_1^{\boldsymbol{m}} \approx \frac{2m_1\pi_1 + 3m_2\pi_2}{2m_1\pi_1 + 2m_2\pi_2}, \quad \phi_2^{\boldsymbol{m}} \approx \frac{m_2\pi_2}{m_1\pi_1 + m_2\pi_2} \tag{8}$$

and

$$\eta_1^{\boldsymbol{m}} \approx \frac{2m_1\pi_1(m_1\pi_1 + m_2\pi_2)}{2m_1\pi_1 + m_2\pi_2}, \quad \eta_2^{\boldsymbol{m}} \approx \frac{m_2\pi_2(m_1\pi_1 + m_2\pi_2)}{2m_1\pi_1 + m_2\pi_2}. \tag{9}$$

Two networks are considered with 25,000 nodes, $K = 10$ and $\boldsymbol{m}$ vectors $(1, 1)$ and $(2, 1)$ in Table 9. In both networks, $\phi_1^{\boldsymbol{m}} > \phi_2^{\boldsymbol{m}}$ and the minorities rank higher under scheme A. The exact values are $\phi_1^{\boldsymbol{m}} \approx 1.31$, $\phi_2^{\boldsymbol{m}} \approx 0.761$ ($m_1 = 1$) and $\phi_1^{\boldsymbol{m}} \approx 1.247$, $\phi_2^{\boldsymbol{m}} \approx 0.609$ ($m_1 = 2$) which are close to the approximations in (8). This also agrees with the degree tail exponents in Fig. 4 with the degree distribution of the minority being more heavy-tailed (higher $\phi_1^{\boldsymbol{m}}$).

Under scheme B, minorities rank higher with $m_1 = 2$ since $2m_1\pi_1 > m_2\pi_2$ in (9) (also $\eta_1^{\boldsymbol{m}} \approx 0.821$, $\eta_2^{\boldsymbol{m}} \approx 0.479$). This means that in a social network, if the arriving majority nodes have almost a neutral attribute preference attachment ($\kappa_{12} = \kappa_{22}$), the minorities can increase their popularity through the number of outgoing edges that connect to other minority nodes.

4.2 Real Networks

We consider two real-world networks with power-law degree distributions to assess the ranking of the minorities under schemes A and B. For the Hate network in Sect. 3.4, the exponents of the fitted degree distributions ($\widehat{\Phi}_a^{\boldsymbol{m}}$) are 2.776 and 3.338; and the normalized sums of the degrees ($\widehat{\eta}_a^{\boldsymbol{m}}$) are 3.223 and 3.617 for the minority and majority, respectively. Table 10 shows that under scheme

A, the minorities rank higher. APS is a scientific network from the American Physical Society where nodes represent articles from two subfields and edges represent citations with homogeneous homophily. Some networks statistics are: 1281 (nodes), 3064 (edges). The minority rank is lower in both schemes where the exponents and normalized sums of the degrees are 3.947 and 3.292, and 1.332 and 3.452 respectively, for subfields 1 (minority) and 2 (majority). For these two real networks, the results on relative ranking of the minority are in line with those for the synthetic networks.

5 Conclusions and Future Work

This paper explored settings where Page-rank and walk-based network sampling schemes favor small minority attribute nodes compared to uniform sampling. We also investigated the conditions for the minority nodes to rank higher in degree-based sampling. To this end, we used an attributed network model with homophily under several network configurations which provided insight into real-world networks.

In follow-up work, we plan to compare and contrast the performance of various centrality measures, including degree and Page-rank centrality, for ranking and attribute reconstruction tasks in the semi-supervised setting, where one has partial information on the attributes and wants to reconstruct it for the rest of the network. In the setting of dynamic and evolving networks, contrary to static networks, preliminary results in [1] seem to suggest starkly different behavior between degree vs Page-rank centrality in such settings.

Acknowledgements. S.Ba. is partially supported by the NSF CAREER award DMS-2141621. S.Bh. and V.P. are partially supported by NSF DMS-2113662. S. Ba., S.Bh. and V.P. are partially supported by NSF RTG grant DMS-2134107.

References

1. Antunes, N., Banerjee, S., Bhamidi, S., Pipiras, V.: Attribute network models, stochastic approximation, and network sampling and ranking. arXiv preprint arXiv:2304.08565v1 (2023)
2. Antunes, N., Bhamidi, S., Guo, T., Pipiras, V., Wang, B.: Sampling based estimation of in-degree distribution for directed complex networks. J. Comput. Graph. Stat. **30**(4), 863–876 (2021)
3. Antunes, N., Guo, T., Pipiras, V.: Sampling methods and estimation of triangle count distributions in large networks. Netw. Sci. **9**(S1), S134–S156 (2021)
4. Banerjee, S., Bhamidi, S.: Persistence of hubs in growing random networks. Probab. Theor. Relat. Fields **180**(3–4), 891–953 (2021)
5. Chebolu, P., Melsted, P.: PageRank and the random surfer model. In: SODA 2008, pp. 1010–1018 (2008)
6. Crawford, F.W., Aronow, P.M., Zeng, L., Li, J.: Identification of homophily and preferential recruitment in respondent-driven sampling. Am. J. Epidemiol. **187**(1), 153–160 (2018)

7. Espín-Noboa, L., Wagner, C., Strohmaier, M., Karimi, F.: Inequality and inequity in network-based ranking and recommendation algorithms. Sci. Rep. **12**(1), 1–14 (2022)
8. Galashin, P.: Existence of a persistent hub in the convex preferential attachment model. arXiv preprint arXiv:1310.7513 (2013)
9. Karimi, F., Génois, M., Wagner, C., Singer, P., Strohmaier, M.: Homophily influences ranking of minorities in social networks. Sci. Rep. **8**(1), 1–12 (2018)
10. Merli, M.G., Verdery, A., Mouw, T., Li, J.: Sampling migrants from their social networks: the demography and social organization of Chinese migrants in Dar es Salaam, Tanzania. Migr. Stud. **4**(2), 182–214 (2016)
11. Mouw, T., Verdery, A.M.: Network sampling with memory: a proposal for more efficient sampling from social networks. Sociol. Methodol. **42**(1), 206–256 (2012)
12. Park, J., Barabási, A.-L.: Distribution of node characteristics in complex networks. Proc. Natl. Acad. Sci. **104**(46), 17916–17920 (2007)
13. Stolte, A., Nagy, G.A., Zhan, C., Mouw, T., Merli, M.G.: The impact of two types of COVID-19-related discrimination and contemporaneous stressors on Chinese immigrants in the US South. SSM Ment. Health **2**, 100159 (2022)

A Framework for Empirically Evaluating Pretrained Link Prediction Models

Emilio Sánchez Olivares, Hanjo D. Boekhout, Akrati Saxena(✉), and Frank W. Takes

Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands
{h.d.boekhout,a.saxena,f.w.takes}@liacs.leidenuniv.nl

Abstract. This paper proposes a novel framework for empirically assessing the effect of network characteristics on the performance of pretrained link prediction models. In link prediction, the task is to predict missing or future links in a given network dataset. We focus on the pretrained setting, in which such a predictive model is trained on one dataset, and employed on another dataset. The framework allows one to overcome a number of nontrivial challenges in adequately testing the performance of such a pretrained model in a proper cross-validated setting. Experiments are performed on a corpus of 49 structurally diverse real-world complex network datasets from various domains with up to hundreds of thousands of nodes and edges. Overall results indicate that the extent to which a network is clustered is strongly related to whether this network is a suitable candidate to create a pretrained model on. Moreover, we systematically assessed the relationship between topological similarity and performance difference of pretrained models and a model trained on the same data. We find that similar network pairs in terms of clustering coefficient, and to a lesser extent degree assortativity and gini coefficient, yield minimal performance difference. The findings presented in this work pave the way for automated model selection based on topological similarity of the networks, as well as larger-scale deployment of pretrained link prediction models for transfer learning.

Keywords: Link Prediction · Transfer Learning · Pretrained Models

1 Introduction

In recent years, researchers have studied complex networks to understand and analyze the intricate relationships that underlie various real-world systems. Complex networks, characterized by their non-trivial topological structures, have applications in diverse fields such as the social sciences, biology, transportation, and information technology [6]. Understanding the dynamics of these different types of networks and predicting the formation of new or missing connections, also known as "link prediction", is a well-known and well-studied problem in the field [17]. Link prediction aims to uncover hidden or potential interactions in a

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 150–161, 2024.
https://doi.org/10.1007/978-3-031-53468-3_13

network; for example, to predict who might connect to whom in a social network or which proteins are likely to interact in a biological network. Furthermore, link prediction is also used in various other applications and tasks, including recommender systems, anomaly detection, privacy control, network routing, and understanding the underlying mechanisms that govern network evolution [9–11,14].

In literature, different types of methods for link prediction have been proposed. Initial methods focused on node pair similarities, such as the Jaccard coefficient, Adamic-Adar index, and resource-allocation index [1,17]. Node pair similarity relies on the notion that if a given pair of nodes has a similarity score higher than some threshold, then this pair is more likely to be connected [11]. Later, researchers proposed other types of methods, including (i) maximum likelihood-based methods that work on maximizing the likelihood of the observed structure so that any missing link can be calculated using the identified rules and parameter [22], (ii) probabilistic models based methods that focus on modeling the underlying network structure and then use the learned model to predict the missing links [24], (iii) machine learning-based methods that train a machine learning model based on node pair features for existing and non-existing links [2,5], and (iv) network embedding-based methods that create a low dimensional representation of the network using word2vec models or matrix-factorization, and then train a machine learning model using these vector representation of nodes to predict missing links [8,16,23]. In literature, it has been shown that the third category, machine learning based methods, outperforms other types of methods and has lately become the focus of link prediction research [12]. An additional advantage is that the use of topological features of the node pairs ensures the interpretability and explainability of resulting models through the analysis of feature importance. However, one limitation of these methods is that a link prediction model must be trained for each new network dataset.

To solve this problem, people have used transfer learning, i.e., a machine learning technique where a model developed for a particular task is reused or adapted as the starting point for a model on a second task [20]. Instead of training a new model from scratch, transfer learning leverages the knowledge gained from solving one problem and applies it to a different but related problem. By using a pretrained model as a starting point, one can save time and resources compared to training a new model from the ground up. In this work, we investigate the feasibility of transfer learning for link prediction in real-world complex networks.

In the remainder of this work, we analyze the characteristics and topology of 49 networks to understand how they affect the ability to train and predict links across networks. Specifically, we first propose a framework to perform cross-validation across multiple datasets, to efficiently test and compare the transfer learning performance of pretrained models for link prediction. Working towards automated pretrained model selection, we subsequently investigate what kind of topological network properties are important for selecting a well-performing pretrained model. Finally, we analyze what topological network similarities between training and testing networks, yield good transfer learning performance. In doing

so, we aim to understand to what extent transfer learning can be applied to predict unseen links in real-world networks by employing pretrained models.

The structure of the remainder of this paper is as follows. In Sect. 2, we discuss the approach followed to train our link prediction model, as well as the framework to test transfer learning. Then, Sect. 3 describes the data, evaluation criteria used, and the experimental setup developed, as well as the experimental results. Finally, we draw conclusions and propose future directions of research in Sect. 4.

2 Methodology

In this section, we first discuss the network features used to train predictive models for link prediction. Then, we give an overview of machine learning algorithms used to predict missing links and explain how we split the datasets for training and testing.

2.1 Features

Working towards a machine learning model that takes node pairs as input, and outputs whether this node pair is likely to be connected in the future, features that describe these node pairs are required.

In this work, we employ features commonly used in link prediction models, focusing on the work presented by Bors [3], to design a good link prediction model and test transfer learning. The chosen features balance simplicity, speed, and performance. We note that this study aims not to design the best link prediction model with the most comprehensive set of features, but instead aims to assess the feasibility of transfer learning in link prediction.

The selected set of features used throughout our experiments are as follows: (i) total neighbors, i.e., the union of all neighbors of the source and target nodes; (ii) common neighbors, i.e., the number of nodes connected to both the source and target nodes; (iii) Jaccard Coefficient [17], i.e., the ratio between the common and total neighbors; (iv) Adamic-Adar [1], which used to compute the closeness of nodes based on their shared neighbors; (v) preferential attachment [17,19], i.e., the multiplication of the number of neighbors of the source and target nodes; (vi) degree of the source node (vii) degree of target nodes, (viii) ratio of degrees of source and target node, (ix) triangle count for the source node, and (x) triangle count for the target nodes, i.e., denoting the number of triangles they are involved in.

2.2 Training and Testing Set Generation

To generate a training dataset from the network, the node pairs with existing links are considered as positive cases, and node pairs with distance two are used as negative cases. We consider only node pairs at a distance of at most two, as links are more likely to be formed between already close nodes. This is reflected in the chosen features, many of which are not applicable to nodes at a distance larger than two. Moreover, it assists in reducing the class imbalance.

2.3 Stacked Classifier

In this work, we use a supervised machine learning model for link prediction classification. Based on the findings by Ghasemian et al. [7], we set up a stacked classifier using *Scikit-Learn* [21], in which we use the most commonly used classifier models in the literature [7,15,18,23]: a random forest classifier, logistic regression, naive-bayes, and quadratic discriminant analysis models. Together, these models serve as base estimators. We then use Logistic Regression as meta-model to combine ("stack") these predictions to make a final prediction.

2.4 Cross-Validation Framework

Here we discuss the proposed framework for adequate cross-validation training. We split the datasets into training and testing subsets to evaluate the performance of the prediction model. The datasets are split using k-fold, and in this work, we set $k = 4$, resulting in a random 75–25% split, to avoid bias in the sample. Additionally, due to the sparsity of networks, we down-sample the majority class of the training dataset to account for class imbalance. We perform cross-validation training as shown in Fig. 1 to validate a model's results on different data portions. Essentially, we train models for each fold per dataset and then validate them on each split of the datasets. Note that in Fig. 1, we only train and validate on the same split when testing on the same network. This is because, within the same split, the data is disjoint, so we do not have the same observations in the train and validation sets. Otherwise, we would encounter the same data in both the training and validation sets, which is not ideal for a machine learning model since it adds bias to the model by predicting previously seen data. Thus, we removed those cases from our testing set.

3 Experiments

This section covers the experimental setup and results. First, in Sect. 3.1, we discuss the datasets and metrics used. Then, in Sect. 3.2, we determine the overall feasibility of using transfer learning for link prediction by studying the AUC (loss) matrix and distributions. Next, in Sect. 3.3, we discuss what the most important topological features are that affect the performance of a pretrained model. Finally, in Sect. 3.4, we determine which structural network similarities yield good transfer learning performance.

3.1 Experimental Setup

Datasets. In order to test the capabilities of transfer learning, we analyze 49 network datasets from the KONECT Project [13]. The networks were chosen such that they cover a variety of topological properties, sizes, and categories. The datasets are presented in Table 1, along with their respective number of nodes and edges. All networks are interpreted as undirected and unweighted. Additionally, we only consider the largest connected component of each network.

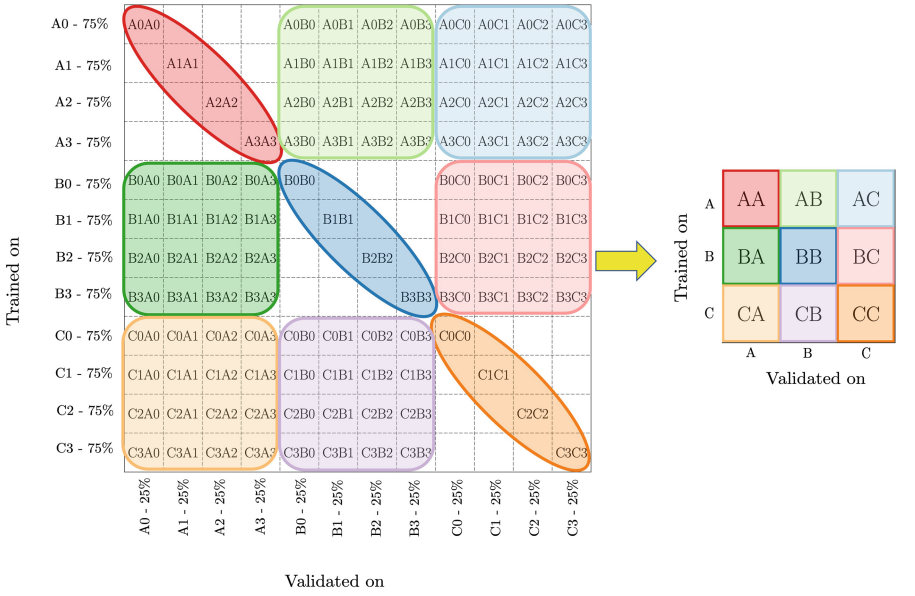


Fig. 1. Cross-validation training assignments. For each split, we generate a training set with 75% of the data and a validation set with the remaining 25%. (Letters indicate the dataset, numbers indicate the split.)

Metrics. Similarly to previous works [4,7], we measure the performance of the classifier using the Area Under the Receiver Operating Characteristic Curve (AUC). Additionally, we define AUC loss, which we use to measure the loss in accuracy resulting from applying transfer learning between two different networks. We calculate loss $\mathcal{L}_{i,j}$ as follows:

$$\mathcal{L}_{i,j} = AUC_{i,i} - AUC_{i,j}$$

Here, $\mathcal{L}_{i,j}$ is the loss of training a model with network i and validating on network j and $AUC_{i,j}$ is the performance score of training a model with network i and validating on network j.

For all pairs of network datasets considered, the resulting AUC score of using a model trained on network i to predict missing links on network j, can be presented in a matrix as shown in Fig. 1. In other words, for each pair of networks, we train on each fold of network i and test each trained model on each fold of network j (provided $i \neq j$), and then aggregate over all combinations of folds by averaging the AUC scores.

3.2 Feasibility of Transfer Learning in Link Prediction

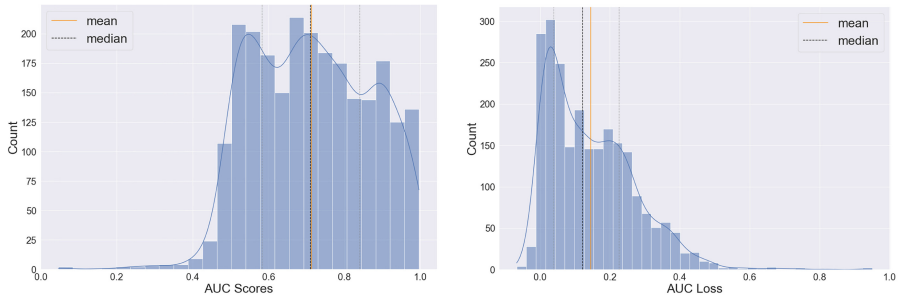
To test whether transfer learning is feasible for link prediction, we assess if it is possible to pretrain a model on one network and test it on another with minimal

Table 1. Datasets sourced from [13], along with the number of nodes and edges.

dolphins (62/159)	residence (217/2,672)	copperfield (112/425)
cora (23,166/91,500)	karate (34/78)	proteins (1,706/6,207)
dblp (12,590/49,759)	adolescent (2,539/12,969)	reactome (6,327/147,547)
hepph (34,546/421,578)	blogs (1,224/19,025)	yeast (1,870/2,277)
hepth (27,770/352,807)	foldoc (13,356/125,207)	asoif (796/32,629)
astroph (18,771/198,050)	airtraffic (1,226/2,615)	sistercities (14,274/20,573)
astrophysics (16,046/121,251)	newyork (264,346/730,100)	lesmis (77/254)
erdos (6,927/11,850)	openflights (3,425/67,633)	pgp (10,680/24,316)
networkscience (1,461/2,742)	contiguous (49/107)	wikipedia (7,118/103,675)
digg (30,398/87,627)	euroroad (1,174/1,417)	hamsters (2,426/16,631)
dnc (2,029/39,264)	chess (7,301/65,053)	twitter (23,370/33,101)
facebook (46,952/876,993)	football (115/613)	filmtrust (874/1,853)
slashdot (51,083/140,778)	congress (219/764)	florida_dry (128/2,137)
uc_irvine (1,899/59,835)	bible (1,773/16,401)	florida_wet (128/2,106)
caida (26,475/53,381)	eat (23,132/511,764)	littlerocklake (183/2,494)
gnutella25 (22,687/54,705)	wordnet (146,005/656,999)	chesapeake (39/170)
routeviews (6,474/13,895)		

AUC loss. Therefore, we applied the cross-validation training procedure detailed in Sect. 2.4 to all 49 datasets. The distribution of the resulting AUC scores and AUC loss are shown in Fig. 2. Since we use the same model and the same features for all networks for link prediction, the AUC scores are, as expected, not particularly high, with an average AUC score of 0.71. More importantly, Fig. 2b shows that AUC loss is very low for many combinations of training and testing networks. However, in many use cases, the average AUC loss of 0.14 can still be considered too significant. This signals the important conclusion that one can not simply choose a random network to pretrain and apply it to any new network. Instead, an appropriate network should be selected to minimize the AUC loss.

To investigate which (types of) networks make transfer learning in link prediction more feasible, Fig. 3 shows the matrix of AUC loss for all pairs of training and testing networks for all 49 networks under consideration. Rows in the matrix depict the training performance of a single network, while columns represent the ease of prediction for a single network. In Fig. 3, we can see that networks from citation and co-authorship categories (i.e., `cora`, `dblp`, `hpph`, `hepth`, `astroph` and `astrophysics`), as well as miscellaneous (`asoif`, `sistercities`, `lesmis`), show favourable training performance with minimal AUC loss, suggesting they are good baseline for pretrained models. Similarly, computer networks (`caida`, `gnutella25`, and `routeviews`), infrastructure networks (`airtraffic`, `newyork`, `euroroad`) and metabolic networks (`proteins`, `yeast`) display good validation performance, meaning they are usually easy to predict regardless of the choice of pretrained model. On the other hand, the bible network is the worst performing training network, with several very large AUC losses. Furthermore, some



(a) AUC scores distribution. (b) AUC losses distribution.

Fig. 2. Distributions of the AUC scores and AUC loss across all networks. Alongside the mean and median the quartiles are depicted with light grey lines. The blue line approximates the trend.

human contact (`residence`, `karate`), human social (`adolescent`), lexical networks (`bible`, `eat`, `wordnet`) and trophic networks, i.e., relating to biological interactions of species (commonly food chains), (`florida_dry`, `florida_wet`, `littlerocklake`, `chesapeake`) show substandard validation performance, with few to no pretrained models providing low AUC loss. As such, there are some networks that often do well for training (pretrained) models and those for whom many pretrained models work well, but there are also some for which no pretrained model appears to perform well. Thus, although transfer learning for link prediction is feasible for most networks given the right choice of pretrained model, it is not effective in all cases, i.e., there is no one-size-fits-all kind of solution.

3.3 Topological Feature Importance for Pretraining Models

Next, we set to understand how a network's topological features might affect a model's learning performance. For this, we train decision tree and random forest algorithms by using the topological features of the networks to fit their average AUC score as pretrained models aggregated over all testing networks. The top splits, i.e., the top discriminating decisions, of the resulting trees are visualized in Fig. 4. By studying the top discriminating decisions of these trees, we can understand which are the most important topological features for good transfer learning performance and how these features affect the performance.

The decision tree depicted in Fig. 4a suggests that a high normalized number of triangles (#triangles/edges) results in, on average, higher AUC scores. Furthermore, for pretrained models from networks with a high maximum degree and few triangles (per link), we observe that the transitivity is a great indicator of either high or low resultant AUC scores, whereas, for lower maximum degree models and few triangles (per link), the average degree can be a good indicator.

The decision tree obtained from the random forest algorithm in Fig. 4b suggests that very low transitivity or relatively high transitivity with high mean

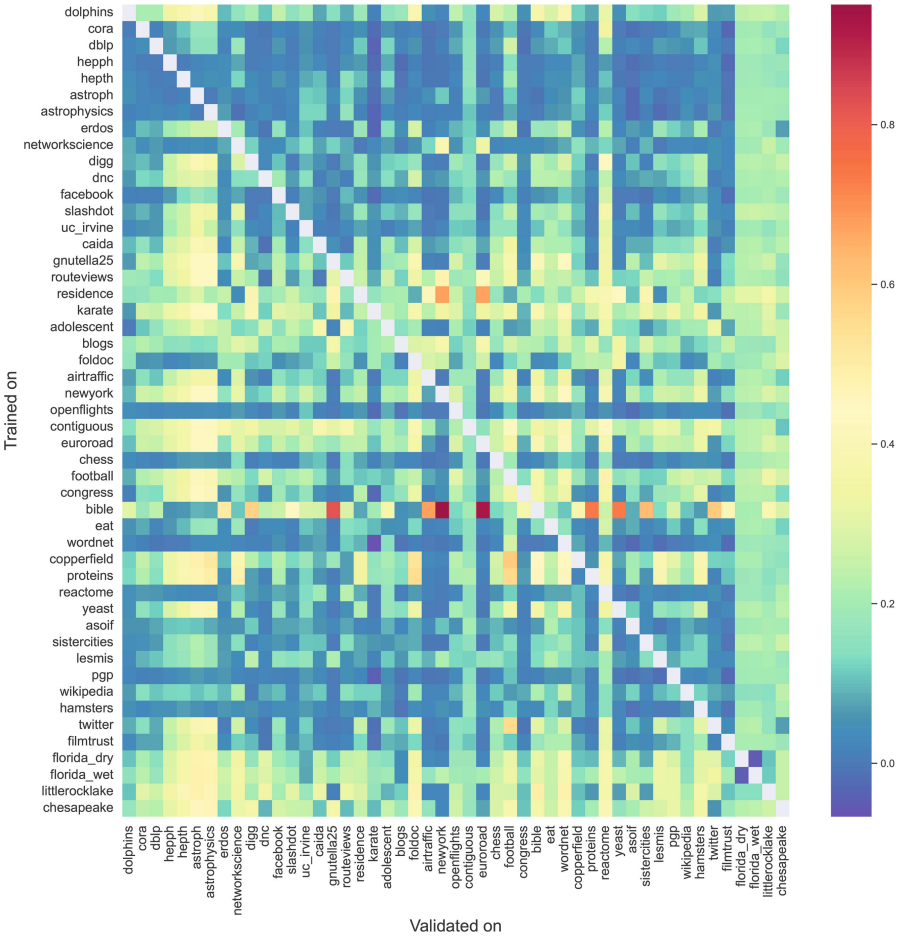
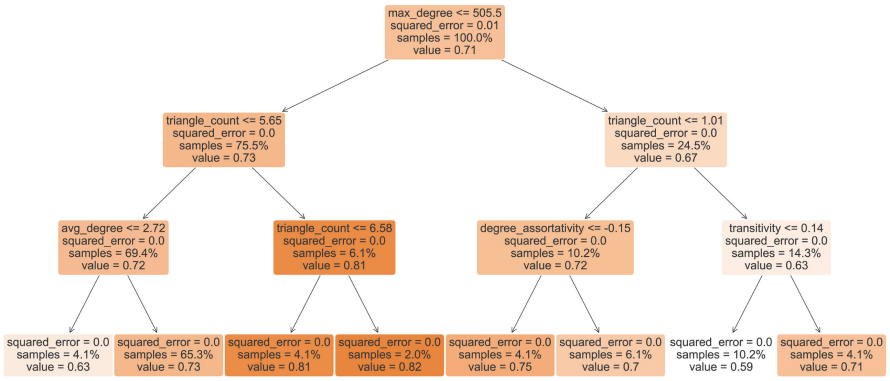
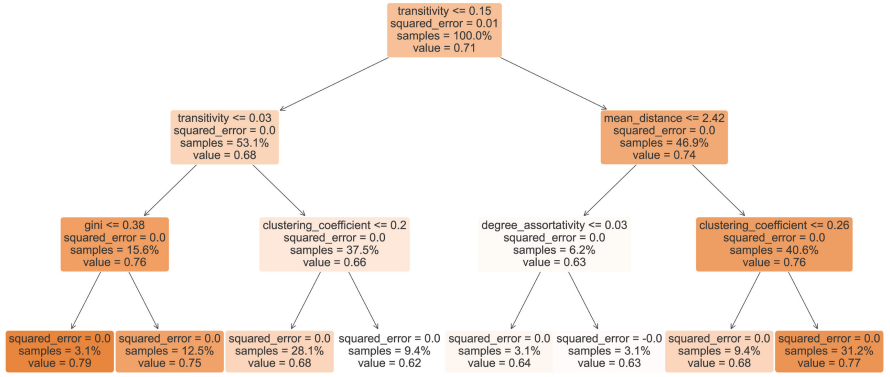


Fig. 3. AUC loss matrix for all 49 datasets.

distance and clustering coefficients tend to result in higher AUC scores. On the contrary, low-to-middle transitivity with higher clustering coefficients and high transitivity with small mean distances or small clustering coefficients result in low AUC scores. As such, only high transitivity or clustering coefficient are not universally good or bad for the transfer learning link prediction performance of a pretrained model. However, note that clustering coefficient and transitivity are usually correlated, so a network with a high clustering coefficient will likely not have low transitivity. Interestingly, our previous observation from Fig. 4b indicates that high AUC scores are obtained when these topological features are indeed correlated for a network, and low AUC scores are obtained when they are not.



(a) Decision tree.



(b) Random forest.

Fig. 4. Tree-based topological feature importance. Darker coloured nodes/leafs indicate higher average AUC scores achieved by the pretrained models of included networks, whereas lighter coloured leafs indicate lower average AUC scores. The average AUC scores of the node/leaf are indicated by 'value'. For each parent node the left child includes the pretrained models from networks that adhere to the condition specified by the parent node and the right child includes those that do not. For example, in figure (a) the left child of the root node includes all pretrained models from networks with a maximum degree ≤ 505.5 while the right child includes all those with a maximum degree > 505.5.

In short, we find that some of the most important topological features influencing the performance of pretrained models are the number of triangles (per link), the transitivity, and the clustering coefficient. Notably, the level of correlation between these features can be especially indicative of the resulting high or low AUC scores of a pretrained model.

3.4 Influence of Network Dissimilarity on Transfer Learning

Finally, we examine what structural network similarities between a training and a testing network lead to good transfer learning performance. We do so by comparing the AUC loss (as defined in Sect. 3.1) to how dissimilar the topological properties are between two networks. This allows us to understand if there is a relationship between their similarity and the performance of the model. Figure 5 illustrates the relation of loss in performance when predicting missing links in one network using a model trained on another, compared to the topological dissimilarities of both networks. It is clear that there is a trend in almost all topological features (except for maximum degree), and if two networks are more different, there is more loss in the performance. Specifically, we observe that if two networks are more similar in terms of clustering coefficient, it leads to the lowest AUC loss, while the most dissimilar networks for this feature have the second highest AUC loss. Furthermore, when it concerns degree assortativity, gini coefficient, and transitivity, we note that highly similar networks also show reasonably low AUC losses. Overall, our results indicate that considering the similarity in terms of clustering coefficient, and to a lesser extent in terms of degree assortativity, gini coefficient, and transitivity is especially important in choosing a pretrained model for link prediction.

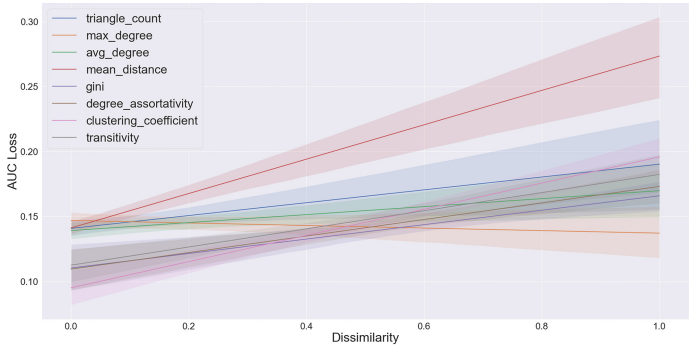


Fig. 5. AUC loss vs. network dissimilarity.

4 Conclusion

In this work, we studied the feasibility of using pretrained link prediction models in complex networks. Moreover, we studied the network characteristics that impact model training, and how these can be used for selecting a well-performing pretrained model. We conducted experimental analysis on a large corpus of structurally diverse networks, including co-authorship, citation friendship, human interaction, biological, and transportation networks. Through our experiments,

we observed that transfer learning for link prediction is a feasible way to move forward, and some network categories perform better as sources for training and others to predict missing links on. Furthermore, we found that network features based on local connectivity, such as clustering coefficient, number of triangles, or transitivity, are important indicators when picking a network for training a predictive model. Specifically, we found that when two networks show very dissimilar topologies in terms of clustering coefficient, but also in terms of degree assortativity, gini coefficient, and transitivity, it is likely that the performance of transfer learning is hindered.

This work demonstrates the feasibility of using pretrained models in link prediction. Future work could focus on designing better transfer learning methods to achieve higher accuracy using topological properties of an unseen network and the network used for pre-training. Additionally, this work opens an avenue to use transfer learning for complex network problems, such as node classification, role identification, and influence maximization.

References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Soc. Netw. **25**(3), 211–230 (2003)
2. Al Hasan, M., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: Workshop on Link Analysis, Counter-Terrorism and Security, SDM 2006, vol. 30, pp. 798–805 (2006)
3. Bors, P.P.: Topology-aware network feature selection in link prediction (2022)
4. de Bruin, G.J., Veenman, C.J., van den Herik, H.J., Takes, F.W.: Supervised temporal link prediction in large-scale real-world networks. Soc. Netw. Anal. Min. **11**(1), 1–16 (2021)
5. van Engelen, J.E., Boekhout, H.D., Takes, F.W.: Explainable and efficient link prediction in real-world network data. In: Boström, H., Knobbe, A., Soares, C., Papapetrou, P. (eds.) IDA 2016. LNCS, vol. 9897, pp. 295–307. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46349-0_26
6. Estrada, E.: The Structure of Complex Networks: Theory and Applications. Oxford University Press, USA (2012)
7. Ghasemian, A., Hosseinmardi, H., Galstyan, A., Airoldi, E.M., Clauset, A.: Stacking models for nearly optimal link prediction in complex networks. Proc. Natl. Acad. Sci. **117**(38), 23393–23400 (2020)
8. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
9. Guimerà, R., Sales-Pardo, M.: Missing and spurious interactions and the reconstruction of complex networks. Proc. Natl. Acad. Sci. **106**(52), 22073–22078 (2009)
10. Huang, Z., Zeng, D.D.: A link prediction approach to anomalous email detection. In: 2006 IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1131–1136. IEEE (2006)
11. Kumar, A., Singh, S.S., Singh, K., Biswas, B.: Link prediction techniques, applications, and performance: a survey. Phys. A Stat. Mech. Appl. **553**, 124,289 (2020)
12. Kumari, A., Behera, R.K., Sahoo, K.S., Nayyar, A., Kumar Luhach, A., Prakash Sahoo, S.: Supervised link prediction using structured-based feature extraction in social network. Concurrency Comput. Pract. Exp. **34**(13), e5839 (2022)

13. Kunegis, J., Staab, S., Dünker, D.: KONECT – the Koblenz network collection. In: Proceedings of the International School and Conference on Network Science (2012)
14. Li, J., Zhang, L., Meng, F., Li, F.: Recommendation algorithm based on link prediction and domain knowledge in retail transactions. Procedia Comput. Sci. **31**, 875–881 (2014)
15. Li, Y., Liu, X., Wang, C.: Research on link prediction under the structural features of attention stream network. In: 2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), pp. 148–154. IEEE (2021)
16. Li, Y., Wang, Y., Zhang, T., Zhang, J., Chang, Y.: Learning network embedding with community structural information. In: IJCAI, pp. 2937–2943 (2019)
17. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: Proceedings of the 12th International Conference on Information and knowledge management, pp. 556–559 (2003)
18. Liu, Z., Zhang, Q.M., Lü, L., Zhou, T.: Link prediction in complex networks: a local Naïve Bayes model. Europhys. Lett. **96**(4), 48,007 (2011)
19. Newman, M.E.: Clustering and preferential attachment in growing networks. Phys. Rev. E **64**(2), 025,102 (2001)
20. Niu, S., Liu, Y., Wang, J., Song, H.: A decade survey of transfer learning (2010–2020). IEEE Trans. Artif. Intell. **1**(2), 151–166 (2020)
21. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
22. Redner, S.: Teasing out the missing links. Nature **453**(7191), 47–48 (2008)
23. Saxena, A., Fletcher, G., Pechenizkiy, M.: NodeSim: node similarity based network embedding for diverse link prediction. EPJ Data Sci. **11**(1), 24 (2022)
24. Wang, C., Satuluri, V., Parthasarathy, S.: Local probabilistic models for link prediction. In: Seventh IEEE International Conference on Data Mining, ICDM 2007, pp. 322–331. IEEE (2007)

Masking Language Model Mechanism with Event-Driven Knowledge Graphs for Temporal Relations Extraction from Clinical Narratives

Kanimozhi Uma(✉), Sumam Francis, and Marie-Francine Moens

Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium
{kanimozhi.uma,sumam.francis,sien.moens}@kuleuven.be

Abstract. For many natural language processing systems, the extraction of temporal links and associations from clinical narratives has been a critical challenge. To understand such processes, we must be aware of the occurrences of events and their time or temporal aspect by constructing a chronology for the sequence of events. The primary objective of temporal relation extraction is to identify relationships and correlations between entities, events, and expressions. We propose a novel architecture leveraging Transformer based graph neural network by combining textual data with event graph embeddings for predicting temporal links across events, entities, document creation time and expressions. We demonstrate our preliminary findings on i2b2 temporal relations corpus for predicting BEFORE, AFTER and OVERLAP links with event graph for correct set of relations. Comparison with various Biomedical-BERT embedding types were benchmarked yielding best performance on PubMed BERT with language model masking (LMM) mechanism on our methodology. This illustrates the effectiveness of our proposed strategy.

Keywords: Natural Language Processing · Information Extraction · Temporal Relations Prediction · Clinical Narratives · Knowledge Graphs · Graph Embeddings · Graph Neural Networks

1 Introduction

It is crucial to extract temporal information from clinical narratives about events, expressions, and their occurrences to better comprehend the past and, to the best of our ability, can predict the future. A clinical event is anything that is pertinent to the clinical timeline, such as clinical concepts, entities, etc., and especially in the medical domain, there are a vast number of texts almost ready to be exploited. The foundation for performing temporal relationship tasks in NLP has traditionally been temporal events and expressions. Temporal information, such as dates, time expressions, durations, and intervals, allows for tracking disease progression, treatment timelines, and event sequencing. Events that occur within a clinical context, such as diagnoses, treatments, procedures, or laboratory

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 162–174, 2024.
https://doi.org/10.1007/978-3-031-53468-3_14

tests, have both temporal and spatial aspects. Clinical reports often contain explicit temporal expressions such as dates, durations, time intervals and extracting these expressions is the first step in identifying temporal information. For instance, identifying phrases like "two weeks ago" or "since last year" as temporal expressions. Once temporal expressions are identified, the next step is to establish relationships between different events or findings mentioned in the clinical reports. This involves determining the order of events, durations, or time intervals between them. For instance, determining whether a specific treatment occurred before or after a diagnosis or the duration between two laboratory test results.

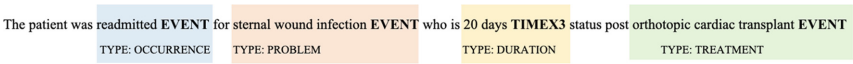


Fig. 1. Sample Entity and Event Temporal Relation Annotation Sentence

In this study, we attempt to extract temporal ordering information by identifying the chronological order or sequence of events or observations mentioned in the clinical narratives. This involves determining whether one event occurs before, after, or concurrently with another event Fig. 1. For example, establishing the sequence of events such as "diagnosis of pneumonia" preceding "administration of antibiotic therapy." Time expressions (TIMEXs) extracting temporal duration, document creation time or time interval between two events or observations involves identifying expressions of time duration, such as "three weeks," "six months," or "since last year" which provide details about when, how long, or how frequently something occurred. In the clinical timeline, temporal relations (such as "before," "after," or "overlap") show how two EVENTS, two TIMEXes, or an EVENT and a TIMEX are related to one another [1]. We provide a novel architecture for extracting temporal relations that combines data from a text document with supplementary data found in the form of a knowledge graph.

Our approach introduces a unique architectural solution for temporal relation extraction by synergizing textual information from documents with complementary knowledge graph data. This combination enriches the context and accuracy of temporal relations. Directly translating text to a timeline in clinical records may not be preferred due to the inherent complexity and ambiguity in clinical texts. Clinical documents often contain intricate details, nuances, and domain-specific terminology that require contextual understanding from a broader knowledge graph to accurately capture temporal relationships. Our approach leverages this hybrid approach to enhance precision and contextual relevance in temporal relation extraction from clinical records. In our study, we concentrate on extracting temporal relationships from medical papers since the automatic recognition of the order of events might lead to a wide range of further applications. We may use such a technique, for instance, to look for trends in the course of symptoms and treatments and such sequences are helpful for tasks like predicting clinical dead ends and identifying illness progression. We use the clinical discharge summaries from the i2b2 2012 corpus to train and test our model [23] and were able to identify the before, after, and overlap relations that the corpus has captured. Relationships are dealt with as a three-class classification problem.

The remainder of this manuscript is structured as follows. We discuss current models for temporal relation extraction in Sect. 2. We present the suggested architecture for our model in Sect. 3. Then, in Sect. 4, we discuss our early findings, which demonstrate the potential of utilizing clinical narratives for temporal relation extraction. Section 5 concludes by summarizing our study and outlining the future directions for our work.

2 Related Work

Rule-based systems, conventional machine learning systems with specialized classifiers and heuristics, and deep learning systems have all been stages in the evolution of temporal relation extraction approaches [5]. Early attempts to address the clinical relation extraction problem make use of traditional machine learning techniques like SVMs, MaxEnt, and CRFs, and neural network-based methods [3, 4, 12, 13, 15, 24]. They either necessitate expensive feature engineering or neglect to consider the dependencies between temporal relations within a document. To model the dependencies, the problem is formulated as a structured prediction problem [7, 18, 19]; however, these approaches were unable to predict temporal relationships globally which is spanning the entire document. The temporal relations at the document level can instead be inferred using our method inspired from [10]. A notable work on time-line construction from the temporal relations were employed [11], which is future scope for this work.

Recent methods use sophisticated deep neural network-based models that can learn high-level representations for temporal relation extraction in the general domain. These techniques can include sophisticated neural language models, such as BERT [27, 28] as well as graph-based architectures that can extract relations at the document level and capture the overall temporal structure of a text. Modern BERT models and variations of this architecture serve as the foundation for state-of-the-art temporal relation extraction systems in the clinical domain [6]. For instance, models proposed in [7, 18] included temporal interactions and tense information, and other works presented in [22, 25] have suggested using graph neural networks (GNNs) to encode dependency structures, which are crucial for extracting temporal relations. Document-creation-time (DCT) is added an attention layer to an R-GCN-based model [17] and further performance optimization of the model was done by choosing the best sentences to feed into neural models using a reinforcement learning framework [14].

Contextualized embeddings were learned using pretrained language models like BERT [12] and the viability of using LLMs in temporal relation extraction has not yet been studied, though. Recent methods, however, frequently lack flexibility and usability because they tend to concentrate on a single task, like relation classification [8, 13, 29] and they are restricted to creating temporal relations from gold standard entities. Some systems evaluated on the Direct Temporal Relations corpus only address explicit intra-sentence temporal relations [9], for example, while others only address a smaller subset of relations [2, 4]. The viability of using LLMs in temporal relation extraction has not yet been studied, though. This work addresses the need for further development of end-to-end strategies, leveraging various of language model variants for temporal relation extraction tasks, which can also handle text, graph input, in document implicit, and cross-sentence temporal relations with larger dependencies.

3 Methodology

More common information is included and displayed as an event graph with already discovered relationships between events as a categorization problem to enhance the performance of temporal relation extraction. A graph structure known as an event graph is one in which events are represented as nodes and temporal links as directed edges. Then it is possible for us to connect the events in the text to be represented as a graph, in which the existing relations are intended to be used as additional factors as they record data regarding the relationships that are typical among various event types. A pretrained language model and knowledge graph are leveraged to derive two sets of relations based on the event embeddings of the input text. Thus, the model may learn rules that apply to relations that are next to one another, such as transitivity. Following that, the model creates a classifier based on each pair of event embeddings, and then it combines both categories to create a single relation prediction. We utilize EntityBERT [16] for text encoding due to its domain knowledge in the learning process by masking entities as a whole and shows superior results on downstream clinical extraction tasks, such as negation detection, document time relation classification, and temporal relation extraction. The Deep Graph Library (DGL) guidelines were used in the construction of our Temporal relations extraction model and are approached as a link prediction problem by classifying whether two nodes are connected by an edge or not. We utilize Relational Graph Convolutional Network (R-GCN) [22] for temporal relation prediction as it allows representation of multiple relations along edges.

In our study, temporal relations are predicted as links using a parameterized score function to reconstruct an edge using an autoencoder architecture. Our Temporal Relational Graph Convolutional Network (TR-GCN) aggregates incoming messages and generates new node representations for each node, calculating outgoing messages for each node using the node representation and an edge type-specific weight matrix. A two-layered Temporal Relation Graph Convolutional Network (TR-GCN) allows the representation of multiple relations along edges by encoding the graph input that is optimized for distinguishing temporal links between events. The first TR-GCN layer served as the input layer followed by projected features (BERT embeddings) into hidden space. Negative sampling methodology is used to compare the scores of nodes connected by edges to the scores of any two random pairs of nodes under the assumption that nodes connected by edges will receive a higher score than nodes that are not connected. To achieve this, a negative graph during the training loop is created, which contained ‘n’ negative examples of each positive edge and used a pairwise dot product predictor to compute the dot product between the node embeddings and to calculate the relevance score between edges. Nodes connected by an edge should have a higher score than nodes that are not connected. In Fig. 2, we display the proposed event-driven knowledge graph (KG) based TR-GCN architecture for temporal relationship extraction. A significant benefit is that we were able to classify using just one type of input for instance, with either text or knowledge graph our architecture still can capture and categorize temporal relations.

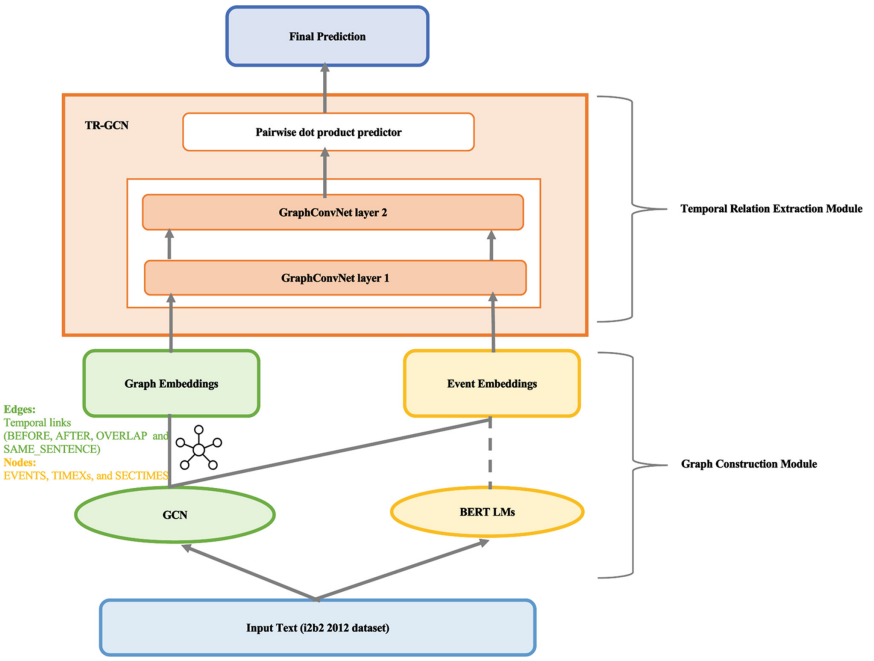


Fig. 2. Event-driven KG Based TR-GCN architecture for temporal relation extraction

3.1 KG Construction: Event Graph

We formulate it as a KG completion task by employing the event graph in the process of temporal relation extraction. To enhance the performance of temporal relation extraction, we introduce domain knowledge into the classification task through the knowledge presented in the form of an event graph that contains already-extracted links between events. The event graph is a graph structure where the temporal interactions are represented as directed edges in the graph and the events are represented as nodes. We catalogue each event using its Unified Medical Language System (UMLS) meta-thesaurus identification, allowing us to connect the events in the text with those present in the graph. Existing relations are intended to be used as extra information since they capture information about the relationships that distinguish certain kinds of events from one another. Thus, the existing relations enable the model to learn rules that are held between neighboring relations, such as transitivity. The most significant events and relations for these extracted relations are those that were taken from the same clinical record.

As a proof of concept, the event graphs are constructed from all relations present in the clinical narrative except for the relations that we are extracting. Although it is implausible for actual application, this type of knowledge graph replicates the ideal situation in which we already have access to all other relations in the clinical document. But in practice, the relations might be gathered by first predicting them with the text model and then constructing the event graph and as a result, it is possible that some of the relations in the event graph may contain inaccuracies, which will negatively impact the outcomes. Nine different types of links between clinical EVENTs, temporal expressions

(TIMEXs), and SECTIMEs (the patient's arrival and departure times) are present in the i2b2 dataset. We use the BEFORE, AFTER, and OVERLAP links from 310 discharge summaries annotated with temporal information.

We used the Python library Deep Graph Library (DGL) [26] to convert each discharge summary into a graph, where EVENTs, TIMEXs, and SECTIMEs serve as the nodes and the BEFORE, AFTER, and OVERLAP links serve as the edges. The SAME_SENTENCE relationship connected all the nodes that were in the same sentence in the raw clinical narratives/discharge summaries. This fourth type of link, which was not included in the i2b2, was added because we thought it would improve the model's prediction accuracy. By doing this, we made sure that when making predictions, we had a relationship that could be automatically added to fresh graphs that still lacked the BEFORE, AFTER and OVERLAP relationships. Graph design is streamlined by giving each node a single type (an "entity") and used node features to store information about their actual types (EVENT, TIMEX, and SECTIME) in a one-hot-encoded vector. Because our graph had three different types of edges in addition to only having one type of node, it was heterogeneous. Then we added BERT contextual embeddings from each token of the plain text reports to our nodes to preserve the contextual information from the raw clinical discharge summary. For multi-token entities, the context is preserved using the mean of the embeddings with the token's components. Eventually, our nodes attributes had 789 dimensions: 786 dimensions from the BERT embeddings vector and 3 dimensions from the one-hot-encoded vector, which represents the entity type.

3.2 Text and Graph Embeddings

We leverage EntityBERT for calculating token embeddings with designated unique tokens in the beginning and end of each event in the text input. The model training aims at identifying entities and events. The event embeddings are created by averaging each token associated with an event as a single embedding which includes details about the event and how it's related to the text. Embedding of both events is integrated by joining two vectors together as one, in which the resultant vector is sent via a graph convolutional layer that categorizes the vector into one of 3 categories of temporal connection. The temporal relation is categorized by using the KG constructed through a graph neural network and graph embeddings are generated. Events and temporal links around the events are constructed as part of KG as nodes and edges and begin each event's embedding with GloVe embedding by aggregating global token-to-token co-occurrence [21]. Three-layer graph convolution is used to combine event embeddings across temporal relations and each event thereafter comprises data about its associated events and the pertinent relations. The observed events' embeddings are combined and then transmitted via two graph convolution layers to carry out the categorization.

3.3 Language Modelling with Masks (LMM) and Prediction

We wanted to see if the quality of the BERT embeddings used as attributes changed after performing Language Model Masking (LMM) on the original models in addition

to testing their effectiveness by contrasting the embeddings of six models. The four models namely ClinicalBERT[1], PubMedBERT[2], BlueBERT[3] and SciBERT[4] were obtained straight from Hugging Face Models Hub[5] which were initially trained on PubMed articles, clinical narratives, MIMIC notes, and electronic health records. Unlike the PubMedBERT (PMB) model, all other models use embedding with a dimension of 1024 rather than 789. The effectiveness of our strategy is demonstrated by conducting an LMM on the PubMedBERT model. This led us to run the PubMedBERT_LMM on Google Colaboratory Notebook (PMB-LMM-GC) and on our servers (PMB-LMM-Serv). We masked 20% of the tokens in the i2b2 dataset for the LMM task and employed the AdamW optimizer with a batch size of 64 and a learning rate of 5e−5. All the models other than PubMedBERT showed decreased performance in the metrics after 2 epochs, hence we only trained our models for PubMedBERT.

We divided our 310 graphs from i2b2 2012 into a train set (80%) and a test set (10%) before training our model and the rest (10%) of the data as the validation set. The model's two primary components are built in such a way to predict the temporal relations and categorize it for text and graph input. We initially concatenate the two prediction vectors, after which we run them through two graph convolution layers that calculate the combined prediction where the model learns to trust one portion of the model over another. The model is trained in three stages. First, by simply using one portion of the network to identify temporal connections, we independently train the text and graph components of the model. This enables us to precisely adjust the training settings for each component separately. We continue training the network using the complete model when both network segments have been trained. A batch size of 20 graphs is batched together to prevent memory issues during the training phase and each input graph is treated as a separate and distinct component of the batched graph in DGL.

Iteratively the negative graph is built during the training loop and calculated the margin loss. We also changed the original negative graph function so that each subgraph only received negative examples from its own subgraph because the DGL guidelines do not use batched graphs. The DGL allows us to predict one type of relationship at a time, and hence we separately trained our model to predict BEFORE, AFTER, and OVERLAP links. Our goal is to compare 6 different types of embeddings, and we have set some hyperparameters to establish the baseline comparison. Five negative samples are generated for each positive edge, and we used hidden and out dimensions of 1280 for BlueBERT, SciBERT, and ClinicalBERT models, and 1024 for the PubMedBERT model. We trained the model in Google collaboratory notebook for 50 epochs with a training batch size of 20 graphs and later optimized some of the hyperparameters for the selected model where we presented the captured metrics in Table 3.

[1] emilyalsentzer/Bio_ClinicalBERT.

[2] microsoft/BiomedNLP-PubMedBERT-base-uncased-abstract-fulltext.

[3] bionlp/bluebert_pubmed_uncased_L-24_H-1024_A-16.

[4] allenai/scibert_scivocab_uncased.

[5] Models - Hugging Face

4 Results and Discussion

We tested our proposed model with the above-mentioned train, test, and validation splits on the i2b2 2012 temporal relations dataset and utilized the test split to assess the model's performance. On the initial set of analyses in forecasting the temporal connections through text modality and knowledge graph, we demonstrate the model's performance with and without knowledge graph input. This outcome informs us of the accuracy of the relations as a problem of knowledge graph completion. Finally, we test the complete model while simultaneously employing both modalities demonstrating the effectiveness of combining them for more accurate and reliable prediction. We also compared the F1 metric of our methodology with some of the baseline models [20, 29].

We also used BERT model [29] with two variants one with single layer for classification and another with soft logic regularizer as a baseline Comparison to our work in which Table 1 shows the outcome of incorporating the event graph in addition to the text alone greatly enhanced the model's performance (14% improvement). Table 2 shows the state-of-the-art performance and other models namely ELMo, BERT base and large, BioALBERT and two other BERT variants were used to compare the effectiveness of our proposed model on the i2b2 corpus for the temporal relation extraction task.

Table 1. Metrics for different input variants on our model for temporal relation extraction on i2b2 2012 test set

Input Scenario	Precision	Recall	F-score
Text + KG	89.6	82.0	85.7
KG	84.0	80.3	82.1
Text	86.4	76.6	81.2

Nevertheless, it is crucial to underscore that the model's achievement was contingent upon utilizing an event graph encompassing all document relations except for the target relations - a scenario that deviates from practical real-world applications. The model would not typically have access to any relations from the observed document in real-world applications. To support such a scenario, we first create the event graph using the text model, and then utilize the created graph to conduct the relation extraction. The obtained results hold promise for further investigation into the use of such temporal event graphs to facilitate the extraction of temporal relations. And to predict the missing links from new graphs, our method is based on training a GNN on the graphs created from the i2b2 dataset.

To improve model performance during training, we used the model margin loss, and this loss is the total number of incorrect predictions made during the training. We used AUC (Area Under the ROC Curve) to assess how well our model performed on the test set (which consisted of 10% of the graphs). AUC is appropriate for link prediction tasks because it provides the likelihood that a positive example will receive a higher score than a negative one. The loss of our six models decreased during training after 50 epochs of batch training, and the evaluation AUC for all three kinds of links gradually

increased in the test set. For BEFORE, AFTER, and OVERLAP relations AUC reached what we think are remarkable levels after 50 epochs. When comparing the two original models, as shown in Table 3, all BERT language models' loss decreased slightly while achieving a slightly higher AUC than PubMedBERT. We believe that this is because its embeddings are larger in size. Additionally, when comparing the performance of the original PubMedBERT model with that of our LMM strategy, the latter two enhanced both the train Loss and the eval AUC of the first one. Notably, out of the six models, the one trained on our server (Nvidia RTX 3090 24 GB) achieved the best Loss and AUC values for both types of relationships. Furthermore, only this final model exceeded 97%, 96%, and 87% of eval AUC for the BEFORE, AFTER and OVERLAP links, respectively.

Table 2. Comparison of Baseline vs our model's performance on i2b2 Clinical Narratives for Temporal Relation Extraction

Models	F1-Score
Ours	
Text input-based temporal relation prediction model	81.2%
Graph input-based temporal relation prediction model	82.1%
Text + Graph based combined prediction model	85.7%
Baseline models	
SOTA [7]	73.7%
ELMo [7]	71.2%
BERT (base) [7]	76.4%
BERT (large) [7]	73.9%
BioALBERT [7]	76.86%
BERT-Linear layer classifier [8]	78.6%
BERT-Linear layer with soft logic regularizer [8]	80.2%

We believe that the findings presented in the preceding section demonstrate both of our hypotheses. First and foremost, it has been demonstrated that Graph Neural Networks and BERT embeddings work well together and produce impressive results when it comes to the prediction of temporal relationships in the clinical domain, which opens numerous avenues for further research in this area. Second, after performing a Masked Language Modelling on the original models, the performance of BERT embeddings improved. Since the OVERLAP relationship continues to present one of the greatest challenges in predicting temporal relationships today, it is a much more laborious prediction than the BEFORE and AFTER relationship, which is obviously more related to the linear nature of time in the text and therefore easier to predict. This seems to be the reasoning behind the significant difference between the results of BEFORE, AFTER, and OVERLAP.

The main issue with our analysis is that, for two main reasons, we are unable to draw a direct comparison with earlier work on the prediction of temporal relationships

from the i2b2 dataset. First, since it was a preliminary study, we only looked at three of the eight relationships identified in the dataset because we thought they were the most fundamental and a good place to start. Second, because the majority of edges in graph data are negative, metrics used in earlier studies of temporal link prediction (accuracy, recall, and F-score) may include noise when predicting links, so DGL suggests using AUC to assess these models. Despite this, we believe that the model's performance, both during training and evaluation, is exceptional. We chose two of eight relationship types for our research due to their fundamental significance and manageable scope. Additionally, the use of AUC as an evaluation metric aligns with the dataset's predominant negative edges.

Table 3. Training Loss and Evaluation AUC

Temporal Relations	Before		After		Overlap	
Model\Metrics	Train loss	Eval AUC	Train loss	Eval AUC	Train loss	Eval AUC
PubMedBERT	1.00	0.967	1.19	0.949	2.28	0.860
MaskedLM_PubMedBERT_GC	1.03	0.969	1.18	0.951	2.28	0.861
MaskedLM_PubMedBERT_Serv	0.83	0.969	1.09	0.962	2.19	0.872
ClinicalBERT	1.67	0.960	1.20	0.936	2.30	0.846
BlueBERT	1.13	0.965	1.00	0.942	2.11	0.852
SciBERT	1.24	0.963	1.08	0.944	2.23	0.849

We are optimistic about the possibilities for the future because the model's continuous improvement during the training loop and the impressive results in the evaluation graphs provide a great area for further research. This model uses embeddings from PubMedBERT after performing an LMM on the i2b2 dataset in our server. We intend to select the most effective model, currently appearing to be the PubMedBERT_LMM_Serv one to continue optimising it and combine it with our own medical NER system as a first step to creating the timeline from any given clinical record.

5 Conclusion

We introduce a novel architecture by extrapolating temporal relationships from text, utilising the nature of graphs, and further leverage the power of BERT embeddings adapted to the clinical domain which offer a great potential when working with temporal relations. Our preliminary experiments show that the proposed architecture greatly outperforms the baseline models which is because we use information present in text and information about other relations captured in a knowledge graph. The current limitation of the proposed approach is that it relies on a knowledge graph to contain correct relations between events. In real-world scenarios, the relations would likely contain errors, as they would come from previously extracted information. When addressing longevity, this earlier examination of clinical narratives' temporality can be incredibly helpful. Our future

scope is to improve the real-world usability of the proposed architecture and assessed in more scenarios as part of our ongoing study. A patient's medical record can be used to create a timeline of their history, which can be used to predict both their future and their past. When discussing multiple patients, this benefit becomes more apparent, and having a large collection of clinical texts with their corresponding illnesses, cures, and side effects all temporally ordered can aid in forecasting and, consequently, encourage the survival of new patients.

Acknowledgement. The authors acknowledge the AIDAVA project financed by Horizon Europe: EU HORIZON-HLTH-2021-TOOL-06-03 and ANTIDOTE project financed by CHIST-ERA and FWO.

Availability of data: https://www.i2b2.org/NLP/DataSets/Main.php.

References

1. Alfattni, G., Peek, N., Nenadic, G.: Extraction of temporal relations from clinical free text: a systematic review of current approaches. J. Biomed. Inform. **108**(2020), 103488 (2020). https://doi.org/10.1016/j.jbi.2020.103488
2. Alfattni, G., Peek, N., Nenadic, G.: Attention-based bidirectional long short-term memory networks for extracting temporal relationships from clinical discharge summaries. J. Biomed. Inform. **123**(2021), 103915 (2021). https://doi.org/10.1016/j.jbi.2021.103915
3. Galvan, D., Okazaki, N., Matsuda, K., Inui, K.: Investigating the challenges of temporal relation extraction from clinical text. In: Lavelli, A., Minard, A.-L., Rinaldi, F. (eds.) Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis, Louhi@EMNLP 2018, Brussels, Belgium, 31 October 2018, pp. 55–64. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/w18-5607
4. Guan, H., Li, J., Xu, H., Devarakonda, M.V.: Robustly pre-trained neural model for direct temporal relation extraction. In: 9th IEEE International Conference on Healthcare Informatics, ICHI 2021, Victoria, BC, Canada, 9–12 August 2021, pp. 501–502. IEEE (2021). https://doi.org/10.1109/ICHI52183.2021.00090
5. Gumiel, Y.B., et al.: Temporal relation extraction in clinical texts: a systematic review. ACM Comput. Surv. **54**(7), 144:1–144:36 (2022). https://doi.org/10.1145/3462475
6. Han, R., Hsu, I.-H., Yang, M., Galstyan, A., Weischedel, R.M., Peng, N.: Deep structured neural network for event temporal relation extraction. In: Bansal, M., Villavicencio, A. (eds.) Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, 3–4 November 2019, pp. 666–106. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/K19-1062
7. Han, R., Ning, Q., Peng, N.: Joint event and temporal relation extraction with shared representations and structured prediction. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019, pp. 434–444. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/D19-1041
8. Ul Haq, H., Kocaman, V., Talby, D.: Deeper clinical document understanding using relation extraction. CoRR abs/2112.13259 (2021). arXiv:2112.13259
9. Lee, H.-J., Zhang, Y., Jiang, M., Xu, J., Tao, C., Xu, H.: Identifying direct temporal relations between time and events from clinical notes. BMC Med. Inform. Decis. Mak. **18**(S-2), 23–34 (2018). https://doi.org/10.1186/s12911-018-0627-5

10. Leeuwenberg, A., Moens, M.-F.: Structured learning for temporal relation extraction from clinical records. In: Lapata, M., Blunsom, P., Koller, A. (eds.) Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Volume 1: Long Papers, Valencia, Spain, 3–7 April 2017, pp. 1150–1158. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/e17-1108
11. Leeuwenberg, A., Moens, M.-F.: Temporal information extraction by predicting relative timelines. In: Riloff, E., Chiang, D., Hockenmaier, J., Tsujii, J. (eds.) Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018, pp. 1237–1246. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/d18-1155
12. Lin, C., Miller, T., Dligach, D., Bethard, S., Savova, G.: A BERT-based universal model for both within-and cross-sentence clinical temporal relation extraction. In: Proceedings of the 2nd Clinical Natural Language Processing Workshop, pp. 65–71 (2019)
13. Lin, C., Miller, T., Dligach, D., Sadeque, F., Bethard, S., Savova, G.: A BERT-based one-pass multi-task model for clinical temporal relation extraction (2020)
14. Lin, C., Miller, T.A., Dligach, D., Amiri, H., Bethard, S., Savova, G.: Self-training improves recurrent neural networks performance for temporal relation extraction. In: Lavelli, A., Minard, A.-L., Rinaldi, F. (eds.) Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis, Louhi@EMNLP 2018, Brussels, Belgium, 31 October 2018, pp. 165–176. Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/w18-5619
15. Lin, C., Miller, T.A., Dligach, D., Bethard, S., Savova, G.: Representations of time expressions for temporal relation extraction with convolutional neural networks. In: Cohen, K.B., Demner-Fushman, D., Ananiadou, S., Tsujii, J. (eds.) BioNLP 2017, Vancouver, Canada, 4 August 2017, pp. 322–327. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/W17-2341
16. Lin, C., Miller, T.A., Dligach, D., Bethard, S., Savova, G.: EntityBERT: entity-centric masking strategy for model pretraining for the clinical domain. In: Demner-Fushman, D., Cohen, K.B., Ananiadou, S., Tsujii, J. (eds.) Proceedings of the 20th Workshop on Biomedical Language Processing, BioNLP@NAACL-HLT 2021, Online, 11 June 2021, pp. 191–201. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.bionlp-1.21
17. Man, H., Ngo, N.T., Van, L.N., Nguyen, T.H.: Selecting optimal context sentences for event-event relation extraction. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022, Virtual Event, 22 February–1 March 2022, pp. 11058–11066. AAAI Press (2022). https://ojs.aaai.org/index.php/AAAI/article/view/21354
18. Mathur, P., Jain, R., Dernoncourt, F., Morariu, V.I., Tran, Q.H., Manocha, D.: TIMERS: document-level temporal relation extraction. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, Volume 2: Short Papers, Virtual Event, 1–6 August 2021, pp. 524–533. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.acl-short.67
19. Ning, Q., Feng, Z., Roth, D.: A structured learning approach to temporal relation extraction. In: Palmer, M., Hwa, R., Riedel, S. (eds.) Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, 9–11 September 2017, pp. 1027–1037. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/d17-1108

20. Peng, Y., Yan, S., Lu, Z.: Transfer learning in biomedical natural language processing: an evaluation of BERT and ELMo on ten benchmarking datasets. In: Demner-Fushman, D., Cohen, K.B., Ananiadou, S., Tsujii, J. (eds.) Proceedings of the 18th BioNLP Workshop and Shared Task, BioNLP@ACL 2019, Florence, Italy, 1 August 2019, pp. 58–65. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/w19-5006
21. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Moschitti, A., Pang, B., Daelemans, W., (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, A meeting of SIGDAT, a Special Interest Group of the ACL, 25–29 October 2014, Doha, Qatar, pp. 1532–1543. ACL (2014). https://doi.org/10.3115/v1/d14-1162
22. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
23. Sun, W., Rumshisky, A., Uzuner, O.: Evaluating temporal relations in clinical text: 2012 i2b2 challenge. J. Am. Med. Inf. Assoc. **20**(5), 806–813 (2013). https://doi.org/10.1136/amiajnl-2013-001628
24. Tourille, J., Ferret, O., Névéol, A., Tannier, X.: Neural architecture for temporal relation extraction: a Bi-LSTM approach for detecting narrative containers. In: Barzilay, R., Kan, M.-Y. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 2: Short Papers, Vancouver, Canada, 30 July–4 August, pp. 224–230. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/P17-2035
25. Wang, L., Li, P., Xu, S.: DCT-centered temporal relation extraction. In: Calzolari, N., et al. (eds.) Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, 12–17 October 2022, pp. 2087–2097. International Committee on Computational Linguistics (2022). https://aclanthology.org/2022.coling-1.182
26. Wang, M., et al.: Deep graph library: a graph-centric, highly-performant package for graph neural networks. arXiv preprint arXiv:1909.01315 (2019)
27. Zhang, S., Ning, Q., Huang, L.: Extracting temporal event relation with syntax-guided graph transformer. In: Carpuat, M., de Marneffe, M.-C., Ruíz, I.V.M. (eds.) Findings of the Association for Computational Linguistics, NAACL 2022, Seattle, WA, United States, 10–15 July 2022, pp. 379–390. Association for Computational Linguistics (2022). https://doi.org/10.18653/v1/2022.findings-naacl.29
28. Zhao, X., Lin, S.-T., Durrett, G.: Effective distant supervision for temporal relation extraction. CoRR abs/2010.12755. arXiv arXiv:2010.12755 (2020)
29. Zhou, Y., et al.: Clinical temporal relation extraction with probabilistic soft logic regularization and global inference. In: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, 2–9 February 2021, pp. 14647–14655. AAAI Press (2021). https://ojs.aaai.org/index.php/AAAI/article/view/177212023-09-1109:46

Machine Learning and Networks

Efficient Approach for Patient Monitoring: ML-Enabled Framework with Smart Connected Systems

G. Dheepak(✉)

Department of Electronics and Communication Engineering, Puducherry Technological University, Pondicherry, Puducherry, India
dheepakyadav2411@pec.edu

Abstract. Healthcare professionals are required to adhere to strict precautions. Thus, continuous monitoring, compliance with safety standards, and prompt care are essential. Hospitals employ various methods and devices to ensure the patient's well-being. However, many of these efforts are compromised when the observation is neglected. This paper presents an effective approach to enable doctors (or) supervisors to track and forecast the patient's health condition during the treatment. The approach leverages the concept of Internet of Things (IoT) for seamless data acquisition, data analytics for visualization, and machine learning (ML) to train models with the acquired data for future prediction of similar conditions. For monitoring, sensors are used to collect data such as the ambient state and the location to verify that the patient (or) person under observation is within the expected range using range detection techniques supported by Bluetooth master-slave communication. Computations are performed in the backend such that the alerts are notified based on the conditions assigned to the respective patients. In case of emergency, we can reliably predict the condition of a patient with improved accuracy. Moreover, the stored data is fed to an ML framework for data training and ML modeling. Therefore, this design can serve as an optimal model to address the much-needed advancement in healthcare.

Keywords: IoT · Machine Learning · Wireless Communication · Bluetooth · Range Detection

1 Introduction

Patients are subjected to adverse conditions due to insufficient/inefficient observation methods in hospitals, which may cause their treatment to deteriorate or even endanger their lives. Medical negligence is one of the most alarming causes of patient death. A survey by Nursing Times found that one in five nurses "rarely" or "never" monitored their wards [1, 2]. Human errors lead to about 5.2 million deaths in India every year. In the US, about 44,000 to 98,000 people are affected. This is not due to the lack of medical skill or knowledge of doctors, but rather the lack of team coordination, observation strategies and communication [3]. Based on this problem statement, we identify three cases and

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 177–188, 2024.
https://doi.org/10.1007/978-3-031-53468-3_15

propose a solution. The solution is motivated by observing the multiple challenges faced by patients. First, due to inadequate observation by doctors, nurses or workers, there is a delay in checking each patient, which poses a serious risk to their lives in hospitals. In domestic scenarios, e.g. at home, elderly people and infants need regular supervision. Second, in recent years, we notice many patients disappearing from hospitals when they are under treatment, due to the misunderstanding of their disease or condition. On the other hand, special and constant care is required for mentally unstable patients, as they tend to escape from their wards [4]. Also, in hospitals, we witness infants being kidnapped, and this trend is consistent [5]. Moreover, studies show that there is a huge impact due to the lack of pre-analysis of health by patients and doctors. Therefore, this paper proposes an efficient way to overcome the limitations of the current approach by developing a connected smart system for children, senior citizens, especially differently-abled individuals, or anyone who needs supervision, such that the concerned person is notified regularly without any service interruption and the patient data is processed and real-time prediction is performed. A unique feature of this proposal is the range detection technique, where the patient is given a limit for movement and beyond which the system makes an alert call. This is achieved by a bluetooth master-slave system to warn the concerned person if the person under observation crosses a specified boundary. The solution is based on the field of IoT, Data Analytics, and Machine Learning and the scope is feasible in the domain of smart home automation, security and data analytics. This paper is structured as follows. The introduction section is followed by a brief overview of the related research works in Sect. 2. Categorical approach for classification of cases and design methodology in Sect. 3. Complete hardware implementations in Sect. 4, followed by software implementation in Sect. 5. Section 6 discusses the mechanism for ML model and data analysis, which covers various models used for training and prediction. Section 7 illustrates the entire workflow explained in the proposal, followed by system deployment & testing in Sect. 8 and Conclusion in Sect. 9.

2 Literature Study and Related Work

The field of remote patient monitoring is rapidly evolving with various applications and methods emerging in domains such as healthcare [6], education [7], agriculture [8, 9], wearable industry [10], etc. Several studies focus on the analysis of chronic diseases [11, 12], which requires continuous observation. Zanaj, E. et al. [13] suggested a method that uses Wireless Sensor Networks (WSNs) to transfer various biometric data such as heart rate, body temperature, SpO2, respiration, ECG for distant monitoring and classification. Wang, P. [14] developed a real-time monitoring system for cardiac in-patients using zigbee as data acquisition device that sends the collected data to a database and evaluates the patient by fuzzy reasoning and this proposal is a distance bound mechanism. Siddik, A.B. et al. [15] demonstrated the use of cloud computing with visualization of the collected data and incorporated a GSM module for notifying the relevant person. Mansfield, S et al. [16] proposed an IoT based system for autonomous patient monitoring focused on pressure injury monitoring and prevention. A Kavak, A. and ˙Inner [17] proposed end to end remote patient monitoring using a framework for data collection and visualization focused on diabetic patients with doctor centric

decision support mechanism. Feng, M et al. [18] proposed an integrated and intelligent system, iSyNCC, to monitor patients and facilitate clinical decision making in Neuro Intensive/Critical Care Units (NICUs). Anifah, L [19] designed a framework that stores data from hardware to backend through MQTT protocol where the data is published and subscribed. Aditya, T.R., et al. [20] proposed a model that uses image processing technique to predict the status of a patient remotely where the system compares, captures and generates alert messages using GSM module while monitoring the body temperature for any anomaly. Sharma, G.K. and Mahesh [21] provided analysis on ESP32 based IoT system for medical monitoring purpose [23] that integrates software and hardware and uses the internet for data transmission and further presented the analysis of percentage of error.

3 Design and Methodology

Our study divides patients into two groups based on their mental and physical state. The first group consists of immobile patients(i) and the second group comprises mobile and mentally unstable patients(ii). The monitoring design provides essential services such as SpO2 (peripheral capillary oxygen saturation) and heart rate monitoring, accelerometer data collection for motion capture, temperature and humidity sensor data collection for monitoring room condition for both groups of patients. Range detection feature is specially provided for mobile and mentally unstable patients(ii), who are likely to exhibit unexpected behaviour. The hardware design has three segments. The prototype with the patient under observation (α) is used for data acquisition using ESP32/Raspberry Pi, The prototype is placed by the doctor within the boundary (near the patient) for range detection and motion capture (β), A monitoring device which will be PC/iOS/Android application (γ). The wearable prototype has SpO2 and Heart rate sensor for health data collection. Bluetooth in prototype (α) and (β) are connected. This pairs module (β) with (α) module for data transfer. Accelerometer module for motion capture, temperature and humidity sensor for monitoring room condition and bluetooth are connected with (β) module. Both the split prototypes are connected to the internet and the information is seamlessly received by the user (γ). The working happens by receiving data collected by the sensors and is sent to database pipeline processing and for alerting doctors/concerned observers. For boundary surveillance, we use bluetooth signals as a key by measuring the distance between the two modules prototype (α) and (β) with signal strength as parameter of analysis [22]. As shown in Fig. 1, random bits are generated in both ends of the device and transmitted with a regular time interval. When these two prototypes separate from each other by a large distance, the communication fails as it starts receiving erroneous bits on one prototype. If erroneous bits are received, the algorithm triggers the alter mechanism and the user is notified of the emergency. Thus, prediction of proximity/location comes under surveillance. The flow of communication is as follows: the master bluetooth generates 8 bytes randomly and transmits them to the slave device, If the data is successfully received the receiver sends the acknowledgment message with the copy of the received data as a checksum to the transmitter (master), The master verifies the received checksum and sends data, if these data match, there is a confirmation that no data is lost between communication so positive acknowledgment is received by

the user and no alert is needed. If the received sum mismatch occurs, it indicates that data is lost in communication so negative acknowledgment is received by the user, and as a loop the master then initiates the next data transmission.

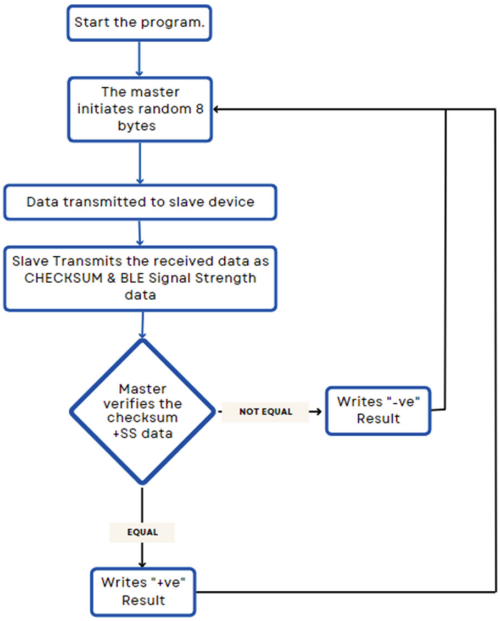


Fig. 1. Flow Chart: Range Detection Mechanism

4 Hardware Design of the System

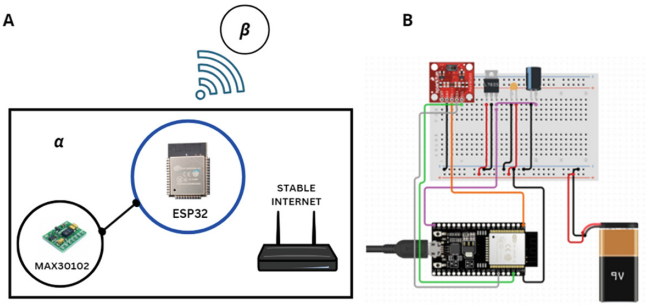


Fig. 2. Prototype with patient (A) Schematic (B) Hardware Implementation

The design in Fig. 2 shows the prototype (α) that is attached to the wrist of the patient/Person under supervision, this records the data for essential health data collection and observation. This serves as the master and is linked with the slave device i.e., the Prototype placed by the supervisor (β) in Fig. 3.

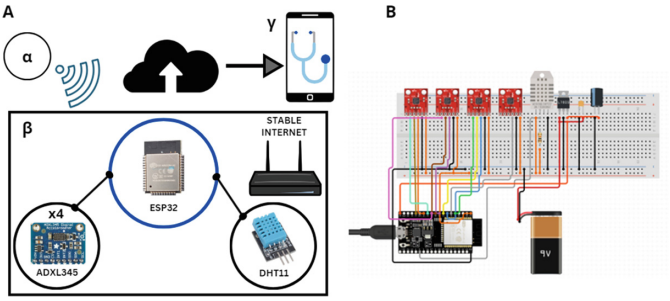


Fig. 3. Prototype placed by supervisor (A) Schematic (B) Hardware Implementation

The schematic and hardware implementation of prototype is placed by the Doctor (β) is shown in Fig. 3. This device gathers the data for Ambience and movement prediction and sends confirmation to the master. Two parts are connected to the internet so that the data is stored for monitoring, alerting, and processing.

5 Software and Mobile Application Design

5.1 Mobile Application

The application has three features as shown in Fig. 4, Main service consists of proximity, SpO2, and heart rate. The second shows ward conditions which gives the real-time Room Humidity in g-m^3 and Room Temperature in Celsius ($°C$) and Fahrenheit(F). Third feature has the parameters to track the movement of patients with axes X, Y, and Z in relation to gravity constant (g).

Fig. 4. Application User Interface (UI)

5.2 Software and Backend Host

The IoT backend is hosted on firebase. As shown in Fig. 5, The data from the sensors are sent to Real-time Database for receiving and transmitting data and updates to application and ML training and deployment.

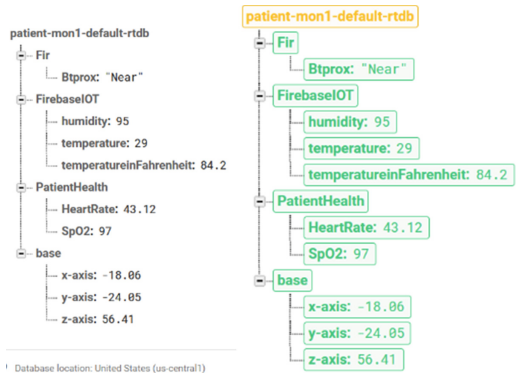


Fig. 5. Real-Time Data Base (RTDB)

6 Data Analysis Framework and Machine Learning Prediction

The patients' data is stored in real-time database, (SQL/Firebase) for analysis and sent to ML containers for post-processing. Data wired to ML Engine/Amazon sage maker where the container is hosted with serverless templates and the process is shown in Fig. 6. Amazon Lambda Production ML Container is a cost-effective, scalable, and reliable way for data scientists to deploy CPU-based machine learning models for inference. The data collected from the database is moved to ML container for training via REST API. The predictions for the wired data are trained, tested and presented to doctors using 3 models, Linear Regression (LR), Decision Tree (DT) and K-NN (K-Nearest Neighbours) Algorithm.

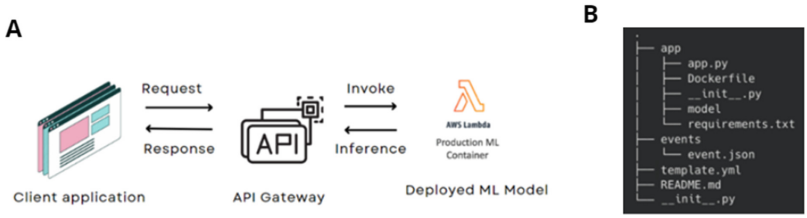


Fig. 6. Post processing scheme (A) ML Model Deployment (B) Node Structure

6.1 Linear Regression (LR)

LR is used in this approach for prediction as this model is extensively used in practical applications which rely linearly on their unknown parameters and are simpler to fit than models which are non-linearly related to their parameters. Also, because of the resulting estimators, the statistical properties are easier to determine. The equation takes the form

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathrm{x}_i^{\mathrm{T}}\beta + \varepsilon_i,\ i = 1, \ldots, n$$

The relationship between the dependent variable y and the vector of regressors x is linear. So, the equation can be simplified as follows,

$$\mathrm{y} = \mathrm{X}\beta + \varepsilon,$$

where

$$\mathrm{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathrm{X} = \begin{bmatrix} \mathrm{x}_1^{\mathrm{T}} \\ \mathrm{x}_2^{\mathrm{T}} \\ \vdots \\ \mathrm{x}_n^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} 1\ x_{11} \ldots x_{1p} \\ 1\ x_{21} \ldots x_{2p} \\ \vdots\ \vdots\ \ddots\ \vdots \\ 1\ x_{n1} \ldots x_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

where $^{\mathrm{T}}$ denotes the transpose, so that $\mathrm{x_i}^{\mathrm{T}}\beta$ is the inner product between vectors $\mathrm{x_i}$ and β. Generally, This relationship is modeled with a disturbance term or error variable ε, an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors.

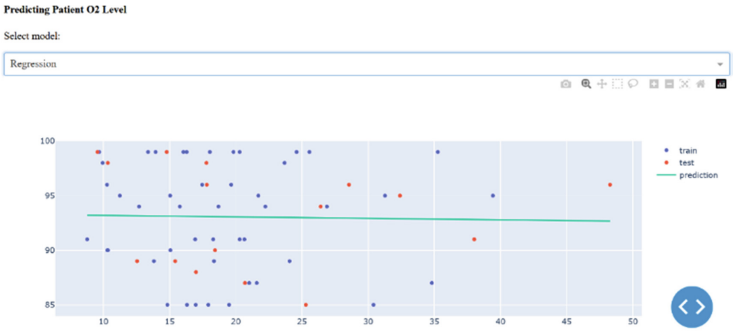


Fig. 7. Linear Regression Prediction [Front-End]

With training accuracy of 88% and the test accuracy of 81%, This provides a preliminary level of understanding of patient's condition. The graphical analysis of realtime and predicted data using Linear Regression displayed in front-end UI of the Web application is shown in Fig. 7.

6.2 Decision Tree Classifier (DT)

Decision trees essentially uses multiple algorithms to decide splitting a node into more sub-node. Increased homogeneity of resulting sub-nodes can be seen with creation of child nodes. The purity of the node increases with the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes which yields the result with maximum score for given objective. The algorithm selection is based on the type of target variables which we focus on. It can be of two types: Continuous Variable Decision Tree: DT that has a continuous target variable, Categorical Variable Decision Tree: DT that has categorical target variable. Here, we use Continuous variable Decision Tree to generate analysis as shown in Fig. 8, This yields training accuracy of 93% and the test accuracy of 85%.

Fig. 8. Decision Tree Prediction

6.3 K-Nearest Neighbour Prediction (KNN)

The KNN algorithm essentially decays down to a majority vote between the K most similar instances to a proposed unseen observation. This is similarity defined according to a distance metric which is between two data points. A popular one is the Euclidean distance metric which is currently used, where x_i, y_i are coordinates in a given plane.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Figure 9 depicts the option for K-NN trained model deployed on Web application which when applied to the collected data generates a trendline which tracks the subsequent status of a patient. With training accuracy of 95% and the test accuracy of 89%, This approach helps satisfy the objective to optimum level. Overall, this expedites the preventive actions that are to be undertaken by doctors. So, in future, Models developed from the collected data can be used to forecast similar condition/ailments, the health condition in the process of receiving treatment, and the survival rate can be obtained which could be used for further research and data modelling.

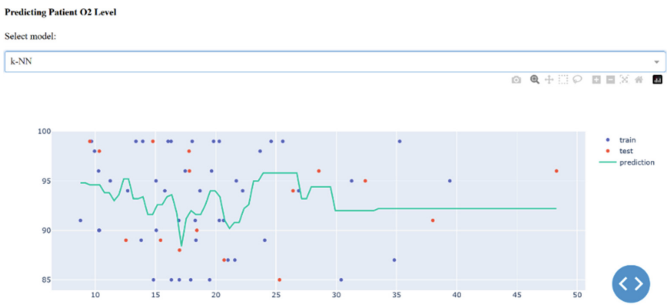


Fig. 9. K-NN Prediction

7 Work Flow

Figure 10 shows the overall workflow of the system. First, data is captured by hardware devices that use bluetooth to measure distance and one device that communicates with the database. The data is then accessed by two clients: ML Framework and Mobile & Web

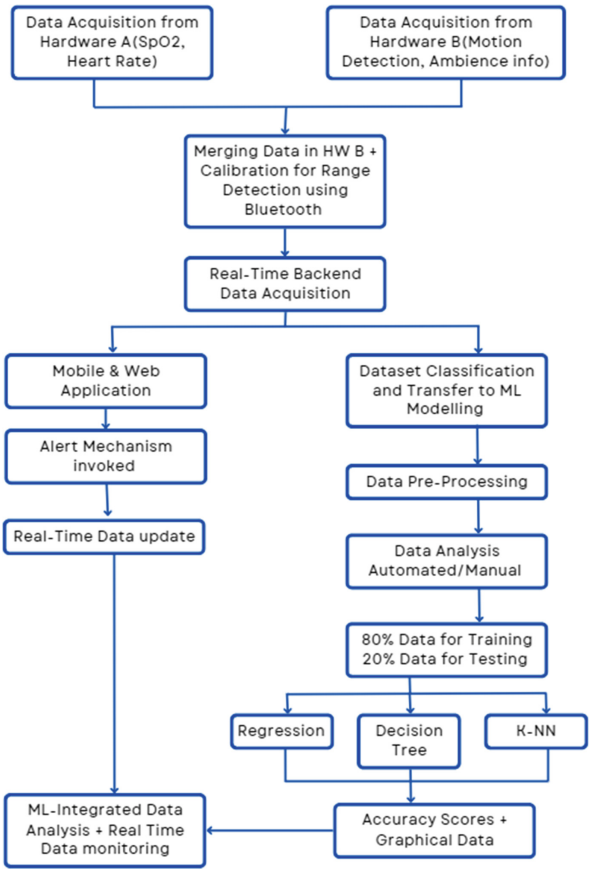


Fig. 10. Flow Chart: Operating Mechanism

Application. The Application monitors the data and sends alerts. The ML framework pre-processes the data for training. The user can also manually examine the data. The Automated mechanism selects the train and test data in a 4:1 ratio for each sample set. The user can choose different algorithms to analyze the data depending on the situation. Doctors are provided with accuracy scores to help them make informed decisions. The graphical data with accuracy is then displayed on the mobile and web application, which offers ML-Integrated data analysis and Real-Time data monitoring.

8 System Deployment and Testing

The prototype was deployed in a domestic environment to evaluate the systems' functionality. The prototype performed remarkably during the deployment and testing phase, obtaining real-time data reliably and transmitting data smoothly with alerts for range monitoring. Moreover, the ML framework supported the overall effort with its accurate prediction, which was demonstrated to be a useful aid in the observation process.

9 Conclusions and Future Research

This paper presented a novel approach to designing and implementing a connected smart system that enables professionals to monitor patients remotely and effectively, using an IoT device with smart real-time predictions based on ML. This system outperforms the existing solutions in terms of accuracy, efficiency, and user-friendliness. As connected smart systems become more prevalent and sophisticated, they offer great potential to address critical problems in various domains, such as healthcare, home automation, and automotive. However, this approach also has some limitations, such as the need for reliable internet connectivity, data security, and user privacy. Therefore, future research should explore how to overcome these challenges and enhance the performance and applicability of the system.

References

1. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6316509/
2. https://www.nursingtimes.net/news/hospital/poor-observation-skills-are-risking-patients-lives-13-10-2009/
3. https://timesofindia.indiatimes.com/life-style/health-fitness/health-news/medical-negligence-70-of-deaths-are-a-result-of-miscommunication/articleshow/51235466.cms
4. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8311958/
5. https://www.missingkids.org/content/dam/missingkids/pdfs/ncmec-analysis/Infant%20Abduction%20Trends_10_10_22.pdf

6. Hu, F., Xie, D., Shen, S.: On the application of the internet of things in the field of medical and health care. In: Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, pp. 2053–2058. IEEE (2013)
7. Gómez, J., Huete, J.F., Hoyos, O., Perez, L., Grigori, D.: Interaction system based on internet of things as support for education. Procedia Comput. Sci. **21**, 132–139 (2013)
8. Sridharani, J., Chowdary, S., Nikhil, K.: Smart farming: the IoT based future agriculture. In 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), January 2022, pp. 150–155. IEEE (2022)
9. Hamdi, M., Rehman, A., Alghamdi, A., Nizamani, M.A., Missen, M.M.S., Memon, M.A.: Internet of Things (IoT) based water ırrigation system. Int. J. Online Biomed. Eng. **17**(5), 69 (2021)
10. Swan, M.: Sensor Mania! the ˙Internet of Things, wearable computing, objective metrics, and the quantified self-2.0. J. Sensor Actuator Netw. **1**(3), 217–253 (2012)
11. Strollo, S.E., Caserotti, P., Ward, R.E., Glynn, N.W., Goodpaster, B.H., Strotmeyer, E.S.: A review of the relationship between leg power and selected chronic disease in older adults. J. Nutr. Health Aging **19**(2), 240–248 (2015)
12. Diet, nutrition and the prevention of chronic diseases: report of a joint WHO/FAO expert consultation. In: WHO Technical Report Series, 916(i-viii) (2003)
13. Zanaj, E., Basha, G., Biberaj, A., Balliu, L.: An intelligent wireless monitoring system in telemedicine using IOT technology. In: 2023 10th International Conference on Modern Power Systems (MPS) , June 2023, pp. 1–5. IEEE (2003)
14. Wang, P.: The real-time monitoring system for in-patient based on Zigbee. In: 2008 Second International Symposium on Intelligent Information Technology Application, December 2008, vol. 1, pp. 587–590. IEEE (2008)
15. Siddik, A.B., et al.: Real-time patient monitoring system to reduce medical error with the help of database system. In: 2022 4th ICECTE, December 2022, pp. 1–4. IEEE (2002)
16. Mansfield, S., Vin, E., Obraczka, K.: An IoT-based system for autonomous, continuous, real-time patient monitoring and ıts application to pressure ınjury management. In: 2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS), July 2021, pp. 66–68. IEEE (2021)
17. Kavak, A., Inner, A.B.: ALTHIS: design of an end to end ıntegrated remote patient monitoring system and a case study for diabetic patients. In: 2018 Medical Technologies National Congress (TIPTEKNO) , November 2018, pp. 1–4. IEEE (2018)
18. Feng, M., et al.: iSyNCC: an intelligent system for patient monitoring & clinical decision support in neuro-critical-care. In: 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, August 2011, pp. 6426–6429. IEEE (2011)
19. Anifah, L.: Smart ıntegrated patient monitoring system based Internet of Things. In: 2022 6th ICITISEE, December 2022, pp. 69–74. IEEE (2022)
20. Aditya, T.R., et al.: Real time patient activity monitoring and alert system. In: 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). IEEE (2020)
21. Sharma, G.K., Mahesh, T.R.: A deep analysis of medical monitoring system based on ESP32 IoT system. In: 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), May 2023, pp. 1848–1852. IEEE (2023)

22. Zhou, P., Ling, X.: HCI-based bluetooth master-slave monitoring system design. In: 2010 International Conference on Computational Problem-Solving, ICCP 2010 (2010)
23. Dhasarathan, C., Shanmugam, M., Kumar, M., Tripathi, D., Khapre, S., Shankar, A.: Anomadic multi-agent based privacy metrics for e-health care: a deep learning approach. Multimedia Tools Appl. **83**, 7249–7272 (2023)

Economic and Health Burdens of HIV and COVID-19: Insights from a Survey of Underserved Communities in Semi-Urban and Rural Illinois

John Matta[1(✉)], Koushik Sinha[2], Cameron Woodard[1], Zachary Sappington[1], and John Philbrick[3]

[1] Southern Illinois University Edwardsville, Edwardsville, IL 62026, USA
jmatta@siue.edu
[2] Southern Illinois University, Carbondale, IL 62901, USA
koushik.sinha@cs.siu.edu
[3] Edwardsville, IL, USA

Abstract. This paper presents findings from the "Estimating the Economic and Health Burdens of HIV in Semi-Urban and Rural Illinois" survey conducted in downstate Illinois, USA. The survey targeted hidden and hard-to-reach communities of HIV-positive individuals and their partners. The study utilizes network science techniques, including community detection and visualization, to analyze the social, medical, and economic forces influencing three underserved communities: African Americans, HIV-positive individuals, and those facing worsened economic situations due to COVID-19. The analysis reveals disparities in healthcare access, discrimination, and economic challenges faced by these communities. The paper highlights the value of network analysis in interpreting smaller datasets and calls for further collaborations and research using the freely available survey data and analysis materials.

Keywords: graph inference · clustering · machine learning · HIV · COVID

1 Introduction

The "Estimating the Economic and Health Burdens of HIV in Semi-Urban and Rural Illinois" survey (BOH) encompassed over 200 questions and targeted hidden and hard-to-reach communities in the St. Louis Metro East area of downstate Illinois, USA. While many studies of disease response are conducted in urban areas, the St. Louis Metro East offers an opportunity to examine individual responses to HIV and COVID-19 in a semi-urban and rural context. The sample consisted of 22 respondents, primarily comprising HIV-positive MSM African

J. Philbrick—Public Health Specialist; Retired.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 189–201, 2024.
https://doi.org/10.1007/978-3-031-53468-3_16

Americans, but also including black cisgender women, Hispanics, and transgender participants. The survey encompassed a range of age groups from 18 to 60, providing valuable insights into the health, economic, and community environments experienced by underserved populations in a post-COVID-19 world.

While statistical analysis poses challenges with small datasets, network analysis has demonstrated successful applications even with limited samples [1]. In this study, we employ network science techniques, including community detection and visualization, to perform a preliminary analysis of how social, medical, and economic forces have influenced three underserved and hard-to-reach communities: African Americans, HIV-positive individuals, and those facing worsened economic situations due to the COVID-19 epidemic.

The survey and this study were conducted under the supervision of the Southern Illinois University Edwardsville Institutional Review Board. All survey results and analysis materials are freely available, and we encourage others to collaborate using this data. Code used in the study can be accessed at https://github.com/SIUEComplexNetworksLab/BOHComplexNetworks. Data are available at https://www.openicpsr.org/openicpsr/project/192186/version/V1/view.

2 Related Work

Network science has a rich history of application in studying various aspects related to HIV, starting with the Potterat et al. [17] HIV transmission network of Colorado Springs, CO, dating back to the early stages of the epidemic in 1985. Recent studies have explored HIV from different perspectives, including investigating the interconnectivity between syphilis and HIV transmission networks [3], identifying intersections among different HIV-adjacent communities to determine optimal locations for intervention efforts [4,7], and combining social and genetic data to infer transmission networks more accurately [21].

Network analysis in the context of HIV is not limited to transmission networks. Adams and Light [2] used bibliographic coupling networks to study interdisciplinary research gaps in HIV/AIDS. Online social networks have also been analyzed to explore support systems for people living with HIV [6] and to identify undiagnosed communities for targeted outreach [10].

3 Methods

3.1 The Burden of HIV Dataset

The BOH survey was conducted from late 2021 to April 2023 by the Applied Research Consultants group at Southern Illinois Carbondale and supervised by Drs. Sinha and Matta. Survey questions covered domains including perceptions of discrimination, and the impact of the COVID-19 pandemic on sexual practices, living conditions, employment, and economic well-being. A dedicated section of the survey addressed mental health and self-esteem, assessing factors such as

partner relationships, frequency of contact with family and friends, pride in gay identity, and level of participation in community organizations.

In the context of this study, responses to individual questions are referred to as "attributes," "features," or "variables." Questions that serve as the primary focus of analysis are referred to as "target" variables. The target variables considered in this study are African American race, HIV positive status, and those with economic situations worsened by COVID-19.

3.2 Data Curation

To ensure data integrity, variables with fewer than 19 responses were removed from the dataset. Multi-valued variables were transformed into binary choices using one-hot encoding. For instance, a variable such as "What sources of transportation discrimination have you experienced," with responses including "Discrimination based on race," and"Discrimination based on sexual orientation," would be transformed into two separate variables using one-hot encoding, denoted as "TransportDiscrim:Race," and "TransportDiscrim:SO."

After data curation, the final survey dataset consisted of 274 variables. Participants who did not finish the interview were removed, resulting in 19 samples. The Recursive Feature Elimination (RFE) class from Python's SciKit Learn library was utilized for feature selection. RFE identifies a fixed number of attributes or attribute combinations that contribute the most to predicting the target attribute. Based on previous studies demonstrating improved clustering results with a reduced number of attributes [11,14,18], feature selection was performed to identify the most important sets of 15 and 30 attributes for each target variable.

3.3 Graph Inference

Based on previous studies demonstrating the effectiveness of the k-Nearest Neighbors (kNN) graph inference method in survey data analysis [11,14], the curated dataset was transformed into kNN graphs. The kNN graph inference approach involves calculating distances between each pair of nodes in the graph. Subsequently, for a given number, k, an edge is created between each node and its k nearest neighbors. Based on previous evidence [15] of the computational superiority of sparse graphs, the number of neighbors parameter was 4 for each sample. The top 15 or 30 features were fed into the kneighbors_graph method found in Python's SciKit Learn Library. A compressed sparse row (CSR) matrix was produced and converted to a graph using the NetworkX Python library.

3.4 Clustering

Community detection was performed using the Leiden algorithm [19]. This algorithm optimizes modularity, a widely used measure for quantifying the quality of communities. The Leiden algorithm allows control over the number of detected

communities through a resolution parameter. This parameter facilitates the capture of both large-scale and fine-grained community structures. Given the small size of the dataset, capturing fine-grained community structures was essential.

Table 1. Modularity quality scores computed for graphs created from either 15 or 30 features, for three target variables.

	15 Features		30 Features	
	2 Clusters	3 Clusters	2 Clusters	3 Clusters
Race:Black	0.4059	0.1548	0.4185	0.3596
CovidFinance:Worse	0.4134	0.2076	0.4380	0.3380
HIV:Positive	0.4387	0.238	0.5116	0.1622

For each target variable, we constructed two graphs using the 15 and 30 most important variables identified through feature selection. To choose which to analyze, we did a preliminary clustering, increasing the Leiden resolution parameter until 2 and 3 clusters were produced. The modularity scores corresponding to each clustering are presented in Table 1. In all but one case, the clusterings based on 30 features exhibited higher scores. Therefore, we conduct in-depth analysis of those clusterings. We did a third clustering on the 30-feature graphs, increasing the resolution parameter until 4 clusters were obtained. This approach allowed us to identify patterns that exist at varying levels of granularity.

To provide insight into the attribute values within the clusterings, Table 2 presents the percentages of each cluster that answered *true* for 40 separate variables. Due to space limitations, we only display results for the 3-cluster groups. The clusterings are visually represented in Figs. 1, 2, and 3. The color at the top of each column in Table 2 corresponds to the node color in the network visualization. In selecting the variables for inclusion in Table 2, we prioritized attributes that exhibited "interesting" characteristics, based on variations between clusters. The intention of this analysis was to qualitatively identify attributes of interest and to observe changes as the number of clusters increased.

4 Results

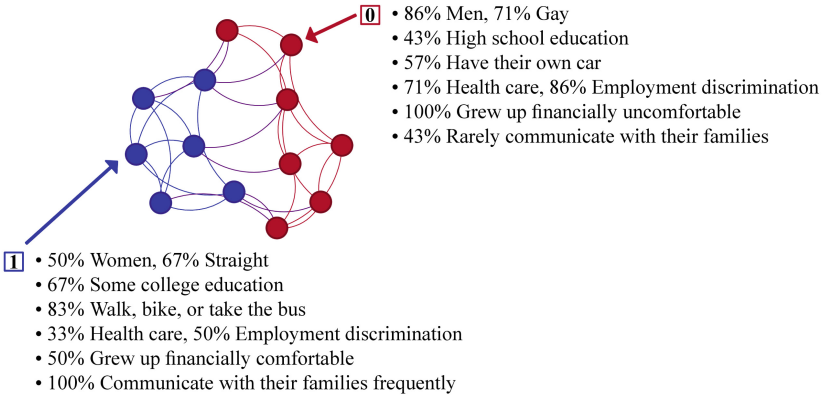
4.1 Selection Based on African American Race

The initial group consists of African American participants. It is important to note that the overall survey sample size is small, and therefore, results should not be considered representative of the entire African American community. They only reflect interviews conducted within a limited region at a specific time.

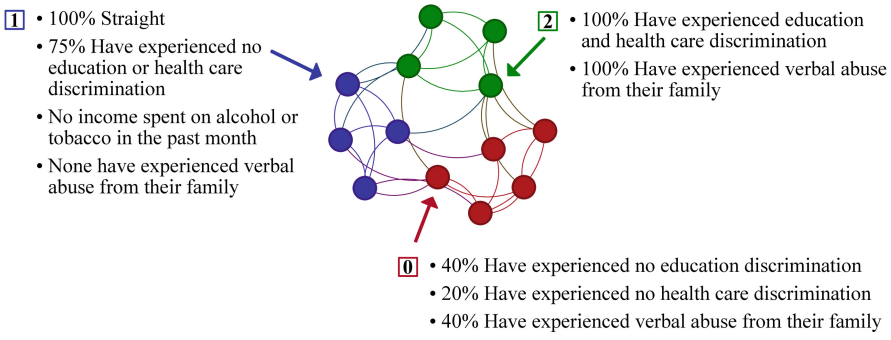
The three stages of graph analysis for African American participants are visualized in Fig. 1. Overall, the constructed graph for this group exhibits robustness

Table 2. Cluster composition for 40 variables. Numbers represent the percentage of cluster members exhibiting the described attribute.

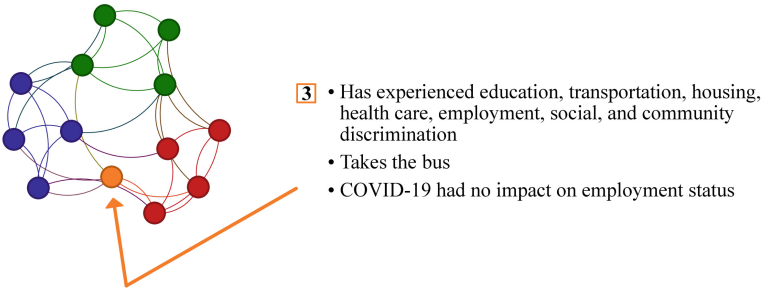
	Race			HIV+			COVID		
Variables / Cluster Number	0	1	2	0	1	2	0	1	2
Gender: Cis Men	100	25	75	50	60	67	20	67	75
Gender: Cis Women	0	50	25	50	20	0	60	17	0
Race: African American	100	100	100	83	80	67	60	67	75
Race: White	0	0	0	0	20	33	0	33	0
Race: Other	0	0	0	17	0	0	40	0	25
Sexuality: Gay/Lesbian	60	0	100	83	60	67	60	50	50
Sexuality: Straight	0	100	0	17	40	0	20	17	0
Income: < 10K	60	25	25	50	40	0	40	17	50
Income: 10-14k	40	0	25	17	20	67	40	50	0
Income: 15-25k	0	50	25	33	40	0	20	17	50
On Income Support	100	50	75	67	100	67	60	83	100
Education: HS	40	0	25	50	0	33	60	33	0
Education: Some College	40	75	0	17	80	33	20	50	25
Transportation: Own Car	40	25	50	33	20	67	0	66	25
Transportation: Bus	40	25	25	17	80	0	40	17	50
Employment: Full Time	0	0	25	0	0	33	0	0	0
Employment: Part Time	20	25	50	67	0	33	40	33	50
Employment: Self-Employed	40	0	0	0	20	33	0	17	25
Employment: Seasonal	20	0	0	17	0	0	20	0	25
Employment: Disabled	0	0	0	0	20	0	0	0	0
Employment: Unemployed	20	75	25	17	60	0	40	50	0
HIV+: Case Manager	80	75	25	50	100	67	20	100	67
HIV+: Meds	60	50	50	33	80	100	20	80	67
Checkup	100	75	50	33	100	100	20	83	100
Treatment: Doctor in Last Year	80	50	50	17	100	100	0	67	100
Covid: Sought Care	80	0	50	17	40	100	0	50	100
Covid: Number with Full Vaccine	40	50	50	17	80	67	0	66	75
Covid: No Vaccine	40	0	50	50	0	33	80	17	25
Covid: Reduced Sex	60	50	75	100	20	0	100	33	100
Covid: Lost Work or Fired	80	75	50	83	60	67	80	66	100
Discrimination: Healthcare(Race)	80	25	50	66	40	33	60	33	50
Discrimination: Education(Race)	40	0	50	33	20	67	40	33	0
Discrimination: Job(race)	100	25	75	67	40	67	60	50	50
Communication w/ Family: Weekly/Monthly	50	75	25	50	60	0	40	33	25
Communication w/ Family: Yearly/Never	20	0	50	50	0	33	40	17	75
Family Abuse: Sometimes/Mostly	40	0	100	83	20	67	60	33	75
Family Abuse: Never	60	0	0	0	40	33	0	33	25
LGBTQ: Feels Part of Community	40	0	0	0	40	0	0	17	25
Understands Them: Weekly/Monthly	20	25	50	50	20	33	60	33	0
Understands Them: Yearly/Never	60	25	50	50	20	33	40	33	100



(a) 2-Cluster Results for African Americans.



(b) 3-Cluster Results for African American Cohort.



(c) 4-Cluster Results for African American Cohort.

Fig. 1. Clustering Progression for African American Cohort from 2 to 4 Clusters

and coherence. For example, in the 4-cluster graph shown in Fig. 1c, the clusters are clearly defined, with the blue and green clusters forming cliques, and the red cluster displaying a high level of cohesion.

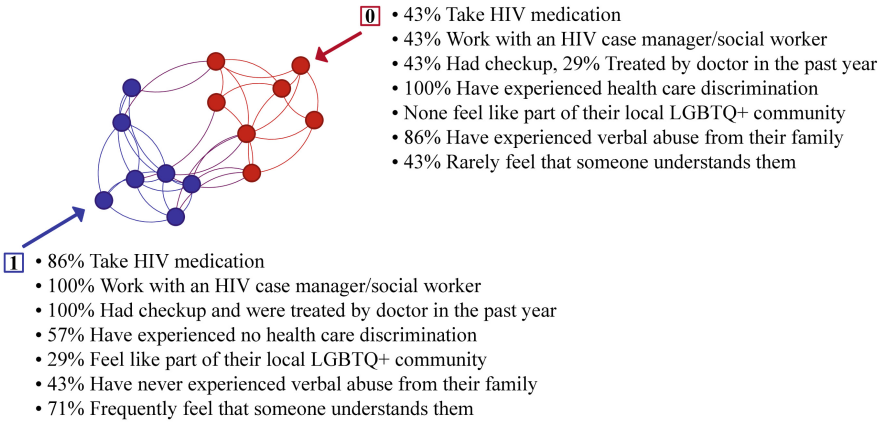
In the initial 2-cluster partition displayed in Fig. 1a, the red cluster primarily comprises participants who identify as gay (71%) and male (86%), whereas the blue cluster consists of 50% women and 67% heterosexuals. The red cluster faces significant economic and social challenges, with 100% of its members growing up in financially difficult circumstances and 57% reporting an annual income below $10,000. In contrast, 50% of the blue cluster grew up financially comfortable. Discrimination is more prevalent among the red cluster, with 71% experiencing healthcare discrimination based on race compared to 33% in blue, and 86% experiencing employment discrimination based on race compared to 50% in blue. The red cluster also exhibits more strained family relationships, as 43% of its members communicate with their family only rarely, whereas all members of the blue cluster maintain monthly contact with their families.

Partitioning into 3 clusters (shown in Fig. 1b), gives additional noteworthy attributes. The blue cluster becomes entirely heterosexual, and has a low prevalence of smoking and alcohol use. Additionally, the blue cluster reports lower levels of name-calling and family abuse, with none experiencing it frequently, unlike the "mostly" or "sometimes" reported cases in other clusters. The newly formed green cluster has 100% exposure to some form of education and healthcare discrimination, in contrast to reduced exposure in the other clusters.

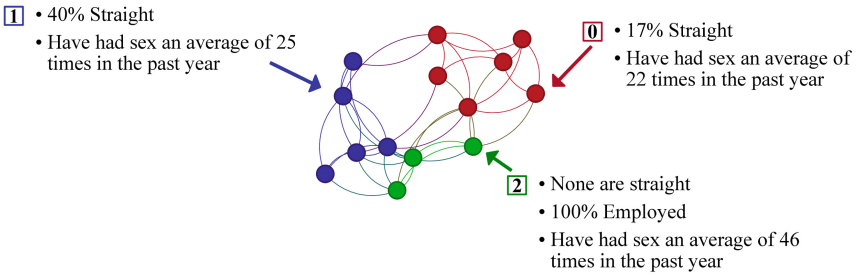
Upon repartitioning into 4 clusters (Fig. 1c), the additional cluster consists of a single node. This individual reports experiencing various forms of discrimination, including educational, transportation, housing (based on credit score), healthcare, job, social and community. Furthermore, this person is unemployed and relies on disability benefits, exacerbating their challenging circumstances.

4.2 Selection Based on HIV Positive Status

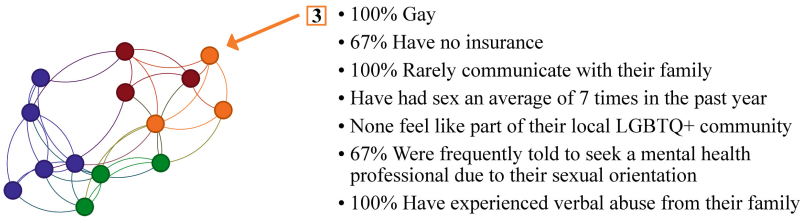
The HIV-positive respondents were predominantly male, Black, and identified as gay, which is reflected in the clusters depicted in Fig. 2. In the 2-cluster partition (Fig. 2a), the red cluster exhibited fewer medical interactions. Specifically, red individuals were less likely to take HIV medications (43% vs. 86%), have an HIV case worker (43% vs. 100%), receive a checkup in the last year (43% vs. 86%), and receive treatment from a doctor in the past year (29% vs. 100%). The red cluster also reported a greater sense of estrangement, experiencing higher levels of race-based healthcare discrimination (71% vs. 29%), with all members of the red cluster having encountered some form of healthcare discrimination. In contrast, 57% of the blue cluster reported no experience with healthcare discrimination. Moreover, the red cluster had a weaker support network. None of its members felt a sense of belonging to their local LGBT+ community (0% vs. 29% in the blue cluster). Additionally, 86% of the red cluster experienced verbal abuse from their family either frequently or occasionally, compared to 43% in the blue cluster who reported never experiencing abuse from their families.



(a) 2-Cluster Results for HIV Positive Status.



(b) 3-Cluster Results for HIV Positive Status.



(c) 4-Cluster Results for HIV Positive Status.

Fig. 2. Clustering Progression for HIV Positive Cohort from 2 to 4 Clusters

When considering 3 clusters, as depicted in Fig. 2b, the new green cluster is notable for its absence of straight members (67% gay and 33% bisexual). All members of this cluster prioritize their health, and they are all employed. Interestingly, this group reported the highest number of sexual encounters in the past year, with an average of 46 partners.

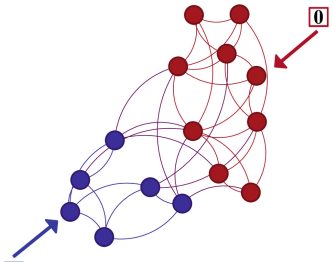
Introducing a fourth cluster, shown in Fig. 2c, reveals that all individuals in this cluster are gay (100%), and 67% of them have no insurance, while all members of other clusters possess some form of insurance. This new cluster faces significant challenges related to lack of support, as 100% of its members communicate with their families on a yearly or infrequent basis, none feel a sense of belonging to the local LGBT+ community, and 100% experience verbal abuse from their families at least occasionally.

4.3 Selection Based on COVID-19 Influence on Finances

The third group is respondents whose financial situations were adversely affected by COVID-19, as depicted in Fig. 3. In the 2-cluster partition illustrated in Fig. 3a, the red cluster demonstrates reduced engagement with healthcare. Notably, red individuals exhibited lower vaccination rates for COVID-19, with 55% remaining unvaccinated, while 83% of the blue individuals were partially or fully vaccinated. Moreover, the red cluster reports lingering brain fog and fatigue (22% vs. 0%), increased interference of COVID-19 with job retention (44% vs. 0%), and higher rates of job loss due to the pandemic (89% vs. 66%). Respondents in the red cluster also reported higher expenditures on alcohol and tobacco.

The three-cluster partition is shown in Fig. 3b. Compared to the new red cluster, green cluster members were more likely to have had a checkup in the past year (100% vs. 20%), sought medical care related to COVID-19 (100% vs. 0%), and have full vaccination for COVID-19 (75% vs. 0%). Green individuals also displayed higher employment rates compared to red (100% vs. 60%). Interestingly, respondents in the green cluster were more likely to receive some form of income support than those in the red cluster (100% vs. 40%).

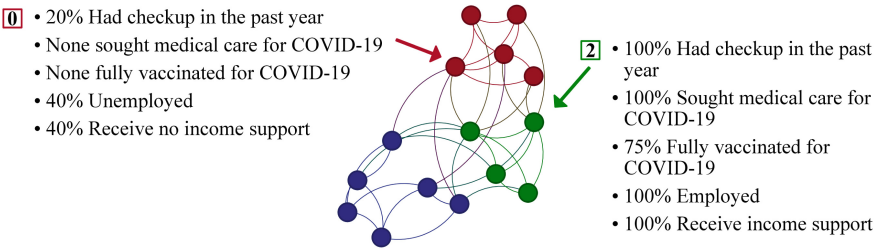
The 4-cluster partition is depicted in Fig. 3c. The orange cluster and the new blue cluster have both lost work due to COVID-19 (67% in both clusters). These clusters also exhibit notable differences. The orange cluster experienced less discrimination than blue, with no orange members reporting education or transportation discrimination, while 100% of the blue members reported experiencing both forms of discrimination. Individuals in the orange cluster were less likely to have experienced worsened living situations due to COVID-19 (33% vs. 100%). Respondents in the blue cluster were more likely to seek medical care related to COVID-19 (100% vs. 0%).



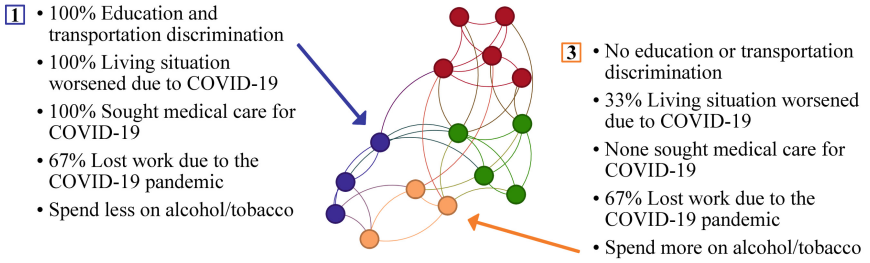
0
- 100% POC
- 38% Work with an HIV case manager/social worker
- 38% Take HIV medication
- 55% Not vaccinated for COVID-19
- 22% Lingering brain fog and fatigue from COVID-19
- 44% COVID-19 interfered with their ability to keep a job
- 89% Lost work due to COVID-19
- Spend more on alcohol/tobacco

1
- 33% White
- 100% Work with an HIV case manager/social worker
- 80% Take HIV medication
- 83% Partially/fully vaccinated for COVID-19
- 66% Lost work due to COVID-19
- Spend less on alcohol/tobacco

(a) 2-Cluster Results for COVID-19 Finance Worse Status.



(b) 3-Cluster Results for COVID-19 Finance Worse Status.



(c) 4-Cluster Results for COVID-19 Finance Worse Status.

Fig. 3. Clustering Progression for COVID-19 Finance Worse Cohort from 2 to 4 Clusters

5 Discussion and Conclusion

Network science provides a valuable approach for analyzing small datasets like the Burden of HIV survey. Our analysis focused on the structure of the data, utilizing successive clustering into 2, 3, and 4 clusters. The extensive range of survey questions enabled us to identify and examine three distinct subgroups, resulting in meaningful clusters that offer insights for further investigations.

Within the African American community, disparities exist in coping with HIV and COVID-19. Sexual minority men who are Black and living with HIV face obstacles including stigma and discrimination. Research has highlighted the detrimental effects of such attitudes on mental health and self-care among affected individuals [8]. These inequities and discriminatory practices contribute to disparities in HIV medical adherence. Consequently, interventions aimed at addressing these issues must be explored, such as this study on behavioral intervention components by Cluesman et al. involving 49 individuals [5]. The disparities also existed with COVID-19, with our study finding a cluster with a predominance of people of color with vaccine hesitancy (estimated at 41.6% in one meta study [9]), and a cluster exhibiting 22% with symptoms of long COVID.

Healthcare discrimination was frequently reported in our study as a significant barrier to proper medical treatment. Research on race-based healthcare disparities (e.g., [13]) is valuable in understanding these challenges, and our clustering approach helps underscore the heterogeneity of needs within the African American community. Additional investigation is warranted to improve the quality of care for subsets of the African American population, as demonstrated by [16]. A focus-group study involving African Americans living with HIV and another chronic disease revealed themes including negative interactions with healthcare professionals, stigma due to prejudice, and bias from healthcare staff [12]. These same issues are reported with regards to COVID [20].

From a network science perspective, our study successfully employed graph inference, community detection, and visualization techniques in a qualitative manner to uncover themes present in a small survey dataset. The analysis primarily utilized straightforward tools such as recursive feature elimination, kNN graph inference, and Leiden clustering. The results of this study are of significance to public health workers and other stakeholders, providing a unique and powerful method for analyzing smaller datasets.

Acknowledgments. This research was conducted through a grant obtained from the Illinois Innovation Network, Sustaining Illinois Seed Grant Program. The authors wish to thank Larry Mayhew and Larry McCulley at SIHF Healthcare for giving support and access to their staff of social workers. Also, thanks to Dr. William Summers and Tawnya Brown at Vivent Health for their support.

References

1. Abbas, M., et al.: Biomarker discovery in inflammatory bowel diseases using network-based feature selection. PloS one **14**(11), e0225, 382 (2019)

2. Adams, J., Light, R.: Mapping interdisciplinary fields: efficiencies, gaps and redundancies in hiv/aids research. PLoS One **9**(12), e115, 092 (2014)
3. Billock, R.M., et al.: Network interconnectivity and community detection in hiv/syphilis contact networks among men who have sex with men. Sex. Transm. Dis. **47**(11), 726 (2020)
4. Clipman, S.J., et al.: Deep learning and social network analysis elucidate drivers of hiv transmission in a high-incidence cohort of people who inject drugs. Sci. Adv. **8**(42), eabf0158 (2022)
5. Cluesman, S.R., et al.: Exploring behavioral intervention components for African American/black and latino persons living with hiv with non-suppressed hiv viral load in the United States: a qualitative study. Inter. J. Equity Health **22**(1), 1–29 (2023)
6. Gao, Z., Shih, P.C.: Communities of support: social support exchange in a hiv online forum. In: Proceedings of the Seventh International Symposium of Chinese CHI, pp. 37–43 (2019)
7. Grubb, J., Lopez, D., Mohan, B., Matta, J.: Network centrality for the identification of biomarkers in respondent-driven sampling datasets. Plos one **16**(8), e0256, 601 (2021)
8. Hong, C., Ochoa, A.M., Wilson, B.D., Wu, E.S., Thomas, D., Holloway, I.W.: The associations between hiv stigma and mental health symptoms, life satisfaction, and quality of life among black sexual minority men with hiv. Quality Life Res., 1–10 (2023)
9. Khubchandani, J., Macias, Y.: Covid-19 vaccination hesitancy in hispanics and african-americans: a review and recommendations for practice. Brain Behav. Immunity-health **15**, 100(277) (2021)
10. Kimbrough, L.W., Fisher, H.E., Jones, K.T., Johnson, W., Thadiparthi, S., Dooley, S.: Accessing social networks with high rates of undiagnosed hiv infection: the social networks demonstration project. Am. J. Public Health **99**(6), 1093–1099 (2009)
11. Kramer, J., Boone, L., Clifford, T., Bruce, J., Matta, J.: Analysis of medical data using community detection on inferred networks. IEEE J. Biomed. Health Inform. **24**(11), 3136–3143 (2020)
12. Mahadevan, M., Amutah-Onukagha, N., Kwong, V.: The healthcare experiences of african americans with a dual diagnosis of hiv/aids and a nutrition-related chronic disease: a pilot study. In: Healthcare, vol. 11, p. 28. Multidisciplinary Digital Publishing Institute (2023)
13. Manning, M., Byrd, D., Lucas, T., Zahodne, L.B.: Complex effects of racism and discrimination on african americans' health and well-being: navigating the status quo. Soc. Sci. Med. **316**, 115, 421 (2023)
14. Matta, J., Singh, V., Auten, T., Sanjel, P.: Inferred networks, machine learning, and health data. Plos one **18**(1), e0280, 910 (2023)
15. Matta, J., Zhao, J., Ercal, G., Obafemi-Ajayi, T.: Applications of node-based resilience graph theoretic framework to clustering autism spectrum disorders phenotypes. Applied Netw. Sci. **3**(1), 1–22 (2018)
16. Mitchell, B.D., et al.: Patient-identified markers of quality care: improving hiv service delivery for older african americans. J. Racial Ethn. Health Disparities **10**(1), 475–486 (2023)
17. Potterat, J.J., et al.: Risk network structure in the early epidemic phase of hiv transmission in colorado springs. Sexually Transmitted Infect. **78**(suppl 1), i159–i163 (2002)

18. Sanjel, P., Matta, J.: Inferred networks and the social determinants of health. In: Complex Networks & Their Applications X: Volume 2, Proceedings of the Tenth International Conference on Complex Networks and Their Applications Complex Networks 2021 10, pp. 703–715. Springer (2022). https://doi.org/10.1007/978-3-030-93413-2_58
19. Traag, V.A., Waltman, L., Van Eck, N.J.: From louvain to leiden: guaranteeing well-connected communities. Sci. Rep. **9**(1), 5233 (2019)
20. Wright, J.E., Merritt, C.C.: Social equity and covid-19: the case of african americans. Public Adm. Rev. **80**(5), 820–826 (2020)
21. Zarrabi, N., Prosperi, M., Belleman, R.G., Colafigli, M., De Luca, A., Sloot, P.M.: Combining epidemiological and genetic networks signifies the importance of early treatment in hiv-1 transmission (2012)

Untangling Emotional Threads: Hallucination Networks of Large Language Models

Mahsa Goodarzi(✉), Radhakrishnan Venkatakrishnan, and M. Abdullah Canbaz

AI in Complex Systems Laboratory Department of Information Sciences and Technology College of Emergency Preparedness, Homeland Security, and Cybersecurity, University at Albany, SUNY, Albany, NY, USA
{mgoodarzi,rvenkatakrishnan,mcanbaz}@albany.edu

Abstract. Generative AI models, known for their capacity to generate intricate and realistic data, have rapidly found applications across various domains. Yet, doubts linger regarding the full scope of their hallucinatory capabilities and the reliability of their outcomes. These concerns underscore the need for rigorous analysis and validation of generative AI models. This study employs network analysis to explore the inherent characteristics of generative AI models, focusing on their deviations and disparities between generated and actual content. Using GPT3.5 and RoBERTa, we analyze tweets, vocabulary, and emotion networks from their outputs. Although network comparison demonstrated hallucination, non-classification, and instability patterns in GPT-3.5 compared to RoBERTa as a baseline, both models exhibit promise and room for improvement.

Keywords: Emotions · Hallucination · GPT · RoBERTa · Generative AI

1 Introduction

In the realm of artificial intelligence, generative models have emerged as remarkable tools capable of producing intricate and realistic data that mimics human creativity and understanding. At the threshold of an impending AI-driven transformation, it is essential to appreciate the ingenuity of generative AI models while subjecting their outputs to meticulous examination.

Generative AI models have been used to generate training data for machine learning models, such as those used for global warming prediction[9], in medical responses [18], and in education [14]. These models have achieved high accuracy, comparable to human-labeled models in various domains [17]. They also automate data labeling in customer service chatbots, enhancing response precision [15]. However, some studies have found varying outcomes in producing data, particularly in AI-generated figurative speech [33,37].

In this study, we delve into the classification prowess of generative AI, examining the extent of the variations between their outputs through the lens of network analysis.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 202–214, 2024.
https://doi.org/10.1007/978-3-031-53468-3_17

Emotions Detection and Classification. In a written context, emotions are often figurative and preserve their uniqueness, but the computational power cannot classify them because of overlapping components across multiple emotions.

From one viewpoint, emotions are perceived as distinct and guided by independent mechanisms, leading to attribute identification for each emotion [24]. According to this outlook, emotions are estimated by examining their specific associations [25]. Furthermore, some research suggests a consensus that all emotions entail shared components with other emotions [32], implying that only a limited number of components genuinely serve as indicators of specific emotions [40]. These findings have fostered a widely embraced perspective that emotions constitute distinct categories with somewhat ambiguous boundaries [12].

Entailing additional research endeavors, in more recent work, researchers conceive emotions as networks comprised of interrelated components, where distinct emotions share common elements and connections [20,35]. To explore this perspective, they introduce network analysis into emotion research, applying it to reanalyze a dataset involving multiple positive emotions. They detail the process of estimating networks and identifying overlapping communities of nodes within these networks. This network-oriented approach carries implications for comprehending different emotions, their co-occurrence, and measurement methods.

Network Analysis on Emotions. Researchers have undertaken emotion network analysis in various contexts. For instance, they have examined COVID-19 era quarantine data through mood networks [23]. Additionally, numerous studies have delved into emotion dynamics, aiming to gain insights into areas such as interpersonal emotion dynamics [6], childhood traumas [16], post-traumatic stress and anxiety disorders [21], emotion and brain correlations [28], motivational and emotional correlations [19], as well as emotion and sentiment analysis [30]. It is worth noting that the pursuit of emotion and sentiment detection have been ongoing endeavors in Natural Language Understanding(NLU) for decades, involving contributions from fields such as psychiatry, psychology, sociology, linguistics, and computer science. However, it is essential to recognize that this work continues, particularly as figurative speech evolves alongside spoken and written language.

Generative AI and Hallucination. On different aspects, Generative AI models exhibit hallucinations when they convincingly generate content that deviates from reality [3]. Such deviations can be attributed to incomplete or inaccurate training data [5]. Additionally, when tasked with generating content about unfamiliar topics, the model resorts to imaginative elements to fill gaps [29]. These models, designed for creativity and diversity in content generation, may produce non-factual outputs. Furthermore, inherent biases or misinformation within the model can be magnified in its generated content.

In recent work, we investigated the effectiveness of emotion detection in large language models (LLMs) [39]. Although one model excels in producing coherent and contextually relevant responses, it faces difficulties in precisely classifying emotions due to its generative nature. In contrast, another model fine-tuned on emotion classification achieves greater accuracy in emotion predictions.

Amidst these issues, in this paper, we ponder the extent to which emotions manifest in the outputs of generative AI models. Is there a more feasible approach for untangling hallucinations from reality by examining emotional variations and shared components through network analysis? In pursuit of this objective, we present the hallucination networks derived from tweets collected between March and April 2023 for the September 16–29, 2022 time frame. Our focus centers on the Zhina Mahsa Amini case in Iran, a context rich in written emotions. The labels for these emotions are provided by generative AI models, namely GPT3.5 and RoBERTa, and are subsequently transformed into a network of emotions incorporating diverse vocabulary perspectives.

2 Methodology

2.1 Dataset

In this paper, we used snscraper [1] to gather data from X [13], formerly Twitter, just before API changes [7]. For our case study, we focused on Zhina Mahsa Amini's case, a 22-year-old Kurdish woman who died in police custody in Tehran, Iran, on September 16, 2022. This incident prompted protests and activism for women's rights in Iran, especially on X against the government, encouraging many to speak out. We searched using popular event hashtags [4] and filtered tweets under 120 characters with memes, images, or videos. From over 6 million tweets in all languages, we randomly selected 5,000 due to the high cost associated with running GPT-3.5.

2.2 Models

The paper "Attention is All You Need," by Vaswani et al. in 2017 [38], introduced the transformative "Transformer" neural network architecture, which has become the basis for state-of-the-art NLP models like GPT and BERT due to its parallelization, scalability, and effective self-attention mechanisms. This study employs two prominent large language models, RoBERTa-2023 and GPT-3.5, for emotion identification.

RoBERTa model is built on BERT's [2] encoder-only architecture [36] trained on 154M tweets [22]. Compared to other post-BERT models, they significantly improve downstream Natural Language Processing (NLP) tasks [2]. The Tweet classification version of RoBERTa fine-tuned on the TweetEval benchmark, initially included four major emotions: Anger, Joy, Optimism, and Sadness in its training data. This RoBERTa-base model was updated in June 2023 [11] with TweetEval benchmark to include 11 emotions. In our analysis, we refer to it as RoBERTa'23. As RoBERTa'23 is purpose-built for Tweet emotion classification, it serves as the baseline model in this paper.

The **GPT-3.5** model represents a refined iteration of the GPT-3 (Generative Pre-Trained Transformer) model. Introduced in January 2022, GPT-3.5 offers three distinct variants, each characterized by parameter counts of 1.3 billion, 6

billion, and 175 billion [26]. A notable feature of GPT-3.5 is its enhanced ability to mitigate the generation of noxious content to a certain extent. In contrast to RoBERTa'23, our approach to eliciting emotions from GPT-3.5 involved employing the zero-shot learning method with prompt engineering techniques [36] and Parrot's emotion binning to obtain ten emotion labels [27]. We have 3691 tweets after processing GPT-3.5 outputs for reasonable consistency.

Please note that GPT's zero-shot learning requires sample prompts for accurate results. Prompt quality is crucial, and we have designed specific prompts to enhance GPT's emotion detection. Prompt design complexity yields diverse outcomes, often amidst the haystack of possibilities. Hence, our study focuses on LLMs' emotion inference, not prompt quality or frameworks, while existing research explores effective, prompt design [31].

3 Experimental Results

3.1 Consistency Analysis of Models

Our initial experiment explored generative AI models' ability to identify emotions when presented with nearly identical or closely resembling inputs. To do this, we selected a subset of tweets from our dataset containing duplicates and substantial similarity. We applied the cosine similarity metric to retain tweets with similarity levels surpassing 90%. Notably, RoBERTa consistently assigned the same emotion label to identical or similar tweets, while GPT-3.5 yielded up to three different emotions. We removed the duplicates from our corpus for the subsequent analysis after accounting for hashtags.

3.2 Distance Measures

In the realm of distance metrics, several methods are employed to quantify the dissimilarity between pairs of networks. One such measure is the Euclidean distance, which computes the square root of the squared differences between the coordinates of two objects. Another widely used metric is the Manhattan distance, also referred to as Manhattan length, which is determined by summing the distances between the x and y coordinates. A variation of the Manhattan distance is the Canberra distance, which introduces weighting factors into the calculation. This weighted approach computes the distance between two points by considering the differences between their corresponding components. These distance metrics serve as valuable tools for quantifying dissimilarity and are applied across various domains of analysis and problem-solving [34].

To this end, we transformed the vocabulary graphs, based on the frequency of words, into adjacency matrices to assess the emotion networks derived from the two models. Note that when the distance values approach zero, it signifies that the nodes' distances in the networks are also zero, essentially indicating their degrees are equivalent in both graphs. As depicted in Table 1, all metrics reveal substantial distance, confirming that the nodes in the networks produced by RoBERTa'23 and GPT-3.5 exhibit significant dissimilarity.

Table 1. Distance Measures between Two Networks

	RoBERTa'23 vs GPT-3.5
Euclidean	834.31
Manhattan	40,130.00
Canberra	0.382

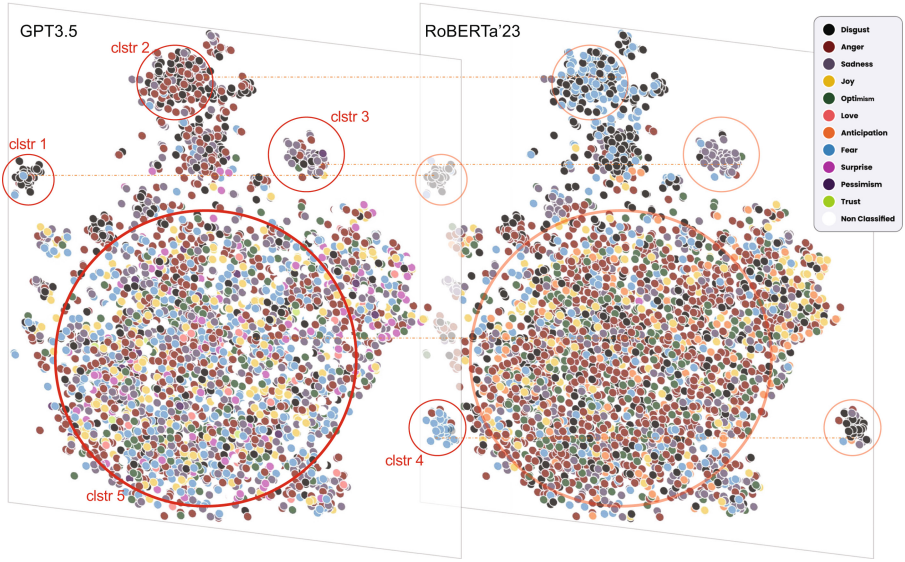


Fig. 1. Visualization of K-Means Clustering for Tweets classified by GPT-3.5 and RoBERTa'23 colored based on its assigned emotion annotation.

3.3 Clustering Analysis

In this section, we investigated the patterns and grouping of tweets by the word embeddings. To this end, we first created word embeddings, dense vector representations of words of the tweets. Subsequently, we performed lemmatization and stemming techniques [8], which aim to reduce words to their base form and truncate words to their root form, respectively. Then, t-SNE (t-distributed Stochastic Neighbor Embedding) was employed to decrease the dimensionality of the data to facilitate visualization. Please note that t-SNE is often used to visualize high-dimensional data in 2D or 3D space while preserving the pairwise similarities between data points. Finally, we implemented k-means clustering, an unsupervised machine learning algorithm, to cluster data points into groups of clusters based on their similarities and labels, e.g., the emotions generated by GPT-3.5 and RoBERTa'23.

In Fig. 1, we mark coordinates to represent single-tweet word embeddings. Closer points suggest similar word embeddings. Despite showing distinct clus-

ters (clstr-1,2, 3, 4, and 5), mixed emotions appear with various colors. This validates prior research in psychology and psychiatry, indicating shared components among emotions, namely vocabulary. However, from a Generative AI perspective, tweets in the figure display different colors and emotions. Emotions mix across clusters and even within them. Considering RoBERTa'23 as our baseline model, we observe that the labels generated by GPT-3.5 are significantly different, revealing its distinct hallucination patterns. It is essential to highlight that out of the 3,691 tweets in this clustering analysis, only 1,266 tweets have been assigned identical emotional labels by the GPT-3.5 and Roberta'23 models.

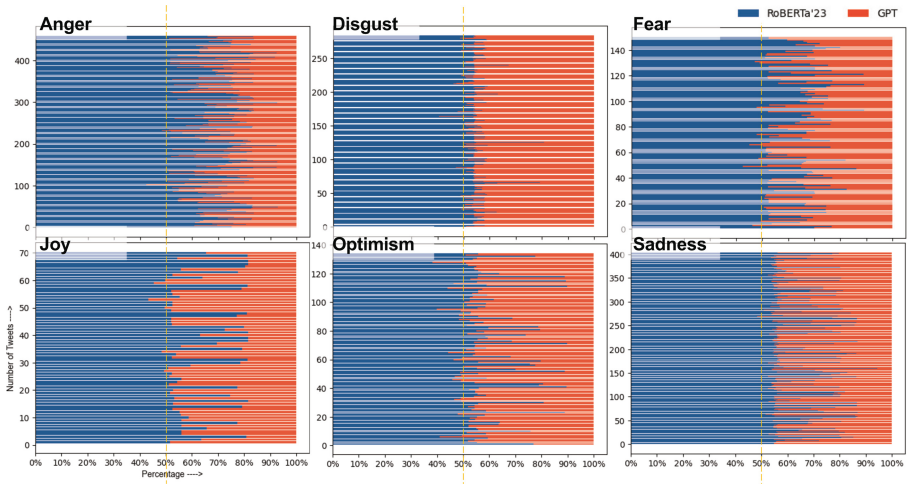


Fig. 2. Distribution of RoBERTa'23 and GPT-3.5 scaled scores for the same tweet in different emotions

3.4 Confidence Level in Labeling

The discernible fact is that tweets can be categorized with distinct emotions. Our interest is assessing the hallucination that occurs, specifically in GPT-3.5 and RoBERTa'23, in ascribing emotional labels and confidence score to tweets. To this end, in Fig. 2, we present the tweets classified with the same emotion by two models. To be able to compare the model scores, we normalized the scores adding the total to 100%. For instance, in the Anger figure (upper-left), there was a total of 459 tweets; both RoBERTa'23 and GPT-3.5 classified them as Anger. Yet, RoBERTa'23 was supremely confident in tweets' emotions being Anger, with an average of 70%, whereas GPT-3.5 was about 30% confident in tweets' emotions being Anger. Similarly, in Fear, Joy, Optimism, and Sadness figures, RoBERTa'23 was confident with averages of 63%, 63%, 58%, and 63%, respectively, whereas GPT-3.5 averaged 37%, 37%, 42%, and 37%, respectively. Interestingly, it was close to 50% for both RoBERTa'23 (54%) and GPT-3.5

(46%) for Disgust. Overall, RoBERTa'23, being fine-tuned for emotion classification, exhibits higher confidence, while GPT-3.5 struggles. This asserts the need for high quality task-specific fine-tuning of foundational models like GPT-3.5 and their hallucination to provide a score not close to reality otherwise.

3.5 MultiGraph of Emotions Labeled by LLM Models

Figure 3 illustrates the network of tweets classified under seven emotions shared between two models, RoBERTa'23 in blue and GPT-3.5 in orange. We observe that 65.7% of tweets are labeled with two emotions (both blue and orange links point to the same node). Moreover, a k-core of two degrees in the vocabulary multigraph eliminates 429 (11.74%) words (not shown in the figure), indicating that about 88.26% of words have been labeled with different emotions by the two models (blue and orange links are connected to different emotions for each word). Although Disgust and Anger emerge as the prevailing emotions in most tweets, in instances where models exhibit uncertainty regarding a specific emotion, they classify it as Fear, Sadness, Optimism, or Joy. It is noteworthy that Surprise is an intriguing emotion observed in both models, often entangled with various other emotions.

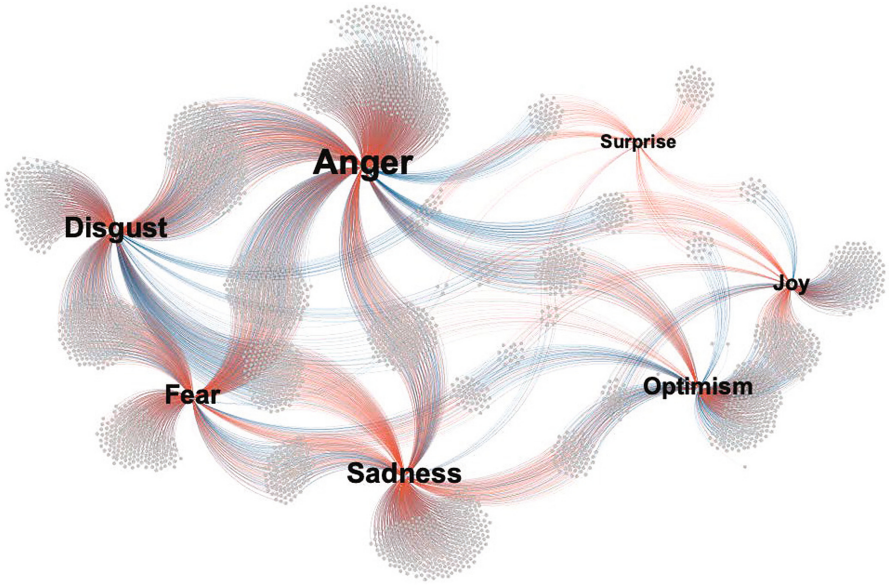


Fig. 3. Network of Tweets classified into seven emotions detected by RoBERTa'23 (shown in Blue edges) and GPT-3.5 (shown in Orange edges)

Similarly, in Fig. 4, we demonstrate the k-core networks of vocabulary classified under seven emotions by the two models, again in RoBERTa'23 in blue

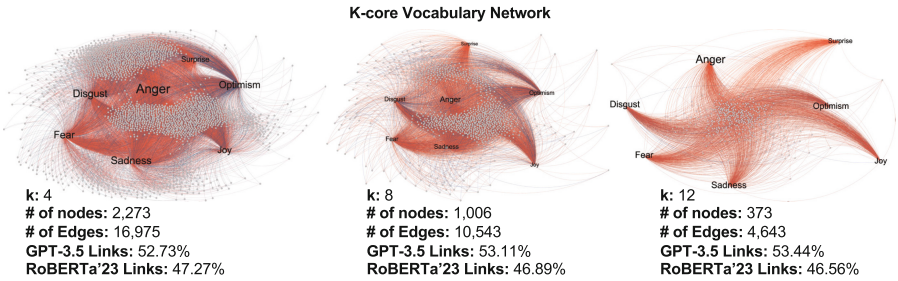


Fig. 4. K-core Vocabulary Network of emotions classified by RoBERTa'23 (in Blue) and GPT-3.5 (in Orange).

and GPT-3.5 in orange. Please note that we thoroughly cleaned word embeddings during the preprocessing step. This process involved lemmatization and stemming to extract the most valuable vocabulary for each tweet. As illustrated in the figure, an increase in k results in a higher percentage of GPT-3.5 links. This indicates that GPT-3.5 categorizes words into more classes than our baseline model, RoBERTa'23. This observation proves GPT-3.5's tendency to generate erroneous inferences based on written context and prompts. Additionally, the largest observable k-core network within the vocabulary consists of thirteen links. The thirteen-core sub-graph (not shown in the figure) reveals that the 230 words connected to all emotions have a higher weighted connection based on frequency to GPT-3.5 (53.78%) than RoBERTa'23 (46.22%).

3.6 Communities Within Vocabulary Networks of LLMs

Figure 5 illustrates the vocabulary network annotated with emotions in two models, GPT-3.5 (on the left) and RoBERTa'23 (on the right). The colors of both nodes and edges in these figures are determined by the community detection algorithm, and modularity scores are computed using the Louvain method implemented by Gephi [10].

The left figure presents GPT-3.5's emotion vocabulary network with eleven emotion classifications featuring six distinct communities. The largest community comprises Anger and Disgust (30.23%), followed by Joy and Optimism (18.45%), Fear and Trust in the same community (17.06%), Sadness and Pessimism (16.8%), Surprise and None Classified (15.7%), and finally, Love as the smallest community (1.75%).

In the right figure, we showcase RoBERTa'23's network, which detects eight emotions and reveals five communities as identified by Gephi. Anger constitutes a single dominant community (36.9%) in this model, whereas Disgust is paired with Fear (19.89%). Meanwhile, Joy, Optimism, and Surprise form the second-largest group (23.31%). Anticipation represents the smallest community (9.21%) in this model.

Among the detected emotions, RoBERTa'23's weakest classification is Surprise, while GPT-3.5 exhibits a relatively weak detection of Love. Additionally,

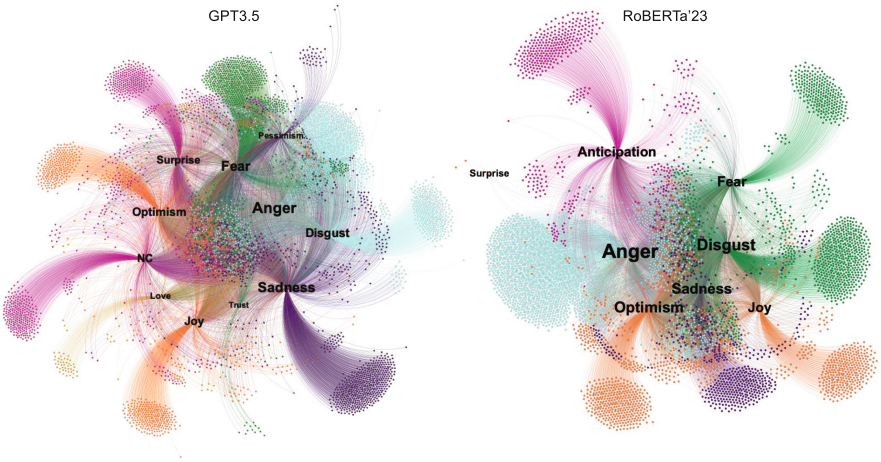


Fig. 5. The vocabulary networks annotated with emotions detected by GPT-3.5 (on the left) and RoBERTa'23 (on the right), colored by Modularity

there are instances in the GPT-3.5 network where the Generative AI model could not assign emotions to tweets and vocabulary items, denoted as "none-classified" (NC) in the figure.

It is important to emphasize that this figure warrants further investigation into the potential clustering of emotions into smaller communities and the distinctions in the vocabulary associated with Anticipation, Surprise, and None-Classified. However, this aspect is reserved for future research, potentially involving collaboration with domain experts.

Regarding emotion detection, it is noteworthy that distinctions are observed in the classification of emotions, particularly in the differentiation between Anger and Fear. On the other hand, similarities exist in the classification of Joy and Optimism, as they appear to share commonalities in their categorization. This suggests that the models may exhibit variability in their interpretation and classification of emotions, with some emotions being more closely aligned than others in the results.

3.7 Networks Metrics

Table 2 provides network metrics for the vocabulary networks shown in Fig. 5. As explained in the previous section, the community detection algorithm provided different hierarchical structures of the networks for the models.

The modularity metric for RoBERTa'23 being 0.261 and GPT-3.5 being 0.231 suggests that some groupings or communities within these networks are not highly distinct or well-defined. The nature of vocabulary, where a word can have multiple meanings, can cause this behavior.

Within the vocabulary networks, the assortativity metrics for GPT-3.5 and RoBERTa'23, being -0.629 and -0.616, respectively, indicate a significant degree

of disassortative mixing. This means that words within the networks, particularly those with dissimilar linguistic or contextual characteristics, tend to form connections. In contrast to assortative networks where similar words cluster together, a disassortative network suggests that antonyms, words with differing grammatical roles, or those used in dissimilar contexts are more likely to connect. This behavior highlights language diversity and contrasting aspects within the analyzed corpus. Additional evidence indicates that emotion detection models, including the baseline model RoBERTA'23, still face significant challenges. It is noteworthy that although RoBERTA'23 achieves a higher classification rate, it also demonstrates the same disassortative mixing behavior, suggesting improvements in emotion detection models are warranted.

Interestingly, while the assortativity metrics of networks are very low, the average clustering coefficients for GPT-3.5 and RoBERTa'23 are 0.584 and 0.603, respectively. The high clustering coefficients, along with a small average shortest path, indicate that words within the network tend to form tight-knit clusters -yet another small-world network- suggesting strong linguistic or contextual associations among words. Overall, the networks exhibit cohesive word clusters and connections between words with differing characteristics, reflecting a diverse and intricate structure encompassing semantic similarities and contrasts in language usage.

Table 2. Network Metrics for Vocabulary Networks of LLM

	Assortativity	Avg. Shortest Path	# of Cliques	# of Communities	Modularity	Avg. Clustering Coefficient
RoBERTa	–0.616	2.919	12,799	5	0.261	0.603
GPT	–0.629	3.023	14,493	6	0.231	0.584

We also observe a high number of cliques in both networks. In this context, cliques represent tightly interconnected groups of words that share strong linguistic or contextual relationships. Each clique encapsulates a distinct semantic or contextual concept, with words within the clique exhibiting similarities in meaning, grammatical roles, or frequent co-occurrence in text.

4 Conclusion

The development of Large Language Models signifies a notable stride toward achieving human-like intelligence. Their capacity to grasp and classify figurative speech is a testament to their growing capabilities. Moreover, as these models continue to enhance their reliability levels, they are poised to play a pivotal role in facilitating the accurate detection of emotions within textual content. This not only expands our understanding of language but also opens up a wealth of possibilities for applications across various domains, underlining the promising future of LLMs in the realm of natural language processing and artificial intelligence. Our research is a step in that direction to point out the issues from a

network analysis standpoint, encouraging practitioners and organizations alike to be critical when using LLMs for NLU tasks. Looking at texts filled with overly negative emotion, the reliability of the fine-tuned model reiterates the notion of fine-tuning LLMs for specific use cases to yield quality results.

In comparing GPT-3.5 with RoBERTa'23 as the baseline model, we discovered significant differences between the outputs of the two models through network analysis. K-means clustering for tweets' word embedding and Confidence level at the emotion labeling reveals hallucination patterns by GPT-3.5 that assigns emotions to tweets or score differently compared to RoBERTa'23. Multigraphs of tweets and associated vocabularies confirmed GPT-3.5's deviation in visualizations and metrics. Less than half the tweets had identical emotion labels, and more than 88% of vocabularies are labeled differently between the models based on k-core subgraphs. The k-core graphlets of the multigraph of vocabularies and shared emotions showcase higher numbers of connections in a wider classification of emotions. This result confirms the statistically inaccurate inference of GPT-3.5's generated content. Despite GPT-3.5's inability to detect an emotion in some tweets, classification similarities exist between the two models. The hierarchical structures of the vocabulary networks are slightly in favor of RoBERTa'23, but the resemblances in community formations provide insights into the challenges of emotion detection and their potential for advancement.

References

1. snscrape. https://github.com/JustAnotherArchivist/snscrape
2. Acheampong, F.A., Nunoo-Mensah, H., Chen, W.: Transformer models for text-based emotion detection: a review of BERT-based approaches (2021)
3. Alkaissi, H., McFarlane, S.I.: Artificial hallucinations in chatgpt: implications in scientific writing. Cureus (2023)
4. Amidi, F.: Hashtags, a viral song and memes empower iran's protesters (2022)
5. Athaluri, S.A., Manthena, S.V. Kesapragada, V.S.R., Yarlagadda, V., Dave, T., Duddumpudi, R.T.S.: Exploring the boundaries of reality: investigating the phenomenon of artificial intelligence hallucination in scientific writing through chatgpt references (2023)
6. Bar-Kalifa, E., Sened, H.: Using network analysis for examining interpersonal emotion dynamics. Multivariate Behav. Res. (2020)
7. Barnes, J.: Twitter ends its free API: Here's who will be affected
8. Bird, S., Klein, E., Loper, E.: Natural language processing with Python. O'Reilly (2009)
9. Biswas, S.S.: Potential use of chat gpt in global warming. Annals Biomed. Eng. (2023)
10. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks (2008)
11. Camacho-Collados, J., et al.: TweetNLP: Cutting-Edge Natural Language Processing for Social Media (2022)
12. Cowen, A.S., Keltner, D.: Self-report captures 27 distinct categories of emotion bridged by continuous gradients. Proc. Nat. Acad. Sci. (2017)
13. Davis, W.: Twitter is being rebranded as x - the verge (2023)

14. Fuchs, K.: Exploring the opportunities and challenges of nlp models in higher education: is chat gpt a blessing or a curse? (2023)
15. Gilardi, F., et al.: Chatgpt outperforms crowd-workers for text-annotation tasks. arXiv: 2303.15056 (2023)
16. Hasmi, L., et al.: Network approach to understanding emotion dynamics in relation to childhood trauma and genetic liability to psychopathology: replication of a prospective experience sampling analysis (2017)
17. Huang, F., Kwak, H., An, J.: Is chatgpt better than human annotators? potential and limitations of chatgpt in explaining implicit hate speech. arXiv:2302.07736 (2023)
18. Johnson, D., Goodman, R., Patrinely, J., Stone, C., Zimmerman, E., et al.: Assessing the accuracy and reliability of ai-generated medical responses: an evaluation of the chat-gpt model (2023)
19. Kinnison, J., Padmala, S., Choi, J-M., Pessoa, L.: Network analysis reveals increased integration during emotional and motivational processing. J. Neurosci. (2012)
20. Lange, J., Zickfeld, J.H.: Emotions as overlapping causal networks of emotion components: implications and methodological approaches (2021)
21. Liang, S., et al.: The relations between emotion regulation, depression and anxiety among medical staff during the late stage of covid-19 pandemic: a network analysis. Psych. Res. (2022)
22. Loureiro, D., et al.: Tweet insights: a visualization platform to extract temporal insights from twitter arXiv:2308.02142 (2023)
23. Martín-Brufau, R., Suso-Ribera, C., Corbalán, J.: Emotion network analysis during covid-19 quarantine-a longitudinal study. Front. Psychol. (2020)
24. Matsumoto, D., Keltner, D., Shiota, M.N., O'Sullivan, M., Frank, M.: Facial expressions of emotion (2008)
25. Mauss, I.B., Levenson, R.W., McCarter, L., Wilhelm, F.H., Gross, J.J.: The tie that binds? coherence among emotion experience, behavior, and physiology (2005)
26. Ouyang, L., et al.: Training language models to follow instructions with human feedback (2022)
27. Parrott, W.G.: Emotions in social psychology: Essential readings. Psychology Press (2001)
28. Pessoa, L.: Understanding emotion with brain networks. Current Opinion Behav. Sci. (2018)
29. Rudolph, J., Tan, S., Tan, S.: Chatgpt: bullshit spewer or the end of traditional assessments in higher education? J. Appl. Learn. Teach. (2023)
30. Sailunaz, K., Alhajj, R.: Emotion and sentiment analysis from twitter text. J. Comput. Sci. (2019)
31. Si, C., et al.: Prompting GPT-3 to be reliable (2023)
32. Siegel, E.H., et al.: Emotion fingerprints or emotion populations? a meta-analytic investigation of autonomic features of emotion categories (2018)
33. Sohail, S.S., et al.: Decoding chatgpt: A taxonomy of existing research, current challenges, and possible future directions (2023)
34. Tantardini, M., Ieva, F., Tajoli, L., Piccardi, C.: Comparing methods for comparing networks. **9**, 17557 (2019)
35. Trampe, D., Quoidbach, J., Taquet, M.: Emotions in everyday life. PloS one (2015)
36. Tunstall, L., von Werra, L., Wolf, T.: Natural language processing with transformers: building language applications with Hugging Face (2022)
37. Vaira, L.A., et al.: Accuracy of chatgpt-generated information on head and neck and oromaxillofacial surgery: a multicenter collaborative analysis (2023)

38. Vaswani, A., et al.: Attention is all you need (2017)
39. Venkatakrishnan, R., Goodarzi, M.,Canbaz, M.A.: Use cases from the middle east, Exploring large language models' emotion detection abilities (2023)
40. Witkower, Z., Tracy, J.L.: A facial-action imposter: How head tilt influences perceptions of dominance from a neutral face (2019)

Analyzing Trendy Twitter Hashtags in the 2022 French Election

Aamir Mandviwalla[1,2], Lake Yin[1,2], and Boleslaw K. Szymanski[1,2(✉)]

[1] Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
[2] Network Science and Technology Center, Rensselaer Polytechnic Institute, Troy, NY 12180, USA
boleslaw.szymanski@gmail.com

Abstract. Regressions trained to predict the future activity of social media users need rich features for accurate predictions. Many advanced models exist to generate such features; however, the time complexities of their computations are often prohibitive when they run on enormous data-sets. Some studies have shown that simple semantic network features can be rich enough to use for regressions without requiring complex computations. We propose a method for using semantic networks as user-level features for machine learning tasks. We conducted an experiment using a semantic network of 1037 Twitter hashtags from a corpus of 3.7 million tweets related to the 2022 French presidential election. A bipartite graph is formed where hashtags are nodes and weighted edges connect the hashtags reflecting the number of Twitter users that interacted with both hashtags. The graph is then transformed into a maximum-spanning tree with the most popular hashtag as its root node to construct a hierarchy amongst the hashtags. We then provide a vector feature for each user based on this tree. To validate the usefulness of our semantic feature we performed a regression experiment to predict the response rate of each user with six emotions like anger, enjoyment, or disgust. Our semantic feature performs well with the regression with most emotions having R^2 above 0.5. These results suggest that our semantic feature could be considered for use in further experiments predicting social media response on big data-sets.

Keywords: Computational social science · Social computing · Network science · 2022 French presidential election · Ukrainian war

1 Introduction

In recent years, social media data has been increasingly used to predict real-world outcomes. Data from platforms like Twitter, Reddit, and Facebook has been shown to be valuable in predicting public sentiment or response towards many different topics. This information has been used across many different fields like predicting stock market price changes or movie popularity [3,15].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 215–224, 2024.
https://doi.org/10.1007/978-3-031-53468-3_18

Social media platforms have continued to get more popular over time. Due to this, the size of social media datasets continues to increase. As the size of these datasets gets bigger and bigger, computational time complexity of the algorithms being used becomes a significant issue. Some of the most popular features used in social media predictions like sentiment analysis become prohibitively expensive when working with larger datasets [1,14].

In this paper we propose a method to generate features that can be used in social media predictions on big datasets. We create a weighted semantic network between Twitter hashtags from a corpus of 3.7 million tweets related to the 2022 French presidential election. A bipartite graph is formed where hashtags are nodes and weighted edges connect the hashtags reflecting the number of Twitter users that interacted with both hashtags. The graph is then transformed into a maximum-spanning tree with the most popular hashtag designated as its root node to construct a hierarchy amongst the hashtags. We then provide a vector feature for each user where the columns represent each of the 1037 hashtags in the filtered dataset and the value for each column is the normalized count of interactions for the user with that hashtag and any children of the hashtag in the tree.

To validate the usefulness of our semantic feature we performed a regression experiment to predict the response rate of each user with six emotions like anger, enjoyment, or disgust. The emotion data was manually annotated by a DARPA team created for the INCAS Program. We provide a baseline simple feature representing the counts the number of times a user interacts with each of the 1037 hashtags. Both the baseline and our semantic feature perform well with the regression with most emotions having R^2 above 0.5. The semantic feature statistically significantly outperforms the baseline feature on five out of six emotions using an F-test with a p value of 0.05.

The rest of the paper is organized as follows. Section 2 details related works. In Sect. 3, we present the dataset used for experimentation. Then, Sect. 4 describes the methodology used in our paper. The design of experiments and their results are presented in Sect. 5, and the conclusions are discussed in Sect. 6.

2 Related Works

Analyzing Twitter using semantic networks has been done in the past with various methods to determine relationships between hashtags and their trends. For example, [18] considered two hashtags to be semantically related if an individual tweet contained both hashtags in the text. Similarly, [9] created a semantic network based on word co-occurrence within tweets. However, [17] presented an approach using a bipartite network between users and hashtags where an edge between a user node and a hashtag node was added if the user tweeted the hashtag at least once. This bipartite network was then projected into a monopartite network of hashtags. This approach is more applicable to our purposes because it captures the latent social network of the dataset. In addition, in [17] the authors focus on a Twitter dataset taken from the 2018 Italian elections which is similar to our 2022 French election dataset.

Many studies have shown that semantic network features can be rich enough to use for regressions in a multitude of situations. In the field of psychology, semantic networks can be used to analyze a person's vocabulary to gain insight on cognitive states [4,7]. In terms of social media semantic networks, [10] used semantic networks generated from sentences as features for a time series regression to capture the volatility of the stock market. In general, these approaches involve creating a semantic network for each person or object in the study. The alternative approach is to create large-scale, singular semantic networks that can be used to describe all users. For example, [12] demonstrated a recommender system which used a large-scale word co-occurrence semantic network created from social media posts to recommend related social media posts to users. Such an approach might be better for analyzing users since it can take advantage of the nuanced relationships between different social media communities, which cannot be done with an approach that only generates an individual semantic network for each user.

3 Data

We applied these enrichments to a dataset provided by the DARPA INCAS program team that comprised 3.7 million French language tweets from 2022. This dataset was collected such that each tweet is relevant to the discussions that arose during the 2022 French presidential election. After pruning, this dataset contains 1037 hashtags and 389,187 users.

4 Methods

4.1 Semantic Network Generation

We performed several steps to prepare the Twitter data and create a semantic network.

Preprocessing. The corpus of Tweets was first cleaned by removing URLs with regular expression and French stop words using the NLTK Python library [5]. Each Tweet was tokenized by converting all words to lowercase, removing digit-only words, and removing punctuation, except for hashtags. After extracting a set of hashtags and corresponding occurrence counts, any hashtags with an occurrence count below the mean were removed from the set to focus on trendy hashtags.

Bipartite Graph Generation. Using this set of hashtags, a bipartite graph was constructed between users and the hashtags where an edge indicates an interaction between a user and a hashtag. Following the technique introduced in [17], we implemented the bipartite graph as an adjacency list where a set of interacted hashtags is stored with each user. A user is said to *interact* with a

trendy hashtag if the user retweets, quote retweets, comments under, or posts a tweet that contains the hashtag word with or without the hashtag symbol. We chose this relaxed approach because we consider situations such as "france" versus "#france" to be semantically identical. The resulting bipartite graph was projected along the hashtags as a weighted semantic network where each node represents a trendy Twitter hashtag, and the weighted edges represent the shared audience of users between two trendy hashtags.

Edge Pruning. Next, the bipartite graph was then converted into a maximum spanning tree (MST) to only consider the most important links between trendy hashtags. We had conducted multiple experiments with and without edge pruning and concluded that some form of edge pruning is essential for removing noisy edges. We tested a flat cutoff approach for excluding edges with a weight below a set cutoff, and the MST approach, achieving the best and most robust results with the MST. All graph operations were performed with the NetworkX Python library [11]. The implementation negates the weights of the edges and then follows Kruskal's [13] algorithm to build a minimum spanning tree. A visualization of the resulting MST can be seen in Fig. 1.

4.2 Semantic User Enrichment

Each user in the dataset $u \in U$ is assigned a set containing the trendy hashtags they had interacted with using the previously described interaction criteria. For each user set S_u, and for each trendy hashtag $t \in S_u$ where $S_u \subset V$ in the semantic network graph $G(V, E)$, each adjacency list corresponding to t is converted into an adjacency vector $\mathbf{a}_t = (a_{t1}, \ldots, a_{tn})$ where $n = |V|$, and

$$a_{ti} = \begin{cases} \frac{w(t,m(i))}{c(t)} & \text{if } e(t, m(i)) \in E \\ 1 & \text{if } m(i) = t \\ 0 & \text{otherwise} \end{cases}$$

where $w : E \to \mathbb{N}$ is the weight of the edge $e(u, v) \in E$, $m : [0, n] \to V$ that maps each index to a trendy hashtag, and $c : V \to \mathbb{N}$ maps each trendy hashtag to the number of users that have interacted with it. The set of vectors for each user is then summed element-wise and then normalized by dividing by the L^2 norm of the summed vector. The result for each user is a vector representing this user's interests in a trendy hashtag and related trendy hashtags weighted by the latent social network.

4.3 Baseline User Enrichment

To judge the utility of our semantic network enrichment, we devised a simpler baseline enrichment for comparison. Each user $u \in U$ is assigned a vector $\mathbf{a}_u = (a_{u1}, \ldots, a_{un})$ where $n = |V|$, and $a_{ui} = k(u, m(i))$ where

$$k(u, t) = \begin{cases} 1 & \text{if } u \text{ interacts with trendy hashtag } t \\ 0 & \text{otherwise} \end{cases}$$

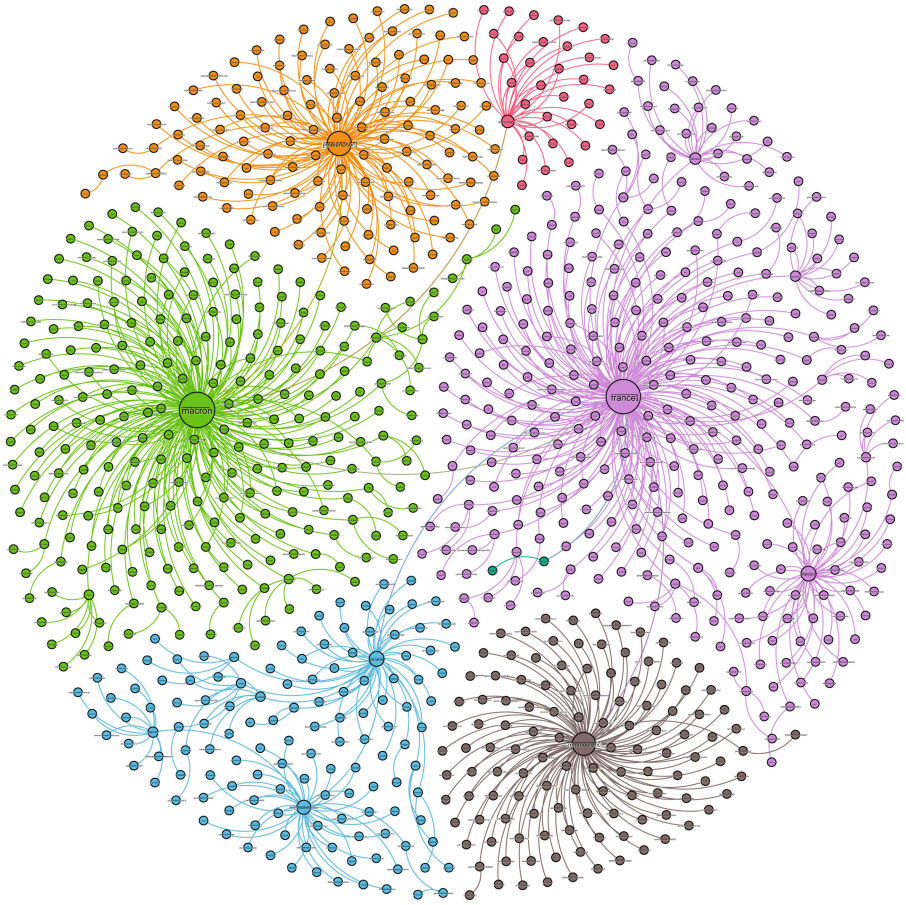


Fig. 1. Maximum-Spanning Tree of trendy hashtags in the 2022 French Presidential Election. Nodes represent trendy hashtags. Weighted edges between nodes represent the number of users that interacted with both trendy hashtags. Node size is based on weighted degree. Node color is based on modularity class after applying Louvain community detection and provided to help distinguish groups of similar nodes visually. From the root node #france, a hierarchy amongst trendy hashtags is formed, with clear distinction between different presidential candidates.

Each vector is normalized by dividing by the sum of the vector elements.

4.4 Regression Experiment

To compare the enrichments, we conducted an experiment to test the performance of the enrichments in a regression task. We decided to test if a user enrichment could be used to predict the average "emotions" for each user. Each tweet in the dataset was annotated by the DARPA team with an array of six

distinct emotions and an "other" value (representing fear, anger, enjoyment, sadness, disgust, surprise, and "none of the above" tag) where the sum of each array equals 1. We split the data into a training and testing period where the testing period is the final 2 weeks of the data and the training period covers the first 10 weeks. For every user's tweets, for both the training and testing periods we summed the emotion arrays, then divided the resulting array by its 1-norm, so that each array element follows $U(0, 1)$ and represents the probability of that user interacting with each emotion. Each user array was split into a set of emotion target variables, and each one was paired with the corresponding user enrichment method as the input variable. Only users with ≥ 10 tweets in the training period were included, resulting in 49,360 entries of input/target pairs for each emotion. Since this experiment is only meant to compare the different methods relative to each other with often minimal differences, we used the Scikit-learn implementation of linear regression [16]. To measure the performance of each regression we compute the R^2 value between the predicted emotion of all the users and the actual emotion of all the users.

5 Results

5.1 Experiment Results

Table 1. Regression experiment results. (*: semantic > baseline; $p < 0.05$)

Emotion	Baseline R^2	Semantic R^2
Fear	0.222	0.229
Anger*	0.567	0.574
Enjoyment*	0.634	0.648
Sadness*	0.266	0.277
Disgust*	0.501	0.514
Surprise*	0.082	0.098
None	0.416	0.423

Overall, there is a clear pattern of improved performance when using the semantic enrichment instead of the baseline as seen in Table 1. All specific emotions, except fear and "none of the above" tag, statistically significantly improved performance using the semantic method versus the baseline. Statistical significance was calculated using an F-test between the two models. The fear and none of the above regressions still had higher R^2 values for the semantic enrichment compared to the baseline enrichment but the increase was just not large enough to pass the F-test.

Interestingly, the baseline linear regression performed poorly on surprise emotion. The regression with the semantic enrichment performed significantly better, but still was weakly correlated with the actual response with an R^2 value

close to zero. Intuitively, people would have various positive or negative views towards certain political trendy hashtags, which would correlate with most of the emotions. However, surprise cannot easily be categorized as on the positive or negative binary spectrum, which could explain why the linear regression performed poorly in those cases. Previous research on sentiment analysis has also shown notably lower performance when predicting surprise [6,19]. A change to our method to distinguish hashtags written in all uppercase or with exclamation points at the end may improve performance on surprise.

Both enrichments had the best performance when being used to predict the enjoyment emotion with R^2 values of 0.634 and 0.648. Social media users are likely to frequently repeatedly mention the topics that give them enjoyment, so this result is not very surprising. Similar logic would apply to anger and disgust emotions and those regressions also performed well with R^2 values greater than 0.5.

5.2 Semantic Analysis

To analyze which trendy hashtags are most associated with improvement with the semantic enrichment, we decided to filter for the top 10% of users that saw the most improvement in prediction accuracy between the baseline regression and the semantic regression. This was determined by the mean absolute error in emotion predictions versus the ground truth. Then we compared trendy hashtag occurrence rates for the top 10% users with the hashtag occurrence rates for the rest of the users. All hashtag occurrence rates were calculated based on direct interactions, like the baseline enrichment. We then selected the top 10 trendy hashtags that saw the largest increase in occurrence rate between the top 10% of users and the rest of the users.

Since the presence of these trendy hashtags in Table 2 result in more accurate emotion predictions with the semantic enrichment, this suggests that the users engaging with these trendy hashtags tend to engage with other emotionally salient trendy hashtags, which would be more useful when predicting emotion levels. Given the severity of war, it would make sense that “Ukrainians” would be strongly connected to other highly emotionally salient trendy hashtags. A previous English language Twitter study about the Ukrainian war found that “Ukrainians” is a significant buzzword, so it is not surprising that the word reappears in French. In addition, that study identified the YouTube twitter account that was frequently mentioned in relation to the war, which would explain why it invokes strongly emotional trendy hashtags [20]. “Paris” on its own may not seem like it would be emotionally charged. However, looking at the children of this hashtag in the tree, “hidalgodemission”, “conseilparis”, and “saccageparis”, they are each related to an emotionally-charged movement to remove the mayor of Paris and rebuild some of the crumbling architecture in the city.

Table 2. Trendy hashtags with the largest increase in occurrence rate between the top 10% of users and the rest of users. Occurrence rate is the proportion of users that directly interacted with that trendy hashtag. The top 10% of users is computed based on the improvement in prediction accuracy between the baseline regression and the semantic regression for those users.

Trendy hashtag	Bottom 90% rate	Top 10% rate
Paris	0.274	0.420
Youtube	0.328	0.470
Europe	0.458	0.600
Lci	0.227	0.348
Passe	0.250	0.370
Ukrainians	0.287	0.404
Jeunes	0.282	0.397
Liberté	0.371	0.485
Nucléaire	0.284	0.394
Immigration	0.248	0.357

6 Conclusions and Future Work

We have connected semantic networks to the area of machine learning, demonstrating using a simple experiment that this can be used to consistently improve results on real-world data. To the best of our knowledge, this is the first time a semantic network feature for machine learning has been explored.

Future work can include specializing in such a framework to tackle specific problems. It is worth noting that the semantic network method used in this paper was designed with the constraints of the INCAS challenge in mind, which might not necessarily be the best way to utilize semantic networks when describing users in other situations. We are reporting these results simply to show that this method is a notable improvement over simpler approaches and it is worth investigating other applications in future work. One of the main drawbacks of this approach is the increased computational time associated with projecting the bipartite graph into a monopartite semantic network. However, there are distributed computing approaches to monopartite projection challenges, so this can be scaled for large scale applications involving many trendy hashtags [2]. Additionally, this computational time scales with the number of hashtags rather than the number of messages or users. Therefore, when the topic scope is kept narrow, this approach should scale better than approaches that rely on the number of messages or users. Conversely, if the topic scope is broadened then our approach may scale more poorly in comparison. In such situations we could explore the possibility of constructing multiple separate topic graphs to break-up the larger overall message-space.

During testing, we found that pruning small edges is important for removing noise from the final enrichments. We used the most aggressive pruning approach,

by using a maximum spanning tree to only retain the strongest edges. This has the disadvantage of removing connections between different communities within the graph. It is possible that using a more sophisticated pruning approach could improve the quality of using semantic networks in this manner. Future work can take inspiration from knowledge graph edge pruning methods, which can account for domain information of the trendy hashtags [8].

Acknowledgements. This work was partially supported by the DARPA INCAS Program under Agreement No. HR001121C0165 and by the NSF Grant No. BSE-2214216.

References

1. Almuayqil, S.N., Humayun, M., Jhanjhi, N.Z., Almufareh, M.F., Khan, N.A.: Enhancing sentiment analysis via random majority under-sampling with reduced time complexity for classifying tweet reviews. Electronics **11**(21) (2022). https://doi.org/10.3390/electronics11213624
2. Asadi, M., Ghadiri, N., Nikbakht, M.A.: A scalable method for one-mode projection of bipartite networks based on hadoop platform. In: 2018 8th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 237–242 (2018). https://doi.org/10.1109/ICCKE.2018.8566259
3. Asur, S., Huberman, B.A.: Predicting the future with social media. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, pp. 492–499 (2010). https://doi.org/10.1109/WI-IAT.2010.63
4. Beckage, N., Smith, L., Hills, T.: Semantic network connectivity is related to vocabulary growth rate in children. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 32 (2010)
5. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc.' (2009)
6. Buechel, S., Hahn, U.: Emotion analysis as a regression problem–dimensional models and their implications on emotion representation and metrical evaluation. In: ECAI 2016, pp. 1114–1122. IOS Press (2016)
7. Chan, A.S., Salmon, D.P., Butters, N., Johnson, S.A.: Semantic network abnormality predicts rate of cognitive decline in patients with probable Alzheimer's disease. J. Int. Neuropsychol. Soc. **1**(3), 297–303 (1995). https://doi.org/10.1017/S1355617700000291
8. Faralli, S., Finocchi, I., Ponzetto, S.P., Velardi, P.: Efficient pruning of large knowledge graphs. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 4055-4063. AAAI Press (2018)
9. Featherstone, J.D., Ruiz, J.B., Barnett, G.A., Millam, B.J.: Exploring childhood vaccination themes and public opinions on Twitter: A semantic network analysis. Telematics Inform. **54**, 101,474 (2020). https://doi.org/10.1016/j.tele.2020.101474. https://www.sciencedirect.com/science/article/pii/S0736585320301337
10. Fronzetti Colladon, A., Grassi, S., Ravazzolo, F., Violante, F.: Forecasting financial markets with semantic network analysis in the COVID-19 crisis. J. Forecasting (2020)
11. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using Network. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, pp. 11 – 15 (2008)

12. He, Y., Tan, J.: Study on SINA micro-blog personalized recommendation based on semantic network. Expert Syst. Appli. **42**(10), 4797–4804 (2015). https://doi.org/10.1016/j.eswa.2015.01.045, https://www.sciencedirect.com/science/article/pii/S0957417415000603
13. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proc. Am. Math. Soc. **7**(1), 48–50 (1956)
14. Kumari, S.: Impact of big data and social media on society. Global J. Res. Anal. **5**, 437–438 (2016)
15. Pagolu, V.S., Challa, K.N.R., Panda, G., Majhi, B.: Sentiment analysis of Twitter data for predicting stock market movements. CoRR abs/ arXiv: 1610.09225 (2016)
16. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
17. Radicioni, T., Saracco, F., Pavan, E., Squartini, T.: Analysing Twitter semantic networks: the case of 2018 Italian elections. Sci. Rep. **11**(1), 1–22 (2021)
18. Shi, W., Fu, H., Wang, P., Chen, C., Xiong, J.: #climatechange vs. #globalwarming: Characterizing two competing climate discourses on Twitter with Semantic Network and temporal analyses. Inter. J. Environ. Res. Public Health **17**(3) (2020). https://doi.org/10.3390/ijerph17031062, https://www.mdpi.com/1660-4601/17/3/1062
19. Tsakalidis, A., et al.: Building and evaluating resources for sentiment analysis in the Greek language. Lang. Resour. Eval. **52**, 1021–1044 (2018)
20. Vyas, P., Vyas, G., Dhiman, G.: RUemo-the classification framework for Russia-Ukraine war-related societal emotions on Twitter through Machine Learning. Algorithms **16**(2) (2023). https://doi.org/10.3390/a16020069

Rewiring Networks for Graph Neural Network Training Using Discrete Geometry

Jakub Bober[1], Anthea Monod[1(✉)], Emil Saucan[2], and Kevin N. Webster[1,3]

[1] Imperial College London, London SW7 2AZ, UK
a.monod@imperial.ac.uk
[2] ORT Braude College of Engineering, Karmiel, Israel
[3] FeedForward Ltd., London, UK

Abstract. Information over-squashing occurs under inefficient information propagation between distant nodes on networks, which can significantly impact graph neural network (GNN) training. Rewiring is a preprocessing procedure applied to the input network to mitigate this problem. In this paper, we investigate discrete analogues of various notions of curvature to model information flow on networks and rewire them. We show that classical notions of curvature achieve state-of-the-art performance in GNN training accuracy on a wide variety of real-world datasets. Moreover, these classical notions exhibit a clear advantage in computational runtime by several orders of magnitude.

Keywords: Discrete curvature · geometric deep learning · graph neural networks · graph rewiring · information over-squashing

1 Introduction

Data captured by structures beyond vectors living in Euclidean space are becoming increasingly abundant, thus, it is becoming increasingly important to develop methods to analyze them. When such data lack a rigorous metric structure, notions of shape and size of the data become useful to incorporate in their analysis—this is the premise of *geometric deep learning* [3]. Networks are an important example of non-Euclidean spaces lacking a natural metric structure, which are the focus of this work.

In this paper, we study the problem of *information over-squashing*, associated with training graph neural networks (GNNs) [1,21], which occurs when information does not flow efficiently between distant nodes on a graph. This problem tends to occur when there is heavy traffic passing through particular edges of a graph, known as the *bottleneck*. *Graph rewiring* is a common mitigating approach, which adds or suppresses edges on the input network to alleviate bottlenecks and improve information flow over a network. Recent pioneering work models network information flow using a new notion of discrete curvature—the *balanced Forman curvature* (BFC)—and uses it to rewire graphs prior to training GNNs, yielding the current state-of-the-art for GNN training in the presence of bottlenecks [21].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 225–236, 2024.
https://doi.org/10.1007/978-3-031-53468-3_19

Discrete curvatures, as discretizations of classical notions from smooth geometry, have been actively studied in recent decades (e.g., [12]) and have been shown to be useful when applying geometric methods to statistics and machine learning tasks for data with discrete geometric structure, such as network learning by sampling (e.g., [2,18]). We return to these original classical notions to study their performance in graph rewiring as in [21], in place of the BFC. We systematically test and compare several classical discrete curvature notions against the BFC on many benchmarking datasets and find that these classical notions achieve state-of-the-art performance in terms of accuracy for GNN training. Moreover, computing these classical discrete curvatures is much quicker, running several orders of magnitude faster than the state-of-the-art.

Related Work. Graph diffusion convolution (GDC) is an alternative graph rewiring approach that uses a discretization of the gas diffusion equation to model the propagation of information on a network [9]. There also exist other non-rewiring bottleneck alleviation methods. Curvature GNNs (CGNNs), in particular, assign specific weights to graph edges as a measure of information flow with weights determined by discrete curvature [10].

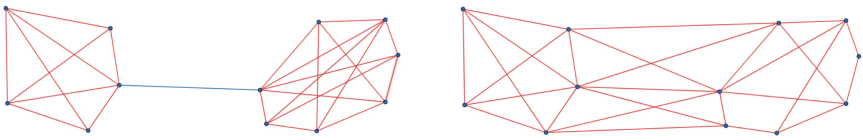
2 Background and Preliminaries

The main difference between the traditional deep neural networks and GNNs has to do with the *message passing* algorithm [7]. In message passing, at each layer and for each node, features from the neighboring nodes are aggregated before updating the features of the target node. The principle concern of over-squashing is that the influence of important node features may be too small and eventually have minimal or no impact on features of distant nodes on the network when message passing over the GNN. When propagating information from a node in a source component to a node in the target component, over-squashing is likely to happen as the information is crowded or "squashed" together with all other node features from the source component, which happens on the edge connecting the two components called a *bottleneck*.

Bottlenecks may be alleviated with *graph rewiring*, which better supports the bottleneck and provides alternative access routes between components to reduce the risk that features become crowded out (over-squashed); see Fig. 1. Edges that have little impact on information flow in the graph can be deleted to control the size of the graph.

2.1 Discrete Geometry and Curvature

The *Ricci curvature* quantifies how much a Riemannian manifold locally differs from a Euclidean space. It determines whether two geodesics shot in parallel from two nearby points on a given manifold converge, remain parallel, or diverge along the manifold. The curvature is positive if the geodesics converge to a single point; zero, if the geodesics remain parallel; and negative, if the geodesics diverge.



(a) (b)

Fig. 1. Graph rewiring reduces over-squashing. (a): A graph with a bottleneck (blue edge). (b): A rewiring that alleviates the bottleneck.

The quicker the convergence or divergence, the larger the Ricci curvature. Ricci curvature can be used to smooth a manifold via the *Ricci flow*, described by a partial differential equation (PDE) [4].

In the discrete setting of meshes or networks, the PDE describing Ricci flow becomes an ordinary differential equation, thus the flow is reversible.

From Manifolds to Graphs. In some cases, there is a natural reduction of manifolds to graphs, e.g., images can be represented in a discrete manner by meshes, which can be seen as 4-regular graphs.

For positive, zero, and negative curvature, there exist natural graph analogies. For a positively curved sphere, a clique is a suitable representation: two parallel geodesics shot from two nearby points on a sphere meet at the top of a sphere, while two edges from two adjacent points in a clique can meet at a common node to create a triangle. For a flat plane, a rectangular grid is an appropriate graphical representation: parallel lines on a plane remain parallel forever. Finally, a negatively curved hyperbolic manifold may be represented by a binary tree.

Discrete Curvature. There is no single established definition of discrete curvature. Here, we outline the first and best-known discrete curvatures historically proposed for networks. The driving motivation is that the bottlenecks will have the lowest discrete curvature in the graph.

We work with undirected networks and list curvatures for undirected networks. We also work with unweighted networks, which allow for combinatorial properties of graphs that lend computational benefits.

1D Forman Curvature. Perhaps the most basic and one of the first notions of discrete curvature is the one proposed by Forman [6,19].

Definition 1. *For two nodes v_1, v_2 in a graph and an edge e between them, the general 1D Forman curvature of e is*

$$F_{\text{full}}(e) = w_e \left(\frac{w_{v_1}}{w_e} + \frac{w_{v_2}}{w_e} - \sum_{e_{v_1} \sim e, e_{v_2} \sim e} \left[\frac{w_{v_1}}{\sqrt{w_e w_{e_{v_1}}}} + \frac{w_{v_2}}{\sqrt{w_e w_{e_{v_2}}}} \right] \right), \quad (1)$$

where $e_{v_1} \sim e$ and $e_{v_2} \sim e$ denote the edges other than e that are adjacent to nodes v_1 and v_2 respectively; w_e, $w_{e_{v_1}}$, and $w_{e_{v_2}}$ denote the weights of edges e,

e_{v_1} and e_{v_2} respectively; and w_{v_1} and w_{v_2} denote the weights of the nodes v_1 and v_2 respectively.

For unweighted graphs, the weights of all nodes and edges are set to 1 and (1) becomes simply

$$F(e) = 4 - (\deg(v_1) + \deg(v_2)), \tag{2}$$

where $\deg(x)$ is the degree of node x. In our work, we compute the 1D Forman curvature using (2) which is a very simple expression and extremely fast to compute, and is concerned only by the degrees of the endpoints of the edge under consideration.

The drawback of this simplicity is that it is not always very descriptive, since under combinatorial weights, the 1D Forman curvature gives information only about the number of edges directly connected to the edge under consideration. It generally assigns lower curvature to clique-like components of the graph rather than tree-like components, since an edge in a clique is connected directly to all of the other edges in the clique, while in a binary tree it is only directly connected to 3 other edges.

Augmented Forman Curvature. The *augmented Forman curvature* [15] aims to mitigate the drawbacks of 1D Forman curvature.

Definition 2. *For two nodes v_1, v_2 in a graph and an edge e between them, the* augmented Forman curvature *or* 2D Forman curvature *is given by:*

$$F^{\#}_{\text{full}}(e) = w_e \left[\left(\sum_{e<f} \frac{w_e}{w_f} + \sum_{v<e} \frac{w_v}{w_e} \right) - \sum_{\hat{e} \| e} \left| \sum_{\hat{e}, e<f} \frac{\sqrt{w_e \cdot w_{\hat{e}}}}{w_f} \sum_{v<e, v<\hat{e}} \frac{w_v}{\sqrt{w_e \cdot w_{\hat{e}}}} \right| \right], \tag{3}$$

where $a \| b$ denotes that a is parallel to b, i.e., a and b have a common higher or lower dimensional graph face; $a < b$ denotes that a is a graph face of b; and the rest of the notation is as in Definition 1 (here the faces are also weighted).

Following [15], we may consider solely 3-cycles and again, under combinatorial weights, (3) reduces to

$$F^{\#}(e) = F(e) + 3t, \tag{4}$$

where $t = |N(v_1) \cap N(v_2)|$ is the number of triangles containing the edge $e = (v_1, v_2)$ under consideration.

The idea is that the curvature $F^{\#}$ (4) increases in relation to F if an edge is contained in some triangles. More precisely, the 3 factor of t in (4) guarantees that edges creating a triangle together with e contribute positively to the curvature. If an edge is not a member a triangle with e, it contributes negatively to the augmented Forman curvature by decreasing it by 1, just as in the 1D version. Hence, the augmented version maintains a balance between the growth of degrees of endpoints and creation of 3-cycles.

Haantjes Curvature. Less common than the Forman curvatures, the *Haantjes curvature* [8] has the simplest and most intuitive definition.

Definition 3. *Consider a graph where all weights are set to 1 (i.e., the combinatorial case). For two nodes v_1, v_2 in a graph and an edge e between them, the* Haantjes curvature *is given by $\kappa_H^2(e) = t$, where t is as in (4).*

The Haantjes curvature is a metric curvature, thus in the network case it takes into account solely edge weights. Definition 3 is commonly used in graphics settings and simply counts the triangles adjacent to a given edge. The Haantjes curvature is typically higher for clique-like components of a graph than for tree-like components. It is trivially nonnegative, which is also in contrast with 1D Forman, where the majority of edges usually have negative curvature. The augmented Forman curvature can be thought of as a balance between 1D Forman and Haantjes curvatures.

Balanced Forman Curvature. The BFC proposed by [21] aims to balance between computational complexity and the richness of structural information associated with neighboring edges. It takes into account 3- and 4-cycles, as well as "loose" neighboring edges, i.e., those that do not create 3- or 4-cycles.

3 Methods and Experimental Design

In this section, we outline the algorithm that we will use to identify network bottlenecks and perform graph rewiring based on discrete curvature; the datasets we will study; and our experimental setup.

3.1 Stochastic Discrete Ricci Flow

Algorithm 1. Stochastic Discrete Ricci Flow (SDRF)

Input: graph G, temperature $\tau > 0$, max number of iterations, discrete curvature Curv, optional Curv upper-bound C^+
repeat
  for edge $i \sim j$ with minimal discrete curvature $\text{Curv}(i, j)$ **do**
    Calcukate vector $\boldsymbol{x}$ where $x_{kl} = \text{Curv}_{kl}(i, j) - \text{Curv}(i, j)$, the improvement to $\text{Curv}(i, j)$ from adding edge $k \sim l$ where $k \in N(i) \cup \{i\}, l \in N(j) \cup \{j\}$;
    Sample index k, l with probability $\text{softmax}(\tau \boldsymbol{x})_{kl}$ and add edge $k \sim l$ to G.
  end for
  Remove edge $i \sim j$ with maximal discrete curvature $\text{Curv}(i, j)$ if $\text{Curv}(i, j) > C^+$.
until convergence or max iterations reached

The *stochastic discrete Ricci flow* (SDRF) algorithm [21] is a discretization of Ricci flow that is a graph rewiring algorithm and will be implemented in our

experimental work. It operates in the same spirit as Ricci flow, where regions of negative or low curvature are identified and compensated by an opposite effect depending on the negativity in order to smooth the manifold. Additionally, it incorporates a mechanism to prevent a blow-up on the graph size. The algorithm intakes a graph and produces another graph where the regions of the most negatively curved edges of the input graph are augmented with additional edges to increase the curvature at those regions.

At each iteration, the algorithm chooses the edge with the lowest curvature; candidate edges to add to support the lowest curvature edge; and the edge to add from candidates with softmax probability (regulated with a temperature parameter τ) to increase curvature, where this latter value is calculated as the difference between curvature of the lowest curvature edge before and after adding the support edge. The algorithm then chooses the edge with the highest curvature and, if this curvature value surpasses a certain threshold, removes this edge from the graph, ensuring a bound on the size of the graph. The process repeats until either the convergence is reached (no additional candidates and no edges to remove) or the maximum number of iterations is reached.

3.2 Datasets

We studied the following 12 benchmarking datasets in our experimental study in a supervised learning task of node classification, whose details are summarized in Table 1: `Cora` [11] and `Citeseer` [16], which are large citations datasets containing information about the presence of specific words in publications; `Pubmed` [13], a large citations dataset containing information about diabetes of patients classified into one of three classes; `Cornell`, `Texas`, and `Wisconson` [5], which are small datasets containing information about webpages collected from computer science departments of corresponding universities; `Chameleon`, `Squirrel` [14], and `Actor` [20], which are large datasets based on the Wikipedia networks; `Computers` and `Photo` [17], which are large e-commerce (Amazon) datasets; and finally, `Coauthor CS` [10], which is a large citation dataset with papers in computer science. The last 3 datasets were not evaluated by [21].

Table 1. Characteristics of datasets studied.

	`Cora`	`Citeseer`	`Pubmed`	`Cornell`	`Texas`	`Wisconsin`
Nodes	2485	2120	19717	140	135	184
Edges	5069	7358	44324	219	251	362
Features	1433	778	500	1703	1703	1703
Classes	7	6	3	5	5	5

	`Chameleon`	`Squirrel`	`Actor`	`Computers`	`Photo`	`Coauthor CS`
Nodes	832	2186	4388	13381	7487	18333
Edges	12355	65224	21907	245778	119043	81894
Features	2323	2089	931	767	745	6805
Classes	5	5	5	10	8	15

3.3 Experimental Setup

We tested the performances of no curvature (i.e., no rewiring); 1D Forman curvature; augmented Forman curvature; Haantjes curvature; and balanced Forman curvature in the SDRF algorithm for graph rewiring. The implementation of the SDRF algorithm was taken from the repository associated with [21]. Other design parameters such as data loading, selection of largest connected component, network type, hyperparameters, and seeds have been set following [21].

4 Results: GNN Training with Graph Rewiring

In this section, we report the results of GNN training with graph rewiring performed with the various discrete curvatures discussed previously in Sect. 2 as well as the BFC proposed by [21] as a comparative benchmark. We discuss performance in terms of accuracy and computational runtime in seconds.

4.1 Accuracy

Each experiment was run for 100 seeds; we report 95% confidence intervals of mean accuracies using a z-score of 1.96. For reference and performance comparison, the 95% confidence intervals for the SDRF-rewiring using BFC reported by [21] are also given for those relevant datasets.

The best two results are highlighted for each dataset in each accuracy table: the best one in red bold, the second best in black bold (excluding the reported BFC results from [21] for reference). The *None* curvature row represents results without any rewiring. OOM indicates that the out of memory error has occurred. N/A in the reference BFC row for `Computers`, `Photo`, and `Coauthor CS` datasets indicates that there are no reference results for these datasets as these datasets were not studied by [21].

We see that SDRF rewiring generally improves training performance. In particular, we note that performance for the classical curvatures is generally better than the performance without any rewiring, and often better than performance of BFC. For some results in Table 2, the simplest form of curvature—the 1D Forman curvature—tends to give the best results. This indicates that edges with large sums of degrees are the graph bottlenecks and suffer from over-squashing. The results for Haantjes curvature are the best for some of the other datasets, which suggests that association with many 3-cycles helps an edge reduce over-squashing. The augmented Forman curvature also yields best results for certain experiments (with less frequency), which suggests that maintaining the balance between the two metrics may reduce over-squashing most effectively.

Note, however, that the experiments upon rerunning yielded results that differ quite significantly, especially for small datasets (`Cornell`, `Texas`, `Wisconsin`). For example, Table 2 shows that the Haantjes curvature seems to generally bring the best results in the first run, while the augmented Forman curvature performs best in the second run. More importantly, it is often the case that the corresponding results (dataset–curvature pairs) for different rewirings for the two runs are

often not within the respective 95% confidence intervals, indicating a lack of robustness of the results. One explanation could be overfitting of the average accuracy to one instance of the SDRF rewiring, which can significantly impact the average performance. This is especially true for the small datasets, for which the rewiring of multiple edges can have a greater impact on the graph structure than for larger datasets. The results for these datasets also differ significantly between each curvature type, and compared to no rewiring. Moreover, the BFC results differ more significantly for these datasets than for others with respect to the reference BFC results.

To further investigate the intuition that adding or deleting edges on smaller graphs impact the overall graph structure more significantly, we re-ran experiments for `Cora`, `Citeseer`, `Cornell`, `Texas` and `Wisconsin` datasets with rewiring for each seed. These datasets were selected given that rewiring was the fastest (as will be discussed further on in considering computational runtime).

Table 2. 95% confidence intervals of mean accuracies for given datasets and curvature types given in percentages of two experimental runs (first two tables for the first run, last two for the second).

	`Cora`	`Citeseer`	`Pubmed`	`Cornell`	`Texas`	`Wisconsin`
None	**81.65 ± 0.25**	72.14 ± 0.31	77.74 ± 0.40	48.50 ± 0.60	59.19 ± 0.38	50.24 ± 0.54
1D	81.15 ± 0.24	**72.17 ± 0.28**	**77.76 ± 0.37**	52.75 ± 0.82	**64.59 ± 1.11**	**52.70 ± 0.71**
Augmented	**81.56 ± 0.24**	72.12 ± 0.30	77.70 ± 0.40	55.43 ± 0.62	**65.48 ± 1.23**	52.62 ± 0.74
Haantjes	81.55 ± 0.25	**72.19 ± 0.30**	**77.75 ± 0.38**	**56.29 ± 0.50**	63.33 ± 0.94	**55.81 ± 0.77**
BFC	81.38 ± 0.25	72.09 ± 0.28	OOM	**58.39 ± 0.64**	61.11 ± 0.68	48.86 ± 0.91
Reference BFC	82.76 ± 0.23	72.58 ± 0.20	79.10 ± 0.11	57.54 ± 0.34	70.35 ± 0.60	61.55 ± 0.86
	`Chameleon`	`Squirrel`	`Actor`	`Computers`	`Photo`	`Coauthor CS`
None	**47.38 ± 0.45**	**38.16 ± 0.32**	27.82 ± 0.24	41.74 ± 1.41	**56.4 ± 2.85**	**90.89 ± 0.11**
1D	44.88 ± 0.43	36.83 ± 0.27	**29.41 ± 0.26**	42.24 ± 1.58	54.93 ± 3.46	90.83 ± 0.11
Augmented	43.54 ± 0.88	36.75 ± 0.25	**29.81 ± 0.30**	**42.93 ± 1.56**	54.44 ± 3.01	**90.89 ± 0.11**
Haantjes	46.14 ± 0.55	36.59 ± 0.26	29.36 ± 0.24	**42.95 ± 1.74**	**56.74 ± 3.12**	90.86 ± 0.10
BFC	**46.92 ± 0.73**	**37.82 ± 0.36**	29.11 ± 0.25	41.55 ± 1.91	54.29 ± 3.13	OOM
Reference BFC	44.46 ± 0.17	37.67 ± 0.23	28.35 ± 0.06	N/A	N/A	N/A
	`Cora`	`Citeseer`	`Pubmed`	`Cornell`	`Texas`	`Wisconsin`
None	81.55 ± 0.23	72.21 ± 0.29	**77.90 ± 0.36**	48.11 ± 0.60	59.33 ± 0.40	49.95 ± 0.49
1D	81.10 ± 0.24	**72.45 ± 0.29**	**77.90 ± 0.35**	51.00 ± 0.88	**68.07 ± 1.09**	54.51 ± 0.84
Augmented	**81.57 ± 0.25**	**72.22 ± 0.27**	77.89 ± 0.38	53.89 ± 0.63	**64.81 ± 1.20**	**56.49 ± 0.79**
Haantjes	**81.56 ± 0.24**	72.10 ± 0.28	77.71 ± 0.41	**57.18 ± 0.57**	64.78 ± 1.15	55.86 ± 0.76
BFC	81.25 ± 0.25	72.04 ± 0.29	OOM	**54.61 ± 0.50**	58.37 ± 0.67	**56.19 ± 0.84**
Reference BFC	82.76 ± 0.23	72.58 ± 0.20	79.10 ± 0.11	57.54 ± 0.34	70.35 ± 0.60	61.55 ± 0.86
	`Chameleon`	`Squirrel`	`Actor`	`Computers`	`Photo`	`Coauthor CS`
None	**46.86 ± 0.44**	**38.25 ± 0.33**	27.69 ± 0.22	**42.45 ± 1.55**	53.39 ± 2.75	**90.90 ± 0.10**
1D	44.99 ± 0.40	36.49 ± 0.29	**29.66 ± 0.26**	41.11 ± 1.86	**55.57 ± 3.14**	90.82 ± 0.10
Augmented	42.69 ± 0.65	36.70 ± 0.26	**29.98 ± 0.25**	41.97 ± 1.71	**56.19 ± 2.96**	**90.90 ± 0.12**
Haantjes	45.97 ± 0.51	36.83 ± 0.24	29.52 ± 0.21	**42.38 ± 1.60**	55.34 ± 2.93	90.88 ± 0.11
BFC	**46.62 ± 0.70**	**37.61 ± 0.34**	29.34 ± 0.28	42.11 ± 1.65	54.51 ± 2.89	OOM
Reference BFC	44.46 ± 0.17	37.67 ± 0.23	28.35 ± 0.06	N/A	N/A	N/A

Table 3 presents results from two runs with rewiring for every seed, which are shown to be significantly more robust. The sizes of the 95% confidence intervals

are comparable to those reported previously in Table 2, but only two pairs of corresponding runs are not contained in the 95% confidence intervals of one another (`Cornell`–Haantjes and `Texas`–BFC). As there are $5 \cdot 5 = 25$ dataset–curvature pairs for which the experiments were run, the mean results are indeed robust and it is reasonable to consider the results as close to being independent and identically distributed (i.i.d.): the probability that two or more out of 25 means of i.i.d. random variables are not within the corresponding 95% confidence intervals is $1 - 25 \cdot 0.05 \cdot 0.95^{24} \approx 0.635 = 63.5\%$, which is high.

Furthermore, the results of these additional experiments are significantly worse than the reference BFC results. This is likely due to the accuracies for differently rewired graphs having been averaged out, as opposed to using the rewiring with the best validation accuracy for the benchmarking. In contrast, the results in Table 2 are slightly better for some dataset–curvature pairs than the reference BFC results, and sometimes slightly worse. When actually using the framework in practice, for the best results, the training can be performed for several different seeds and the model with the best validation accuracy can be chosen with the most effective rewired graph structure.

Table 3. 95% confidence intervals of selected datasets with graph rewiring for each seed given in percentages run twice.

	`Cora`	`Citeseer`	`Cornell`	`Texas`	`Wisconsin`
None	81.63 ± 0.24	72.13 ± 0.29	48.04 ± 0.60	59.74 ± 0.36	50.11 ± 0.53
1D	81.15 ± 0.26	**72.14 ± 0.31**	53.39 ± 0.81	**67.00 ± 1.28**	**55.54 ± 0.89**
Augmented	**81.64 ± 0.25**	72.05 ± 0.29	**54.93 ± 0.59**	**64.56 ± 1.15**	55.49 ± 0.82
Haantjes	**81.64 ± 0.24**	**72.19 ± 0.33**	**56.50 ± 0.60**	62.96 ± 0.92	**55.95 ± 0.72**
BFC	81.18 ± 0.27	72.12 ± 0.29	53.07 ± 0.74	59.19 ± 0.69	54.24 ± 0.93
Reference BFC	82.76 ± 0.23	72.58 ± 0.20	57.54 ± 0.34	70.35 ± 0.60	61.55 ± 0.86
	`Cora`	`Citeseer`	`Cornell`	`Texas`	`Wisconsin`
None	**81.56 ± 0.23**	**72.24 ± 0.29**	48.46 ± 0.56	59.48 ± 0.40	49.97 ± 0.52
1D	81.24 ± 0.23	**72.30 ± 0.29**	52.75 ± 0.80	**67.74 ± 1.26**	**55.62 ± 0.79**
Augmented	**81.69 ± 0.22**	72.23 ± 0.30	**55.39 ± 0.68**	**64.93 ± 1.10**	55.27 ± 0.79
Haantjes	81.49 ± 0.24	72.21 ± 0.27	**55.61 ± 0.58**	63.11 ± 1.05	**56.08 ± 0.82**
BFC	81.07 ± 0.25	72.01 ± 0.32	53.00 ± 0.73	60.30 ± 0.80	54.59 ± 0.88
Reference BFC	82.76 ± 0.23	72.58 ± 0.20	57.54 ± 0.34	70.35 ± 0.60	61.55 ± 0.86

We summarize the test results for rewiring instances and model parameters pairs that achieved the best validation accuracy in the experiments reported in the second run from Table 3 in Table 4. Only the second run is considered, but this does not have a significantly negative impact on the robustness of the results, since, as previously justified, the results in Table 3 are robust.

The main conclusion from these experiments is that there is no clear curvature that has overall better mean performance across the multiple datasets, but it is reasonable to deduce that using the classical curvatures for SDRF-based rewiring can lead to significant performance improvement, often achieving better

Table 4. Percentage accuracy for the best rewiring cases from Table 3.

	Cora	Citeseer	Cornell	Texas	Wisconsin
None	82.34	**74.19**	50.0	51.85	51.35
1D	82.23	70.48	**60.71**	**74.07**	54.05
Augmented	**83.35**	**74.19**	**57.14**	**74.07**	**59.46**
Haantjes	**83.55**	73.23	53.57	66.67	**59.46**
BFC	82.84	74.03	56.94	62.96	54.05
Reference BFC	82.76	72.58	57.54	70.35	61.55

results than the BFC. For every dataset, performing the SDRF-based rewiring almost always yields the best test accuracy when using one of the three classical curvature types, compared to BFC, although no rewiring may also yield the best results. Often, the best test accuracies were achieved using classical curvatures.

4.2 Computational Runtime

We measure the runtime for one rewiring process per curvature type and per dataset; the measurements are given in Table 5. The runtimes here are reported for only one instance for each dataset and each curvature type, in order to avoid influences of spurious computational issues such as CPU and GPU occupancy with other processes, which would become much more significant with repeated iterations. Here, the interest is rather in the comparison between longer computation times which shows the difference in computational complexity at scale.

From these results, we see that all of the classical discrete curvatures studied have a significantly shorter computation time than the BFC. The slowest among the three classical curvatures is the augmented Forman curvature, at scale. This is expected, as it essentially needs to do the same calculations as 1D Forman and Haantjes curvatures combined (computation of degrees of endpoints and adjacent triangles for each edge).

For the `Computers` and `Photo` datasets, however, the computation of the augmented Forman curvature took longer than the computation of BFC. This suggests that for some types of graphs, possibly for bigger or more dense graphs (notice from Table 1 that the edges to nodes ratio is very high for these two datasets), the BFC computation can outperform the augmented Forman curvature computation in terms of computation time. Nevertheless, the 1D Forman and Haantjes curvatures are still quicker to compute.

Table 5. Computation times of the SDRF rewiring given in seconds.

	Cora	Citeseer	Pubmed	Cornell	Texas	Wisconsin
1D	5.86	5.31	53.57	0.34	0.41	0.61
Augmented	6.16	5.48	107.55	0.19	0.21	0.49
Haantjes	2.88	4.27	39.65	0.13	0.13	0.14
BFC	27.64	12.26	OOM	34.95	21.2	21.24

	Chameleon	Squirrel	Actor	Computers	Photo	Coauthor CS
1D	86.51	900	418.06	4369.07	853.52	47.15
Augmented	251.25	901.71	872.78	10504.34	2262.72	88.10
Haantjes	53.58	531.31	105.36	462.14	139.97	30.12
BFC	1627.61	2006.78	5121.31	6431.32	1274.44	OOM

5 Discussion

We systematically and comprehensively studied various classical and novel discrete curvatures in mitigating the over-squashing problem in training GNNs. Following [21], we adapted discretizations of Ricci curvature and Ricci flow to model information flow and bottleneckness of a network, respectively. In [21], the BFC was proposed as a discretization of Ricci curvature, while the SDRF algorithm was proposed as a discretization of Ricci flow. We tested a wide range of classical discrete curvatures against the BFC in implementing the SDRF algorithm. We found that more classical curvatures were able to achieve performance of the same order as the BFC in training accuracy and, at times, outperformed the BFC. Moreover, they far outperformed it in computational runtime. We thus found that the impact of the contribution by [21] lies in the SDRF algorithm, rather than the BFC. We conclude that almost any of the more classical discrete curvatures may be used over the BFC together with the SDRF algorithm for more efficient computational runtimes.

Building on our study, future work may take into account directedness of the graphs in the SDRF and other rewiring methods. Also, alternative non-rewiring, discrete geometric approaches to mitigating over-squashing may be explored, such as CGNNs [10]. Here, other computational geometric notions for networks may be investigated, such as those arising from topological data analysis, where persistent homology concurrently captures the topology of data as well as its integral geometry. Such an approach would be particularly interesting when the goal is to preserve the topology of a graph, as the CGNN does.

References

1. Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications. arXiv preprint arXiv:2006.05205 (2020)
2. Barkanass, V., Jost, J., Saucan, E.: Geometric sampling of networks. J. Complex Netw. **10**(4), cnac014 (2022)

3. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017)
4. Chow, B., Knopf, D.: The Ricci Flow: An Introduction: An Introduction, vol. 1. American Mathematical Soc. (2004)
5. Craven, M., McCallum, A., PiPasquo, D., Mitchell, T., Freitag, D.: Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science (1998)
6. Forman, R.: Bochner's method for cell complexes and combinatorial ricci curvature. Discret. Comput. Geom. **29**(3), 323–374 (2003)
7. Gilmer, J., , S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y.W. (eds.), Proceedings of the 34th International Conference on Machine Learning, vol. 70. Proceedings of Machine Learning Research, 06–11 Aug, pages 1263–1272. PMLR (2017)
8. Haantjes, J.: Distance geometry. curvature in abstract metric spaces. Proc. Kon. Ned. Akad. V. Wetenseh., Amsterdam **50**, 302–314 (194
9. Klicpera, J., Weißenberger, S., Günnemann., S.: Diffusion improves graph learning. arXiv preprint arXiv:1911.05485 (2019)
10. Li, H., Cao, J., Zhu, J., Liu, Y., Zhu, Q., Wu, G.: Curvature graph neural network. arXiv preprint arXiv:2106.15762 (2021)
11. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Inf. Retrieval **3**(2), 127–163 (2000)
12. Najman, L., Romon, P. (eds.): Modern Approaches to Discrete Curvature. LNM, vol. 2184. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58002-9
13. Namata, G., London, B., Getoor, L., Huang, B., Edu, U.: Query-driven active surveying for collective classification. In: 10th International Workshop on Mining and Learning with Graphs, vol. 8, p. 1 (2012)
14. Rozemberczki, B., Allen, C., Sarkar, R.: Multi-scale attributed node embedding. J. Complex Netw. **9**(2), cnab014 (2021)
15. Samal, A., Sreejith, R., Gu, J., Liu, S., Saucan, E.: Comparative analysis of two discretizations of ricci curvature for complex networks. Sci. Rep. **8**, 8650 (2018)
16. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
17. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of graph neural network evaluation. arXiv preprint arXiv:1811.05868 (2018)
18. Sigbeku, J., Saucan, E., Monod, A.: Curved markov chain monte carlo for network learning. In: Benito, R.M., Cherifi, C., Cherifi, H., Moro, E., Rocha, L.M., Sales-Pardo, M. (eds.) Complex Networks & Their Applications X. pp, pp. 461–473. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-93413-2_39
19. Sreejith, R., Mohanraj, K., Jost, J., Saucan, E., Samal, A.: Forman curvature for complex networks. J. Stat. Mech: Theory Exp. **2016**(6), 063206 (2016)
20. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 807–816 (2009)
21. Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522 (2021)

Rigid Clusters, Flexible Networks

Gail Gilboa Freedman(✉)

Reichman University, Herzliya, Israel
gail.gilboa@runi.ac.il
https://www.runi.ac.il/en/faculty/ggilboa

Abstract. In scenarios where objects are characterized by a combination of rigid and flexible features, we consider the problem of identifying a natural set of rigid clusters, along with a network model of flexible states per cluster. Our approach proves effective within the Allais paradox context. Our algorithm, applied to data collected in an experiment, identified personality clusters and emotion states within each cluster. This model outperforms alternative clustering models in capturing information regarding participants' choices. Beyond the current scope, our approach is applicable to other data-sets with combined rigid-flexible attributes. Beyond prediction, a strategy that aims to achieve a result by influencing a flexible state holds the promise of enhanced effectiveness when it is tailored to a cluster.

Keywords: clustering · partition · community · modeling · state model · machine learning · Symmetric Uncertainty · Allais paradox

1 Introduction

Predictive models hold significance across behavioral, operational, financial, and various other domains. Machine Learning algorithms construct a predictive model by utilizing a training dataset to identify meaningful relationships between predicting features and a predicted label.

To illustrate, consider the realm of decision-making. Human choices are influenced by various factors, including the decision-maker's personality and the expected emotional response to potential decision results. A conventional approach to solving this decision prediction problem entails uncovering distinct causal relationships and elucidating how personality leads to specific emotional expectations, which subsequently influence a decision. Nevertheless, such an approach tends to obscure the inquiry into how personality anticipates patterns of emotional fluctuations and the resulting array of potential decisions.

Consider, for example, a decision-maker who confronts two job options-one guarantees a fixed salary, while the other offers higher yet uncertain earnings. The decision involves a mental analysis of how the decision-maker will feel when earning a low salary, a high salary, or no salary. The expected emotions influence their decision. It is challenging to predict decisions that are emotion-related because the decision-maker expects different emotions to result from his decision. Moreover, expected emotions are often mixed, and their precise effect

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 237–249, 2024.
https://doi.org/10.1007/978-3-031-53468-3_20

on decision-making is unclear. Initially, modeling emotion-driven decisions may seem daunting. We explore whether it is possible to identify an underlying structure in this seeming chaos.

The case of modeling the personality-emotion-decision process calls for the use of personality traits to cluster individuals into distinct types, showcasing diverse emotion-decision patterns-instead of showcasing a generalized emotion-decision pattern for the whole population. The current research specifically revolves around this concept, and its essence can be encapsulated by the expression 'model of models,' which entails creating a clustering model, in which each cluster illustrates how observations that are assigned into this cluster shift across different states and how this arrangement impacts the labeling process.

Using a naive approach involving clustering based on rigid features and then clustering each subgroup based on flexible features frequently leads to the neglect of organic patterns of flexible states within distinct clusters. Instead, our algorithm begins by identifying states and subsequently grouping them into clusters based on rigid features.

Our proposed *R*&*F* algorithm assumes training data in which each data point includes 'stable parameters', 'fluctuating parameters', and 'label'. The algorithm identifies a clustering model that ensures that objects within each cluster are not merely similar but are grouped for the efficiency of subsequent modeling. The algorithm uses 'stable parameters' and identifies 'type clusters,' each encompassing a unique 'network of states.' The states are defined in terms of the remaining 'fluctuating' parameters.'

To illustrate the practicality of our proposed algorithm, we turn to the example of the personality-emotion-decision problem discussed earlier, specifically within the context of the Allais paradox. This paradox highlights the 'certainty effect,' where the presence of a certain outcome amplifies risk aversion. In such dilemmas, individuals exhibit irrational choices, which prompts an exploration of patterns connecting this bounded rationality with personality and emotions.

We conducted an experiment, gathering data in the context of the Allais-like narrative, and presenting choices between hypothetical games with uncertain results. In this context, participants were asked to report their preferences, alongside their anticipated emotions for each potential outcome. Participants also completed two psychological questionnaires assessing their personality using the prominent Big Five model, as well as their Locus of Control (LoC), which is a measure of how much individuals believe outcomes hinge on their decisions.

Within this framework, our algorithm demonstrates a significant outcome: It produces a model of distinctive clusters of personality types, each characterized by a unique network of emotional states when confronting the Allais-narrative decisions. These cluster-related models shed light on the emotional states in which the certainty effect is manifested as discussed in the paradox, and conversely, the states in which this effect is absent. Importantly, these insights differ across personality types.

The remainder of the paper is structured as follows: In the Related Literature section, we briefly review the relevant literature. In Sect. 3, Methods, we delineate the extended Allais paradox experimentation and data analysis. The

Algorithm section outlines the proposed *R*&*A* algorithm for solving the clustering into network models problem. In the Results section, we showcase the efficacy of our algorithm in predicting decisions. We summarize our findings and conclude in the Summary and Conclusions section.

2 Literature Review

The Clustering Problem. Our proposed methodology employs the *R*&*F* algorithm, which commences by utilizing the Machine Learning algorithm Kmeans [22] to cluster projects into states. This is followed by an application of the Agglomerating Hierarchical algorithm [23] to aggregate the states into clusters. Both algorithms address the clustering problem, which has deep roots in the Machine Learning literature (for surveys see [31,35]). The primary objective of clustering is to identify groups of objects, where objects within the same cluster exhibit greater similarity to one another compared to those in different clusters, based on specific similarity measures. In line with this definition, the solution involves a function that maps each object to a cluster. Clustering techniques may use various approaches, including the partitioning of networks [36] or graphs [13], and hierarchical clustering [29], (refer to surveys such as [16,33]), each possessing unique advantages and limitations [3]. These techniques find applications in areas such as pattern recognition [16] and insight extraction [16], among others.

Symmetric Uncertainty. The intended application of the results (a model of clusters of networks) is to predict a label, and the model's performance depends on how informative these networks are regarding the label value. To quantify the information content of a network, we borrow a measure from the literature of information theory [8,24]. Specifically, we compute the distribution of objects among the states of the network. and the distribution of their choice; We then use these distributions to compute Symmetric Uncertainty [10,27], a measure of how much information is shared between two features relative to the entire information content. The Symmetric Uncertainty of two random variables measures the degree to which knowing the value of either random variable reduces uncertainty regarding the value of the other [26]. It is derived by normalizing the information gain to the entropies of the random variables (see [30] for definitions of these measures). Normalization induces values in the range [0,1] and assures symmetry. A value of zero means that the two variables are independent. For a formal definition of Symmetric Uncertainty, see [34]. In our analysis, a higher (vs. lower) level of Symmetric Uncertainty represents that knowing the state of an object is more (vs. less) informative for predicting its expected label.

Network of Networks. In the current article, we map each object to a cluster, wherein we identify a network structure. Partitioning into clusters is designed to achieve a set of network structures that are effective in predicting a predefined label. Our 'clustering into networks' approach draws parallels with techniques found in the scientific literature concerning the development of 'networks of networks,' as evident from various reviews such as [14], and [15]. Our approach

differs from the classical 'network of networks' approach in two fundamental aspects. The first is that we focus on identifying partitions into a set of state models (vs. static sub-networks). This approach has been applied to diverse areas including traffic [5], medicine [2], communication [6], and many others. The second difference pertains to the model development methodology. Aligned with the aforementioned motivation, rather than segmenting the objects into parts and subdividing them, we segment the objects into states and subsequently reassemble them into state models.

Allais Paradox. We demonstrate the applicability of our methodology to the Allais paradox [1]. The paradox regards the violations of von Neumann & Morgenstern's expected utility for the objective risk paradox. The paradox demonstrates the 'certainty effect,' whereby when a certain outcome is available, it enhances risk aversion. The influence of personality and emotions is stronger when the required judgment or decision is complex [11], and in such cases, people demonstrate irrational choices. It is therefore interesting to identify patterns of how such bounded rationality is associated with personality [20] and emotions [12].

Personality and Emotions. We conducted an experiment, gathering data by requesting participants to describe their decisions in the extensively studied paradigm of the Allais Paradox. Participants also rated the intensity of their anticipated emotions for each potential outcome, using six emotion scales [28]: Anxiety-Confidence, Boredom-Fascination, Frustration-Euphoria, Dispirited-Encouraged, Terror-Enchantment, and Humiliation-Pride. Finally, participants completed two psychological questionnaires assessing their personality. The first questionnaire evaluated their personality in terms of the prominent Big Five model [18]. This model is widely popular with cross-cultural applicability [7], and is used extensively to simplify the description of individual personality. The second psychological test evaluated Locus of Control [32], which refers to the degree to which people expect that an outcome is contingent on their behavior or is simply unpredictable. The motivation for adding data about expected emotions and personality is motivated by research showing that decisions are influenced by the emotional state of the decision-maker [19], her personality [9,17], or both [4].

3 Methods

We propose an algorithm that produces rigid clusters of flexible networks. The algorithm is called 'Rigid & Flexible' ($R\&F$). It assumes the availability of training data in which each data point includes 'stable parameters' and 'flexible parameters' and uses Machine Learning techniques to identify a clustering model that ensures that the objects in each cluster are not merely similar but are grouped for the efficiency of modeling as a flexible model of states. Specifically, it uses Kmeans [21] to cluster participants into states, and a hierarchical algorithm [23] to agglomerate the states into clusters.

We demonstrate the implementation of the $R\&F$ algorithm on the Allais paradox [2], to show how producing rigid personality clusters of flexible emotion

networks is efficient for predicting decision-makers' tendency to demonstrate a 'certainty effect' in the Paradox scheme.

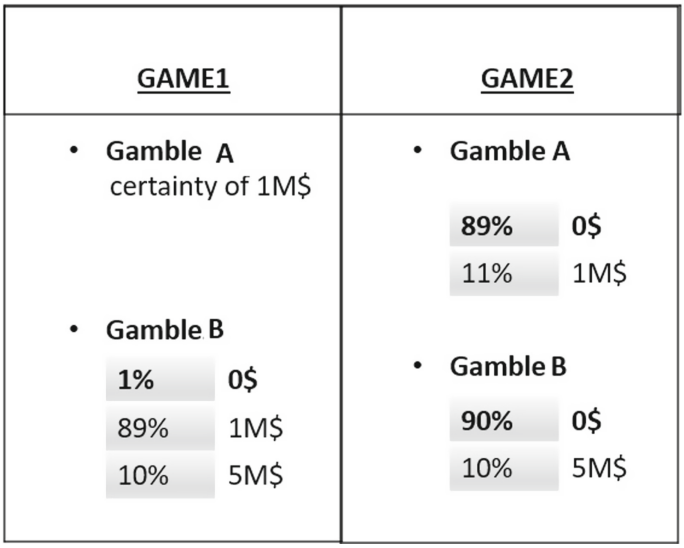


Fig. 1. Allais Experiment. In the first game, the choice was between a specific outcome and a gamble, while in the second game, it was between two gambles with different risks and outcomes.

3.1 Preliminary Definition of Demonstrating a Certainty Effect (DCE)

We investigated the application of our methodology to the problem of predicting 'Who demonstrates a certainty effect?' Our investigation starts with the definition of 'demonstrating a certainty effect' in the context of the Allais paradox experiment (see Fig. 1). The Allais paradox deals with the interesting observation that when presented with GAME1, most people choose GambleA, but when presented with GAME2 most people choose GambleB. This phenomenon is known as the 'certainty effect' and it is paradoxical as it violates the independence axiom.

The 'certainty effect' induces the definition of a human property (that any participant in the Allais paradox experiment may or may not have). The behavior of participants in the Allais experiment is represented by the combination of their two choices and thus classified into four types:

1. choosing 1A, and then 2A.
2. choosing 1A, and then 2B - 'demonstrating a certainty effect'.
3. choosing 1B, and then 2A.
4. choosing 1B, and then 2B.

It is only the second type of behavior that is paradoxical for being inconsistent with the expected utility theory (by which the participant should choose either 1A and 2A or 1B and 2B). This behavior is associated with the *certainty effect* and leads to the following definition:

Definition
We say that a decision-maker is *'demonstrating a certainty effect'* and denote it by *DCE*, if and only if he is 'choosing Gamble A in GAME1' and 'choosing Gamble B in GAME2.'

4 Algorithm

We present an algorithm that identifies 'type clusters,' along with the 'network of states' in each cluster.

4.1 Experimentation

To collect data, we conducted an experiment through Prolific (www.prolific.co), a crowd-working platform with high transparency and functionality, which supports the recruitment and performance of online tasks and completion of questionnaires [25]. We asked 200 participants (gender-biased) to report their choice when facing the paradox (see Fig. 1), and their expected emotions in each optional gamble.

Participants also completed two psychological questionnaires 2. The first questionnaire evaluated their personality in terms of the prominent Big Five model. The second psychological test evaluated their Locus of Control, which refers to the degree to which people expect that an outcome of their behavior is contingent on their behavior or is simply unpredictable. Each data point represents a participant in the experiment and includes their:

1. **6 'stable parameters'**: personality trait scores (locus of control, Big5: openness, consensuses, extroversion, agreeability, stability.
2. **24 'flexible parameters'**: 4 sets of the 6 reported levels of the following emotions on a scale from 1 to 5: Anxiety-Confidence, Boredom-Fascination, Frustration-Euphoria, Dispirited-Encouraged, Terror-Enchantment, Humiliation-Pride. Each set refers to one of the 4 Allais Paradox gambles.
3. **1 'label'**: a binary indicator of whether the participant is 'demonstrating a certainty effect'

Algorithm 1. Rigid Flexible (R&F) Algorithm

Input: D D is a data-set with m data points.
Each point holds the values of n_r *rigid parameters*, and n_f *flexible parameters*.

Phase I (compute D^*): Identifying states. Building a clustering model (which serves as a tool for the algorithm, rather than being the final outcome).

1. $M = Kmeans(D)$
   M is a clustering model, built with *Kmeans* algorithm using data D.
2. M_{num}
   M_{num} is the number of clusters in model M, each represents a 'state'.
3. $G : D \rightarrow [0, 1, \cdots, M_{num} - 1]$
   G is a mapping function of each data point (in D) to state (cluster in model M).
4. $D^* = centroids(M)$
   D^* is a data-set with the M_{num} centroids (cluster centers) of M.

Phase II (compute N): Identifying the list of 'state networks'. Agglomerating states into networks.

5. $D_r^* = D_r^*[:, 0 : n_r - 1]$
   D_r^* is a data-set with only the r 'rigid parameters' of D^*.
6. $N = Heirarchical(D_r^*)$
   N is a set of 'states networks', built with *Heirarchical* algorithm using data D_r^*

Phase III (compute G^{**}): Identifying the clustering model of 'type clusters'.

7. N_{num} is the number of clusters in model M
8. $G^* : D^* \rightarrow [0, 1, \cdots, N_{num} - 1]$
   G^* is the mapping function of each data point in D^* (centroids of M) to 'states network' (cluster in model N).
9. $G^{**} : D \rightarrow [0, 1, \cdots, N_{num} - 1] = G^*(G(p)), for pin D$
   G^{**} is the mapping function of each data point in D to 'states network' (cluster in model N).

Output: N and G^{**}
N is the set of 'networks of states'. Specifically for the i^{th} network, the nodes are all the states in D^* that the mapping function G^* assigns to the i^{th} cluster. We emphasize that a state is a centroid (center of a cluster) in model M and a node in model N.
G^{**} assigns each point p to a 'type cluster' which is one of the networks described above.

4.2 Analysis

As a prior step, we standardized the data as a common practice for the following analysis. Data analysis involves two phases: The first phase investigates the application of the $R\&F$ algorithm to the personality-emotions data. For each participant, the result is an assignment to a state model and to a state within this model. The second phase examines the application of the results, utilizing the set of state models to gain information on decisions. For each state model,

we consider the participants assigned to the cluster of this model and compute 2 random variables. The first random variable represents the distribution among the states, and the second random variable represents the distribution of the decision property of either demonstrating a certainty effect or not. These two random variables are used to compute Symmetric Uncertainty which measures how much information is portaged between them. In our analysis, a higher (vs. lower) level of Symmetric Uncertainty represents that knowing a participant's mapping onto a state (in the flexible emotion network) is more (vs. less) informative for knowing its label (demonstrating a certainty effect or not).

5 Results

The findings are divided into two sections, corresponding to the two phases of the data analysis process discussed in Sect. 4.2.

5.1 Part I - *R&F* Algorithm Implementation

In this section, we present the results of the application of the *R&F* algorithm (as referenced in Sect. 4) to the data-set gathered in our extended Allais Paradox experiment (refer to Sect. 4.1). For each participant, the outcomes offer a dual assignment: one to a cluster representing their personality type and the other to a specific emotion state within the state model that is associated with the cluster. The assignment to the Rigid Cluster is determined by the six rigid properties representing the participant's personality. On the other hand, the assignment to the state is delineated by the 24 flexible attributes that capture the participant's

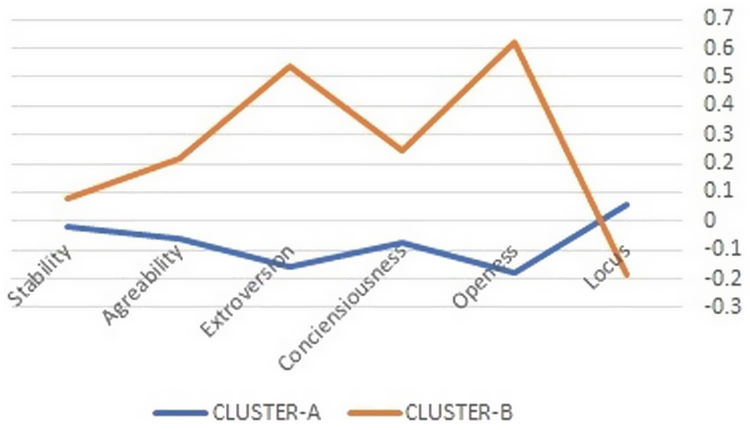


Fig. 2. Rigid Clusters (of personality types): Application of the *R&F* algorithm to the Allais experiment data identified 10 states and aggregated them into two clusters. This figure demonstrates the average properties (personality traits) of the participants assigned to each cluster (Cluster-A and Cluster-B).

anticipated emotions. The analysis of this dual assignment provides insights into the flexibility of emotions when facing a decision, and how the influence of emotions is different for different personalities. The algorithm identifies 10 states ($State-0, State-1$, to $State-9$) and aggregates them into 2 Rigid Clusters: $Cluster-A$ and $Cluster-B$, each containing a Flexible Network of states. The number of clusters was determined using the elbow curve and silhouette score. Cluster- A comprises eight states, while Cluster- B consists of two states ($State-2$ and $State-4$). Refer to Fig. 2 for an illustration of participants' personality traits within each cluster. Additionally, Fig. 3 delves into Cluster-B, highlighting the emotional characteristics across its states.

5.2 Part II - Model Application

This section illustrates the pragmatic implementation of the model generated in the first part of the analysis. Following the Allais Paradox example that we used in this article, we delve into the extent to which information is conveyed between an individual's emotional state and their propensity to exhibit the certainty effect. We continue to focus on Cluster-B which was described in 3, and consider participants who are assigned to Cluster-B. Table 1 demonstrates the counters for their assignment to the states in Cluster-B and the property of their choice (either DCE or not).

Table 1. States in Cluster-B and demonstrating a certainty Effect (DCE): This table presents participants assigned to Cluster-B, displaying counters for their assignment to Cluster-B states and the property of their choice (either demonstrating a certainty Effect (DCE) or not demonstrating a certainty Effect (nonDCE)). Table shows the difference between the distribution among DCE or nonDCE for participants in State-2 vs. State-4. The difference indicates the potential information gained by knowing participants' emotion state for predicting their tendency to demonstrate a certainty effect.

	State-2	State-4
nonDCE	4	5
DCE	7	25

Symmetric Uncertainty and Evaluation of the Model

To evaluate the informativeness of an individual's emotional state for predicting their inclination to display a certainty effect, we define two random variables: The first variable characterizes the distribution of participants across the states, and the second variable represents the distribution of whether they demonstrate the certainty effect or not. We utilized Python code from 2 to compute the Symmetric Uncertainty for these two variables. The performance of our model is evaluated against alternative data partitioning methods. Specifically, we assess the informativeness of an individual's cluster assignment across three variations of the

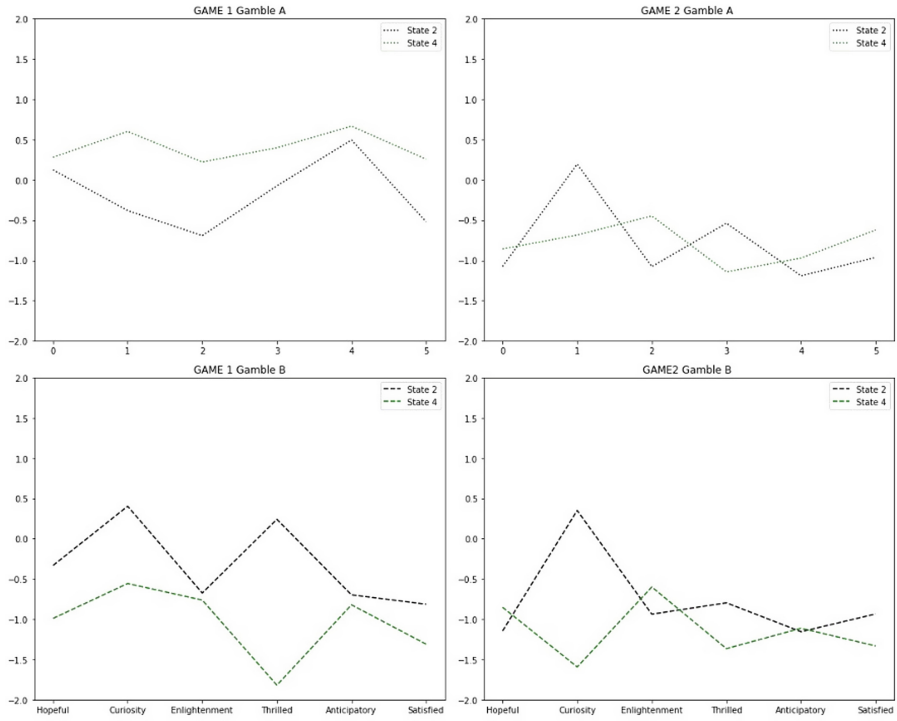


Fig. 3. Flexible States (of emotions) within Cluster-B: the *R&F* algorithm implementation on the Allais experiment data, identified 10 states and aggregated them into two clusters. This figure considers Cluster-B and demonstrates the average properties of the participants that are assigned to each of its states (State-2 and State-4) in terms of their reported emotions.

Kmeans algorithm, which are based on rigid parameters (personality traits), flexible parameters (reported emotions), or both. Additionally, we include a random assignment of participants into two clusters. Similar to our model, we calculate Symmetric Uncertainty for these benchmark models. 2 The results of this analysis highlight the efficacy of our methodology in uncovering patterns rich in informational value. Notably, our *R&F* algorithm efficiently identifies groups

Table 2. Symmetric Uncertainty: We evaluate our model's performance against various data partitioning techniques, including variations of the Kmeans algorithm using rigid and flexible parameters, as well as random assignment. We quantify the informativeness of an individual's cluster assignment, calculating Symmetric Uncertainty

	R&F	Kmeans(personality)	Kmeans(emotions)	Kmeans(all)	random
Symmetric Uncertainty	0.063	0.014	0.016	0.07	0.02

of individuals where knowledge of their emotional states proves exceptionally informative.

6 Summary and Discussions

Our 'Rigid & Flexible' algorithm harnesses the power of both Machine Learning and Network Analysis to carve out an effective model of flexible states within rigid clusters. The Allais paradox serves as our experimental ground, highlighting the efficacy of our approach for understanding how an individual's characteristics are associated with their inclination to exhibit the intriguing certainty effect. Unlike traditional methods that mostly focus on either rigid or flexible characteristics, we chose to decode the decision-making complexity by segmenting objects into flexible states and then reconstructing them into rigid clusters. This shift in perspective offers remarkable insights and links rigid personality traits to explain the valuable predictive patterns of flexible emotional states. Looking beyond, our methodology finds applications in diverse domains. Imagine classifying machines into distinct functional categories, each operating within its unique state space. This approach could also be used to categorize products based on satisfaction-related or weather-related states, segment employees according to task-related states, and more. By pioneering a clustering approach that truly captures shared state spaces, we are opening doors to a new realm of predictive insights.

References

1. Allais, M.L.: comportement de l'homme rationnel devant le risque: Critique des postulats et axiomes de l'école américaine. Econometrica **21**, 503–546 (1953)
2. Andersen, P.K., Keiding, N.: Multi-state models for event history analysis. Stat. Methods Med. Res. **11**(2), 91–115 (2002)
3. Arabie, P., et al.: Hierarchical classification Clustering and classification, pp. 65-121 (1996):
4. Aren, S., Hamamci, H.N.: Relationship between risk aversion, risky investment intention, investment choices: impact of personality traits and emotion'. Kybernetes **49**(11), 2651–2682 (2020)
5. Chandrasekaran, B.: Survey of network traffic models. Washington University in St. Louis CSE 567 (2009)
6. Lai, R.: A survey of communication protocol testing. J. Syst. Softw. **62**(1), 21–46 (2002)
7. McCrae, R.R., Costa, P.T., Jr.: Personality trait structure as a human universal. Am. Psychol. **52**, 509–516 (1997)
8. Dieck, R.H.: Measurement uncertainty: methods and applications. ISA (2007)
9. Dougherty, L.R., Guillette, L.M.: Linking personality and cognition: a meta-analysis. Philos. Trans. Royal Soc. B: Biol. Sci. **373**(1756), 20170282 (2018)
10. Edwards, W.: Methods for computing uncertainties. Am. J. Psychol. **67**(1), 164–170 (1954)
11. Finucane, M.L., Alhakami, A., Slovic, P., Johnson, S.M.: The affect heuristic in judgments of risks and benefits. J. Behav. Decis. Mak. **2000**(13), 1–17 (2000)

12. Moors, A., Fischer, M.: Demystifying the role of emotion in behaviour: toward a goal-directed account. Cogn. Emot. **33**(1), 94–100 (2019)
13. Fjällström, P.: Algorithms for graph partitioning: a survey. Linköping University Electronic Press (1998)
14. Gao, J., Li, D., Havlin, S.: From a single network to a network of networks. Natl. Sci. Rev. **1**(3), 346–356 (2014)
15. Gu, S., et al.: Modeling multi-scale data via a network of networks. Bioinformatics **38**(9), 2544–2553 (2022)
16. Han, J., Pei, J., Tong, H.: Data mining: concepts and techniques. Morgan kaufmann (2022)
17. Kassarjian, H.H.: Personality and consumer behavior: a review. J. Mark. Res. **8**(4), 409–418 (1971)
18. John, O.P., Srivastava, S.: The Big five trait taxonomy: history, measurement, and theoretical perspectives. Handbook Personal. Theory Res. **2**, 102–138 (1999)
19. Lerner, J.S., Keltner, D.: Beyond valence: toward a model of emotion-specific influences on judgement and choice. Cogn. Emot. **14**, 473–493 (2000)
20. Samar, S.M., Walton, K.E., McDermut, W.: Personality traits predict irrational beliefs. J. Rational-Emotive Cognit.-Behav. Therapy **31**, 231–242 (2013)
21. Sarwar, M.G., et al.: Machine learning at the network edge: a survey. ACM Comput. Surv. (CSUR) **54**(8), 1–37 (2021)
22. Ahmed, M., Seraj, R., Islam, S.M.S.: The Kmeans algorithm: a comprehensive survey and performance evaluation. Electronics **9**(8), 1295 (2020)
23. Murtagh, F., Contreras, P.: Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Rev. Data Mining Knowl. Dis. **2**(1), 86–97 (2012)
24. Namdari, A., Zhaojun L.: A review of entropy measures for uncertainty quantification of stochastic processes. Adv. Mech. Eng. **11**(6) (2019)
25. Palan, S., Schitter, C.: Prolific. ac - A subject pool for online experiments. J. Behav. Experim. Finance **17**, 22-27 (2018)
26. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. Cambridge University Press (1988)
27. Piao, M., Piao, Y., Lee, J.Y.: Symmetrical uncertainty-based feature subset generation and ensemble learning for electricity customer classification. Symmetry **11**(4), 498 (2019)
28. Kort, B., Rob R., Picard R.W.: An affective model of interplay between emotions and learning: Reengineering educational pedagogy-building a learning companion. In: Proceedings IEEE International Conference on Advanced Learning Technologies. IEEE (2001)
29. Reddy, C.K., Vinzamuri, B.: A survey of partitional and hierarchical clustering algorithms. Data Clustering: Alg. Appli. **87** (2013)
30. Renyi, A.: On measures of entropy and information. In: Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics, pp. 547 -561. University of California Press (1961)
31. Rokach, L.: A survey of clustering algorithms. Data Mining Knowl. Dis. Handbook, 269–298 (2010)
32. Rotter, J.B., Mulry, R.C.: Internal versus external control of reinforcement and decision time. J. Pers. Soc. Psychol. **2**, 598–604 (1965)
33. Sisodia, D., et al.: Clustering techniques: a brief survey of different clustering algorithms. Inter. J. Latest Trends Eng. Technol. (IJLTET) **1**(3), 82–87 (2012)
34. Song, Q., Ni, J., Wang, G.: A fast clustering-based feature subset selection algorithm for high-dimensional data. IEEE Trans. Knowl. Data Eng. **25**(1), 1–14 (2013)

35. Xu, D., Tian, Y.: A comprehensive survey of clustering algorithms. Annals Data Sci. **2**, 165–193 (2015)
36. Xu, X., et al.: Scan: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2007)

Beyond Following: Augmenting Bot Detection with the Integration of Behavioral Patterns

Sebastian Reiche[1(✉)], Sarel Cohen[2], Kirill Simonov[1], and Tobias Friedrich[1]

[1] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
sebastian.reiche@student.hpi.de, {kirill.simonov,tobias.friedrich}@hpi.de
[2] The Academic College of Tel Aviv-Yaffo, Tel Aviv, Israel
sarelco@mta.ac.il

Abstract. Social media platforms like Twitter revolutionized online communication. But this new era of interaction has brought with it a challenge—the widespread presence and influence of bot accounts. These bots are rapidly evolving, making traditional detection methods increasingly ineffective and allowing malicious actors to influence public discourse. While existing bot detection methods report high performance, such results might actually be connected to shortcomings in dataset collection and labeling practices, rather than reflecting their true ability to detect bots, casting doubt on their true reliability. Our study introduces higher-order behavior-based relations, including Co-Retweet, and Co-Hashtag, derived from the TwiBot-22 dataset. By leveraging these new relations in the BotRGCN architecture, we shift the emphasis from isolated accounts to coordinated group dynamics, making it more challenging for bot developers to evade detection. This strategy not only acknowledges the limitations and inherent biases presented in existing bot detection techniques, but also presents a way to address them. Our experiments support this approach as a promising way forward to tackle challenges in bot detection.

Keywords: bot detection · graph neural network · relation enhancement

1 Introduction

Social networks like Twitter — currently in the process of rebranding to X — have become an integral part of our social lives. They revolutionized the way we communicate online, shape public discourse, and provide access to the latest news and opinions. One major issue within social networks is the prevalence of bot accounts, which have been known to influence public opinion, especially in critical areas like politics or financial markets [2]. It is notoriously hard to estimate the true extent of the presense of bots on social media platforms, and platforms may be incentivized to misrepresent them, as it could negatively impact

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 250–259, 2024.
https://doi.org/10.1007/978-3-031-53468-3_21

revenue[1]. In 2017, Varol et al. estimated that bots may make up to 15% of all Twitter accounts [13]. In another study, Cresci et al. analyzed Twitter dicussions concerning the US stock market, and concluded that up to 71% of the engaged users might be bots [4].

Furthermore, bots seem to become more sophisticated over time [2,6], a phenomenon often referred to as *bot evolution*. This term describes the adversarial cycle in which newer bots evade increasingly more sophisticated bot detection measures, by becoming progressively indistinguishable from real humans. An illustrative example of this effect are the results reported in early 2017 by Cresci et al. [3]. In this experiment, the users were tasked to tell bots apart from legitimate users, only being able to correctly identify newer bots with a 24% accuracy, compared to 91% on older bots. Cresci [2] points out that bot detection methods must be able to distinguish between genuine users and bots, who disguise as genuine users through stolen profile pictures and neutral messages. This complexity has been further intensified by the advancement of artificial intelligence, particularly generative AI, which makes it more difficult to separate individual bot accounts from genuine users. The increasing difficulty in distinguishing between human-written and AI-generated text underscores the complexity of the issue. This is highlighted by OpenAI's decision to disable their AI classifier as of July 2023 due to low rate of accuracy in distinguishing between AI-generated and human-generated content.[2]

In response to these challenges with feature-based methods, graph-based methods are emerging as an alternative, due to their proven effectiveness in recognizing coordinated, synchronized activities [6]. By leveraging these techniques it is not only possible to study how users interact with content, but also how they interact with other users. The rationale behind these approaches stems from the assumption that human-guided and authentic activities typically display more variability than their automated, inauthentic counterparts. This emphasizes the need to move beyond analyzing individual accounts to focusing on patterns of suspicious coordination within groups.

However, research by Elmas et al. [5] on retweet bots, utilizing data from services previously purchased on black market sites, discovered discrepancies in common assumptions about bot characteristics. This included, but was not limited to, areas of *volume of activity*, *diversity*, *following and followers* and *temporality*. They illustrated that bots may emerge from compromised accounts, acting as bots only for certain period of time, and did not find a single case of one bot following another one. Such insights should prompt researchers to critically assess, whether the metrics used to evaluate the performance of bot detection methods are in fact contributing to improving downstream applications. Hays et al. [8] argued that this is currently not the case for Twitter bot detection tools, attributing high performance to simplistic collection and labeling practices of the datasets employed. Separately, Martini et al. [10] observed that different methods

[1] https://storage.courtlistener.com/recap/gov.uscourts.cand.330648/gov.uscourts.cand.330648.257.0.pdf.

[2] https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text.

yield remarkably different results in comparison. This implies that current tools may not be ready for downstream usage and may result in the misclassification of many users [11].

With the heightened difficulty in identifying individual bot accounts, we focus our efforts on group activities and their coordinated behavior patterns. Our work is in line with trends in recent research that focuses more on actions and behavior of groups of accounts rather than on the classification of individual accounts [1].

We investigate the potential of new sets of relations that are challenging to circumvent; any attempts to do so could drastically limit the functionality of organized automated actions by restricting their common operational patterns. The goal of our research is to determine the feasibility of utilizing coordination patterns for the purpose of bot detection, with due consideration to both the inherent complexities and data restrictions. By recognizing these challenges, we contrast first-order behavior-based relations, such as retweets (a user sharing a tweet), with higher-order relations like co-retweet (two users retweet the same tweet) and co-hashtag (two users tweet the same hashtag more than a certain number of times). The former highlights direct user behavior, while the latter reveals shared interests or subjects, uncovering subtler collective actions. This approach is set against the current conventional method of utilizing follow relations, which are more static. Utilizing the same dataset and graph neural network architecture across our experiments, we conduct a comparative study between the conventional follow relations and those centered around behavioral patterns to assess their impact on bot detection, avoiding the introduction of new uncertainties through algorithmic changes or dataset variations. Though our results did not surpass the conventional approach, they remain competitive in terms of accuracy and F1-score, demonstrating the viability of this approach. To the best of our knowledge, this is the first work that integrates higher-order relations in a behavior-based approach for bot detection.

2 Methodology

2.1 Dataset

We utilize the `TwiBot-22` dataset for our experiments. Compared to previous datasets, `TwiBot-22` includes a broader and more diverse range of relations. For an in-depth exploration of the dataset's conceptual framework, we refer the readers to the work of Feng at al. [6] that introduced `TwiBot-22`. Previous bot detection methods were constrained to rely only on FOLLOWER/FOLLOWING relationships between user entities and an implicit relation between users and their tweets. The `TwiBot-22` dataset encompasses extensive 14 different kinds of relations. In this work we leverage the FOLLOWER (user A is followed by user B), FOLLOWING (user A follows user B), RETWEET (tweet A retweets tweet B), POST (user A posts tweet B), and DISCUSS (tweet A discusses hashtag B) relations. We believe that this range of relations offers a lot of potential for future development of more sophisticated and accurate bot detection methods. The accessibility of these diverse relations not only enhances our analytical capabilities

but also allows us to reveal hidden connections between users, cross-referencing entities in ways previously unattainable. We refer the reader to Table 1 for an overview of `TwiBot-22`, comprising both statistics as well as an exploration of some of the characteristics that differentiate humans from bots. The left side of the table provides a quantitative overview of the dataset. On the right side, a more nuanced analysis of the variances in human and bot behavior. Key contrasts include variations in tweet and following/follower count[3] as well as ratios like hashtag-to-tweet, revealing discrepancies between the two types of accounts. This comparative analysis offers valuable insight that guides the process of deriving new relations. We explore these aspects further and delve into more detail in subsequent sections, specifically in Subsect. 2.3.

Table 1. Statistics (left) and in-depth analysis (right) of human and bot characteristics in `TwiBot-22`. ∗users with at least 1 tweet. † with at least 1 follower / following.

Measurement	Human	Bot	Total
Users (all)	860,057	139,943	1,000,000
Users (min. 1 tweet)	818,613	115,259	933,872
Tweet	81,250,102	6,967,355	88,217,457
Following	1,038,302	78,353	1,116,655
Followers	2,383,574	243,405	2,626,979
Retweet	1,482,911	97,732	1,580,643
Hashtags	56,353,776	9,646,857	66,000,633

Measurement	Human	Bot	Diff.
Mean tweet count*	99.25	60.45	−48.59%
Median tweet count*	56.00	40.00	−33.33%
Mean following count†	200.22	170.21	−16.20%
Mean follower count†	124.59	59.40	−70.86%
Mean retweet count*	200.39	104.98	−62.49%
Ratio: tweet / retweet	54.79	71.29	+26.17%
Ratio: hashtag / tweet	0.69	1.38	+66.66%

While `TwiBot-22` is believed to contain high-quality labels, it is important to recognize that we cannot entirely dismiss the possibility of underlying biases towards older notions of bot characteristics. A potential bias could be introduced by the use of non-transparent hand-crafted labeling functions and dependence on existing bot detection methods. These methods are often trained on follow relationships, an assumption we challenged in the introduction. This reliance on possibly flawed assumptions may further deviate bot detection in the wrong direction. In addition, recent evidence indicates that classifiers performing exceptionally within one dataset may significantly underperform when applied to others, even when employing more sophisticated models [8]. This may be attributed to the reliance on inherently unstable features present in the initial training data. Therefore, although `TwiBot-22`'s expert-guide process signals a marked improvement, the broader methodology might compromise the dataset's overall effectiveness. Nevertheless, we assume the labels in the data to be the ground truth. This assumption is made due to the lack of better annotation methods and inherent difficulty of this problem.

[3] Somewhat counter-intuitively, the total following and follower counts do not match. This is due to specifics of data collection, see [6] for insights into the process.

2.2 BotRGCN

`BotRGCN` (Bot detection with Relational Graph Convolutional Networks) [7] is a graph-based method for Twitter bot detection. The model first creates a multimodal encoding by jointly encoding multiple numerical and categorical user properties, as well as encoding user tweets and descriptions using a pre-trained RoBERTa model. These encodings serve to represent individual users, capturing diverse aspects of their behavior and characteristics. A heterogeneous graph is constructed by defining multiple relational neighborhoods for each Twitter user. `BotRGCN` applies relational graph convolutional networks (RGCN), which support a variable number of relations, allowing the model to capture complex patterns of interactions between users. We chose to work with `BotRGCN` due to its modular and well-designed architecture that allows for easy modification and experimentation. The model was used with the initialization of hyperparameters as found in the original implementation, available at the corresponding Github repository.[4] Adjustments were made to accommodate the specific number of categorical and numerical properties in `TwiBot-22`. The architecture and specific components of `BotRGCN` are further detailed in Table 2.

Table 2. Architecture of the `BotRGCN` model. Variables: D: embedding size, D_s: description size, T_s: tweet size, N_s: numerical properties size, C_s: categorical properties size. The input layers' outputs are concatenated before processing through the hidden layers. A dropout regularization technique is applied between the RGCN layers. The model is used with the CrossEntropyLoss, which implicitly includes a Softmax activation on the output.

BotRGCN Architecture		
Input Layers	Description Embedding	Linear ($\mathbb{R}^{D_s \times \frac{D}{4}}$) + LeakyReLU
	Tweet Embedding	Linear ($\mathbb{R}^{T_s \times \frac{D}{4}}$) + LeakyReLU
	Numerical Properties Embedding	Linear ($\mathbb{R}^{N_s \times \frac{D}{4}}$) + LeakyReLU
	Categorical Properties Embedding	Linear ($\mathbb{R}^{C_s \times \frac{D}{4}}$) + LeakyReLU
Hidden Layers	Input Transformation	Linear ($\mathbb{R}^{D \times D}$) + LeakyReLU
	RGCN 1st Layer	RGCN Convolution ($\mathbb{R}^{D \times D}$)
	RGCN 2nd Layer	RGCN Convolution ($\mathbb{R}^{D \times D}$)
	Hidden Transformation	Linear ($\mathbb{R}^{D \times D}$)+ LeakyReLU
Output Layer	Final Output	Linear ($\mathbb{R}^{D \times 2}$)

2.3 Derived Relations

Elmas et al. [5] argue that a significant challenge in bot detection is the non-intuitive nature of bot characteristics. For instance, their analysis revealed that

[4] https://github.com/BunsenFeng/BotRGCN.

the majority of bot accounts in their dataset had more followers than accounts they were following, and no two bots followed each other.

Moreover, the authors also observed different retweet behaviour for bots, both temporal as well as quantitative. This insight, coupled with the observation of bot evolution, led us to investigate the potential offered by new sets of relations.

Inspired by work from Vargas et al. [12], which builds upon coordination patterns from [9] we introduce the following relations:

- RETWEET: a user retweeted the tweet of another user.
- CO-RETWEET: two users retweeted the same tweet.
- CO-HASHTAG: two users tweet the same hashtag above a certain threshold.

These relations are behavior-based, which makes them harder to manipulate than, e.g., FOLLOWER and FOLLOWING relations. We believe that this approach has the potential to reveal additional patterns of coordinated behavior among users. However, none of these are readily usable for us out-of-the-box and require some data transformation steps.

RETWEET: Our analysis showed that bots tend to retweet disproportionately. In order to take advantage of this, we first need to transform the existing RETWEET relation from tweet→tweet to user→user. By cross-referencing the given RETWEET relation with the POST relation (user→tweet), we are able to associate a user for each tweet and subsequently derive the RETWEET relation in the form of user→user. This process is illustrated in Fig. 1.

CO-RETWEET: We introduce this relation to emphasize instances where two users retweeted the same tweet. To achieve this, we map a user to each tweet that retweets another tweet, similar to the process laid out in RETWEET above. Then, we group these users by their retweeted target tweet. From these groups, we create all possible combinations of users (excluding pairs with the same user twice) and export them as our new CO-RETWEET relation.

CO-HASHTAG: Using a similar grouping and pairing approach as with the CO-RETWEET relation, we focus on the DISCUSS relation (tweet→hashtag). Prior to the pairing step, we filter out hashtags with an unusually large number of users to decrease computational demands and filter out those hashtags that do not offer any reasonable insight. After this step, we create pairs of users who tweeted the same hashtag a minimum of n times. The choice of n can be regarded as a hyperparameter itself and is detailed further, in the subsequent experiments section and Table 3.

3 Experiments

To determine the feasibility of utilizing coordination patterns for bot detection we conducted sensitivity and ablation studies. We kept hyperparameters constant across all experiments. The model is initialized with the same parameters as mentioned in Subsect. 2.2. We further fixed the dropout rate at 0.3, the learning rate at 0.001, and weight decay at 0.005. Furthermore, we standardized the

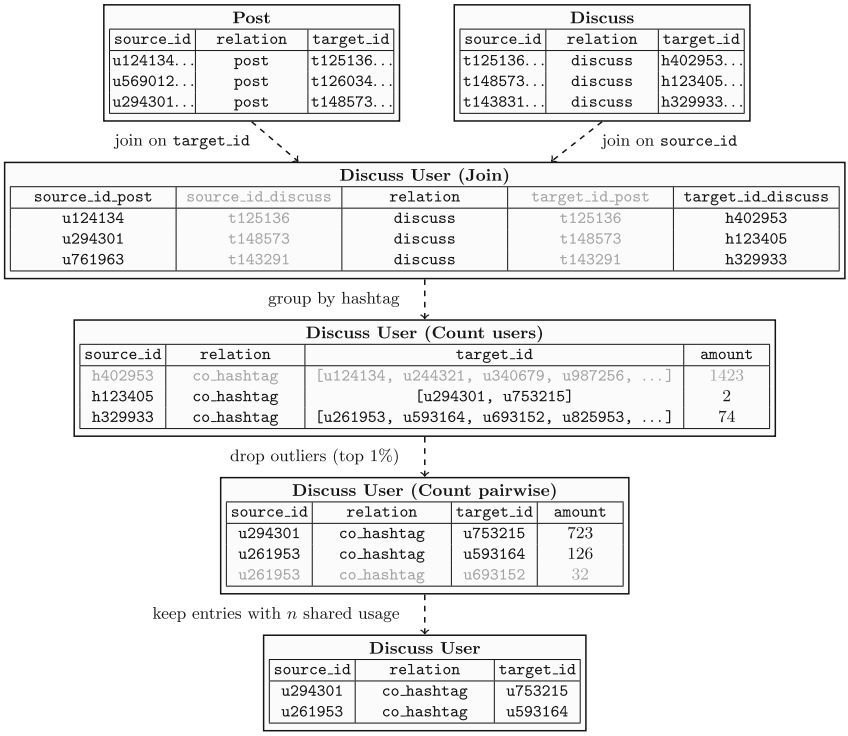


Fig. 1. Visualization of the process of deriving the new Co-Hashtag (`co_hashtag`) relation. Initially, the edge file is split into individual relations (not depicted). We then join the `post` and `discuss` relation to associate user-ids with each hashtag in the `discuss` table. In this example we assume a threshold `amount` value of 100, below which `co_hashtag` occurrences are discarded. We then create pairs of users with the respective count of how often they share a hashtag. Lastly, we keep only those with at least n shared hashtags and discard the `amount` column to get the expected format.

number of training epochs to 200 across all experimental runs. We reused the train/test split that comes with `TwiBot-22`, for comparability with prior work.

First, we defined a threshold for the Co-Hashtag relation. The threshold was set to three standard deviations above the mean, with values provided in Table 3. Since the differences between the thresholds were minor, we chose the one that achieved the highest F1-score, indicating the most reliable predictions. Additional experimentation with the sets and quantities of relations can be referenced in Table 4. Notably, the follower relation yielded the best results, as opposed to the common follower+following combination. It matches the intuition that this relation can be a strong indicator. Our main interest, however, was on the newly derived behavioral relations, with *follow* relationships serving as a baseline for comparison.

Table 3. Sensitivity study of the `co-hashtag` edge creation threshold. The Amount column corresponds to the parameter n, representing the minimum number of times pairs of users tweeted the same hashtag. We run each experiment five times and report the average value as well as the standard deviation in parentheses.

Threshold	Amount	Accuracy	F1-score
mean + 1 SD	467	76.02 (0.65)	42.64 (3.53)
mean + 2 SD	907	75.59 (0.23)	41.03 (1.85)
mean + 3 SD	1347	75.91 (0.39)	43.06 (2.15)
mean + 4 SD	1787	75.36 (0.60)	39.76 (3.52)
mean + 5 SD	2226	75.61 (0.45)	41.07 (2.19)

Our findings necessitate contextual interpretation, contrasting our approach with the conventional use of FOLLOWER+FOLLOWING relations. Instead, we leverage the higher-order CO-RETWEETED and CO-HASHTAG relations to capture more complex user behaviors like mutual affinity for retweeting particular content or using the same hashtags above a certain level. However, we do not dismiss the RETWEET relation and still consider it valuable for future exploration. Though we did not outperform the conventional approach, our results are closely competitive, with differences of less than 1.22 percent points lower in accuracy and 3.78 percent points in F1-score.

Table 4. Sensitivity analysis of `BotRGCN` to different edge types in the graph. We run each experiment five times and report the average value as well as the standard deviation in parentheses.

Category	Sensitivity Settings	Accuracy	F1-score
=single relation type	`follower`	77.63 (0.47)	50.70 (2.03)
	`following`	75.38 (0.59)	37.19 (3.76)
	`retweeted`	75.78 (0.69)	41.75 (4.20)
	`co-retweeted`	75.56 (0.80)	40.43 (4.63)
	`co-hashtag`	75.91 (0.39)	43.06 (2.15)
=two relation types	`follower+following`	76.99 (0.43)	46.06 (2.31)
	`retweeted+co-retweeted`	75.43 (0.34)	39.70 (2.23)
	`co-retweeted+co-hashtag`	75.77 (0.13)	42.28 (1.08)
=three relation types	`following+follower+retweeted`	77.55 (0.57)	48.92 (3.24)
	`retweeted+co-retweeted+co-hashtag`	75.81 (0.52)	41.51 (2.42)
five relation types	all of the above	77.11 (0.32)	46.72 (1.63)

This gap, although initially discouraging, reveals upon closer examination the capability to make predictions, avoiding biases that might have characterized previous approaches. Despite the notable perfomance of the single `follower`

relation, there's evident improvement when using three or five relations instead of two. Our concerns regarding these biases are outlined in Subsect. 2.1 dedicated to the dataset. This highlights the potential of a multi-rational approach, but it is essential to note that inherent characteristics of the used dataset might influence these observations. Such results are particularly significant, as bot developers may find it challenging to avoid behavior-based detection without substantially constraining their capabilities. Building on the findings from Feng et al. [7], where it was confirmed that the optimal performance is achieved with 2 layers of RGCN, we have carried out an ablation study of BotRGCN, utilizing the same layer configuration. Our experiments, as detailed in Table 5, prove that the integration of all available modalities remains essential for robust bot detectors. The challenge requires a multi-faceted approach, integrating various modalities. This approach must then model the aggregation of these signals, aiming to ensure a clear distinction between accounts involved in automated coordinated efforts and those demonstrating authentic behavior, which may stem from social initiatives.

Table 5. Ablation Study of `BotRGCN` under different relation types using 2 layers of `RGCN`. Abbreviations used: T = User Tweets; N = User Numerical Properties; C = User Categorical Properties; D = User Descriptions. We run each experiment five times and report the average value as well as the standard deviation in parentheses.

	`follower + following`		`co-retweeted + co-hashtag`	
Ablation Setting	Accuracy	F1-score	Accuracy	F1-score
`RGCN` + T	70.51 (0.01)	1.34 (0.23)	70.54 (0.02)	0.89 (0.34)
`RGCN` + T, N	70.83 (0.18)	7.05 (2.69)	70.81 (0.28)	6.72 (3.90)
`RGCN` + T, N, C	73.07 (0.34)	25.67 (3.18)	72.70 (0.34)	20.79 (3.02)
`RGCN` + T, N, C, D (`BotRGCN`)	76.99 (0.43)	46.06 (2.31)	75.77 (0.13)	42.28 (1.08)

4 Conclusion

The complexity of bots continues to evolve, making the task of bot detection a critical challenge. Our investigation into alternative higher-order, behavioral-based relations emphasizes a different approach in detecting automated coordinated group activities. Although not surpassing the conventional approach, the competitiveness of our results suggest a reliable method without falling into suspected biases of traditional techniques. Bot developers seeking to avoid detection may find it increasingly difficult without limiting their capacities. `TwiBot-22`, the dataset used in this study, has been instrumental in establishing these new relations. Yet, as we look into further research, the incorporation of temporal patterns into these newly established relations seems promising. This direction, however, necessitates datasets that support this, a limitation we currently face. We are optimistic that pursuits into this direction can foster the development of more robust and reliable detection methods.

Acknowledgements. The authors thank Ali Alhosseini for his guidance during the early conceptual phase and Lukas Drews for his collaboration in the initial experiments.

References

1. Cinelli, M., Cresci, S., Quattrociocchi, W., Tesconi, M., Zola, P.: Coordinated inauthentic behavior and information spreading on twitter. Decision Support Syst. **160**, 113,819 (2022)
2. Cresci, S.: A decade of social bot detection. Commun. ACM **63**(10), 72–83 (2020)
3. Cresci, S., Di Pietro, R., Petrocchi, M., Spognardi, A., Tesconi, M.: The paradigm-shift of social spambots: evidence, theories, and tools for the arms race. In: Proceedings of the 26th International Conference on World Wide Web Companion, pp. 963–972 (2017)
4. Cresci, S., Lillo, F., Regoli, D., Tardelli, S., Tesconi, M.: Cashtag piggybacking: uncovering spam and bot activity in stock microblogs on twitter. ACM Trans. Web (TWEB) **13**(2), 1–27 (2019)
5. Elmas, T., Overdorf, R., Aberer, K.: Characterizing retweet bots: The case of black market accounts. In: Proceedings of the International AAAI Conference on Web and Social Media, vol. 16, pp. 171–182 (2022)
6. Feng, S., Tan, Z., Wan, H., Wang, N., Chen, Z., Zhang, B., Zheng, Q., Zhang, W., Lei, Z., Yang, S., et al.: Twibot-22: towards graph-based twitter bot detection. Adv. Neural. Inf. Process. Syst. **35**, 35254–35269 (2022)
7. Feng, S., Wan, H., Wang, N., Luo, M.: Botrgcn: Twitter bot detection with relational graph convolutional networks. In: Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 236–239 (2021)
8. Hays, C., Schutzman, Z., Raghavan, M., Walk, E., Zimmer, P.: Simplistic collection and labeling practices limit the utility of benchmark datasets for twitter bot detection. In: Proceedings of the ACM Web Conference 2023, pp. 3660–3669 (2023)
9. Keller, F.B., Schoch, D., Stier, S., Yang, J.: Political astroturfing on twitter: how to coordinate a disinformation campaign. Polit. Commun. **37**(2), 256–280 (2020)
10. Martini, F., Samula, P., Keller, T.R., Klinger, U.: Bot, or not? comparing three methods for detecting social bots in five political discourses. Big data & society **8**(2), 20539517211033,566 (2021)
11. Rauchfleisch, A., Kaiser, J.: The false positive problem of automatic bot detection in social science research. PloS one **15**(10), e0241,045 (2020)
12. Vargas, L., Emami, P., Traynor, P.: On the detection of disinformation campaign activity with network analysis. In: Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop, pp. 133–146 (2020)
13. Varol, O., Ferrara, E., Davis, C., Menczer, F., Flammini, A.: Online human-bot interactions: Detection, estimation, and characterization. In: Proceedings of the International AAAI Conference on Web and social media, vol. 11, pp. 280–289 (2017)

Graph Completion Through Local Pattern Generalization

Zhang Zhang[1,2], Ruyi Tao[1,2], Yongzai Tao[3], Mingze Qi[4], and Jiang Zhang[1,2(✉)]

[1] School of Systems Science, Beijing Normal University, Beijing, China
zhangjiang@bnu.edu.cn
[2] Swarma Research, Beijing, China
[3] College of Computer Science and Technology, Zhejiang University, Hangzhou, China
[4] College of Science, National University of Defense Technology, Changsha, China

Abstract. Network completion is more challenging than link prediction, as it aims to infer both missing links and nodes. Although various methods exist for this problem, few utilize structural information-specifically, the similarity of local connection patterns. In this study, we introduce a model called C-GIN, which captures local structural patterns in the observed portions of a network using a Graph Auto-Encoder equipped with a Graph Isomorphism Network. This model generalizes these patterns to complete the entire graph. Experimental results on both synthetic and real-world networks across diverse domains indicate that C-GIN not only requires less information but also outperforms baseline prediction models in most cases. Additionally, we propose a metric known as "Reachable Clustering Coefficient (RCC)" based on network structure. Experiments reveal that C-GIN performs better on networks with higher Reachable CC values.

Keywords: Network Completion · Graph Auto-Encoder · Structural Patterns

1 Introduction

Networks form the underlying structures of numerous systems and hold significant implications in both scientific research and everyday life [1,2]. One approach to understanding these complex systems involves studying the properties of the networks that underlie them. However, obtaining a complete network structure is often infeasible due to factors such as measurement errors, privacy concerns, and other limitations [3–5]. For instance, while online social network data can be readily collected, there are 'offline' nodes that may exert significant influence at certain times but remain difficult to capture due to the unavailability of offline data. Consequently, there is a pressing need for methodologies capable of inferring missing information in incomplete networks. The methodologies

R. Tao and Y. Tao—Those author contribute equally

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 260–271, 2024.
https://doi.org/10.1007/978-3-031-53468-3_22

developed for network completion could thus have wide-ranging applications, including identifying hidden nodes in networks [6], uncovering the property of the whole network from partially observed structure [7] and discovering unobserved variables [8] in causal inference [9].

While link prediction [10–12] has been extensively studied in the field of network structural inference and serves as a basis for various downstream applications such as node classification [13–15] and graph dynamics learning [16,17], it generally operates under the assumption that all nodes are observable, with only some edges missing. In network completion tasks, the objective is not only to infer missing links but also to identify missing nodes. This makes the task considerably more challenging than link prediction, particularly due to the presence of 'naked nodes'-nodes that are isolated from the rest of the network because they lack any connecting links. Therefore, some methods commonly used in link prediction tasks, such as those based on common neighbors [18,19] or mutual information [20] cannot be directly applied in network completion tasks.

Previous works have used diverse strategies for network completion. For instance, G-GCN [21] focuses on network growth, which isn't applicable to all networks, such as more static gene regulatory networks. Wei's model [22] relies on node attributes, limiting its use in scenarios without such data. Other high-accuracy methods like Gumbel Graph Network [8] and DeepNC [23] require extensive time-series data or similar graph sets, making them impractical for many real-world applications. KronEM [24], to the best of our knowledge, is the sole model specifically designed for network completion without extra node data. It leverages self-similarity in many networks to infer a kernel from observed sections. However, self-similarity is not universal, and KronEM's limited parameters may not capture the complexity of diverse networks.

We introduce the Completion Graph Isomorphism Network (C-GIN) to solve network completion without extra node data. C-GIN uses a Graph Auto-Encoder to learn and apply local connection patterns from the known part of the adjacency matrix.

C-GIN has distinct advantages over existing models. Unlike G-GCN [21] and GGN [8], which require node features and time-series data, C-GIN only needs the visible adjacency matrix, widening its applicability. Unlike DeepNC [23], which requires a dataset of similar, smaller networks, C-GIN exploits the local structure of a single, larger network, making it ideal for incomplete networks. Compared to KronEM [24], it captures complex local structures more effectively due to the flexibility of graph neural networks. Tests confirm C-GIN outperforms current models in network completion.

2 Results

2.1 The Network Completion Problem

Problem Definition. Suppose we have a undirected network $G(V, E)$ with an adjacency matrix A. This network cannot be fully observed, as information about some nodes and their corresponding edges is missing. Instead, we can only observe a sub-graph G_o of G, which contains some observed vertices V_o and

edges E_o between them. Assume we know the number of missing nodes N_m. Our objective is to infer the missing part of the network, denoted as $G_m = G - G_o$, which includes the missing nodes V_m and the missing edges E_m. We can reorder the nodes such that the observable nodes are placed at the beginning of the sequence. This allows us to divide the adjacency matrix into two sub-matrices: one for the observable nodes (A_o) and another for the connections related to unobserved nodes (A_m). Thus, the task is to reconstruct the whole adjacency matrix $A = A_o + A_m$ based on A_o, where A_m is an inverted L-shaped matrix describing the connections between V_o and V_m, as well as between V_m themselves, as shown in Fig. 1.

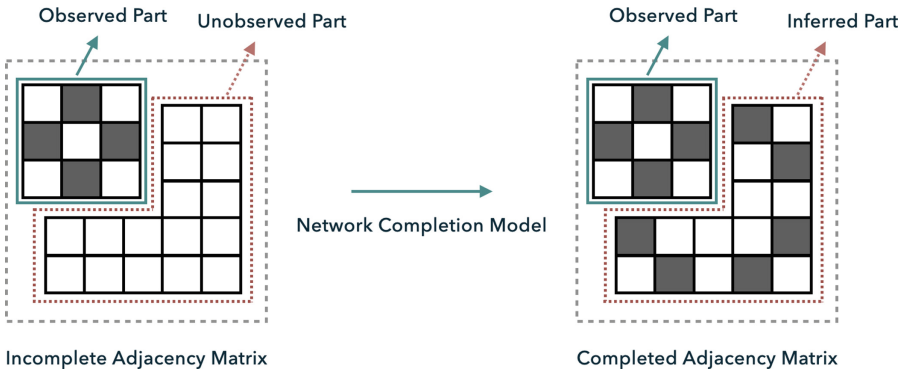


Fig. 1. The illustration of the network completion problem. A network completion problem is to infer the potential nodes and links according to the observed sub-graph. Left: suppose we can re-arrange all vertices such that the observed nodes come first. Then the entire adjacency matrix A contains two areas: the squared sub-matrix of the observed part A_o and the inverted L-shaped area A_m with the blank entries around A_o. Right: the complete adjacency matrix $\hat{A} = A_o + \hat{A}_m$, where $\hat{A}_m$ is inferred by the network completion algorithm.

Overview of the C-GIN Framework. The C-GIN model assumes that a network's different areas share common connection patterns. Using a Graph Auto-Encoder, C-GIN captures these patterns from the partially observed subgraph to fill in the missing network portions. The Graph Neural Network (GNN) within the Auto-Encoder's Encoder learns the formation of local structures around nodes in two steps: first, by generating initial node embeddings from a linear layer, and then by obtaining final embeddings through message-passing layers. We selected the Graph Isomorphism Network (GIN) [25] for its superior expressive power among various GNN models like GCN [13], GAT [26], and GraphSAGE [27].

To extend learned patterns to the unobserved area, the Graph Auto-Encoder encodes the network structures into node embeddings. The observed part are used to compute the loss function, thereby compelling the Graph Neural Network

to assign appropriate embedding vectors to unobserved nodes, ensuring that the represented network structure exhibits patterns consistent with the observed parts. Figure 2 outlines our model's workflow.

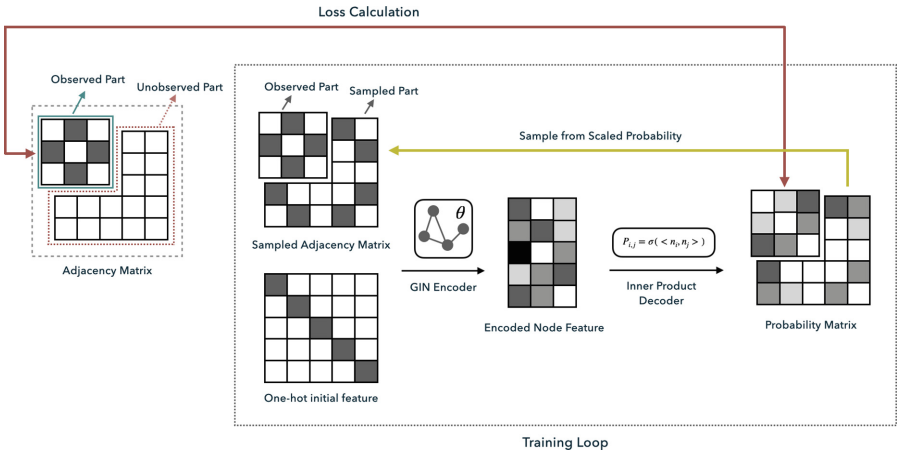


Fig. 2. The Overview of our C-GIN model: Our model employs GIN to learn the local connection patterns of the network's known portion and iteratively complete the missing elements. Each iteration comprises an encoding stage and a decoding stage. During the encoding stage, initial node features represented by one-hot vectors, along with the inferred complete network from the previous iteration, are fed into GIN. This results in updated node feature vectors as output. In the decoding stage, these updated feature vectors are used to generate a matrix that represents the probabilities of connections between each node pair. The observable portion of this probability matrix is utilized to calculate the loss. The remainder of the matrix is rescaled by multiplying it with a scaling factor γ. Finally, the adjacency matrix for the next epoch is sampled according to these adjusted probabilities.

Next, we delve into the specific execution flow of the algorithm. According to the problem definition, we have partial information about how the observed nodes are connected, represented by A_o, which forms the upper-left quadrant of the adjacency matrix. We also assume knowledge of the number of missing nodes, N_m. In the initial stage, we populate the adjacency matrix of the unknown part with zeros to obtain an $N \times N$ matrix, denoted as $\hat{A}$. We use one-hot vectors as the initial features, $X = I$, for all nodes, allowing the neural network's linear layers to independently determine each node's embedding vector. This initialized matrix $\hat{A}$ and feature vector X are then fed into the GIN encoder, which is known to excel in learning local connection patterns." [25], to update the node feature vector H. The initial vector $X = I$ remains unchanged during the training process. The GIN gradually learns how to map from one-hot vectors to appropriate node embedding vectors to represent the network structure. This operation can be expressed by the Eq. 1:

$$H = GIN_\theta(\hat{A}, X). \tag{1}$$

In this context, θ denotes the parameters of the GIN encoder. Once we have the encoded node features, they are then fed into the decoder as described by Eq. 2. This results in a decoded probability matrix $P_{N\times N}$. Each element $P_{i,j}$ represents the probability that nodes v_i and v_j are connected.

$$P_{i,j} = \frac{1}{1 + \exp(-\langle H^i, H^j \rangle)}. \tag{2}$$

Note that the probability matrix P is divided into two sections. The upper-left subsection, a $N_o \times N_o$ matrix, represents the connection probabilities for the observed part of the network. This subsection is used to calculate the loss function. The loss function, in turn, helps to optimize the parameters θ within the GIN encoder. The specific definition of the loss function is provided in Eq. 3.

$$\mathcal{L}(\theta) = -\sum_{i=0}^{N_o-2} \sum_{j=i+1}^{N_o} A_{i,j} \log(P_{i,j}) + (1 - A_{i,j}) \log(1 - P_{i,j}). \tag{3}$$

To ensure edge density consistency between the observed and unobserved network parts, we introduce another training stage. In this stage, probabilities in the "inverted L-shaped region" of matrix P are scaled using a factor γ. This adjustment is crucial for controlling sparsity in the predicted adjacency matrix, as high edge density can impact the graph auto-encoder's performance. The scaling factor γ is calculated using Eq. 4 and helps align edge densities across the network.

$$\gamma = \frac{N^2 - N_o^2}{N_o^2} \times \frac{\sum_{i<N_o, j<N_o} A_{i,j}}{\sum_{i<N, j<N} P_{i,j} - \sum_{i<N_o, j<N_o} P_{i,j}}. \tag{4}$$

2.2 Experiments

Baselines Models. The baselines were chosen from three different types of network completion algorithms for comparison:

Preferential Attachment and Random-De: Preferential Attachment (PA) serves as a conventional baseline in the link prediction task. Intuitively, in this model the unobserved nodes are more inclined to link with observed nodes that have a greater number of neighbors. Random-De employs the same framework as our proposed approach but it use randomly generated node embeddings of the same dimensions as those input into the decoder. Thus serves essentially as a random guess baseline.

G-GCN: As outlined in the Introduction, G-GCN tackles the network completion problem by considering network growth. The original G-GCN model requires node feature information for its input. However, this does not align with our problem definition. Therefore, we modify the input to use one-hot vectors, making it consistent with our own model's input. Experimental results indicate that even without node feature information, the modified G-GCN model remains competitive in certain tasks.

KronEM: KronEM is an algorithm for network completion that leverages the network's self-similarity property. It employs the Kronecker graph model to characterize the network and estimates its missing parts. The optimization of the Kronecker model parameters is carried out using the Expectation Maximization (EM) framework and a scalable Metropolized Gibbs sampling approach. It's worth noting that we do not evaluate this approach on real-world networks. This is because the model requires that the number of nodes in the complete graph be a power of the size of the Kronecker product kernel, thereby limiting its applicability.

2.3 Metrics

We evaluate the effectiveness of our model and the baseline methods in network completion tasks by treating the completion of the inverted L-shaped region of the adjacency matrix as a binary classification problem. Performance is assessed using the area under the Receiver Operating Characteristic (ROC) curve (AUC). To ensure a balanced evaluation, we follow previous literature [21] in randomly sampling an equal number of positive and negative edges. Specifically, we introduce two sets of these metrics-namely, $AUC_{Observed-Unobserved}$ and $AUC_{Unobserved-Unobserved}$ to rigorously evaluate the accuracy of inferred connections between both observed and unobserved nodes, and exclusively unobserved nodes, respectively.

To calculate AUC, it is necessary to compare the predicted connection probability matrix with a ground-truth adjacency matrix. However, a direct comparison is not feasible due to the alignment requirement for the inferred unobserved nodes with the actual ones. Specifically, a permutation matrix is needed to reorder the rows and columns corresponding to the unobserved nodes in the probability matrix generated by the network completion algorithm. This reordering aims to make the rearranged probability matrix resemble the ground-truth adjacency matrix as closely as possible. Considering there are $N_m!$ possible permutations, we opt for the best-performing permutation in the performance comparison for fairness. This problem is known as sub-graph matching [28]. We employ the Seeded Graph Matching [29] algorithm to address this challenge. According to existing literature, this algorithm achieves a matching accuracy exceeding 90% when the similarity between the two matrices in question is greater than 90% and the number of nodes to be matched exceeds 15. Details can be referred to [29].

Note that we did not use the Seeded Graph Matching method to reorder the probability matrix returned by KronEM, because this algorithm cannot only output the learned matrix of A_m but also the node alignment.

Performances on Completing Synthetic Graphs. In this section we test the performances on completing the synthetic networks which includes 4 types of networks generated by well-known models, namely, Barabási-Albert(BA) model, Watts-Strogatz(WS) model, Forest Fire(FF) model, and Kronecker

graphs(Kron) model, respectively. In each network, the number of nodes is 1,024 to satisfy the requirement of KronEM algorithm. In BA network, each new node is added to the network with two connections. In WS network, each node has four neighbors, and the reconnection probability is 0.2. In FF network the probability of both forward and backword of an edge is 0.33. In Kron network, just like the KronEM paper [24], we use the $[[0.9, 0.7], [, 0.5, 0.2]]$ as the Kronecker Kernel to generate the network. We randomly choose 25% of nodes and the relevant edges to remove and we ran five repeated experiments to get the mean and the standard deviation to fill the table. Results are shown in Table 1.

Table 1. AUC on unobserved part of synthetic networks: In this table we show the experimental results of different methods on the synthetic networks. Listed in the table is the overall AUC value of the unobserved region of the adjacency matrix.

Networks	PA	Random-De	KronEM	G-GCN	C-GIN
BA	58.67 ± 1.5	59.33 ± 1.5	64.1 ± 0.6	74.67 ± 2.1	**76.49 ± 1.8**
WS	35.45 ± 0.5	35.91 ± 0.4	80.42 ± 1.2	81.20 ± 0.6	**85.22 ± 0.3**
Kron	71.60 ± 0.6	71.13 ± 1.1	**83.9 ± 0.4**	68.38 ± 1.1	71.55 ± 1.4
FF	79.33 ± 0.9	74.16 ± 0.7	62.1 ± 0.4	**82.75 ± 0.7**	80.17 ± 1.8

In Table 1, it's evident that C-GIN model outperforms comparative models in BA and WS cases. Specifically, the KronEM model excels only in networks generated by the Kron network generator, which feature self-similar properties. The G-GCN model achieves the highest scores on FF networks generated by the forest fire model. Among these two datasets, the C-GIN model performs second best. Interestingly, despite being generated by the preferential attachment mechanism, the BA network is not well-complemented by this method. This discrepancy arises because our experiments involve randomly deleting a subset of nodes, rather than specifically eliminating the most recently added ones. As a result, some high-degree nodes may be removed, which in turn alters the connections to their smaller-degree counterparts.

Performances on Completing Empirical Graphs. In this section we test the performances of C-GIN model on real-world networks. Different types of real-world networks are selected, The basic information of them is listed below, and their structural statistics are shown in Table 2. In all networks, we randomly removed 20% of the nodes and the corresponding edges. Results are shown in Table 3.

– **Bio_S and Bio_D** are two undirected and weighted networks of gene interactions extracted from C. elegans. The nodes are genes and the edges are Inferred links by genetic interactions. In which Bio_S is more sparse and Bio_D is more dense. To create an unweighted network, we set all the weights greater than zero to be one. Both of them were collected from Network Repository(NR) [30]

– **Co-Author** network represents the co-authorship of researchers in network theory & experiments, where a node is a researcher and an edge represents the co-author relationship. If two authors both contribute to at least one paper, a connection between them will be added in the network. The above data is collected from [31].
– **Cora** is a dataset composed of machine learning papers. It is one of the most popular data sets for graph deep learning in recent years, each paper in this dataset has at least one citation. Cora was originally a directed graph. As a conventional operation [13], we remove all link directions to make it an undirected graph.

Table 2. The statistics of different empirical networks: In this table we show the statistical properties of the networks, they are the number of nodes, the number of edges, the density and the clustering coefficient.

Network	Nodes	Edges	Density	Clustering Coefficient
Bio_S	924	3239	0.0076	0.6051
Bio_D	636	3959	0.0196	0.4712
Cora	2708	5278	0.0014	0.2406
Co-Author	379	914	0.0128	0.7412

Table 3. AUC on unobserved part of real-world networks: In this table we show the experiment result of different methods on real-world networks. In most cases G-GIN model outperform other competetors.

Networks	AUC	Models			
		PA	G-GCN	C-GIN	Random-De
Bio_S	All	72.89 ± 1.3	83.23 ± 1.8	**88.71 ± 2.1**	70.22 ± 0.7
	Observed-Unobserved	73.5 ± 1.4	88.83 ± 1.7	**90.16 ± 1.7**	72.41 ± 0.9
	Unobserved-Unobserved	-	57.19 ± 3.3	**74.05 ± 3.4**	54.34 ± 2.2
Bio_D	All	75.18 ± 0.9	81.35 ± 1.0	**85.79 ± 4.0**	62.43 ± 1.6
	Observed-Unobserved	77.01 ± 0.8	85.86 ± 1.0	**88.48 ± 2.6**	63.99 ± 1.8
	Unobserved-Unobserved	-	57.07 ± 1.1	**66.71 ± 11.2**	54.66 ± 2.7
Cora	All	60.01 ± 0.9	86.02 ± 0.9	**87.37 ± 0.3**	78.28 ± 1.4
	Observed-Unobserved	60.13 ± 0.6	**92.07 ± 0.9**	89.10 ± 0.8	81.28 ± 1.1
	Unobserved-Unobserved	-	55.58 ± 2.3	**74.70 ± 2.2**	53.78 ± 2.1
Co-Author	All	58.88 ± 2.7	85.82 ± 1.7	**91.61 ± 1.6**	73.93 ± 1.5
	Observed Uunobserved	59.07 ± 2.2	**97.17 ± 1.1**	93.78 ± 1.8	76.27 ± 1.6
	Unobserved-Unobserved	-	57.36 ± 7.0	**75.29 ± 7.6**	60.66 ± 4.1

In Table 1, it is evident that the C-GIN model outperforms all competing models in completing the 'unobserved-unobserved' section of the adjacency

matrix. This indicates that C-GIN model excels at modeling connections between unobserved nodes. In the 'observed-unobserved' section, C-GIN model also attained the best results for two biological networks. However, G-GCN achieved superior performance on the the citation network(Cora). This discrepancy may stem from the inherent aptitude of G-GCN for modeling growing networks such as citation network. To further investigate which types of networks C-GIN model is most effective for, we will examine the relationship between model performance and structural features in the following section.

Performance Vs. Reachable Clustering Coefficient. In our model's encoding stage, we employ the GIN model to aggregate neighbor information through message passing, allowing for superior performance on networks with intricate local connection patterns. We illustrate this by completing a small-world network and examining how the reconnecting probability p, p influences model performance and CC. When $p = 0$, the network has high CC; at $p = 1$, it has low CC. Crucially, variations in p only affect local connection patterns, not network density or degree distribution. We measure performance using the AUC difference between the C-GIN and a random baseline model.

Figure 3's left and middle panel show a decline in the AUC difference as p increases, reaching near-zero when p approaches 1, indicating that the C-GIN model is not better than a random guess in this scenario. Similarly, CC also decreases as p increases, confirming our model excels in networks with higher CCs. This suggests our model captures the local connectivity pattern effectively, outperforming random guesses for networks with higher CC.

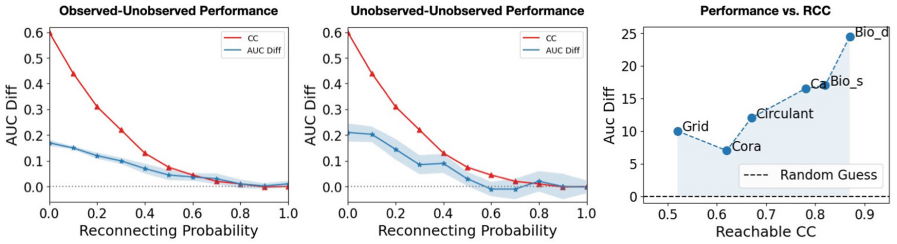


Fig. 3. C-GIN Performance on W-S and Empirical Networks: The left 2 panel are the experiments on W-S network. The x-axis is the reconnecting probability p of the WS network. This figure has two y-axes, one is CC and the other is the AUC difference, in which we replace the output of the GIN encoder with the randomly generated matrix that has the same shape. The right panel is the model relation of model performance and Reachable CC on several empirical networks.

While the Clustering Coefficient (CC) offers some insight into local connectivity, it falls short in capturing more complex structures. For instance, in a 2D grid network, first-order neighbors are not interconnected, maintaining a CC of zero despite complex local connections involving second-order neighbors. To

address this, we introduce the Reachable CC metric. In our experiments, higher Reachable CC scores correlate with better model performance.

To compute Reachable CC, we form a new network by connecting a node's higher-order neighbors and then calculate the clustering coefficient. The adjacency matrix of this new network, A_n, is derived from Eq. 5 and Eq. 6:

$$A^{(n)} = sgn(\sum_{i=1}^{n} A^i) - \sum_{i=1}^{n-1} A^{(i)}, \tag{5}$$

$$A_n = \sum_{i=1}^{n} A^{(i)} * (1-\lambda)^{i-1}, \tag{6}$$

where $A^{(1)} = A$. In Eq. 5, we get a matrix $A^{(n)}$ that represents the n-order connection between nodes, specifically, if there is a path of length n between node i and node j, then $A^{(n)}_{i,j} = 1$, otherwise $A^{(n)}_{i,j} = 0$. In Eq. 6, we get A_n by weighted summation of $A^{(i)}$. where λ refers to the decay index, which can also be understood as the reaching cost corresponding to the path length. If λ is 0, it means that node i can reach node j at no cost. If λ is 1, it means that node i cannot reach any second-order and above neighbors, and then A_n degenerates into the original adjacency matrix A. We calculate the clustering coefficient on the newly obtained A_n, resulting in Reachable CC.

In Fig. 3's right panel, we demonstrate a positive correlation between edge completion performance and Reachable CC across various networks, including real, Grid, and Circulant networks. This trend can be understood through the model's mechanism: it learns local structure patterns from observed nodes and extrapolates to unobserved nodes. With a high Reachable CC, higher-order neighbors are more interconnected, making it easier for the GIN-based encoder to learn the network's structural pattern. As a result, C-GIN performs better in networks with higher Reachable CC.

2.4 Discussion

In this study, we introduce C-GIN model to address the network completion problem when certain nodes and their edges are unobserved. Utilizing a Graph Auto-Encoder, C-GIN learns the local connectivity patterns within the observed portions of the network and generalizes these patterns to unknown regions of the adjacency matrix. Experimental results confirm that C-GIN outperforms benchmark models on both synthetic and real-world networks, particularly excelling at completing edges between unobserved nodes. To delve deeper into the model's nature, we introduce the Reachable CC metric, which gauges the likelihood of connecting edges between higher-order neighbors within a network. C-GIN performs notably better on networks with higher Reachable CC values.

Despite its contributions, the model has limitations. For instance, it presupposes the number of unobserved nodes, a parameter often unknown in real-world scenarios. Future research could focus on estimating this number. Moreover, certain networks, like Cora, offer node-specific features, while others, such as those

modeling infectious diseases, present temporal dynamics. Incorporating these features and dynamics into our network completion approach could be a promising direction for future studies.

References

1. Marsden, P.V.: Network data and measurement. Ann. Rev. Sociol. **16**(1), 435–463 (1990)
2. Newman, M.E.: Communities, modules and large-scale structure in networks. Nat. Phys. **8**(1), 25–31 (2012)
3. Cimini, G., Squartini, T., Garlaschelli, D., Gabrielli, A.: Systemic risk analysis on reconstructed economic and financial networks. Sci. Rep. **5**(1), 1–12 (2015)
4. Kossinets, G.: Effects of missing data in social networks. Social networks **28**(3), 247–268 (2006)
5. Anand, K., et al.: The missing links: a global study on uncovering financial network structures from partial data. J. Financ. Stab. **35**, 107–119 (2018)
6. Su, R.-Q., Wang, W.-X., Lai, Y.-C.: Detecting hidden nodes in complex networks from time series. Phys. Rev. E **85**(6), 065201 (2012)
7. Haehne, H., Casadiego, J., Peinke, J., Timme, M.: Detecting hidden units and network size from perceptible dynamics. Phys. Rev. Lett. **122**(15), 158301 (2019)
8. Chen, M., Zhang, Y., Zhang, Z., Du, L., Wang, S., Zhang, J.: Inferring network structure with unobservable nodes from time series data. Chaos: Interdiscip. J. Nonlinear Sci. **32**(1), 1 013126 (2022)
9. Pearl, J.: Causal inference.: Causality: objectives and assessment, pp. 39–58 (2010)
10. Lichtenwalter, R.N., Lussier, J.T., Chawla, N.V.: New perspectives and methods in link prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 243–252 (2010)
11. Lü, L., Zhou, T.: Link prediction in complex networks: a survey. Phys. A: A **390**(6), 1150–1170 (2011)
12. Wang, P., Xu, B., Wu, Y., Zhou, X.: Link prediction in social networks: the state-of-the-art. SCIENCE CHINA Inf. Sci. **58**(1), 1–38 (2015)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social network data analytics. Springer, pp. 115–148 (2017)
15. Rong, Y., Huang, W., Xu, T., Huang, J.: Dropedge: Towards deep graph convolutional networks on node classification. arXiv preprint arXiv:1907.10903 (2019)
16. Cai, H., Zheng, V.W., Chang, K.C.-C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
17. Bunke, H., Riesen, K.: Graph classification based on dissimilarity space embedding. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) Structural, Syntactic, and Statistical Pattern Recognition, pp. 996–1007. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89689-0_103
18. Zhou, T., Lü, L., Zhang, Y.-C.: Predicting missing links via local information. Europ. Phys. J. B **71**(4), 623–630 (2009)
19. Liu, Z., Zhang, Q.-M., Lü, L., Zhou, T.: Link prediction in complex networks: A local naïve bayes model. EPL (Europhysics Letters) **96**(4), 48007 (2011)

20. Tan, F., Xia, Y., Zhu, B.: Link prediction in complex networks: a mutual information perspective. PLoS ONE **9**(9), e107056 (2014)
21. Xu, D., Ruan, C., Motwani, K., Korpeoglu, E., Kumar, S. and Achan, K.: Generative graph convolutional network for growing graphs. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 3167–3171 (2019)
22. Wei, Q., Hu, G.: Unifying node labels, features, and distances for deep network completion. Entropy **23**(6), 771 (2021)
23. Tran, C., Shin, W.-Y., Spitz, A., Gertz, M.: Deepnc: Deep generative network completion. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
24. Kim, M., Leskovec, J.: The network completion problem: inferring missing nodes and edges in networks. In: Proceedings of the 2011 SIAM International Conference on Data Mining, SIAM, 2011, pp. 47–58 (2011)
25. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
26. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
27. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
28. Koutra, D., Parikh, A., Ramdas, A., Xiang, J.: Algorithms for graph similarity and subgraph matching. In: Proceedings of Ecol. inference conf, vol. 17. Citeseer (2011)
29. Fishkind, D.E., et al.: Seeded graph matching. Pattern Recogn. **87**, 203–215 (2019)
30. Rossi, R., Ahmed, N.: The network data repository with interactive graph analytics and visualization,. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29, no. 1 (2015)
31. Rossi, R., Ahmed, N.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). https://networkrepository.com

A Consistent Diffusion-Based Algorithm for Semi-Supervised Graph Learning

Thomas Bonald(✉) and Nathan De Lara

Institut Polytechnique de Paris, Paris, France
thomas.bonald@telecom-paris.fr

Abstract. The task of semi-supervised classification aims at assigning labels to all nodes of a graph based on the labels known for a few nodes, called the seeds. One of the most popular algorithms relies on the principle of heat diffusion, where the labels of the seeds are spread by thermoconductance and the temperature of each node at equilibrium is used as a score function for each label. In this paper, we prove that this algorithm is not consistent unless the temperatures of the nodes at equilibrium are centered before scoring. This crucial step does not only make the algorithm provably consistent on a block model but brings significant performance gains on real graphs.

1 Introduction

The principle of heat diffusion has proved instrumental in graph mining [5]. It has been applied for many different tasks, including pattern matching [10], ranking [7], embedding [4], clustering [11], classification [2,6,13,14] and feature propagation [9]. In this paper, we focus on the task of semi-supervised node classification: given labels known for a few nodes of the graph, referred to as the *seeds*, how to infer the labels of the other nodes? A popular approach consists in using diffusion in the graph, under boundary constraints, a problem known in physics as the Dirichlet problem [14]. Specifically, one Dirichlet problem is solved per label, setting at 1 the temperature of the seeds with this label and at 0 the temperature of the other seeds. Each node is then assigned the label with the highest temperature over the different Dirichlet problems. In this paper, we prove using a simple block model that this algorithm is actually not consistent, unless the temperatures are *centered* before label assignment. This step of temperature centering does not only make the algorithm consistent but also brings substantial performance gains on real datasets. This is a crucial observation given the popularity of the algorithm[1].

The rest of this paper is organized as follows. In Sect. 2, we introduce the Dirichlet problem on graphs. Section 3 describes our algorithm for node classification. The analysis showing the consistency of our algorithm on a simple block model is presented in Sect. 4. Section 5 presents some experimental results and Sect. 6 concludes the paper.

[1] The number of citations of the paper [14] exceeds 4 000 in 2023, according to Google Scholar.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 272–282, 2024.
https://doi.org/10.1007/978-3-031-53468-3_23

2 Dirichlet Problem on Graphs

In this section, we introduce the Dirichlet problem on graphs and characterize the solution, used later in the analysis.

2.1 Heat Equation

Consider an undirected graph G with n nodes indexed from 1 to n. Denote by A its adjacency matrix. This is a symmetric matrix with non-negative entries. Let $d = A1$ be the degree vector, which is assumed positive, and $D = \mathrm{diag}(d)$. The Laplacian matrix is defined by:

$$L = D - A.$$

Now let S be some strict subset of $\{1, \ldots, n\}$ and assume that each node $i \in S$ is assigned some fixed temperature T_i. We are interested in the evolution of the temperatures of the other nodes, we refer to as the *free* nodes. We assume that heat exchanges occur through each edge of the graph proportionally to the temperature difference between the corresponding nodes, so that:

$$\forall i \notin S, \quad \frac{dT_i}{dt} = \sum_{j=1}^{n} A_{ij}(T_j - T_i),$$

that is,

$$\forall i \notin S, \quad \frac{dT_i}{dt} = -(LT)_i,$$

where T is the vector of temperatures, of dimension n. This is the heat equation in discrete space. At equilibrium, the vector T satisfies Laplace's equation:

$$\forall i \notin S, \quad (LT)_i = 0. \tag{1}$$

With the boundary constraint giving the temperature T_i for each node $i \in S$, this defines a Dirichlet problem. Observe that Laplace's equation (1) can be written equivalently:

$$\forall i \notin S, \quad T_i = (PT)_i, \tag{2}$$

where $P = D^{-1}A$ is the transition matrix of the random walk in the graph.

2.2 Solution to the Dirichlet Problem

We now characterize the solution to the Dirichlet problem (1). Without any loss of generality, we assume that free nodes (i.e., not in S) are indexed from 1 to $n - s$ so that the vector of temperatures can be written

$$T = \begin{bmatrix} X \\ Y \end{bmatrix},$$

where X is the vector of temperatures of free nodes at equilibrium, of dimension $n-s$, and Y is the vector of temperatures of the seeds, of dimension s. Writing the transition matrix in block form as

$$P = \begin{bmatrix} Q & R \\ \cdot & \cdot \end{bmatrix},$$

it follows from (2) that:

$$X = QX + RY, \tag{3}$$

so that:

$$X = (I - Q)^{-1} RY. \tag{4}$$

Note that the inverse of $I - Q$ exists whenever the graph is connected [3]. The solution to the Dirichlet problem exists and is unique.

3 Node Classification Algorithm

In this section, we introduce a node classification algorithm based on the Dirichlet problem. The objective is to infer the labels of all nodes given the labels of a few nodes called the *seeds*. Our algorithm is a simple modification of the popular method proposed by [14]. Specifically, we propose to *center* temperatures before label assignment.

3.1 Binary Classification

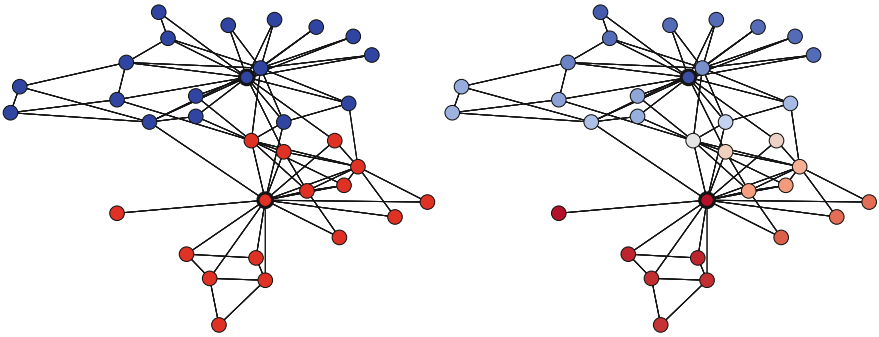
When there are only two different labels, say 0 and 1, the classification follows from the solution of a single Dirichlet problem. The idea is to set at 0 the temperature of seeds with label 0 and at 1 the temperature of seeds with label 1. The solution to this Dirichlet problem gives temperatures between 0 and 1 to the free nodes, as illustrated by Fig. 1 for the Karate Club graph [12].

A natural decision rule is to use a threshold of $1/2$ for classification: any free node with temperature above $1/2$ at equilibrium is assigned label 1, while any other free node is assigned label 0. The analysis of Sect. 4 suggests that it is preferable to set the threshold to the mean temperature at equilibrium,

$$\bar{T} = \frac{1}{n} \sum_{i=1}^{n} T_i. \tag{5}$$

Specifically, any free node with temperature above $\bar{T}$ at equilibrium is assigned label 1, while any other free node is assigned label 0. Equivalently, temperatures are *centered* by their mean before classification: after centering, free nodes with positive temperature are assigned label 1, the others are assigned label 0.

It is worth noting that the threshold (5) is the mean temperature of *all* nodes at equilibrium, including seed nodes. Another option, suggested by the *class mass normalization* step of [14] for instance, is to set the threshold at the mean temperature of *free* nodes at equilibrium. This variant of the algorithm is not provably consistent, however.



(a) Ground truth (b) Solution to the Dirichlet problem

Fig. 1. Binary classification of the Karate Club graph with 2 seeds (indicated with a black circle). Blue nodes have label 0, red nodes have label 1. (Color figure online)

3.2 Multi-class Classification

In the general case with K labels, we use a *one-against-all* strategy: the seeds of each label alternately serve as hot sources (temperature 1) while all the other seeds serve as cold sources (temperature 0). After centering the temperatures (so that the mean temperature of each diffusion is equal to 0), each node is assigned the label that maximizes its temperature. This algorithm, we refer to as the Dirichlet classifier, is parameter-free.

Algorithm 1. Dirichlet classifier

Require: Seed set S and associated labels $y \in \{1, \ldots, K\}$
1: **for** k in $\{1, \ldots, K\}$ **do**
2: $T = 0$
3: **for** $i \in S$ **do**
4: **if** $y_i = k$ **then**
5: $T_i = 1$
6: **end if**
7: **end for**
8: $T \leftarrow \text{Dirichlet}(S, T)$
9: $\Delta(k) \leftarrow T - \frac{1}{n} \sum_{i=1}^{n} T_i$
10: **end for**
11: **for** $i \notin S$ **do**
12: $\hat{y}_i = \arg\max_{k=1,\ldots,K}(\Delta_i(k))$
13: **end for**
14: **return** $\hat{y}$, predicted labels of free nodes (outside S)

The solution to the Dirichlet problem (line 8 of the algorithm) can be obtained either from (4) or from iterations of the fixed-point equation (3).

4 Analysis

In this section, we prove the consistency of Algorithm 1 on a simple block model. In particular, we highlight the importance of temperature centering (line 9 of the algorithm) for the consistency of the algorithm.

4.1 Block Model

Consider a graph of n nodes consisting of K blocks of respective sizes $n_1, \dots, n_K$, forming a partition of the set of nodes. There are $s_1, \dots, s_K$ seeds in these blocks, which have labels $1, \dots, K$, respectively. Intra-block edges have weight p and inter-block edges have weight q. We expect the algorithm to assign label k to all nodes of block k, for all $k = 1, \dots, K$, whenever $p > q$, i.e., the blocks are *assortative* [8].

4.2 Dirichlet Problem

Consider the Dirichlet problem when the temperature of the s_1 seeds of block 1 is set to 1 and the temperature of the other seeds is set to 0. We have an explicit solution to this Dirichlet problem, given by Lemma 1. All proofs are deferred to the appendix.

Lemma 1. *Let T_k be the temperature of free nodes of block k at equilibrium. We have:*

$$\begin{aligned}(s_1(p-q)+nq)T_1 &= s_1(p-q)+n\bar{T}q,\\ (s_k(p-q)+nq)T_k &= n\bar{T}q \qquad k=2,\dots,K,\end{aligned}$$

where $\bar{T}$ is the average temperature, given by:

$$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i = \left(\frac{s_1}{n}\frac{n_1(p-q)+nq}{s_1(p-q)+nq}\right) \Big/ \left(1 - \sum_{k=1}^{K} \frac{(n_k - s_k)q}{s_k(p-q)+nq}\right).$$

4.3 Classification

We now state the main result of the paper: the Dirichlet classifier is a consistent algorithm for the block model, in the sense that all nodes are correctly classified whenever $p > q$.

Theorem 1. *If $p > q$, then the predicted label of each free node i of block k is $\hat{y}_i = k$, for any $n_1, \dots, n_K$ (label distribution) and $s_1, \dots, s_K$ (seed distribution).*

Observe that the temperature centering is critical for consistency. In the absence of centering, free nodes of block 1 are correctly classified if and only if

their temperature is the highest in the Dirichlet problem associated with label 1. In view of Lemma 1, this means that for all $k = 2, \ldots, K$,

$$s_1 q \frac{n_1(p-q)+nq}{s_1(p-q)+nq} + s_1(p-q)\left(1 - \sum_{j=1}^{K} \frac{(n_j - s_j)q}{s_j(p-q)+nq}\right) > s_k q \frac{n_k(p-q)+nq}{s_k(p-q)+nq}.$$

This condition might be violated even if $p > q$, depending on the parameters $n_1, \ldots, n_K$ and $s_1, \ldots, s_K$. In the simplest case of $K = 2$ blocks, with $p = 10^{-1}$ and $q = 10^{-2}$ for instance, the classification is incorrect in the following two asymmetric cases:

Seed asymmetry (blocks of same size but different number of seeds):
$n_1 = n_2 = 100; s_1 = 10, s_2 = 5$,
Label asymmetry (blocks with the same number of seeds but different sizes):
$n_1 = 100, n_2 = 10; s_1 = s_2 = 5$.

This sensitivity of the algorithm to both forms of asymmetry will be confirmed by the experiments. The step of temperature centering is crucial for consistency.

5 Experiments

In this section, we show the impact of temperature centering on the quality of classification using both synthetic and real data. The Python code is available as a Jupyter notebook in Python[2], making the experiments fully reproducible.

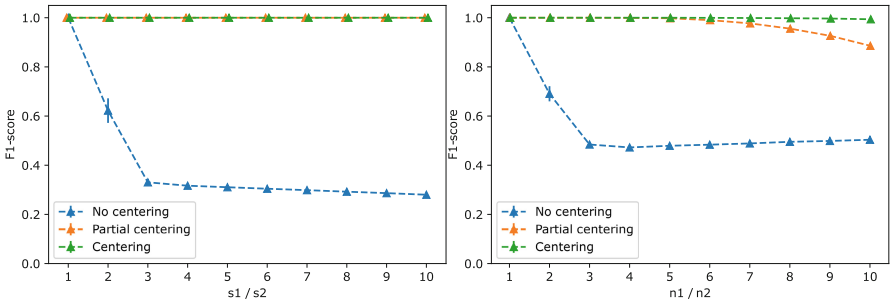
5.1 Synthetic Data

We first use the stochastic block model [1] to generate graphs with an underlying structure in clusters. This is the stochastic version of the block model used in the analysis. There are K blocks of respective sizes $n_1, \ldots, n_K$. Nodes of the same block are connected with probability p while nodes in different blocks are connected probability q. Nodes in block k have label k. We denote by s_k the number of seeds in block k and by s the total number of seeds.

We first compare the performance of the algorithms on a binary classification task ($K = 2$) for a graph of $n = 10\,000$ nodes with $p = 10^{-2}$ and $q = 10^{-3}$, in two different settings:

Seed asymmetry (blocks of same size but different number of seeds):
$n_1 = n_2 = 5000$, $s_2 = 250$, ratio $s_1/s_2 \in \{1, 2, \ldots, 10\}$.
(5% of nodes in block 2 are seeds)
Label asymmetry (blocks with the same number of seeds but different sizes):
number of nodes $n = 10\,000$, ratio $n_1/n_2 \in \{1, 2, \ldots, 10\}$, $s_1 = s_2 = 250$.
(5% of all nodes are seeds)

[2] https://perso.telecom-paris.fr/bonald/notebooks/diffusion.ipynb.

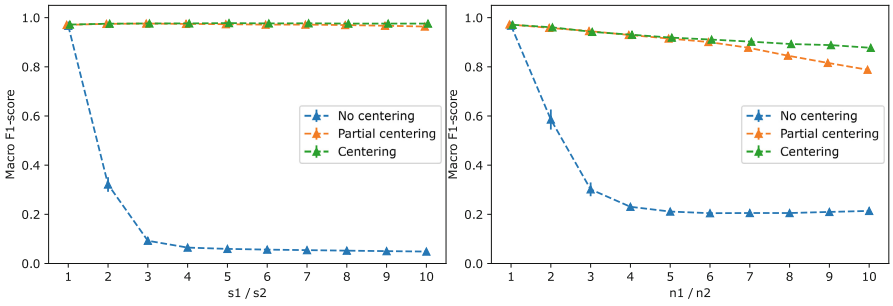


(a) Seed asymmetry (b) Label asymmetry

Fig. 2. F1 scores on the stochastic block model (2 labels).

For each configuration, the experiment is repeated 100 times. Randomness comes both from the generation of the graph and from the selection of the seeds. We report the F1-scores in Fig. 2 (mean $\pm$ standard deviation). Observe that the variability of the results is very low due to the relatively large size of the graph. As expected, the centered version is much more robust to both forms of asymmetry. The variant called *partial centering*, where the mean temperature is computed over free nodes only, tends to be less robust to label asymmetry.

We show in Fig. 3 the same results for $K = 5$ blocks, still with $n = 10\,000$ nodes, $p = 10^{-2}$ and $q = 10^{-3}$. Blocks $2, 3, 4, 5$ have the same size and the same number of seeds. For the experiments on seed asymmetry, each block has $2\,000$ nodes and 5% of nodes in blocks $2, 3, 4, 5$ are seeds; we only vary the number of seeds in block 1. For the experiments on label asymmetry, there is the same number of seeds for each label, corresponding to an average proportion of 5% of all nodes. The performance metric is the F1-score averaged over the 5 labels. The conclusions are the same as with 2 labels.



(a) Seed asymmetry (b) Label asymmetry

Fig. 3. Macro F1-scores on the stochastic block model (5 labels).

5.2 Real Data

We now focus on real datasets available from the SNAP collection[3] and the NetSet[4] collection, restricting to graphs having ground-truth labels. All graphs are considered undirected.

Table 1. Overview of the datasets.

Dataset	#nodes	#edges	#classes
Cora	2 708	5 278	7
Citeseer	3 264	4 536	6
PubMed	19 717	44 325	3
Email	1 005	16 385	42
PolBlogs	1 490	16 716	2
WikiSchools	4 403	100 329	16
WikiVitals	10 011	654 502	11
WikiVitals+	45 179	3 079 335	11

For each dataset, we select seeds uniformly at random. The process is repeated 100 times. The macro-F1 scores (i.e., F1-scores averaged over all classes) are shown in Table 2 for seeds representing $5\%, 10\%$ or 20% of the nodes. We see that the centered version outperforms the standard version over all datasets. The performance gains are substantial for the largest graphs, extracted from Wikipedia. The variance is also lower in all cases, showing the robustness of the algorithm. Additional results, not reported here, tend to show that the variant selected for temperature centering (based on either all nodes or free nodes) has a marginal impact on performance.

[3] https://snap.stanford.edu/.
[4] https://netset.telecom-paris.fr/.

Table 2. Macro-F1 scores (mean ± standard deviation) without and with temperature centering.

(a) 5% of seeds

Dataset	No centering	Centering	Variation
Cora	0.69 ± 0.02	**0.71** ± 0.02	+2%
Citeseer	**0.48** ± 0.01	**0.48** ± 0.01	0%
PubMed	0.76 ± 0.01	**0.78** ± 0.01	+2%
Email	0.12 ± 0.04	**0.22** ± 0.03	+85%
PolBlogs	0.82 ± 0.12	**0.87** ± 0.01	+7%
WikiSchools	0.08 ± 0.06	**0.44** ± 0.03	+472%
WikiVitals	0.29 ± 0.06	**0.63** ± 0.02	+116%
WikiVitals+	0.31 ± 0.03	**0.65** ± 0.01	+112%

(b) 10% of seeds

Dataset	No centering	Centering	Variation
Cora	0.74 ± 0.02	**0.75** ± 0.01	+1%
Citeseer	**0.52** ± 0.01	**0.52** ± 0.01	0%
PubMed	0.78 ± 0.01	**0.79** ± 0.00	+1%
Email	0.21 ± 0.04	**0.31** ± 0.03	+43%
PolBlogs	0.86 ± 0.02	**0.87** ± 0.01	+1%
WikiSchools	0.13 ± 0.04	**0.50** ± 0.02	+295%
WikiVitals	0.43 ± 0.04	**0.67** ± 0.01	+57%
WikiVitals+	0.61 ± 0.01	**0.68** ± 0.01	+12%

(c) 20% of seeds

Dataset	No centering	Centering	Variation
Cora	**0.78** ± 0.01	**0.78** ± 0.01	0%
Citeseer	**0.57** ± 0.01	**0.57** ± 0.01	0%
PubMed	**0.80** ± 0.00	**0.80** ± 0.00	0%
Email	0.32 ± 0.03	**0.40** ± 0.02	+24%
PolBlogs	**0.87** ± 0.01	**0.87** ± 0.01	0%
WikiSchools	0.27 ± 0.03	**0.57** ± 0.02	+110%
WikiVitals	0.58 ± 0.02	**0.70** ± 0.01	+22%
WikiVitals+	0.65 ± 0.01	**0.71** ± 0.00	+9%

6 Conclusion

We have proposed a novel approach to node classification based on heat diffusion. Specifically, our technique consists in centering the temperatures of each solution to the Dirichlet problem before classification. We have proved the consistency of this algorithm on a simple block model and shown that the temperature centering brings significant performance gains on real datasets. This is a crucial observation given the popularity of the algorithm.

The question of the consistency of the algorithm when the mean temperature is computed over free nodes (instead of all nodes) remains open. Another interesting research perspective is to extend our proof of consistency of the algorithm to *stochastic* block models, where edges are drawn at random [1].

Appendix

A Proof of Lemma 1

Proof. In view of (2), we have:

$$(n_1(p-q)+nq)T_1 = s_1 p + (n_1 - s_1)pT_1 + \sum_{j\neq 1}(n_j - s_j)qT_j,$$
$$(n_k(p-q)+nq)T_k = s_1 q + (n_k - s_k)pT_k + \sum_{j\neq k}(n_j - s_j)qT_j,$$

for $k = 2, \ldots, K$. We deduce:

$$\begin{aligned}(s_1(p-q)+nq)T_1 &= s_1 p + Uq,\\ (s_k(p-q)+nq)T_k &= s_1 q + Uq \qquad \forall k = 2,\ldots,K,\end{aligned}$$

with

$$U = \sum_{j=1}^{K}(n_j - s_j)T_j.$$

The proof then follows from the fact that

$$n\bar{T} = s_1 + \sum_{j=1}^{K}(n_j - s_j)T_j = s_1 + U.$$

B Proof of Theorem 1

Proof. Let $\Delta_k^{(1)} = T_k - \bar{T}$ be the deviation of temperature of non-seed nodes of block k for the Dirichlet problem associated with label 1. In view of Lemma 1, we have:

$$\begin{aligned}(s_1(p-q)+nq)\Delta_1^{(1)} &= s_1(p-q)(1-\bar{T}),\\ (s_k(p-q)+nq)\Delta_k^{(1)} &= -s_k(p-q)\bar{T} \qquad k = 2,\ldots,K,\end{aligned}$$

For $p > q$, using the fact that $\bar{T} \in (0,1)$, we get $\Delta_1^{(1)} > 0$ and $\Delta_k^{(1)} < 0$ for all $k = 2, \ldots, K$. By symmetry, for each label $l = 1, \ldots, K$, $\Delta_l^{(l)} > 0$ and $\Delta_k^{(l)} < 0$ for all $k \neq l$. We deduce that for each block k, $\hat{y}_i = \arg\max_l \Delta_k^{(l)} = k$ for each free node i of block k.

References

1. Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. J. Mach. Learn. Res. (2008)
2. Berberidis, D., Nikolakopoulos, A.N., Giannakis, G.B.: Adadif: Adaptive diffusions for efficient semi-supervised learning over graphs. In: International Conference on Big Data. IEEE (2018)
3. Chung, F.R.: Spectral graph theory. American Mathematical Soc. (1997)
4. Donnat, C., Zitnik, M., Hallac, D., Leskovec, J.: Learning structural node embeddings via diffusion wavelets. In: International Conference on Knowledge Discovery & Data Mining. In: ACM (2018)
5. Kondor, R.I., Lafferty, J.: Diffusion kernels on graphs and other discrete structures. In: Proceedings of the 19th international conference on machine learning (2002)
6. Li, Q., An, S., Li, L., Liu, W.: Semi-supervised learning on graph with an alternating diffusion process. CoRR (2019)
7. Ma, H., King, I., Lyu, M.R.: Mining web graphs for recommendations. IEEE Transactions on Knowledge and Data Engineering (2011)
8. Newman, M.E.J., Girvan, M.: Mixing patterns and community structure in networks. In: Pastor-Satorras, R., Rubi, M., Diaz-Guilera, A. (eds.) Statistical Mechanics of Complex Networks, pp. 66–87. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/978-3-540-44943-0_5
9. Rossi, E., Kenlay, H., Gorinova, M.I., Chamberlain, B.P., Dong, X., Bronstein, M.M.: On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. In: Proceedings of Machine Learning Research (2022)
10. Thanou, D., Dong, X., Kressner, D., Frossard, P.: Learning heat diffusion graphs. IEEE Transactions on Signal and Information Processing over Networks (2017)
11. Tremblay, N., Borgnat, P.: Graph wavelets for multiscale community mining. IEEE Transactions on Signal Processing (2014)
12. Zachary, W.W.: An information flow model for conflict and fission in small groups. J. Anthropol. Res. (1977)
13. Zhu, X.: Semi-supervised learning with graphs. Ph.D. thesis, Carnegie Mellon University (2005)
14. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th International conference on Machine learning (2003)

Leveraging the Power of Signatures for the Construction of Topological Complexes for the Analysis of Multivariate Complex Dynamics

Stéphane Chrétien[1,4(✉)], Ben Gao[1,2], Astrid Thébault Guiochon[3], and Rémi Vaucher[1,2]

[1] Laboratoire ERIC, Université Lyon 2, Lyon, France
{stephane.chretien,ben.gao,remi.vaucher}@univ-lyon2.fr
[2] HALIAS, Paris, France
{ben.gao,remi.vaucher}@halias.fr
[3] Université Lyon 2, Lyon, France
a.thebaultguiochon@univ-lyon2.fr
[4] The Alan Turing Institute, London, UK
https://sites.google.com/site/stephanegchretien/home ,
https://remivaucher.github.io , https://linktr.ee/astridthebaultg

Abstract. Topological Data Analysis is a field of great interest in many applications such as finance or neuroscience. The goal of the present paper is to propose a novel approach to building simplicial complexes that capture the multiway ordered interactions in the components of high-dimensional time series using the theory of Signatures. Signatures represent one of the most powerful transforms for extracting group-wise structural features and we put them to work in the task of discovering statistically meaningful simplices from a complex that we estimate sequentially. Numerical experiments on an fMRI dataset illustrates the efficiency and relevance of our approach.

Keywords: Topological Data Analysis · Signatures · simplicial complex · multiway interactions · high-dimensional time series · fMRI data analysis

1 Introduction

Topological Data Analysis (TDA) is a new field with a wide range of application in fields such as finance, neuroscience, medicine, etc. TDA addresses the problem of accounting for groupwise interactions in the data and therefore opens very promising prospects to better apprehend complex phenomena than models relying on pairwise interactions only. Several tools from algebraic topology, such as homology groups, homotopy groups, Betti numbers, etc. can be put to work in building a set of relevant features that can capture the intricate nature of dependencies.

The original version of the chapter has been revised. A correction to this chapter can be found at https://doi.org/10.1007/978-3-031-53468-3_39

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 283–294, 2024.
https://doi.org/10.1007/978-3-031-53468-3_24

Some compelling examples of the benefits of using topological features appear regularly in the literature. In [17] the homological features of brain functional networks are shown to take different values in two states depending on the absorption of some drug. More generally, it is shown in [21] that homological cycles in structural brain networks finds connections between regions of early and late evolutionary origin. TDA can also be efficiently used in dynamical settings as well. One very intriguing example is change detection as illustrated in the study of functional brain networks conditioned in different tasks [22]. In [12] it is investigated how speech-related brain regions connectivity changes in different scenarios of speech perception.

In the present paper, we focus on the analysis of dynamical high dimensional phenomena and on the problem of constructing associated relevant topological structures with the aim of proposing news computational tools for deepening our understanding of the higher order structures hidden in time series data. Our main contribution is a new approach to building statistically informed simplicial complexes and possibly more general structures.

Our two main tools will be the basic objects of TDA and Signature theory. Signatures were recently proposed as a very powerfull feature map for time series and dynamical systems in [5,7,14]. The introduction of Signatures for building topological structures for high dimensional dynamical phenomena is new and appears as a key and very natural ingredient that can accurately account for orientations of the various simplices in the complex at hand while capturing the main shape features from the dynamics. Using Signature in a statistical/machine learning context is an approach which is adopted in a growing number of applications nowadays [7] and our work is also intended to illustrate the relevance of Signature theory combined with statistics/machine learning for building a higher order interaction modelling framework.

In mathematical terms, our proposal is based on the assumption that k-simplices are simply sets of k nodes with there time series attached to them, with an orientation prescribed by the ordering of the nodes in computing their associated k-Signature. Recall that the orientation encoded in the computation of the associated signature carries potentially very interesting interpretation about the causal dependencies of the times series [11]. In the next step, the relevance of incorporating a simplex into our simplicial complex is assessed using a purely statistical procedure: each oriented k-simplex is associated with a corresponding k-Signature that is included into a set of multivariate features that is used to predict the Signatures of all the other potential simplices. More precisely, our construction is a generalisation of the approach developed by Meinshausen and Bühlmann in [16] for Gaussian Graphical Models. Simplices that are selected from the set of potential Simplices whose Signature can predict the Signature of a target simplex in terms of confidently regressing or predicting[1] the Signature associated with this target are included as candidates for being considered as adjacent to this target. Using this procedure, we obtain a construction of a simplicial complex that accurately incorporates the statistical relationships between all the simplices in terms of regression or prediction, while keeping track of the inherent orientations of the simplices.

[1] for time dependent Signatures.

The plan of the paper is as follows. In Sect. 2, we recall the necessary background on topological data analysis and Signature theory. In Sect. 3, we present our method for constructing the simplicial complex using the Signatures of the simplices and the LASSO algorithm. In Sect. 4, we present our numerical experiments on real datasets. A conclusion section completes the paper.

2 Background on Signatures and Topology

In this section, we summarise the mathematical prerequisites from topology and the theory of Signatures.

2.1 Recalls on the Theory of Signatures

The theory of Signatures is a new topic of growing interest that emerged as a sub-branch of the theory of rough paths [9,10,15] which has a long history in mathematics and control that may have started with the work of Chen [4]. Rough paths provide a new framework for the analysis of stochastic processes and permitted to resolve various open problems, including the existence of a solution to the KPZ equation in mathematical physics, a result for which Martin Heirer was awarded the Fields medal [6]. Recently, this theory developed as a new tool for the analysis of signals in the area of Machine Learning [5,7,14] where remarkable performance was achieved for a series of difficult practical problems including the analysis of financial data, medical data and textual data [13,14]. Lately, an intriguing relationship with recurrent neural networks was exhibited using the viewpoint of control theory [8].

Let us now turn to the definition and some interesting properties of Signatures. Consider a d-dimensional path $X = (X^1, X^2, ..., X^d) : \mathbb{R} \to \mathbb{R}^d$. Then, the (truncated)[2] **signature** of X on $[a, b]$ is an object in $\mathcal{T}(\mathbb{R}^d) = \mathbb{R}^d \oplus \mathbb{R}^{d\times d} \oplus \mathbb{R}^{d\times d\times d} \oplus \dots$, defined, for $j \in \mathbb{N}^*$ by

$$\begin{aligned}(S_{[a,b]}(X))_{i_1,i_2,\dots,i_j} &:= S^{i_1 i_2 \dots i_j}_{[a,b]}(X) \\ &= \int\limits_{a\le s_1\le s_2\le\dots\le s_j\le b} dX^{i_1}_{s_1} dX^{i_2}_{s_2} \dots dX^{i_j}_{s_j} \end{aligned} \tag{1}$$

which lies in $\mathbb{R}^{\overbrace{d\times d\times\dots\times d}^{j}}$.

Chen's identity is a very useful result that allows to compute the Signature recursively based on linear interpolation of observed values of a trajectory.

Theorem 1 (Chen's identity). *Let $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$. Consider the **concatenation** of X and Y (noted $X * Y$) defined by:*

$$\begin{aligned}(X * Y) : [a, c] &\to \mathbb{R}^d \\ t &\mapsto \begin{cases} X(t) & , \quad t \in [a, b] \\ X(b) - Y(b) + Y(t) & , \quad t \in [b, c]. \end{cases}\end{aligned}$$

[2] The k-truncated version of the signature is $S^{(1)}(X) \oplus S^{(2)}(X) \oplus \dots \oplus S^{(k)}(X)$.

Then:

$$S_{[a,c]}(X * Y) = S_{[a,b]}(X) \otimes S_{[b,c]}(Y)$$

Augmentation of a Path. The Signature defines X uniquely on $[a, b]$ close to *tree-like equivalence* (i.e. there exist $I, J \subset [a, b]$, such that $X|_I(t) = X|_J(b-t)$).

Proposition 1. *$S_{[a,b]}(X)$ define X uniquely on $[a, b]$ if there exists $1 \leq i \leq d$ such that X^i is strictly monotonic on $[a, b]$.*

This result leads to consider the **time-augmented path** $\tilde{X}$ associated with X, defined as $\tilde{X} = (t, X^1, X^2, \ldots, X^d)$ in order to ensure the unicity of $S(X)$. Another augmentation will be useful for our work, namely the Lead-Lag augmentation.

Definition 1. *Consider a d-dimensional path X with $T+1$ timesteps. The **lead-lag augmentation** of X is a $2d$-dimensional path $X_{lead,lag} = (X^{Lead}, X^{Lag})$ with $2T + 1$ time steps such that:*

$$\begin{aligned} X^{Lead} &= \{X(0), X(1), X(1), X(2), \ldots, X(T), X(T)\} \\ X^{Lag} &= \{X(0), X(0), X(1), X(1), \ldots, X(T-1), X(T)\} \end{aligned}$$

The Lead-Lag augmentation was found to play an important role in many machine learning applications [7].

2.2 Recalls on Topology

We now turn to some useful definitions from topology. Consider a set of n vertices $V = \{v_1, \ldots, v_n\}$.

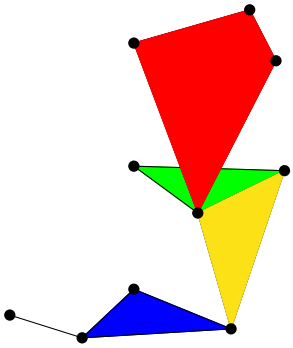
Definition 2. *For $k < n$, a **k-simplex** σ_k of V is the collection of a subset V_k of length $k+1$ and all its subsets. The **geometric realization** of a k-simplex is the convex hull C of $k + 1$ points, such that $\dim(C) = k$. A face (of dimension l) σ_k is a collection of set in σ_k that form a l-simplex ($l \leq k$).*

Definition 3. *A **simplicial complex** $\mathcal{C}$ on V is a collection of simplices (of V) such that for every $\sigma^i \in \mathcal{C}$, there exists j with $\sigma^i \cap \sigma^j$ a sub-simplex of both σ^i and σ^j.*

Definition 4. *The **dimension** of $\mathcal{C}$ is the dimension of the highest simplex in $\mathcal{C}$ (i.e. the highest k such that there exist a k-simplex in $\mathcal{C}$).*

Definition 5. *An **orientation** on $\mathcal{C}$ is a order of the vertices for every simplicies in $\mathcal{C}$. We use the notation $[v_1, v_2, v_3]$ to denote an oriented 3-simplex.*

From a geometric point of view, $\mathcal{C}$ is constructed by attaching a group of simplices to each other by binding them with a shared face.



Definition 6. *Let Δ be a simplicial complex and σ one of its k-sub-simplices. The* ***neighborhood*** *of v in Δ is the set $Lk(\sigma, \Delta)$ of all the k-sub-simplices τ of Δ such that:*

- $\sigma \cap \tau = \emptyset$
- $\sigma \cup \tau$ *is a face of* Δ.

Especially, the neighborhood of a vertex v is all the vertices v_i such that the edge $\{v, v_i\} \in \Delta$.

The goal of our work is to build a simplicial complex encoding the groupwise interactions explaining the dependencies inside high dimensional time series. The main ingredient in our approach is to make use of relevance measures in predicting *simplex indexed*-groups of time series using other *simplex indexed*-groups of time series as a criterion for selecting potential higher and higher dimensional simplices for (or for the sake of mitigating the computational complexity, sequential greedy) aggregations. We turn to the description of our approach in the next section.

3 Building Simplicial Complexes Between Time Series

3.1 Presentation of the Method

In the same spirit as for Gaussian graphical models [16], our goal is to infer a higher order model from data using a model selection based numerical procedure. One brute force method is to use sparse solutions of LASSO type estimators that can be employed to predict all sub-groups of times series based on all other groups of time series.

In the present paper, our goal is to propose a better structured solution to the problem of capturing interesting structures in the dependency relationship among the various components of high dimensional time series. Interesting types of structure often come from Topological Data Analysis (TDA) as presented in e.g. [3]. Inferring such structures is however extremely computationally extensive, and even more so if we account for the necessity of using Cross-Validation types of hyper parameter calibration procedures.

Our proposal is to adopt a principled sequential approach to simplicial complexes estimation. In our approach, the simplices that will be incorporated into the simplicial complex are chosen among completions of existing simplices into one order higher simplices, hence allowing to stratify the construction by the complexity of the interactions. The selection is performed using the LASSO, complemented by inspection of the R^2 criterion. One key ingredient of our construction is the use of the Signature transform as features for prediction. Signatures bring an essential information to the construction of our simplicial complex, namely orientation, since, Signatures encode the order in which the times series are integrated in (1). Using the orientation can be instrumental for the interpretation of the interactions among the various components of high dimensional time series, and noticing any difference in the prediction capabilities of two different orderings might be extremely useful in practice.

We now turn to the details of the implementation.

3.2 Our Greedy Order Stratified Algorithm:

Consider a group of d (augmented) time series $\mathcal{G} = \{X^1, X^2, \ldots, X^d\}$ with $T+1$ timesteps each of the form $X^i = [(0, X^i(0)), (t_1, X^i(t_1)), \ldots, (t_T, X^i(t_T))]$.

In this section, we introduce our main contribution, namely the construction of a simplicial complex that encodes the multiway dependencies of the various dimensions in a high-dimensional time series. We also discuss a greedy algorithm for sequentially building the sought for simplicial complex that mitigates the computational complexity. As mentioned earlier in the introduction, the main principle of the our algorithm is to build consistent simplices within groups of time series using regression or prediction error measures. In our implementation, we chose to present the *Signature prediction* version, consisting of predicting the Signature at a simplex as a linear function of the signatures at other simplices of various orders.

In order to keep control on the computational complexity of the method, we now present a lighter greedy algorithm. For any $t \in [\![0, T-L]\!]$ with L to be specified.

Closure of a simplicial complex.

In our framework, we need to introduce the following definition.

Definition 7. *Let $\mathcal{C}$ denote a simplicial complex. We denote by $\overline{\mathcal{C}}$ the simplicial complex consisting in appending all k-simplices whose **single** incorporation results in creating a simplex of order $k+1$ using the simplices already present in $\mathcal{C}$ only.*

The sequential algorithm.

We now define the steps of our sequential greedy method as follows.

Data: Set $\ell = 1$ and set $\mathcal{C}^{(1)} = \{1, \ldots, d\}$.
Result: The time series interaction simplicial complex
while *No more simplex is selected* **do**
  Select an (augmented) k-subset of nodes $C^J = \{X^{j_1}, X^{j_2}, \ldots, X^{j_k}\}$, with $J = \{j_1, \ldots, j_k\}$ in $\mathcal{C}^{(\ell)}$, and compute $S_{[t,t+L]}(C^J)$.;
  For every (augmented) k'-combination $C^{J'} = \{X^{j'_1}, X^{j'_2}, \ldots, X^{j'_{k'}}\}$ with $J' = \{j'_1, \ldots, j'_{k'}\}$ in $\overline{\mathcal{C}^{(l)}}$, compute $S_{[t,t+L]}(C^{J'})$;
  Predict $S_{[t,t+L]}(C^J)$ from $\left(S_{[t,t+L]}(C^J)\right)_{\left(^{1\leq j\leq d-k-1}_{k+1}\right)}$ with LASSO;
  if $R^2 > 0,67$ **then**
    Select all non-zero β_j LASSO coefficients
  else
    Set $\beta_j = 0$ for all j.
  end
end

Algorithm 1: Sequential construction of the simplicial complex

Each non-zero β_j coefficient represents a k-simplex whose vertices are $\{X_i, X_{j_1}, X_{j_2}, \ldots, X_{j_{k+1}}\}$. This simplex comes with a natural weight w. As this k-simplex can be produced multiple time (by predicting X_i with $\{X_{j_1}, \ldots, X_{j_{k+1}}\}$ or X_{j_l} by $\{X_i, X_{j_1}, \ldots, X_{j_{k+1}}\}_{k \neq l}$), this weight is produced as the sum of all the non-zero LASSO coefficients obtained.

By iterating the procedure for every possible simplices whose signature is a statistically interesting quantity to predict, and every dimension of simplex $k \leq K$ (for a chosen k), one can produce a simplicial complex among $\mathcal{G}$.

Remarks

Let us now address some technical question that arise from the proposed construction.

Time dependancy: The algorithm is applied to evolving time series for which the computation of the Signatures is updated incrementally and prediction is performed using these updated Signatures as time increases.

Orientation: This algorithm gives a natural orientation on every simplex, as $S(X_i, X_j) \neq S(X_j, X_i)$ which is often of great potential use for interpretability.

4 Numerical Experiments and Applications

Multivariate times series are omnipresent and high order correlation occurred frequently in e.g. the domains of finance and neuroscience. We evaluated our method on two public data sets analysed by [19] : the fMRI resting-state data from the HCP https://www.humanconnectome.org/.

4.1 Practical Choices and Hyper Parameters

We consider only simplices up to triangles $k \leq 2$ for now. Due to the concerns about complexity, the depth of signature is set to $depth = 2$ for the construction of both 1-simplex and 2-simplex. More precisely, we use time-augmented path to calculate 2-truncated signatures of 0-simplices, in order to construct 1-simplices. As a design choice, for 2-simplices the predictors (1-simplices composed of two times series) are not augmented when applying LASSO regression. Clearly, many different choices could be imposed on how we model the dependencies between subgroups of time series.

The regularisation term of LASSO is crucial for our method since it directly controls the sparsity in prediction using the linear models on signature features. Recall that the selected groups of time series will immediately be translated into new simplices in our sequentially growing simplicial complex.

In the present numerical experiments, we show that coarsely selected values for these hyperparameters already provide interesting results on a real dataset. In practice, $\lambda_{1-simplex}$ and $\lambda_{2-simplex}$ have been empirically tuned to the values $\lambda_{1-simplex} = 1000$ and $\lambda_{2-simplex} = 3$ for the fMRI dataset. More experiments based on extensive comparisons over a refined grid will be tested in an extended version of the present paper. The latest version of our implementation is available on our GitHub page :

https://github.com/ben2022lo/conf-complex-network

4.2 Modelling Interactions in Functional MRI Datasets

Functional connectivity is a neuroscience approach aimed at understanding the organization of the human brain based not solely on spatial proximity and structural factors, but rather on its functionality, i.e. its connectivity patterns between different brain regions and networks. For instance, even seemingly routine tasks such as paying attention during a lecture have been found to activate regions like the pulvinar (within the thalamus), the superior colliculus (in the midbrain), and the posterior parietal cortex [18]. In this perspective, and given that functional brain imaging data can be regarded as time series, the theory of Signatures could prove to be particularly useful.

In the absence of specific tasks or external stimulation (resting, meditating, sleeping, etc.), the brain enters what is known as resting-state. The Default Mode Network (DMN) becomes prominently active during this resting state. This neural network includes key regions such as the medial prefrontal cortex (mPFC), the posterior parietal lobe (PTL), the posterior cingulate cortex (PCC), and the precuneus [2].

We tested our method on resting-state fMRI(rs-fMRI) data[3] preprocessed by the same pipeline in [19]. The dataset contains 100 cortical (Schaefer100 [20]) and 19 subcortical ROIs (Regions of Interest). In order to evaluate the quality of identified interactions, we have selected a subset of 15 ROIs of which

[3] HCP, http://www.humanconnectome.org/.

functional connectivities during resting-state are well known. We constructed simplicial complex on all 1200 timesteps, and analysed the top 10 1-simplices and 2-simplices that are the most persistent, i.e.that occurred on most time-steps. Besides, we observed that the life duration distribution of 1-simplices is centred and symmetric, whereas the distribution of 2-simplices is positively skewed (Fig. 1).

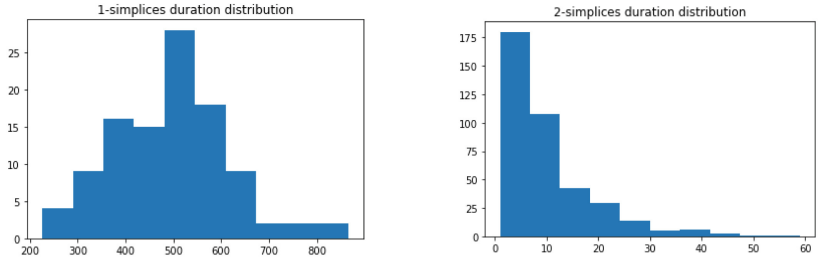


Fig. 1. Histograms of the observed duration for the discovered 1 and 2-simplices

The 1-simplex representing the interaction between *7Networks LH Vis 9* and *7Networks LH SomMot 4* occurred on most occasions (865 time steps). The most persistent 2-simplex (59 time steps) represents the interaction among *LH Cont Par 1*, *RH Default PFCdPFCm 2* and *RH Default pCunPCC 2*.

Most of the persistent 1-simplices involve the prefrontal cortex, the parietal lobe and the precuneus, which is consistent with literature as all three regions are active during resting-state [2]. The most persistent interaction include sub-regions of the left hemisphere's visual and somatomotor networks. Component *LH SomMot 4* has previously been associated with components of the left (*LH Default PFC*) and right (*RH Default PFCv 2*) PFC in a study proposing an age prediction pipeline Ayu using rs-fMRI data [1]. In the same study, several visual areas (*RH Vis 1, 3* and *4*) are linked to the PFC areas during rs-fMRI (*LH Default PFC 1, 2,* and *3*), although they are located in the right hemisphere.

The top 10 interactions that occurred the most include components from the same three recurrent brain areas that are the PFC, parietal regions and the precuneus, with the latter taking par in all 10 of them. This aligns with previous work [2] as all three are indeed involved in the Default Mode Network which is active during rs-fMRI.

The most persistent simplicies and matching ROIs are represented in Fig. 3. In particular, the interactions discovered using our approach show excellent coherence with well identified spatial activity zones (Fig. 2).

5 Discussion and Future Work

The qualification of high-order interaction of time series is a relatively new research area. Previous work, such as [19], tried to estimate the higher

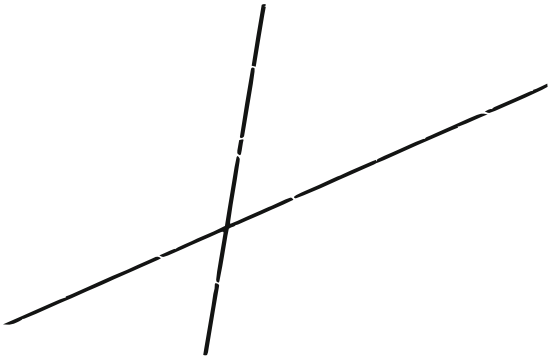


Fig. 2. Simplicial complex constructed with persistent simplicies. 1-simplices are blue, 2-simplicies are defined by their orange 1-simplex faces. The gray signifies the coincidence of a 1-simplex and one face of a 2-simplex. The selected ROI and numerated 0-simplices are matched by the following dictionary: 0 - *LH Vis 9*, 1 - *LH SomMot 4*, 2 - *LH DorsAttn Post 4*, 3 - *Cont Par 1*, 4 - *LH Cont pCun 1*, 5 - *LH Default pCunPCC 1*, 6 - *LH Default pCunPCC 2*, 7 - *RH Cont Par 1*, 8 - *RH Cont PFCl 1*, 9 - *RH Cont pCun 1*, 10 - *RH Default Par 1*, 11 - *RH Default PFCdPFCm 1*, 12 - *RH Default PFCdPFCm 2*, 13 - *RH Default PFCdPFCm 3*, 14 - *RH Default pCunPCC 2*.

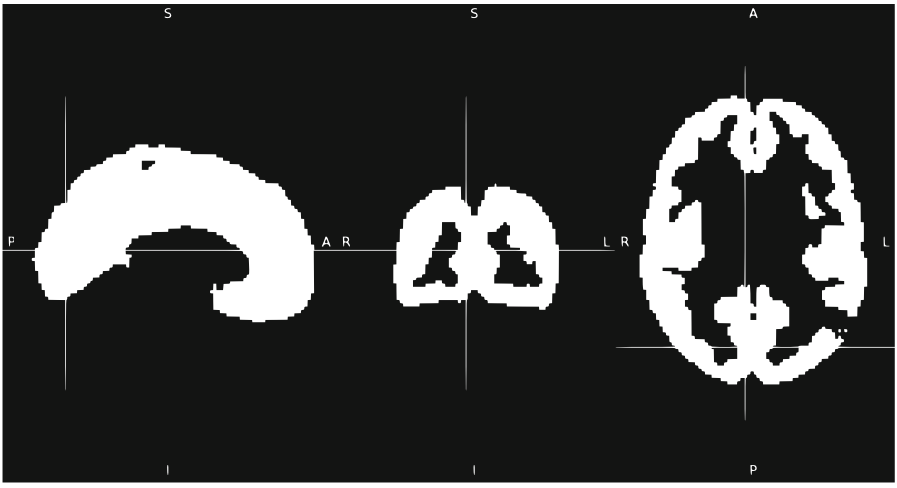


Fig. 3. Anatomical representation of the 10 most persistent 1-simplices and 2-simplices matched to their corresponding network in the 7-network parcellation by [23]. Only parcels that are part of the most persistent simplices are colored. Each color corresponds to a network as per the following color legend: yellow - *Visual*, red - *Somatomor*, purple - *Dorsal Attention*, green - *Control*, blue - *Default*.

order interactions in high dimensional signals. Our method, based on the theory of Signatures that captures higher dimensional interactions together with

what can be encoded as a simplex orientations, is able to leverage much more refined information about the mutual behaviour of the observed phenomena.

From a theoretical point of view, although the orientation of the various simplices discovered in the sequential construction was not used proper, it could be fruitfully exploited in the future. Secondly, the problem of dimension consistency could be appropriately tackled using group LASSO types of techniques or SLOPE-based approaches.

To conclude, we mention that the method we just presented could also be handily put to work on nonstationary problems for e.g. change point detection such as in early detection of epidemics, using human in the loop validation steps. The simplicial complex could be sequentially updated as a function of time as well, leading to a dynamical topological structure whose characteristics and abrupt potential changes could help extract valuable information about the emergence of certain interesting phenomena.

References

1. Borkar, K., Chaturvedi, A., Vinod, P.K., Bapi, R.S.: Ayu-characterization of healthy aging from neuroimaging data with deep learning and rsfmri. Front. Comput. Neurosci. **16**, 940922 (2022)
2. Broyd, S.J., Demanuele, C., Debener, S., Helps, S.K., James, C.J., Sonuga-Barke, E.J.: Default-mode brain dysfunction in mental disorders: a systematic review. Neurosci. Biobehav. Rev. **33**, 279–96 (Oct 2008)
3. Chazal, F., Michel, B.: An introduction to topological data analysis: fundamental and practical aspects for data scientists. Front. Artif. Intell. **4**, 108 (2021)
4. Chen, K.-T.: Integration of paths, geometric invariants and a generalized baker-hausdorff formula. Ann. Math. **65**(1), 163–178 (1957)
5. Chevyrev, I., Kormilitzin, A.: A primer on the signature method in machine learning. arXiv preprint arXiv:1603.03788 (2016)
6. Eckmann, J.P., Genève, U.: Martin Hairer got the fields medal for his study of the KPZ equation
7. Fermanian, A.: Embedding and learning with signatures. Comput. Stat. Data Anal. **157**, 107148 (2021)
8. Fermanian, A., Marion, P., Vert, J.-P., Biau, G.: Framing RNN as a kernel method: A neural ode approach. Adv. Neural. Inf. Process. Syst. **34**, 3121–3134 (2021)
9. Friz, P.K., Hairer, M.: A Course on Rough Paths: With an Introduction to Regularity Structures. Springer International Publishing, Cham (2020)
10. Friz, P.K., Victoir, N.B.: Multidimensional stochastic processes as rough paths: theory and applications, vol. 120 Cambridge University Press (2010)
11. Giusti, C., Lee, D.: Iterated integrals and population time series analysis. In: Baas, N.A., Carlsson, G.E., Quick, G., Szymik, M., Thaule, M. (eds.) Topological Data Analysis: The Abel Symposium 2018, pp. 219–246. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-43408-3_9
12. Kim, H., Hahm, J., Lee, H., Kang, E., Kang, H., Lee, D.S.: Brain networks engaged in audiovisual integration during speech perception revealed by persistent homology-based network filtration. Brain Connectivity **5**(4), 245–258 (2015)
13. Kormilitzin, A., Vaci, N., Liu, Q., Ni, H., Nenadic, G., Nevado-Holgado, A.: An efficient representation of chronological events in medical texts. arXiv preprint arXiv:2010.08433 (2020)

14. Lyons, T., McLeod, A.D.: Signature methods in machine learning. arXiv preprint arXiv:2206.14674 (2022)
15. Lyons, T., Qian, Z.: System control and rough paths. Oxford University Press (2002)
16. Meinshausen, N., Bühlmann, P.: High-dimensional graphs and variable selection with the lasso (2006)
17. Petri, G., et al.: Homological scaffolds of brain functional networks. J. Royal Society Interface. **11**(101), 20140873 (2014)
18. Posner, M.I., Petersen, S.E.: The attention system of the human brain. Annual Rev. Neurosci. **13**, 25–42 (Feb 1990)
19. Santoro, A., Battiston, F., Petri, G., Amico, E.: Higher-order organization of multivariate time series. Nat. Phys. **19**(2), 221–229 (2023)
20. Schaefer, A., et al.: Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity MRI. (July 2017)
21. Sizemore, A.E., Giusti, C., Kahn, A., Vettel, J.M., Betzel, R.F., Bassett, D.S.: Cliques and cavities in the human connectome. J. Comput. Neurosci. **44**(1), 115–145 (2018). https://doi.org/10.1007/s10827-017-0672-6
22. Stolz, B.J., Harrington, H.A., Porter, M.A.: Persistent homology of time-dependent functional networks constructed from coupled time series. Chaos: An Interdiscip. J. Nonlinear Sci. **27**(4) (2017)
23. Yeo, B.T., et al.: The organization of the human cerebral cortex estimated by functional correlation. J. Neurophysiol. **106**, 1125–1165 (June 2011)

Empirical Study of Graph Spectra and Their Limitations

Pierre Miasnikof[1(✉)], Alexander Y. Shestopaloff[2,3], Cristián Bravo[4], and Yuri Lawryshyn[1]

[1] University of Toronto, Toronto, ON, Canada
p.miasnikof@mail.utoronto.ca
[2] Queen Mary University of London, London, UK
[3] Memorial University of Newfoundland, St, John's, NL, Canada
[4] University of Western Ontario, London, ON, Canada

Abstract. We examine the sensitivity of community-structured graph spectra to graph size, block size and inter-block edge probability. We use the Planted Partition Model because of its transparency. While this generative model may seem simplistic, it allows us to isolate the effects of graph and block size, edge probabilities and, consequently, vertex degree distribution on spectra. These sensitivities to key graph characteristics also generalize beyond Planted Partition Model graphs, because they are based on graph structure. Notably, our results show that eigenvalues converge to those of a complete graph, with increases in graph size or inter-block edge probability. Such convergence severely limits the use of spectral techniques.

Keywords: Spectral graph theory · eigengap · community detection

1 Introduction

Graphs have several matrix representations. The adjacency and the several Laplacian matrices are instances of these representations. Spectral decomposition of these matrices is used for many tasks in the study of graphs, especially those with community structure (complex networks) [5]. For example, it is used for vertex clustering [4,7,10,11,19,21,22] and in the "eigengap heuristic" [4,19]. In this short article, we identify the limitations of spectral decomposition of these graph matrix representations. In doing so, we also compare results obtained by decomposing the two most commonly used matrix representations, the adjacency and the symmetric normalized Laplacian matrices.

Our empirical investigations reveal that normalized Laplacian eigenvalues (eigenvectors) are extremely sensitive to noise and scale. This noise manifests itself in the form of edge randomness. This randomness is a consequence of inter-block edge probability or small block size. We find that increases in noise or graph size lead to eigenvalue uniformity, which renders spectral methods of limited use. Indeed, spectral techniques rely on eigenvalue differentiability. Obviously,

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 295–307, 2024.
https://doi.org/10.1007/978-3-031-53468-3_25

as eigenvalues move to uniformity, differentiability vanishes. Unfortunately, we also find the adjacency matrix does not provide a viable alternative in cases where Laplacian spectra are uninformative.

We investigate the behavior of the spectra of graphs displaying community structure, in order to help target the application of spectral techniques to instances where they are more likely be informative and to yield meaningful results. The work in this article is focused on the link between a few key graph properties and those of the corresponding symmetric normalized Laplacian matrix. Although we focus our attention on the widely used normalized symmetric Laplacian matrix, we also examine the eigenvalues of the adjacency matrix, for the sake of completeness.

Of course, spectral techniques are also limited by computational constraints (e.g., memory, precision, computation times, etc.). However, in this article, we do not examine these computational issues. Nevertheless, we do note that the limitations identified in this work are amplified by the various computational and tractability constraints. For the sake of brevity, the scope of this article is also restricted to simple graphs, unweighted undirected graphs with no self-loops or multiple edges and graphs with only a single connected component.

2 Previous Work

As mentioned earlier, spectral techniques are widely used in the study of graphs. The foundations of this area of study were laid by Chung [6] and later by Spielman [24]. In this short work, we focus on three areas of the spectral graph analysis literature. First, our experiments motivate us to examine past inquiries into the convergence of eigenvalues, under the stochastic block model (SBM) [23]. Then, in order to gain a better understanding of this convergence, we also survey studies in which authors have established a link between Laplacian eigenvalues and vertex degree [5,27]. Finally, we are also motivated by authors who have highlighted the differences in the conclusions of analyses based on adjacency and Laplacian matrix representations [18,21].

Asymptotic convergence of eigenvectors (and consequently eigenvalues) under the SBM was identified by Rohe et al. [23]. These authors posit the existence of a "population Laplacian" towards which the empirical Laplacian converges, as the graph grows. While our results do not agree with these authors' conclusions, we also document convergence with increased graph size and noise in connectivity.

Under random graph models, like the SBM or the Erdös-Rényi-Gilbert (ERG) model [9,13], edge probabilities are independent of each other and only depend on the nodes they are connecting. These models have often been described as too simplistic to represent real world networks [1,2,15,20,23]. In particular, the degree uniformity yielded by these generative models, its lack of skewness or heavy right tail, has been identified as a weakness as models of real world networks.

Nevertheless, random graph models have been found to be adequate in many empirical cases [10,11,16,20,22]. For example, Newman et al. [20] state that

"in some cases random graphs with appropriate distributions of vertex degree predict with surprising accuracy the behavior of the real world". In closing, while a detailed examination of this debate on realistic models of real world networks is beyond the scope of our work, we do note that some authors claim that networks with power-law degree distributions are rare (e.g., [3]). We also note that the SBM is still used as a model of real world networks in the recent literature (e.g., [12])

3 Mathematical Background

As discussed earlier, there are several matrix representations of graphs. The two most commonly used are the adjacency matrix (A) and the symmetric normalized Laplacian ($\mathcal{L}$). Because of its symmetry and specific properties, we use the symmetric normalized Laplacian, instead of the unnormalized or random walk Laplacians, in this work.

In order to study the link between graph characteristics and spectra, we generate several synthetic random graphs with known structure. Indeed, by modifying the parameters of random graph generative models, we are able to isolate and unambiguously observe the sensitivities of the spectra. We use the Planted Partition Model (PPM), a special case of the SBM, for its clarity. Using the PPM also allows us to compare our conclusions to those reported in the literature (e.g., [23]). We use the Python NetworkX library [14,25,26] to generate our graphs.

For the remainder of this article, we will use the following naming conventions:

- $N = |V|$ is the total number of vertices,
- n_k is the number of vertices in block k,
- d_i is the degree of the $i-$th vertex and
- $\lambda_0 = 0 < \lambda_1 \leq \ldots \leq \lambda_N$ are the eigenvalues of the $(N \times N)$ normalized symmetric matrix $\mathcal{L}$,
- Because we only consider graphs with one connected component, only the first eigenvalue is equal to 0, all others are strictly positive.

3.1 Matrices Under Consideration

As mentioned earlier, a graph can be represented by various matrices. In this work, we focus on two, the adjacency matrix (A) and the normalized symmetric Laplacian ($\mathcal{L}$). The latter is a simple transformation of the former.

Adjacency Matrix. The adjacency matrix A for a graph with N vertices is defined as

$$\underset{(N\times N)}{A} = [w_{ij}]$$

with,

$$w_{ij} \in \{0, 1\} \text{ and}$$
$$w_{ii} = 0, \forall i\,.$$

Throughout this article, the matrix A is symmetric and all matrix elements (edge weights w_{ij}) are binary (i.e., graphs are unweighted & undirected). Also, because we only consider simple graphs, all diagonal elements are equal to zero (i.e., no self-loops $\Leftrightarrow A_{ii} = w_{ii} = 0$).

Normalized Symmetric Laplacian. The (un-normalized) Laplacian matrix is defined as

$$L = D - A$$

where,

$$D = \begin{bmatrix} d_1 & 0 & \dots \\ 0 & \ddots & \dots \\ 0 & \dots & d_n \end{bmatrix} .$$

Here, d_i is the degree of the $i-th$ node. It is obtained by summing the $i-th$ row of the adjacency matrix:

$$d_i = \sum_j A_{ij} .$$

The corresponding symmetric normalized Laplacian $\mathcal{L}$ is defined as

$$\begin{aligned} \mathcal{L} &= D^{-1/2} L D^{-1/2} \\ &= D^{-1/2}(D - A)D^{-1/2} \\ &= I - D^{-1/2} A D^{-1/2} . \end{aligned}$$

(by convention, $\frac{1}{\sqrt{d_i}} = 0$, if $d_i = 0$)

The spectra of normalized symmetric Laplacian matrices have several appealing properties. Symmetry, which guarantees real-number eigenvalues, is the most obvious. However, there are several other very informative features. In the next section, we will review the most important of these properties.

3.2 Eigenvalues of Normalized Symmetric Laplacian Matrices

While the topic of spectral graph theory is very vast, here, we only review the few properties of the eigenvalues which are used in this work. For a thorough treatment of this topic, we refer the reader to Fan Chung's seminal text "Spectral graph theory" [6]. Here, we assume a graph with only one connected component.

- Naturally, all eigenvalues of the normalized symmetric Laplacian are real.
- The normalized symmetric Laplacian is positive semidefinite, all its eigenvalues are non-negative: $\lambda_i \geq 0,\ \forall i$.
- Since all graphs considered in this work have only one connected component, in this work the following properties hold:
  - $\diamond\ \sum_i \lambda_i = |V| = N,$
  - $\diamond\ \lambda_0 = 0 < \lambda_1 \leq \ldots \leq 2$ (i.e., only one zero eigenvalue).
- In a complete graph, $\lambda_i = N/(N-1) \gtrapprox 1,\ \forall i > 0$.

3.3 Eigengap Heuristic

Very commonly, the spectra of normalized symmetric Laplacians are used to obtain the so-called "eigengap heuristic" [4,19]. This heuristic provides a reliable estimate of the number of clusters (communities) in a graph. Many graph clustering techniques that require the number of clusters as input rely on this technique.

The eigengap heuristic consists of the following steps: (one connected component case)

- compute eigenvalues of the normalized symmetric Laplacian matrix,
- sort the eigenvalues in ascending order, i.e., $\lambda_0 < \lambda_1 \leq \ldots \leq \lambda_{N-1}$,
- the number of communities on the graph (K) is approximately equal to the index at which the eigenvalues (of the Laplacian) display their first spike.

In other words, this heuristic tells us: if $\lambda_i \ll \lambda_{i+1} \Rightarrow K \approx i$ (where, $i \in \{1, \ldots, N-1\}$).

In our empirical experiments, we use PPM graphs with a known number of blocks (i in the case above). To examine the sensitivities of the eigengap, we compute the ratios $\left(\frac{\lambda_{i+1}}{\lambda_i}\right)$. (N.B.: in all our PPM graph examples, $K = i$ is known)

3.4 Graphs Under Consideration

While the ERG model is the prototypical random graph generative model, there exist other models that yield less trivial structures. The Stochastic Block Model (SBM) and a special case of it called the Planted Partition Model (PPM) [8,10,11] are two such instances. These two models are often used to generate examples of networks with community structure [10,11]. Indeed, such graphs are composed of blocks (communities, clusters) of vertices that are densely connected while only being sparsely connected to the remaining graph.

The PPM is a special case of the more general SBM. Under the SBM each block of vertices has its own within-block edge probability, each block pair has its own inter-block edge probability and blocks have varying numbers of vertices. In the PPM, all blocks have the same number of nodes, while within-block and between-block edge probabilities are fixed for all blocks and block pairs.

Vertex Degrees Under SBM. For an arbitrary vertex i belonging to a block k containing n_k nodes, the expected degree is

$$\mathbf{E}(d_i) = \underbrace{P_k \times (n_k - 1)}_{\text{expected intra-block edges}} + \underbrace{\sum_{m \neq k} P_{km}(n_k \times n_m)}_{\text{expected inter-block edges}} . \tag{1}$$

Here, P_k denotes the probability that two arbitrary nodes in block k are connected by an edge. Similarly, P_{km} denotes the probability that an arbitrary node in block k is connected to an arbitrary node in block m. (With, $P_k > P_{km}$, typically.)

Vertex Degree Under PPM. For the remainder of this document, we will use the following variable naming conventions to describe the PPM graphs:

- $d_i = d_i^{in} + d_i^{out}$ is the degree of a node i.
- d_i is the sum of connections to nodes within the same block (d_i^{in}) and to nodes in other blocks (d_i^{out}).
- P_{in}, P_{out} are the within/between-block edge probabilities,
- N is the total number of vertices,
- n is the number of vertices within any given block and, finally,
- $K = N/n$ is the total number of blocks or partitions (under the PPM, all blocks have the same number of nodes).

$$\begin{aligned}\mathbf{E}(d_i) &= \underbrace{P_{in} \times (n-1)}_{\mathbf{E}(d_i^{in})} + \underbrace{P_{out} \times (N-n)}_{\mathbf{E}(d_i^{out})} \\ &= \underbrace{P_{in} \times (n-1)}_{\mathbf{E}(d_i^{in})} + \underbrace{P_{out} \times N\left(1 - \frac{1}{K}\right)}_{\mathbf{E}(d_i^{out})} \end{aligned} \tag{2}$$

In Eqs. 1 and 2, we clearly see how the (expected) degree of a vertex can be partitioned. Degree can be understood as the cardinality of the union of a set of connections to nodes within the same block and to nodes on the remaining graph. We use this partition to examine the sensitivity of graph spectra to overall degree, but also specific generative model parameters. Specifically, we examine the relationships between spectra and graph size, block size and inter-block edge probability. The analysis of these relationships is a useful tool in understanding the applicability and limitations of spectral graph techniques. While we use the PPM for its transparency, our conclusions reveal critical information about the relationship between graph structures and spectra. This relationship transcends generative models, because they study the links between graph structure (esp. degree) and spectra.

4 Empirical Tests

To examine the sensitivity of graph spectra to graph size, block size and to noise from increased inter-block edge probability, we conduct four sets of experiments using the Planted Partition Model. In all experiments, we begin with a graph with small block size, number of blocks or inter-block probability. We then gradually increase these parameters and observe the effect on spectra. We stop our sensitivity tests, when a pattern appears (or disappears, e.g., eigengap) in the spectra. For completeness, we examine the spectra of both the symmetric normalized Laplacian and adjacency matrices. While most studies of graph spectra examine the normalized symmetric Laplacian (e.g., [6,19,24]), the adjacency matrix remains the most basic matrix representation of a graph.

In our experiments, growth in size occurs in two different ways. In the first case, growth occurs in the sizes of blocks, while the number of blocks remains constant. We start with a graph containing 50 blocks of five nodes each and then expand to graphs with 50 blocks of 50 and 500 nodes. In the second case, growth occurs in the numbers of blocks, while sizes of blocks remains constant. We begin with a graph of five blocks of 50 nodes and expand to 50 blocks and 500 blocks of 50 nodes. These results are presented in Sect. 4.1. We also isolate the effect of block size, within a fixed size graph. The goal of these numerical experiments is to identify the (in)ability of spectra to detect the presence of densely connected blocks of varying sizes, within a sparser graph with identical characteristics (edge probability, number of blocks and overall size). These results are presented in Sect. 4.2. In all the above-mentioned experiments, intra-/inter–block edge probabilities are held constant ($P_{in} = 0.9, P_{out} = 0.1$).

Our last batch of experiments, is an examination sensitivity to edge probability. In these experiments, we generate PPM graphs with intra-block edge probability of 0.9 but with varying inter-block edge probabilities. Here again, we isolate the effect of block size. We begin by generating graphs with a relatively large block size ($n = 500$) and relatively small number of blocks ($K = 50$). We then repeat the same experiments with graphs containing a relatively large number ($K = 500$) of relatively small ($n = 50$) blocks.

4.1 Sensitivity to Graph Size

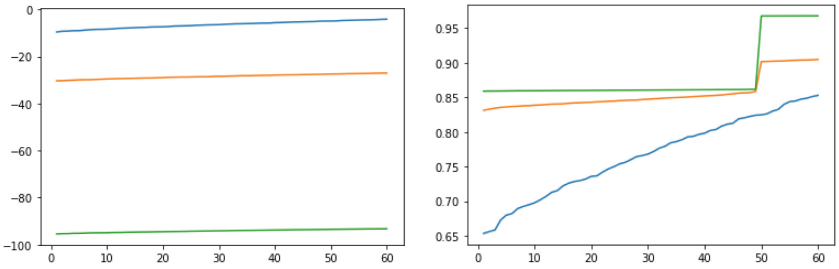
In this first set of experiments, we vary graph size by increasing block size (n), while keeping the number of blocks constant ($K = 50$). We compute the eigenvalues for the adjacency and normalized Laplacian matrices for graphs with:

- $n = 5 \Rightarrow N = 5 \times 50 = 250$,
- $n = 50 \Rightarrow N = 50 \times 50 = 2,500$,
- $n = 500 \Rightarrow N = 500 \times 50 = 25,000$,
- $P_{in} = 0.9, P_{out} = 0.1$.

Results are shown in Fig. 1. The blue curve shows the $(N - 1)$ non-zero eigenvalues, sorted in ascending order, for the graph with 250 nodes. The orange curve shows the same, for the graph with $2,500$ nodes. Finally, the green curve shows the eigenvalues of the graph with $25,000$ nodes. In order to focus on the theoretical location of the eigengap, we adjust the x-axis accordingly.

In a separate set of experiments, we generate graphs with the same characteristics as those in Fig. 1. We then record the range of eigenvalues and the eigengap of the normalized Laplacian as the graph grows in size. Results are reported in Table 1.

In our second set of experiments, we vary graph size by increasing the number of blocks (K), while keeping the block size constant ($n = 50$). It has been argued that this growth model is more realistic and consistent with real world networks [17,23]. We compute the eigenvalues for the adjacency and normalized Laplacian matrices for graphs with:



(a) Adj mat (b) Lap mat

Fig. 1. Varying N, number of blocks is constant ($K = 50$) (blue 250 nodes, orange 2,500 nodes, green 25,000 nodes)

Table 1. Eigenvalue range, number of blocks is constant ($K = 50$)

	$N = 250$	$N = 2500$	$N = 25K$
Min	0.66	0.83	0.86
Max	1.34	1.11	1.03
Eigengap $(\lambda_{i+1}/\lambda_i)$	1.00	1.05	1.12

- $K = 5 \Rightarrow N = 5 \times 50 = 250$,
- $K = 50 \Rightarrow N = 50 \times 50 = 2,500$,
- $K = 500 \Rightarrow N = 500 \times 50 = 25,000$,
- $P_{in} = 0.9, P_{out} = 0.1$.

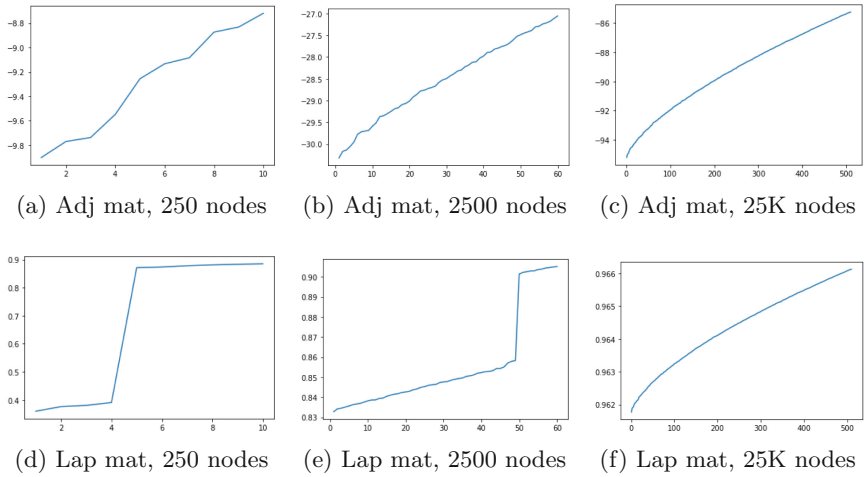
Results are shown in Fig. 2. Here too, in order to focus on the theoretical location of the eigengap, we adjust the x-axis accordingly.

Once more, we also generate a new set of graphs with the same characteristics as those in Fig. 2. We record the range of eigenvalues and the eigengap of the normalized Laplacian as the graph grows in size. Results are reported in Table 2.

Table 2. Eigenvalue range, block size is constant ($n = 50$)

	$N = 250$	$N = 2500$	$N = 25K$
Min	0.35	0.83	0.96
Max	1.16	1.11	1.04
Eigengap $(\lambda_{i+1}/\lambda_i)$	2.31	1.05	1.00

Results from all four experiments in this section highlight the relationships between Laplacian eigenvalues and graph and block sizes. Our observations are consistent with and extend prior work that has linked vertex degree and spectra [5,27]. In particular, we isolate the effect of increases in graph and block sizes on



(a) Adj mat, 250 nodes (b) Adj mat, 2500 nodes (c) Adj mat, 25K nodes

(d) Lap mat, 250 nodes (e) Lap mat, 2500 nodes (f) Lap mat, 25K nodes

Fig. 2. Varying N, number of blocks varies, block size constant ($n = 50$)

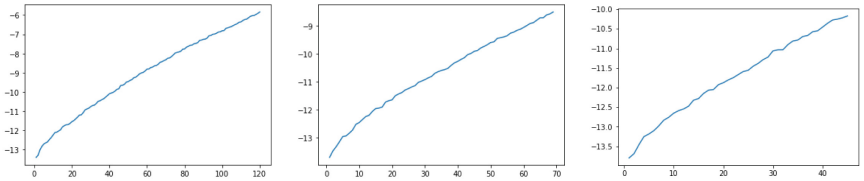
degree and, consequently, eigenvalues, as shown in Eqs. 1 and 2. Indeed, Fig. 1 and Table 1 reveal that while block size may remain a constant proportion of graph size, blocks with a small number of nodes are undetectable. For example, in the first column of Table 1 blocks have only five nodes. Meanwhile, in the third column, blocks have 500 nodes. In the first column, no eigengap is detected, while in the second and third columns, where blocks are larger in absolute terms, a small eigengap is present. Nevertheless, we do note a significant monotonic narrowing of the range of eigenvalues with increases in graph size.

The narrowing of the range of eigenvalues is marked even more in the case of a graph with constant block size, as seen in Fig. 2 and Table 2. In these same experiments, we also observe a vanishing eigengap with increases in graph sizes. Here, it is important to note that these experiments follow a pattern of network growth has been found to be more realistic and consistent with real world networks [17,23]. Clearly, these results highlight the limitations of spectral techniques in the case of large real world networks.

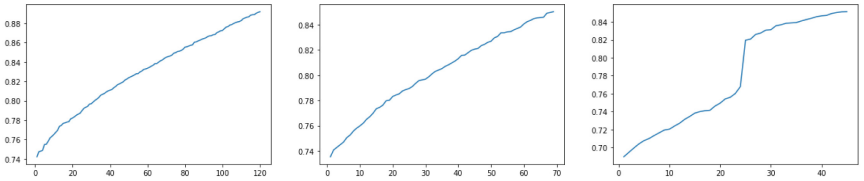
4.2 Sensitivity to Block Size

To further isolate the effect of block size, we keep the number of nodes constant ($N = 500$), but vary block size ($n \in \{5, 10, 20\}$). Once again, we adjust the x-axis to focus on the eigengap. Results are shown in Fig. 3.

Once again, we observe that smaller block sizes lead to increased uniformity in eigenvalues. In fact, the eigengap is non-existent, except in the very last experiment ($n = 20$).



(a) Adj mat, $n = 5$ (b) Adj mat, $n = 10$ (c) Adj mat, $n = 20$



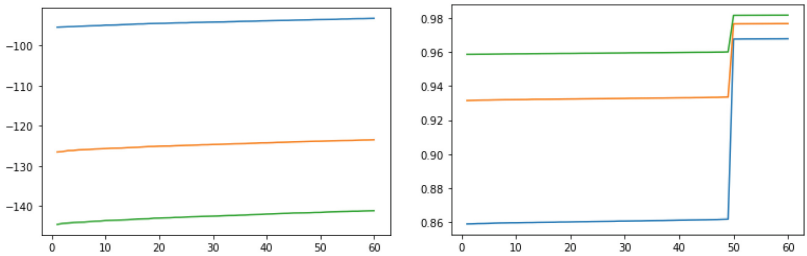
(d) Lap mat, $n = 5$ (e) Lap mat, $n = 10$ (f) Lap mat, $n = 20$

Fig. 3. Sensitivity to block size (n), graph size constant ($N = 500$)

4.3 Sensitivity to Inter-block Edge Probability

In this set of experiments, we keep the intra-block edge probability fixed ($P_{in} = 0.9$). To simulate a noisy network, we vary inter-block edge probability ($P_{out} \in \{0.1, 0.2, 0.3\}$). In the experiments shown in Fig. 4, the graphs contain $K = 50$ blocks of $n = 500$ vertices. Meanwhile, for the ones in Fig. 5, the graphs contain $K = 500$ blocks of $n = 50$ nodes. Arguably, this latter case is more representative of real world networks, which typically have smaller blocks (clusters, communities) [17,23]. In both figures, the blue curve represents the eigenvalues of graph with $P_{out} = 0.1$, the orange curve is for a graph with $P_{out} = 0.2$ and the green curve is for a graph with $P_{out} = 0.3$. Finally, as in Sect. 4.1, we also reproduce the experiments in our figures (Figs. 4 and 5) and record the eigenvalue ranges and eigengap ratios. These results are shown is Tables 3 and 4.

Once again, these experiments highlight the link between degree and spectra. Specifically, these experiments highlight the link between increases in P_{out}

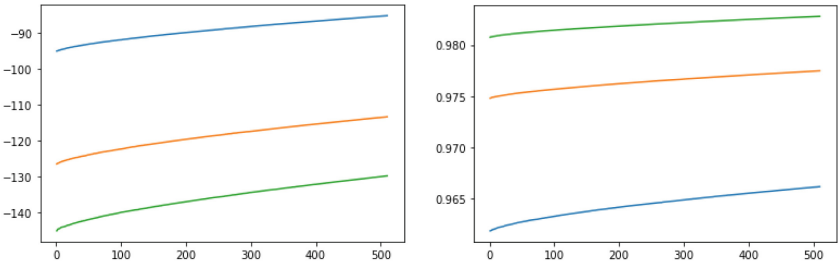


(a) Adj mat (b) Lap mat

Fig. 4. Increasing inter-block edge probability ($N = 25,000, K = 50, n = 500$) (blue $P_{out} = 0.1$, orange $P_{out} = 0.2$, green $P_{out} = 0.3$)

Table 3. Eigenvalue range, with increasing inter-block edge probability ($N = 25,000, K = 50, n = 500$)

	$P_{out} = 0.1$	$P_{out} = 0.2$	$P_{out} = 0.3$
Min	0.86	0.93	0.96
Max	1.03	1.02	1.02
Eigengap (λ_{i+1}/λ_i)	1.12	1.05	1.02



(a) Adj mat (b) Lap mat

Fig. 5. Increasing inter-block edge probability ($N = 25,000, K = 500, n = 50$) (blue $P_{out} = 0.1$, orange $P_{out} = 0.2$, green $P_{out} = 0.3$)

Table 4. Eigenvalue range, with increasing inter-block edge probability ($N = 25,000, K = 500, n = 50$)

	$P_{out} = 0.1$	$P_{out} = 0.2$	$P_{out} = 0.3$
Min	0.96	0.97	0.98
Max	1.04	1.03	1.02
Eigengap (λ_{i+1}/λ_i)	1.00	1.00	1.00

and consequently inter-block degree and spectra. In all experiments, we observe narrowing eigenvalue ranges and vanishing eigengaps.

5 Conclusion and Future Work

In this article, we identify the limitations of spectral graph analysis. We show that eigenvalues are sensitive to noise in connectivity and converge to uniformity as graph size increases. This noise is a function of both block size and inter-block edge probability. While we use the PPM to illustrate these sensitivities, we argue that our conclusions extend to other generative models as well. Indeed, vertex degree, edge probability and block (community) sizes are variables that are present in all graphs with clustered structure, regardless of generative model. Naturally, graphs with power law degree distributions are less sensitive to the variations discussed in this paper. However, we reserve a detailed examination of their spectra for future work.

On the basis of our numerical experiments, we recommend against the use of spectral techniques for large graphs or in cases where blocks (communities) are expected to be small. Our future work will focus on identifying clear thresholds for the applicability of spectral techniques.

Acknowledgments. – The work of P.M. was funded by a MITACS grant (grant# IT33832).

– The authors thank Prof. Valery A. Kalyagin of the Laboratory of Algorithms and Technologies for Networks Analysis in Nizhny Novogorod, Russia, for his comments on the early stages of this work.

References

1. Albert, R., Barabási, A.L.: Statistical mechanics of complex networks. Rev. Mod. Phys. **74**, 47–97 (2002). https://doi.org/10.1103/RevModPhys.74.47
2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**, 509–512 (1999)
3. Broido, A.D., Clauset, A.: Scale-free networks are rare. Nature Commun. **10**(1), 1017 (2019)
4. Bruneau, P., Parisot, O., Otjacques, B.: A heuristic for the automatic parametrization of the spectral clustering algorithm. In: 2014 22nd International Conference on Pattern Recognition, pp. 1313–1318 (2014). https://doi.org/10.1109/ICPR.2014.235
5. Chen, J., Lu, J., Zhan, C., Chen, G.: Laplacian Spectra and Synchronization Processes on Complex Networks, pp. 81–113. Springer US, Boston, MA (2012). https://doi.org/10.1007/978-1-4614-0754-6_4. URL https://doi.org/10.1007/978-1-4614-0754-6_4
6. Chung, F.R.K.: Spectral graph theory. American Mathematical Soc. (1997)
7. Coja-Oghlan, A., Goerdt, A., Lanka, A.: Spectral partitioning of random graphs with given expected degrees. In: Navarro, G., Bertossi, L., Kohayakawa, Y. (eds.) Fourth IFIP International Conference on Theoretical Computer Science- TCS 2006, pp. 271–282. Springer, US, Boston, MA (2006)
8. Condon, A., Karp, R.: Algorithms for graph partitioning on the planted partition model. Random Struct. Algorithms **18**(2), 116–140 (2001). https://doi.org/10.1002/1098-2418(200103)18:2⟨116::AID-RSA1001⟩3.0.CO;2-2
9. Erdös, P., Rényi, A.: On random graphs I. Publicationes Mathematicae Debrecen **6**, 290–297 (1959)
10. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**, 75–174 (2010). https://doi.org/10.1016/j.physrep.2009.11.002
11. Fortunato, S., Hric, D.: Community detection in networks: A user guide. arXiv (2016)
12. Gan, L., Wan, X., Ma, Y., Lev, B.: Efficiency evaluation for urban industrial metabolism through the methodologies of emergy analysis and dynamic network stochastic block model. Sustainable Cities and Society, p. 104396 (2023)
13. Gilbert, E.: Random graphs. Ann. Math. Statist. **30**(4), 1141–1144 (1959). https://doi.org/10.1214/aoms/1177706098.
14. Hagberg, A., Schult, D., Swart, P.: Exploring Network Structure, Dynamics, and Function using NetworkX. In: G. Varoquaux, T. Vaught, J. Millman (eds.) Proceedings of the 7th Python in Science Conference, pp. 11–15. Pasadena, CA USA (2008)

15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. arXiv **78**(4), 046110 (2008). https://doi.org/10.1103/PhysRevE.78.046110
16. Lee, C., Wilkinson, D.J.: A review of stochastic block models and extensions for graph clustering. Appl. Netw. Sci. **4**(1), 1–50 (2019)
17. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters. Internet Mathematics **6**(1), 29–123 (2009). https://doi.org/10.1080/15427951.2009.10129177
18. Lutzeyer, J.F., Walden, A.T.: Comparing Graph Spectra of Adjacency and Laplacian Matrices. arXiv e-prints arXiv:1712.03769 (2017). https://doi.org/10.48550/arXiv.1712.03769
19. von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**, 395–416 (2007)
20. Newman, M.E.J., Strogatz, S., Watts, D.J.: Random graphs with arbitrary degree distributions and their applications. Phys. Rev. E **64**, 026,118 (2001). https://doi.org/10.1103/PhysRevE.64.026118.
21. Priebe, C.E., et al.: On a two-truths phenomenon in spectral graph clustering. Proc. Natl. Acad. Sci. **116**(13), 5995–6000 (2019)
22. Rao Nadakuditi, R., Newman, M.E.J.: Graph spectra and the detectability of community structure in networks. arXiv e-prints arXiv:1205.1813 (2012). https://doi.org/10.48550/arXiv.1205.1813
23. Rohe, K., Chatterjee, S., Yu, B.: Spectral clustering and the high-dimensional stochastic blockmodel. Ann. Stat. **39**(4), 1878–1915 (2011). https://doi.org/10.1214/11-AOS887.
24. Spielman, D.A.: Spectral graph theory and its applications. In: 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07), pp. 29–38 (2007). https://doi.org/10.1109/FOCS.2007.56
25. documentaton page (author unknown), O.: Planted partition model. https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.planted_partition_graph.html
26. documentaton page (author unknown), O.: Stochastic block model. https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.stochastic_block_model.html
27. Zhan, C., Chen, G., Yeung, L.F.: On the distributions of Laplacian eigenvalues versus node degrees in complex networks. Physica A: Statistical Mechanics and its Applications **389**(8), 1779–1788 (2010). https://doi.org/10.1016/j.physa.2009.12.005. URL https://www.sciencedirect.com/science/article/pii/S0378437109010012

FakEDAMR: Fake News Detection Using Abstract Meaning Representation Network

Shubham Gupta[1], Narendra Yadav[1(✉)], Suman Kundu[1(✉)], and Sainathreddy Sankepally[2(✉)]

[1] IIT Jodhpur, Jodhpur, India
{gupta.37,yadav.42,suman}@iitj.ac.in
[2] IIIT Naya-Raipur, Naya Raipur, India
sankepallysainathreddy@gmail.com

Abstract. Given the rising prevalence of disinformation and fake news online, the detection of fake news in social media posts has become an essential task in the field of social network analysis and NLP. In this paper, we propose a fake detection model named, FakEDAMR that encodes textual content using the Abstract Meaning Representation (AMR) graph, a semantic representation of natural language that captures the underlying meaning of a sentence. The graphical representation of textual content holds longer relation dependency in very few distances. A new fake news dataset, FauxNSA, has been created using tweets from the Twitter platform related to 'Nupur Sharma' and 'Agniveer' political controversy. We embed each sentence of the tweet using an AMR graph and then use this in combination with textual features to classify fake news. Experimental results on publicly and proposed datasets with two different sets show that adding AMR graph features improves F1-score and accuracy significantly. (Code and Dataset: https://github.com/shubhamgpt007/FakedAMR)

Keywords: Fake News Detection · Machine Learning · AMR Network · Network Embedding

1 Introduction

Social media became essential for communication and information sharing. However, news shared over social media platforms lacks cross-referencing, allowing the spread of misinformation. Interestingly, it appears that the rate at which fake news is shared on Twitter exceeds that of genuine news [18]. Figure 1 presents some examples of fake news that spread through various media platforms, including Twitter. Many ML/DL methods were proposed to identify fake news from social media [7]. These existing methods focused on syntactic features and did not investigate how semantic features of news content affect ML models. However, complex semantic features are seen to improve the performance of different

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 308–319, 2024.
https://doi.org/10.1007/978-3-031-53468-3_26

NLP tasks such as event detection [6], abstractive summarization [14], and question answering [13] in machine learning. Considering this one may ask "Does incorporating complex semantic features of sentences enhance the performance of fake news detection models too?"

Fig. 1. Examples of false information related to topics 'Nupur Sharma' and 'Agniveer' controversy showcased in the images (Courtesy: Boomlive). The images depict various misleading claims, including a) Russia, Netherlands, France and 34 other countries are supporting India and Nupur Sharma. b) Nupur Sharma is arrested and in jail. c) Oppressors are damaging the railway line in the protest of Aginveer scheme.

The present study proposed a fake news detection model, FakEDAMR, that classifies tweets as genuine and fake information, by introducing graph-based semantic features with syntactic and lexical features of the sentences. The main contribution of our work is to use deep semantic representation from the features of the Abstract Meaning Representation (AMR) graph. AMR helps to better extract the relationships between entities far apart in the text with minimum cost. This approach reduces the emphasis on syntactic features and collapses certain elements of word category, such as verbs and nouns, as well as word order and morphological variations. To the best of our knowledge, this research rigorously investigates the semantic features of Abstract Meaning Representation (AMR) graphs in comparison to other studies focused on identifying fake news. We curated a fake news dataset, namely, FauxNSA, related to the well-known controversies *Nupur Sharma* and *Agniveer* in India. Tweets with a list of curated hastags (Table 1) on said topics are collected from the Twitter platform, in two different languages - Hindi and English. We extracted AMR graphs from each text document by using STOG model [24]. We encoded AMR graphs using graph embedding and combined them with the syntactic features of the text used in state-of-the-art model [22]. Finally, the resulting embedding vector, which includes both semantic and syntactic features, is fed into a deep-learning model to predict the probability of fake and real. We have experimented with our model on two publicly available datasets (Covid19-FND[19], KFN[12]) and FauxNSA. Our experiments demonstrated an improvement in accuracy of 2–3% over all the datasets when the AMR graph features were included with existing textual features in the model.

The rest of the paper is organized as follows: Sect. 2 reports the related work. Section 3 and 4 describe the working methodology and experimental setup.

Section 5 reports the results with comparative analysis. Ablation study is presented in Sect. 6. Finally, Sect. 7 concludes the research outcome.

2 Related Work

Fake news detection has been extensively studied recently using Natural Language Processing. Oshikawa et al. [16] clarify the distinction between detecting fake news and related concepts, including rumor detection, and provide an overview of current data sets, features, and models. As mentioned in the introduction, Castillo et al. [3] created a set of 68 features in the identification of false information. They used a propagation tree over the feature set to identify whether the news is false or not. An extension to the lexical-based analysis model is used in [15] by incorporating speaker profile details into an attention-based long short-term memory (LSTM) model. Zervopoulos et al. [23] created a set of 37 handcrafted features that includes morphological (e.g., part of speech), vocabulary (e.g., type-to-token ratio), semantic (e.g., text and emoji sentiment), and lexical features (e.g., number of pronouns) to predict the false news using traditional ML algorithms. Further, in 2022 [22], they have extended the research to run different feature sets with complex deep learning models.

AMR is a graph-based representation of natural language that accurately captures the complex semantics of a sentence in a way that is both language-independent and computationally tractable. A growing number of researchers are investigating how to use the information stored in the AMR graphs and its representations to assist in the resolution of other NLP problems. AMR has been successfully applied to more advanced semantic tasks such as entity linking [17], question answering [13], and machine translation [10]. Garg et al. [5] were the first to employ AMR representation for extracting interactions from biomedical text, utilizing graph kernel methods to determine if a given AMR subgraph expresses an interaction. Aguilar et al. [1] and Huang et al. [8] had conducted research and indicated that the semantic structures of sentences, such as AMR introduced in [2], encompass extensive and varied semantic and structural information concerning events. AMR graphs in Fake News detection has been relatively unexplored, however, considering its capability to determine the trigger words by extracting complex semantic information, AMR graphs have the potential to improve the efficiency of existing fake news detection methodologies. Recently, Zhang et al. [25] extracted fact queries based on AMR to verify the factual information in multimodality. Some evidence-based GNN fake news detection models [4,9,21] are also proposed for misinformation detection.

3 Methodology

The methodology in this study is divided into two parts: 1) curation of the proposed data set FauxNSA, and 2) fake news detection model FakEDAMR. Figure 2 shows the methodology, and the description of each step is provided in the following sections.

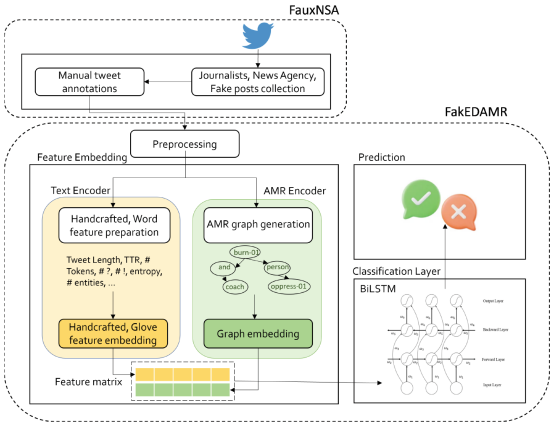


Fig. 2. Structural outline of proposed fake news detection methodology.

Table 1. List of curated hashtags used to scrap tweets from Twitter platform.

Category	Hashtags
Nupur Sharma Controversy	#NupurSharmaControversy, #Jamamasjid, #Nupur_Sharma, #NupurSharma, #HinduRashtra, #HindusUnderAttack, #SarTanSeJuda, #KanahiyaLal, #NupurSharmaBJP, #IsupportNupurSharma
Agniveer Controversy	#AgnipathRecruitmentScheme, #Agnipath, #Agniveer, #AgnipathProtests

FauxNSA: Fake News Dataset on Nupur Sharma and Agniveer Dispute

Fake news dataset The data set was gathered from the Twitter platform between May and September 2022 using the Twitter Academic API's full-archive search over the political controversy 'Nupur Sharama' and 'Agniveer'. This controversy holds the data related to religion, political, and terrorist issues. The methodology to collect the tweets can be broken down as follows. First, a list of curated hashtags mentioned in Table 1 related to the topics 'Nupur Sharama' and 'Agniveer' controversy is manually constructed. Tweets were captured through Twitter API consisting of at least one hashtag from the list. Total 31,889 tweets including 31 features such as account information (display name, # of followers), tweet information (text, hashtag, URLs), and network information (quote, like, reply) are collected. We used popular fact-checking websites such as BoomLive[1], NewsChecker[2], AltNews[3], etc. to annotate the data for fake news which were then manually searched over various social media platforms and carefully annotated by two human annotators. We have also collected tweets from the 42 verified fact-checker Twitter accounts (PIBFackCheck, ABCFactCheck, etc.) from Twitter platform. Subsequently, this comprehensive approach ensures the

[1] https://www.boomlive.in/.
[2] https://newschecker.in/.
[3] https://www.altnews.in/.

reliability and credibility of the data used in our study. After performing all the filtering process, we have 4632 tweets that are fake news.

Real news dataset In the collection of real news, we have used the same approach discussed in [22]. That is, we have considered journalists and news agency as trustworthy source and collected tweets related to topics *Nupur Sharama* and *Agniveer* controversy. Overall, the account of 34 news agencies[4] and 82 account of journalists[5] with a global outreach are identified and gathered. Total 4657 tweets are collected and verified with the human annotators to make data set for the real news. Figure 3 shows the word cloud of the collected true and fake data for Hindi and English languages. We can observe from that all the keywords are related to 'Nupur Sharama' and 'Agniveer' controversy only. After gathering tweets from news agencies, journalists, and fake news sources, a comparison of their characteristics was made to determine their similarities. Specifically, the average number of hashtags per tweet was found to be 2.95 in the fake news data set, 3.12 for tweets posted by journalists, and 2.7 for those posted by news agencies. Additionally, the mean number of URLs per tweet was found to be 0.42 in the fake news data set, 0.55 for tweets posted by journalists, and 1.18 for those posted by news agencies. The statistics show that collected data from all the sources shows almost similar properties. Finally, the data set consists of 9289 tweets with 4632 fake and 4657 real tweets.

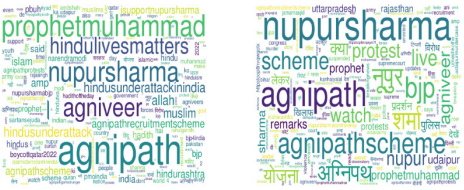


Fig. 3. Frequency word clouds of a) fake and b) true tweets collected from Twitter over 'Nupur Sharama' and 'Agniveer' controversy.

3.1 FakEDAMR: Fake nEws Detection Using Abstract Meaning Representation

Proposed model, FakEDAMR, takes preprocessed text as input and predicts whether the text document is real or fake. FakEDAMR comprises three primary components:

Text Encoder. Research in the field of Natural Language Processing (NLP) has long focused on effectively representing sequential data. In line with previous studies, we have employed two different approaches to encode the sequence of

[4] https://www.similarweb.com/top-websites/india/news-and-media/.
[5] https://www.listofpopular.com/tv/top-journalist-of-india/.

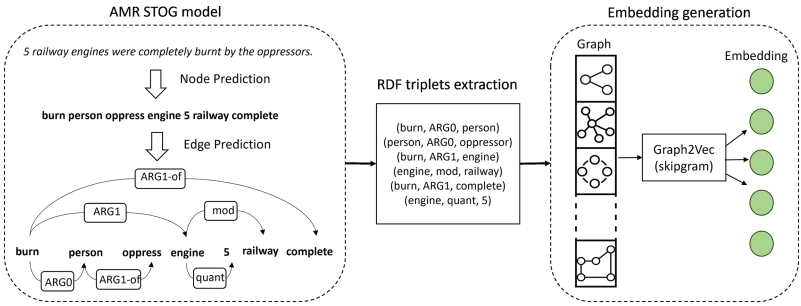


Fig. 4. Process flow of AMR Network Encoder: Text-to-Graph Conversion, RDF Triplet Extraction, and Graph Embedding Generation.

tokens. The first approach involves the use of handcrafted features, consisting of 37 specific features outlined in [22]. The second approach utilizes GloVe embedding [20], which is pre-trained using a Twitter-based corpus comprising 27 billion tokens. This embedding maps each word to a d-dimensional vector. Mathematically, the cost function of the GloVe embedding for a word in a word sequence (represented as $w =< w_1, ..., w_k >$) can be expressed as follows:

$$J = \sum_{p,q=1}^{V} f\left(N_{pq}\right)\left(w_p^T \tilde{w}_q + b_p + \tilde{b}_q - \log N_{pq}\right)^2 \tag{1}$$

Here, $f\left(N_{pq}\right)$ is a weighting function, w_p^T is a context word vector and $\tilde{w}_q$ is out of context word vector and b_p, $\tilde{b}_q$ are bias terms. In Eq. 1, bias terms are also learned along with weight vector. Finally, we get the text embedding vector $t = ([t_i]_{i=1}^{i=d};\ t_i \in \mathbb{R}^{1\times m})$, where d is the fixed dimension and m is the maximum number of tokens.

AMR Encoder. It is important to understand the complex semantics of the sentences which is not explored much in previous work on fake news. We use AMR graphs for the same. AMR is a sembanking language that utilizes a rooted, directed, labeled, and mostly acyclic graph structure to capture the complete meanings of sentences. It employs multi-layer linguistic analysis techniques, including PropBank frames, non-core semantic roles, co-reference, named entity annotation, modality, and negation, to express the semantic structure of a sentence. AMR graph is composed of nodes representing semantic concepts such as entities, events, and attributes, and edges represent the relationships between those concepts, labeled with semantic roles such as agent, patient, location, time, and manner. Our approach involves several steps to generate an AMR graph from each text document (Fig. 4) and extraction of RDF (Resource Description Framework) triplets from the generated AMR graph. These triplets are represented in the form of *(subject, relation, object)*. Subsequently, we create a final graph using the extracted edges from the RDF triplets. Finally, we fed this converted graph into the Graph2Vec model to obtain the AMR graph embedding.

The AMR graph conversion process of each text in the document utilizes STOG model [24]. STOG model breaks down the sequence-to-graph task into two main components: node prediction and edge prediction.

In node prediction, the model takes an input sequence $w =< w_1, \ldots, w_k >$, where each word w_a is part of the sentence. It sequentially decodes a list of nodes $v =< v_1, \ldots, v_k >$ and assigns their indices $i =< i_1, \ldots, i_k >$ deterministically using the equation:

$$P(\boldsymbol{v}) = \prod_{a=1}^{k} P\left(v_a \mid v_{<a}, i_{<a}, w\right) \tag{2}$$

For edge prediction, given an input sequence w, a list of nodes v, and indices i, the model searches for the highest scoring parse tree y within the space $\mathcal{Y}$ of valid trees over v, while adhering to the constraint of i. A parse tree y represents a collection of directed head-modifier edges, depicted as:

$$y = \{(v_a, v_b) | 1 \leq a, b \leq k\} \tag{3}$$

To efficiently find the highest scoring parse tree (i.e., maximum spanning arborescence), the model utilizes a scoring mechanism used in [11].

$$\text{parse}(\boldsymbol{v}) = \underset{y \in \mathcal{Y}(\boldsymbol{v})}{\arg\max} \sum_{(v_a, v_b) \in y} \text{score}\left(v_a, v_b\right) \tag{4}$$

After obtaining the parse tree, the model proceeds with a merging operation to reconstruct the standard Abstract Meaning Representation (AMR) graph by combining nodes that share identical indices. Once we have the AMR tree denoted as y, we extract the RDF triplets from it. These triplets are represented as $t = \{(v_1, r_1, v_2), \ldots, (v_{k-1}, r_j, v_k)\}$. Each triplet consists of a subject v_a, a concept r_k, and an object v_b.

Using the extracted RDF triplets, we construct the final graph denoted as $g = (v, e, r)$. Here, v represents the set of vertices, specifically $v = \{v_1, \ldots, v_k\}$, r corresponds to the set of concepts obtained from the RDF triplets, i.e., $r = \{r_1, \ldots, r_j\}$. Lastly, e represents the set of edges in the graph, which is defined as $e = \{(v_a, r_j, v_b) | \exists\, v_a, v_b \in v \,\text{and}\, r_j \in r\}$. In other words, the edges in e connect the vertices v_a and v_b using the relation r_j. This process of extracting RDF triplets and constructing the final graph enables the representation and analysis of the AMR graph, capturing the semantic relationships between entities and facilitating further processing and interpretation.

Afterward, a list of graph G, where each graph $g \in G$ represents one text, is passed as input to the Graph2Vec model, specifically the skip-gram model, to obtain the final embedding. The Graph2Vec model processes the AMR graph and generates embeddings by considering the graph structure and the relationships between its elements. The resulting embedding is obtained from the last hidden layer of the model, capturing the learned representation of the AMR graph in a vector form. Finally, we get the graph embedding vector $u = ([u_i]_{i=1}^{i=d}; u_i \in \mathbb{R}^{n \times 1})$, where d is the fixed dimension and n is number of sentence in the document.

Classification Layer. After getting the text embedding $t_{(m\times d)}$ and graph embedding $u_{(n\times d)}$, we get final embedding x by Eq. 5, where | represents concatenation:

$$x = (t_{(m\times d)} \mid u_{(n\times d)})_{(m+n)\times d} \tag{5}$$

Finally, BiLSTM model is used as classification layer to identify tweets in fake or real news using the prepared feature embedding x.

$$f_t = \text{sigmoid}\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \tag{6}$$

$$i_t = \text{sigmoid}\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \tag{7}$$

$$c_t = c_{t-1} \odot f_t + i_t \odot \tanh\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right) \tag{8}$$

$$o_t = \text{sigmoid}\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right),\ h_t = o_t \odot \tanh\left(c_t\right) \tag{9}$$

Where $x_t \in \mathbb{R}^{n\times d}$ is the input vector, $W \in \mathbb{R}^{l\times n}, b \in \mathbb{R}^v$ and the superscripts n and l depict the dimension of the input vector and the number of words in the dataset or vocabulary at any time t, respectively. For an input vector x_t, h_{t-1} and c_{t-1} are previously hidden and cell state, whereas the current hidden and cell state h_t and c_t. The above output represents the LSTM network. Finally, output of the BiLSTM can be summarized by concatenating the forward and backward state as $h_t = \left[\overrightarrow{h}_t, \overleftarrow{h}_t\right]$. At the output layer, it employed binary cross-entropy as the loss function to identify probability of true label $p(y_i)$ for real/fake classification.

4 Experimental Setup

Dataset: Other than our dataset, we have also used two publicly available datasets Covid-FND [19] and KFN [12] for our experiments. Covid-FND dataset consists social media posts and articles related to COVID-19 and KFN dataset includes 20,387 news items which spans the fields of politics, commerce, and technology, contains an evenly distributed mix of real and fake news pieces. Statistics of the data used in our experiments is given in the Table 2.

Table 2. Distribution of data for: a) Covid19-FND, b) KFN, and c) FauxNSA (ours)

Dataset	Covid19-FND	KFN	FauxNSA
# Real	5100	10387	4657
# Fake	5600	10413	4632
# Total	**10700**	**20800**	**9289**

Implementation Details: We have developed and tested our code in Keras (Python library). We partitioned each data set into training, validation, and testing sets, following a 70:20:10 split. This approach maintains the proportional representation of classes within both the train, validation, and test sets, enabling a robust evaluation of the model performance across various data sets. We have

used basic preprocessing, like removing URLs, stopwords, etc., on each text document of the data set. We have incorporated AMR graph features on the feature sets proposed by [22]. They used two feature sets: Feature-set 1 adopts a feature engineering approach, where the chosen features are hand-crafted, including various categories such as morphological, vocabulary, and lexical features. Feature-set 2 employs tokenization of each tweet's text and conversion into word embedding. GloVe embedding [20], pre-trained with a Twitter-based corpus of 27 billion tokens, is used to map each word to a 100-dimensional vector. Despite each word being mapped to a fixed-size vector, tweet length still varies; to address this issue, post-padding (i.e., padding at the end of a tweet) is used to match the longest tweet (approximately 100 tokens). Therefore, a tweet in Feature-set 2 is presented as a 100×100 matrix. Although the size of Graph2Vec can vary based on the length of the AMR graph, we have fixed the dimension to 100, considering the length of the tweet is fixed in the Twitter platform. We evaluated Feature-set 1 on Naive Bayes, SVM, C4.5, random forests, and Feature-set 2 on CNN, C-LSTM, and BiLSTM. For the purpose of training each model on the data sets, we carried out three distinct trials with various seed values. The performance metric was then generated using the test data set findings, taking into account the best-performing trial. Four performance metrics, namely, Precision, Recall, F1-score, and Accuracy are considered for comparative study. Model configuration, such as the number of hyper-parameters and number of layers used in the model, is kept the same as in the research [22].

Table 3. Comparison of the performance of different models on Feature-sets (FS) 1 and 2 for datasets Covid19-FND, KFN, and FauxNSA (ours).

Model	Feature-set	Covid19-FND				KFN				FauxNSA			
		Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
Naive Bayes	FS1	76.06	67.81	71.67	71.55	67.25	79.90	73.03	69.42	82.19	80.85	81.51	80.11
	FS1+AMR	78.83	66.93	72.39	72.58	67.46	79.59	73.10	70.13	84.41	82.06	83.21	80.81
SVM	FS1	82.05	81.47	81.76	80.71	85.91	83.80	84.84	83.71	84.58	82.87	83.71	84.35
	FS1+AMR	81.70	82.39	82.04	82.24	85.66	83.84	84.73	84.65	83.71	83.63	83.67	85.31
C4.5	FS1	81.94	79.57	80.74	79.85	80.81	79.56	80.18	80.28	86.08	83.16	84.59	83.32
	FS1+AMR	82.70	80.04	81.34	80.03	81.42	80.87	81.14	81.67	87.64	83.47	85.50	86.04
Random Forest	FS1	86.01	90.02	87.96	88.26	88.86	84.89	86.83	86.92	87.37	84.05	85.67	86.35
	FS1+AMR	**86.25**	**91.28**	**88.69**	**89.48**	**88.90**	**85.12**	**86.70**	**87.09**	**87.72**	**84.21**	**85.92**	**88.90**
CNN	FS2	91.16	91.30	91.20	91.21	91.74	91.27	91.50	91.52	89.25	84.35	86.34	89.64
	FS2+AMR	92.65	92.75	92.69	92.71	92.42	92.18	92.29	92.11	90.14	89.99	90.06	92.10
C-LSTM	FS2	91.49	91.46	91.47	91.51	91.58	91.58	91.57	91.57	91.61	86.13	88.34	91.11
	FS2+AMR	92.87	92.93	92.90	92.95	93.38	93.28	93.23	93.24	91.85	88.68	90.23	91.89
BiLSTM	FS2	91.55	91.71	91.54	91.55	92.44	92.40	92.41	92.36	90.60	86.83	88.45	91.12
	FS2+AMR	**93.43**	**93.08**	**93.20**	**93.26**	**93.55**	**93.54**	**93.52**	**93.52**	**92.25**	**91.67**	**91.96**	**93.96**

5 Results

We evaluated the performance of AMR with different feature sets on different ML/DL algorithms. Table 3 presents the comparison results of different models.

It is evident that incorporating AMR semantic features into the feature sets significantly improves the performance of the models. Among the models evaluated using Feature-set 1, Random Forest with AMR-encoded feature sets achieves the highest accuracy of 88.90% and an F1-score of 85.92% on the FauxNSA (proposed) dataset. Furthermore, it also achieves the highest accuracy of 89.48% and 87.09%, along with F1-scores of 88.69% and 86.70%, on the publicly available datasets Covid19-FND and KFN, respectively.

BiLSTM with AMR-encoded features outperforms other models in the case of Feature-set 2. The model achieved an accuracy of 93.96% and an F1-score of 91.96% on our data set. Similar performance is observed on the other two publicly available data sets as well, where the accuracy and F1-scores of 93.26% and 93.20% on Covid19-FND, and 93.52% and 93.52% on KFN, respectively, are achieved.

6 Ablation Study: Effect of AMR Graph Features

We conducted an investigation to understand why AMR features enhance the accuracy of the model. To provide evidence for our hypothesis, we visualized the features of the concatenation layer in the model, which merges the AMR features with textual features. In our experiment, we randomly selected three samples each from three datasets that demonstrated improved prediction accuracy when using both textual and AMR features compared to using only textual features in the model. Figure 5 clearly illustrates the impact of AMR features on the model's performance. It shows that the AMR creates a distinct and decisive boundary which helps the model to understand and create a boundary between target classes.

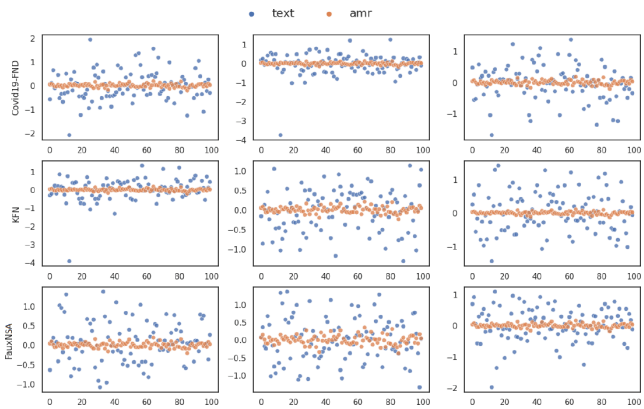


Fig. 5. Comparative analysis of AMR and text features in three datasets: Covid19-FND, KFN, and FauxNSA. Graph is plotted for three correct predicted samples by model where x-axis represents the feature index and y-axis represents its corresponding value.

7 Conclusion

In this paper, we show that detecting fake news requires a more sophisticated understanding of the semantic relationships between trigger words and entities in the text. We demonstrated how Abstract Meaning Representation (AMR) graph improves the fake news detection model and we concluded that semantic features are just as important as linguistic and syntactic features for identifying fake news in posts. In the future, we are exploring way to embed AMR graphs with pre-trained transformer-based models such as Bert, XLM-Roberta, Electra, etc. Also, we are interested in exploring more ways to encode AMR knowledge in order to increase the performance of existing fake news models.

References

1. Aguilar, J., Beller, C., McNamee, P., Van Durme, B., Strassel, S., Song, Z., Ellis, J.: A comparison of the events and relations across ACE, ERE, TAC-KBP, and FrameNet annotation standards. In: Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation, pp. 45–53. Association for Computational Linguistics, Baltimore (2014)
2. Banarescu, L., et al.: Abstract meaning representation for sembanking. In: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, Sofia, Bulgaria, pp. 178–186 (2013)
3. Castillo, C., Mendoza, M., Poblete, B.: Predicting information credibility in time-sensitive social media. Internet Res. Electron. Network. Appl. Policy **23**, 560–588 (2013)
4. Dun, Y., Tu, K., Chen, C., Hou, C., Yuan, X.: Kan: knowledge-aware attention network for fake news detection. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 1, pp. 81–89 (2021)
5. Garg, S., Galstyan, A., Hermjakob, U., Marcu, D.: Extracting biomolecular interactions using semantic parsing of biomedical text. In: AAAI, AAAI 2016, pp. 2718–2726. AAAI Press (2016)
6. Gupta, S., Kundu, S.: Interaction graph, topical communities, and efficient local event detection from social streams. Expert Syst. Appl. **232**, 120890 (2023)
7. Hu, L., Wei, S., Zhao, Z., Wu, B.: Deep learning for fake news detection: a comprehensive survey. AI Open **3**, 133–155 (2022)
8. Huang, L., Cassidy, T., Feng, X., Ji, H., Voss, C.R., Han, J., Sil, A.: Liberal event extraction and event schema induction. In: ACL, vol. 1: Long Papers, pp. 258–268. ACL, Berlin (2016)
9. Jin, Y., et al.: Towards fine-grained reasoning for fake news detection. In: AAAI, vol. 36, pp. 5746–5754 (2022)
10. Jones, B., Andreas, J., Bauer, D., Hermann, K.M., Knight, K.: Semantics-based machine translation with hyperedge replacement grammars. In: Proceedings of COLING 2012, pp. 1359–1376 (2012)
11. Kiperwasser, E., Goldberg, Y.: Simple and accurate dependency parsing using bidirectional LSTM feature representations. Trans. Assoc. Comput. Linguist. **4**, 313–327 (2016)
12. Lifferth, W.: Fake news (2018). https://kaggle.com/competitions/fake-news

13. Lim, J., Oh, D., Jang, Y., Yang, K., Lim, H.: I know what you asked: graph path learning using AMR for commonsense reasoning. In: ICCL, pp. 2459–2471. International Committee on Computational Linguistics, Barcelona (2020)
14. Liu, F., Flanigan, J., Thomson, S., Sadeh, N., Smith, N.A.: Toward abstractive summarization using semantic representations. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1077–1086. Association for Computational Linguistics, Denver (2015)
15. Long, Y., Lu, Q., Xiang, R., Li, M., Huang, C.R.: Fake news detection through multi-perspective speaker profiles. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing, vol. 2: Short Papers, pp. 252–256. Asian Federation of Natural Language Processing, Taipei (2017)
16. Oshikawa, R., Qian, J., Wang, W.Y.: A survey on natural language processing for fake news detection. CoRR arxiv:1811.00770 (2018)
17. Pan, X., Cassidy, T., Hermjakob, U., Ji, H., Knight, K.: Unsupervised entity linking with abstract meaning representation. In: NAACL: Human Language Technologies, pp. 1130–1139 (2015)
18. Parmelee, J.H., Bichard, S.L.: Politics and the Twitter revolution: how tweets influence the relationship between political leaders and the public. Lexington books (2011)
19. Patwa, P., et al.: Fighting an infodemic: Covid-19 fake news dataset. In: Chakraborty, T., Shu, K., Bernard, H.R., Liu, H., Akhtar, M.S. (eds.) CONSTRAINT 2021., vol. 1402, pp. 21–29. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-73696-5_3
20. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: EMNLP, pp. 1532–1543. Association for Computational Linguistics, Doha (2014)
21. Xu, W., Wu, J., Liu, Q., Wu, S., Wang, L.: Evidence-aware fake news detection with graph neural networks. In: WWW, WWW 2022, pp. 2501-2510. ACM, New York (2022)
22. Zervopoulos, A., Alvanou, A., Bezas, K., Papamichail, A., Maragoudakis, M., Kermanidis, K.: Deep learning for fake news detection on twitter regarding the 2019 Hong Kong protests. Neural Comput. Appl. **34**, 1–14 (2022)
23. Zervopoulos, A., Alvanou, A.G., Bezas, K., Papamichail, A., Maragoudakis, M., Kermanidis, K.: Hong Kong protests: using natural language processing for fake news detection on twitter. In: Maglogiannis, I., Iliadis, L., Pimenidis, E. (eds.) AIAI 2020, vol. 584, pp. 408–419. Springer, Cham (2020)
24. Zhang, S., Ma, X., Duh, K., Van Durme, B.: AMR parsing as sequence-to-graph transduction. In: ACL, pp. 80–94. ACL, Florence (2019)
25. Zhang, Y., Trinh, L., Cao, D., Cui, Z., Liu, Y.: Detecting out-of-context multimodal misinformation with interpretable neural-symbolic model (2023)

Visual Mesh Quality Assessment Using Weighted Network Representation

Mohammed El Hassouni[1(✉)] and Hocine Cherifi[2]

[1] FLSH, Mohammed V University in Rabat, Rabat, Morocco
mohamed.elhassouni@flsh.um5.ac.ma
[2] ICB UMR 6303 CNRS, University of Burgundy, Dijon, France

Abstract. This paper addresses the critical task of evaluating the visual quality of triangular mesh models. We introduce an innovative approach that leverages weighted graphs for this purpose. Motivated by the growing need for accurate quality assessment in various fields, including computer graphics and 3D modeling, our methodology begins by generating saliency maps for each distorted mesh model. These models are subsequently transformed into a network representation, where mesh vertices are nodes and mesh edges are edges in the graph. The determination of vertex weights relies on the salience values. We then extract a wide range of topological properties and compute statistical measures to create a signature vector. To predict the quality score, we rigorously evaluate the performance of three regression algorithms. Experiments span four publicly available databases designed for mesh model quality assessment. Results demonstrate that the proposed approach excels in this task, showcasing remarkable correlations with subjective evaluations. This preliminary analysis paves the way for further research to address potential limitations and explore additional applications of mesh network representation.

Keywords: Weighted graph · topological properties · statistical measures · quality assessment · triangular mesh · saliency

1 Introduction

Recently, the utilization of 3D models has expanded significantly across various application domains, including virtual and mixed reality, computer-aided diagnosis, architecture, and cultural heritage preservation. However, processing these 3D models through operations like simplification and compression introduces the potential for various distortions that can adversely affect the visual quality [1,2]. Addressing this issue, there is a growing demand for the development of robust methods to assess perceived quality.

Traditionally, assessing distortion levels in 3D models has relied on human observers, a time-consuming and resource-intensive endeavor. To streamline this process, objective methods have emerged as a practical solution. These methods involve the implementation of automated metrics that aim to replicate the

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 320–330, 2024.
https://doi.org/10.1007/978-3-031-53468-3_27

judgments of an ideal human observer [3]. These metrics generally fall into three categories: full reference [4–7], reduced reference [8–10], and blind methods [11–14]. Among these, blind methods, which do not rely on reference models, have gained particular significance, especially in real-world applications.

3D meshes are very complex structures consisting of vertices, edges, and faces that collectively shape a 3D model. Selecting an appropriate data structure to represent this extensive volume of data and relationships is paramount. It's worth noting that the effectiveness of any quality assessment method greatly relies on the chosen data structure.

Graphs stand out as remarkably versatile data structures, capable of intuitively representing 3D triangular meshes. The degree of connectivity between a vertex and its neighboring vertices provides valuable insights into the perceptual characteristics of a mesh. Additionally, the geometric structure of a mesh can be effectively described in terms of the network's topological properties. The application of graph representations has yielded considerable success in addressing various challenges in computer vision, including tasks like image segmentation [15–17], classification [18–20] and denoising [1,21].

Nevertheless, it is noteworthy that the literature contains relatively few studies that have delved into assessing 3D mesh quality using graph-based approaches. Lin et al. proposed a novel method that relies on learning the graph spectrum's entropy and the mesh's spatial characteristics, as described in [22]. Similarly, Abouelaziz et al. introduced the concept of convolutional graph networks to estimate mesh quality, as proposed in [23]. These two methods, although distinct in their approaches, share a common limitation in that they separately learn the geometric and perceptual attributes without integrating them into the weighted graph construction process.

In this context, we present a novel approach in this paper to assess the visual quality of 3D meshes. Our proposed approach relies on a weighted graph constructed from the geometric coordinates and the saliency values of the mesh vertices. Subsequently, the graph's characteristics are trained with a machine learning-based regression method to forecast the quality score.

The remainder of this paper is organized as follows. We present in Sect. 2 a description of the proposed method. Section 3 is devoted to the experimental results. Finally, we present some concluding remarks and perspectives in Sect. 4.

2 Proposed Method

A common representation of a 3D mesh object uses polygonal or triangular meshes. In the case of a triangular mesh denoted as $\mathbb{M}$, it can be defined as a triple $\mathbb{M} = (\mathcal{V}, \mathcal{E}, \mathcal{T})$. Here, $\mathcal{V} = v_1, ..., v_k$ represents the set of vertices, $\mathcal{E} = e_{ij}$ represents the set of edges, and $\mathcal{T} = t_1, ..., t_n$ represents the set of triangles. Our approach primarily relies on extracting geometric and perceptual features from fundamental mesh components, including vertices, edges, and triangles. We start by computing the saliency map for each mesh. Next, we transform the 3D mesh into a weighted graph. Then, we determine the topological characteristics

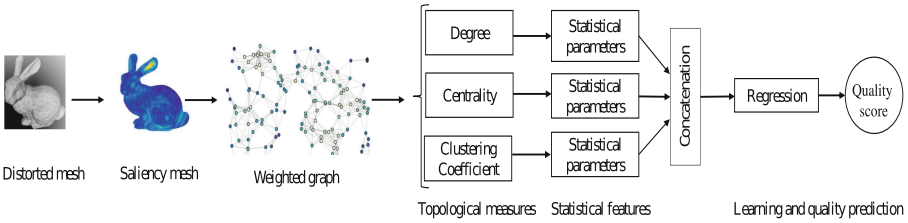


Fig. 1. The overall scheme of the proposed method

of the graph. Based on these findings, we derive signatures for each mesh by computing statistical parameters related to these properties. These resultant signatures serve as inputs for the regression module, enabling us to predict the quality score. In this study, we opted for three regression methods: Random Forest, Support Vector Regression, and Generalized Regression Neural Networks due to their proven utility and appropriateness for learning-based applications.

2.1 3D Mesh Saliency

Saliency is a perceptual concept that describes the attention of our HVS to some regions due to its specificities (curvature, orientation and so on). This work employs mesh visual saliency to compute the graph weights using the method proposed in [24]. The first step to obtain the mesh saliency is to compute the mean curvature at mesh vertices. After that, fine and coarse Gaussians filter the mean curvatures. The saliency is obtained by computing the difference between the filtered mean curvatures within different scales. Finally, a non-linear normalization sum of all the multi-scale saliency maps is applied to compute the final map.

2.2 Weighted Graph Representation of 3D Mesh

Weighted graphs can be derived from mesh descriptions for geometrical domains. An enormous amount of literature exists on techniques for mesh generation and manipulation. These structures find widespread use in computer graphics and computer vision applications. The main concept here relies on graphs representing 3D triangular meshes, enabling us to leverage graph properties to develop a model-based approach to evaluate visual quality without bias.

A graph, denoted as $G = E, V$, representing a triangular mesh, undergoes a straightforward transformation. In this transformation, mesh vertices become graph nodes (V), and mesh edges become graph edges (E).

Additionally, we introduce the concept of weighted graphs. A weighted graph, represented as $G = E, V, \omega$, comprises a set of nodes (V), a set of edges (E), and a weight function ($\omega(e)$) that assigns a positive weight to each edge ($e =$

$(u, v) \in E$). The weight function is defined as follows:

$$\omega(u,v) = \begin{cases} \frac{dist(u,v)+r^2\frac{\|S(u)-S(v)\|}{L}}{2r^2} & dist(u,v) \leq E \\ 0 \quad \text{otherwise} \end{cases} \tag{1}$$

Here, $dist(u, v)$ represents the Euclidean distance between vertices u and v at a radial distance r from either u or v. L is the maximum value of saliency at a radial distance r from either u and v. $S(.)$ is the saliency value of a vertex.

Associated with the weighted graph G is its adjacency matrix A, defined as:

$$A(u,v) = \begin{cases} \omega(u,v) \quad \forall (u,v) \in E \\ 0 \quad \text{otherwise} \end{cases} \tag{2}$$

2.3 Graph Local Characteristics

The topological measures provide valuable insights into the structural properties of a graph and can be used to analyze and understand the characteristics of various types of networks. Here, we will focus on local topological measures of a graph that characterize the properties of individual vertices.

For each mesh, we derive a topology-based features vector TF which is expressed as:

$$\text{TF} = \{\text{d}, \text{dc}, \text{cc}\} \tag{3}$$

where d, dc, and cc are the degree, centrality, and clustering coefficient vectors for mesh mesh-weighted graph, respectively.

Degree: The fundamental and most basic metrics of graphs are the vertex degree. In a weighted graph, the degree of an individual vertex, denoted as $\text{d}(v)$, is explicitly characterized as the summation of the weights associated with all edges connected to it.

$$\text{d}(v) = \sum_{u \in V \setminus \{v\}} w(u,v) \tag{4}$$

Where $V \setminus \{v\}$ denotes the neighboring set of the vertex v.

Degree Centrality: This metric assigns an importance score solely determined by the count of links each node holds. In other words, it is the percentage of the network that the particular node is connected to meaning being similar to.

$$\text{dc}(v) = \frac{\sum_{u \in V \setminus \{v\}} w(u,v)}{\sum\limits_{(i,j) \in E} w(i,j)}. \tag{5}$$

That means that the higher the degree centrality of a node is, the more edges are connected to the particular node and thus the more neighbor nodes this node has.

The Clustering Coefficient: This measure of a node is defined as the probability that two randomly selected similar nodes of are nodes with each other. It is defined by:

$$\mathrm{cc}(v) = \frac{1}{\mathrm{d}(v)(\mathrm{d}(v)-1)} \sum_{u,h} \frac{w_(u,v) + w(v,h)}{2} * \\ A(u,v)A(v,h)A(u,h) \tag{6}$$

As a result, the average clustering coefficient is the average of clustering coefficients of all the nodes. The closer the average clustering coefficient is, the more complete the graph will be because there is just one giant component.

2.4 Statistical Parameters Estimation

The histogram is a precise and straightforward representation that imparts relevant statistical insights into the network structure. Here, we propose using a statistical distribution known as the Gamma distribution to infer statistical parameters. This approach facilitates the reduction of extensive training datasets into smaller subsets. As a result, we can readily use the estimated model parameters instead of the voluminous training data. The parameters are estimated using the maximum likelihood (ML) method. A random variable x conforms to the Gamma distribution with shape parameter α and scale parameter β if it satisfies the following probability density equation:

$$p(x;\alpha,\beta) = \frac{\beta^{-\alpha}x^{\alpha-1}}{\Gamma(\alpha)} \exp\left(-\frac{x}{\beta}\right) \qquad 0 < x < \infty \tag{7}$$

Where $\Gamma(.)$ denotes the Gamma function.

The estimated parameters are then employed as input feature vectors for the regression module that will be used for the learning and the quality score estimation. The final statistical measure vector SM is a concatenation of degree statistics $\mathrm{SM_d}$, degree centrality statistics $\mathrm{SM_{dc}}$ and clustering coefficient statistics $\mathrm{SM_{cc}}$ of the mesh network. For example, $\mathrm{SM_d}$ is defined by:

$$\mathrm{SM} = \{\hat{\alpha}_\mathrm{d}, \hat{\beta}_\mathrm{d}, \hat{\alpha}_\mathrm{dc}, \hat{\beta}_\mathrm{dc}, \hat{\alpha}_\mathrm{cc}, \hat{\beta}_\mathrm{cc}\} \tag{8}$$

where $\hat{b}_d, \hat{\theta}_d, \hat{b}_{dc}, \hat{\theta}_{dc}, \hat{b}_{cc}, \hat{\theta}_{cc}$ are the estimated Gamma parameters of the probability density function of the mesh network, respectively.

2.5 Feature Learning and Regression

In this work, we compare the performances of three machine learning-based methods regression algorithms(Random Forest (RF), Support Vector Regression (SVR), and Generalized Regression Neural Network (GRNN)) to forecast an intermediate quality score through leave-one-out cross-validation (LOOCV). This entails constructing a training regression model using all 3D objects in the repository, excluding one object and its distorted variants at a time. The omitted subset is subsequently employed as the test set, using the established regression model. This iterative procedure is performed for each 3D object within the repository.

3 Experimental Results

This section deals with our experimental methodology, including the studied databases, the evaluation criteria, and a brief overview of the results obtained through comparison with the latest state-of-the-art methods.

3.1 Datasets

The goal of mesh visual quality assessment is to provide quality predictions correlated with the human observer's opinions, named the mean opinion score (MOS). Therefore, a dataset of distorted meshes graded by human observers is needed to evaluate the algorithms. As the meshes' geometric aspect significantly influences the evaluation process, care must be taken when choosing the dataset. It must contain meshes that reflect adequate diversity in their content, and generated distortions should reflect a broad range of mesh degradation. The experiments and tests are conducted on four datasets. These databases, specially designed for quality metrics evaluation, are made of original and distorted mesh. Figure 2 shows the reference objects of the four databases briefly described below.

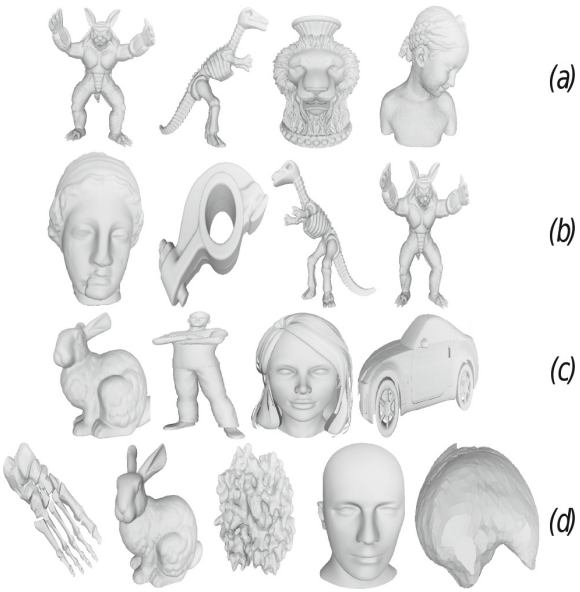


Fig. 2. The reference models from: the LIRIS masking database (a) the general-purpose database (b) and the UWB compression database (c) and the IEETA simplification database (d).

- LIRIS/EPFL General-Purpose database [25]: This database contains four reference meshes and 84 distorted models (88 models in total). Two types of distortion are applied on the reference mesh: smoothing and noise addition either locally or globally.
- LIRIS Masking database [26]: This database comprises four reference meshes and 24 distorted models with local noise addition. This database's specific objective is to test the ability of MVQA methods in capturing the visual masking effect.
- UWB compression database [10] includes five reference models and 63 distorted models. From Twelve to thirteen distorted versions, obtained by a compression algorithm, are associated with each reference model.
- The IEETA simplification database [27] comprises 35 models: five reference meshes and six simplified models for each reference mesh. The simplified models are obtained using three simplification algorithms with two different vertex reduction ratios.

3.2 Evaluation Criteria

To evaluate the obtained predicted MOS and the MOS provided by the database. We used the following two criteria that are Pearson Linear Correlation Coefficient (PLCC) to measure the prediction monotonicity and Spearman Rank-Order Correlation Coefficient (SRCC) to measure the prediction accuracy. A better correlation with human visual system perception a value close to one for PLCC and SRCC. These measures are defined as follows:

$$\text{PLCC} = \frac{\sum_{i=1}^{n}(Qs_i - \bar{Q}s)(\text{MOS}_i - \bar{\text{MOS}})}{\sqrt{\sum_{i=1}^{n}(Qs_i - \bar{Q}s)^2}\sqrt{\sum_{i=1}^{n}(\text{MOS}_i - \bar{\text{MOS}})^2}} \tag{9}$$

$$\text{SRCC} = 1 - \frac{\sum_{i=1}^{n}(\text{rank}(\text{MOS}_i) - \text{rank}(Qs_i))^2}{n(n^2-1)} \tag{10}$$

where n denotes the number of distortions in a given database. The mean opinion scores in the database are defined by MOS_i, and Qs_i is the objective quality score obtained by a given method. MOS and Qs are the mean values of MOS_i and Qs_i, respectively.

3.3 Effect of Weighting Graph

We begin our analysis by highlighting the influence of saliency on quality score prediction, drawing a comparison between the proposed method and the unweighted graph-based method. Then, we also assess the usefulness of regression techniques in predicting quality scores. To achieve this, we evaluate three distinct regression algorithms: Support Vector Regression (SVR), Random Forest (RF), and Generalized Regression Neural Network (GRNN).

Table 1 presents the PLCC and SRCC correlation coefficients achieved by these algorithms on the four databases. Notably, our results demonstrate a significant performance advantage for methods utilizing weighted graphs over those

relying on unweighted graphs across all mesh types and entire datasets. Regarding regression methods, we observe substantial disparities between RF as compared to SVR, and GRNN. In summary, the most effective combination emerges as the one employing the RF regression method within the framework of weighted graphs.

3.4 Comparison with the State-of-the-art

Table 2 presents a comparative analysis of PLCC and SRCC scores achieved by our proposed method against state-of-the-art alternatives. Due to the limited literature options, we compare with only two graph-based no-reference methods in mesh graph assessment. The first, BMQA-GSES, leverages graph spectral entropy and spatial features, while the second employs a graph convolutional network in combination with spatial features. Our study also includes comparisons with other methods, categorizing them as Full-Reference (FR), Reduced-Reference (RR), and No-Reference (NR). We introduce two NR methods, NR-SVR and NR-GRNN, which we have previously proposed. To ensure fairness, we directly source the scores for competing methods from their respective research publications. Among the NR methods, MVQ-GCN consistently offers regular performance across all four databases, primarily attributable to its robust training process. However, it is worth noting that MVQ-GCN relies on an unweighted graph and formulates its prediction process as a node classification problem. In addition, the meshes are trained in the graph convolutional networks without any reduction which makes this method time-consuming. In contrast, BMQA-GSES lags slightly behind our proposed method regarding quality scores. This method's drawback lies in its reliance on numerous spectral and spatial features, while our approach primarily centers on the saliency component, considered a powerful perceptual attribute. The recorded correlation coefficients show that NR-SVR and NR-GRNN also yield respectable prediction scores. These methods rely on roughness measures, a pivotal perceptual feature in mesh processing. The TPDM method achieves a relatively high score, as it is a full-reference approach. In summary, our graph-based method, introduced in this study, outperforms many no-reference methods, and the resulting scores are highly promising, often exceeding or closely approaching the 92% quality score rate.

Table 1. Correlation coefficients SRCC (%) and PLCC (%) with weighted and unweighted graphs on the four databases.

Regression	Method	Masking		General		Compression		Simplification	
		SRCC	PLCC	SRCC	PLCC	SRCC	PLCC	SRCC	PLCC
GRNN	Unweighted Graph	70.6	63.2	63.8	61.4	64.5	60.0	59.4	65.5
	Weighted Graph	82.8	81.2	83.8	80.1	82.0	79.0	78.3	80.4
SVR	Unweighted Graph	74.2	62.4	63.9	65.0	63.1	61.4	62.2	67.8
	Weighted Graph	87.6	86.3	89.4	82.5	87.2	82.6	80.9	83.1
RF	Unweighted Graph	78.1	68.6	66.6	72.2	75.6	73.5	70.5	75.9
	Weighted Graph	91.1	91.9	90.3	88.7	91.2	89.4	88.6	90.8

Table 2. Correlation coefficients SRCC (%) and PLCC (%) of different objective methods on LIRIS masking database, LIRIS/EPFL general-purpose database and the UWB compression database.

Type	Method	Masking		General		Compression		Simplification	
		SRCC	PLCC	SRCC	PLCC	SRCC	PLCC	SRCC	PLCC
FR	HD [5]	26.6	20.2	13.8	11.4	24.5	14.0	49.4	25.5
	RMS [4]	48.8	41.2	26.8	28.1	52.0	49.0	64.3	34.4
	MSDM2 [6]	89.6	87.3	80.4	81.4	78.0	89.3	86.7	79.6
	TPDM [7]	90.0	88.6	89.6	86.2	82.9	**91.5**	86.9	88.2
RR	FMPD [9]	80.2	80.8	81.9	83.5	81.8	88.8	87.2	89.3
NR	NR-SVR [28]	91.1	89.1	84.6	86.8	85.5	88.1	88.9	87.6
	NR-GRNN [29]	90.2	82.4	86.2	88.7	86.3	86.7	87.7	88.0
	BMQA-GSES [22]	91.3	84.1	87.9	**90.5**	87.3	87.4	89.1	89.6
	MVQ-GCN [23]	**91.7**	90.9	89.3	88.6	90.5	87.7	**89.9**	89.4
	Our method	91.1	**91.9**	**90.3**	88.7	**91.2**	89.4	88.6	**90.8**

4 Conclusion

In this paper, we introduced a novel approach for evaluating the visual quality of meshes using graph feature learning methodology. Our method focused on analyzing 3D meshes, treating them as weighted graphs while considering topological features and statistical characteristics. The construction of these graphs relied on the saliency of vertices and their geometric coordinates. We compared the performance of three distinct machine learning methods and found that the Random Forest regression model excels in predicting quality scores, primarily attributed to its suitability for handling graph-structured data. Compared to existing methods for assessing mesh visual quality, including full reference, reduced reference, and no-reference methods, our proposed method exhibits robust correlations with human visual perception. Remarkably, this high accuracy level is achieved by using a straightforward machine learning algorithm. This pioneering exploration into the application of graph representation for mesh quality assessment

holds significant promise and paves the way for numerous future research openings.

Acknowledgment. This work is partially supported by the research grant of the Hassan II Academy of Sciences and Technology of Morocco.

References

1. Pastrana-Vidal, R.R., Gicquel, J.-C., Colomes, C., Cherifi, H.: Frame dropping effects on user quality perception. In: Proceedings of 5th International WIAMIS (2004)
2. Pastrana-Vidal, R.R., Gicquel, J.C., Blin, J.L., Cherifi, H.: Predicting subjective video quality from separated spatial and temporal assessment. SPIE Human Vision Electron. Imaging XI **6057**, 276–286 (2006)
3. Corsini, M., Larabi, M.-C., Lavoué, G., Petrik, O., Vasa, L., Wang, K.: Perceptual metrics for static and dynamic triangle meshes. In: Computer Graphics Forum, vol. 32, no. 1, pp. 101–125. Wiley Online Library (2013)
4. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. In: Computer Graphics Forum, vol. 17, pp. 167–174. Wiley Online Library (1998)
5. Aspert, N., Santa-Cruz, D., Ebrahimi, T.: Mesh: measuring errors between surfaces using the Hausdorff distance. In: IEEE International Conference on. Multimedia and Expo, vol. 1, pp. 705–708 (2002)
6. Lavoué, G.: A multiscale metric for 3d mesh visual quality assessment. In: Computer Graphics Forum, vol. 30, pp. 1427–1437. Wiley Online Library (2011)
7. Torkhani, F., Wang, K., Chassery, J.M.: A curvature tensor distance for mesh visual quality assessment. In: Computer Vision and Graphics, pp. 253–263 (2012)
8. Corsini, M., Gelasca, E.D., Ebrahimi, T., Barni, M.: Watermarked 3-d mesh quality assessment. IEEE Trans. Multimedia **9**(2), 247–256 (2007)
9. Wang, K., Torkhani, F., Montanvert, A.: A fast roughness-based approach to the assessment of 3D mesh visual quality. Comput. Graph. **36**(7), 808–818 (2012)
10. Váša, L., Rus, J.: Dihedral angle mesh error: a fast perception correlated distortion measure for fixed connectivity triangle meshes. In: Computer Graphics Forum, vol. 31, no. 5, pp. 1715–1724. Blackwell Publishing Ltd., Oxford (2012)
11. Abouelaziz, I., El Hassouni, M., Cherifi, H.: A convolutional neural network framework for blind mesh visual quality assessment. In: IEEE International Conference on Image Processing (ICIP), pp. 755–759 (2017)
12. Abouelaziz, I., Chetouani, A., El Hassouni, M., Latecki, L.J., Cherifi, H.: Convolutional neural network for blind mesh visual quality assessment using 3d visual saliency. In: 25th IEEE International Conference on Image Processing (ICIP), pp. 3533–3537 (2018)
13. Hamidi, M., Chetouani, A., El Haziti, M., El Hassouni, M., Cherifi, H.: Blind robust 3D mesh watermarking based on mesh saliency and wavelet transform for copyright protection. Information **10**(2), 67 (2019)
14. Abouelaziz, I., Chetouani, A., El Hassouni, M., Latecki, L.J., Cherifi, H.: No-reference mesh visual quality assessment via ensemble of convolutional neural networks and compact multi-linear pooling. Pattern Recogn. **100**, 107174 (2020)
15. Mourchid, Y., El Hassouni, M., Cherifi, H.: A general framework for complex network-based image segmentation. Multimedia Tools Appl. **78**, 20191–20216 (2019)

16. Rital, S., Bretto, A., Cherifi, H., Aboutajdine, D.: A combinatorial edge detection algorithm on noisy images. In: International IEEE Symposium on VIPromCom Video/Image Processing and Multimedia Communications, pp. 351–355 (2002)
17. Rital, S., Cherifi, H., Miguet, S.: Weighted adaptive neighborhood hypergraph partitioning for image segmentation. In: Singh, S., Singh, M., Apte, C., Perner, P. (eds.) Pattern Recognition and Image Analysis: Third International Conference on Advances in Pattern Recognition, ICAPR 2005, Bath, UK, 22–25 August 2005, Proceedings, Part II, vol. 3, pp. 522–531. Springer, Heidelberg (2005). https://doi.org/10.1007/11552499_58
18. Ribas, L.C., Riad, R., Jennane, R., Bruno, O.M.: A complex network-based approach for knee Osteoarthritis detection: data from the Osteoarthritis initiative. Biomed. Signal Process. Control. **71**, 103133 (2022)
19. Lasfar, A., Mouline, S., Aboutajdine, D., Cherifi, H.: Content-based retrieval in fractal coded image databases. In: Proceedings 15th International Conference on Pattern Recognition, vol. 1, pp. 1031–1034. IEEE ICPR (2000)
20. Demirkesen, C., Cherifi, H.: A comparison of multiclass SVM methods for real-world natural scenes. In: Blanc-Talon, J., Bourennane, S., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2008, vol. 5259. Springer, Heidelberg (2008)
21. Hassouni, M.E., Cherifi, H., Aboutajdine, D.: HOS-based image sequence noise removal. IEEE Trans. Image Process. **15**(3), 572–581 (2006)
22. Lin, Y., Yu, M., Chen, K., Jiang, G., Chen, F., Peng, Z.: Blind mesh assessment based on graph spectral entropy and spatial features. Entropy **22**(2), 190 (2020)
23. Abouelaziz, I., Chetouani, A., Hassouni, M.E., Cherifi, H., Latecki, L.J.: Learning graph convolutional network for blind mesh visual quality assessment. IEEE Access **9**, 108200–108211 (2021)
24. Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. In: ACM SIGGRAPH 2005 Papers, pp. 659–666 (2005)
25. Lavoué, G., Gelasca, E.D., Dupont, F., Baskurt, A., Ebrahimi, T.: Perceptually driven 3D distance metrics with application to watermarking. In: Applications of Digital Image Processing XXIX, vol. 6312, p. 63120L. International Society for Optics and Photonics (2006)
26. Lavoué, G., Larabi, M.C., Váša, L.: On the efficiency of image metrics for evaluating the visual quality of 3D models. IEEE Trans. Visualization Comput. Graph. **22**(8), 1987–1999 (2015)
27. Silva, S., Santos, B.S., Ferreira, C., Madeira, J.: A perceptual data repository for polygonal meshes. In: 2009 Second International Conference in Visualisation, pp. 207–212. IEEE (2009)
28. Abouelaziz, I., El Hassouni, M., Cherifi, H.: No-reference 3d mesh quality assessment based on dihedral angles model and support vector regression. In: Mansouri, A., Nouboud, F., Chalifour, A., Mammass, D., Meunier, J., Elmoataz, A. (eds.) International Conference on Image and Signal Processing, pp. 369–377. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33618-3_37
29. Abouelaziz, I., El Hassouni, M., Cherifi, H.: A curvature based method for blind mesh visual quality assessment using a general regression neural network. In: 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), pp. 793–797. IEEE (2016)

Multi-class Classification Performance Improvements Through High Sparsity Strategies

Lucia Cavallaro(✉), Tommaso Serafin, and Antonio Liotta

Free University of Bozen-Bolzano, Piazza Domenicani 3, 39100 Bolzano, Italy
{lucia.cavallaro,tserafin,antonio.liotta}@unibz.it

Abstract. A fast and accurate stand-alone face recognition system is crucial for surveillance purposes; however, it is also important to keep the costs as low as possible. Herein, we address this issue by proposing a preliminary analysis of embedded Machine Learning techniques by using Erdős-Rényi sparse random networks. The idea is to develop a compact and reliable ANN to conduct a multi-class classification of low-resolution face images to simulate the scenario of having cheap security cameras with an embedded ANN. The study considered two architectures (ResNet, and AlexNet inspired CNNs) with a sparsity level varied up to 90%. To achieve comparable results, the image resolution was varied from 32×32 up to 96×96. The analyses unveiled that for low-resolution images, the best trade-off between accuracy and sparsity level has been achieved with ResNet architectures and a sparsity level of 70% outperforming the benchmark (*i.e.*, with no sparsity).

Keywords: Embedded Machine Learning · Sparse Artificial Neural Networks · Complex Networks Compression

1 Introduction

Despite the first embedded system was born way back in the 60 s of the last century [1] for NASA's Apollo missions, there is nowadays an exponentially growing interest for its application on IoT (*i.e.*, Internet of Things) [2] that, thanks to the easier and cheaper availability of small-sized sensors, allow a smoother integration of smart devices in our society. The applications of such technologies are the most diverse: mobility, grids, domotics, environmental monitoring, industrial processing, healthcare, and security, to cite a few [3].

This latter aspect was the application domain of the present work. Indeed, in the context of security issues, it became crucial to develop smarter IoT devices for cheap embedded lightweight CCTVs, and a few steps are already moved in this direction in the state-of-art [4]. Lightweight cameras are a perfect example of how Edge AI can revolutionise an established system architecture. By moving the inference on the edge of the system's network (*i.e.*, the deployed devices), it is possible to avoid data breaches and obtain an overall more secure architecture [5].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 331–338, 2024.
https://doi.org/10.1007/978-3-031-53468-3_28

However, we are prone to push further the compression of those technologies to allow higher performances with strict hardware constraints. To do so, we consider the concept of sparse random networks borrowed from the Graph Theory and Network Science domains.

In Mocanu *et al.* [6], we unveiled that choosing an initial Erdős-Rényi sparse random graph as ANN instead of having the classic fully-connected layers, there was a quadratic reduction of the number of parameters, with no decrease in accuracy; additionally, there was also significant optimisation both in terms of storage space and computational complexity.

Despite there being plenty of studies about techniques able to compress the models (*e.g.*, knowledge distillation processes that transfer learned knowledge from a larger model to a smaller one), our contribution differs from them. Our aim is to make embedded devices learn and get trained autonomously; hence, we want all the computation to be done in a 'stand-alone' environment as a final goal. For this reason, we want to investigate to what extent it is possible to compress ANNs for multi-class classification problems in the context of security.

To perform our analyses we have chosen a publicly available face images dataset and we used eight classes based on three factors, which are: age (*i.e.*, young/old), gender (*i.e.*, male/female), and skin tone (*i.e.*, pale/dark). We considered as of primary importance the fact that this dataset was the largest collection of muti-labelled face images. The high number of labelled features allowed us to investigate different aspects of the human face by expanding the number of classes for our algorithm.

Next, we utilised two Convolutional Neural Networks (CNNs) architectures, namely ResNet and AlexNet, on which we varied both image resolution (from the highest of 96×96 up to smaller images of 32×32) and sparsity level of the ANNs (up to 90%). The trade-off was evaluated by the combined factors of high accuracy and compact size of the models.

The most relevant outcome was that our analyses confirmed what was found in our previous work on binary classification decease detection [7] that is that a high sparsity configuration (*i.e.*, equals to 70%) can both reduce significantly the size of the models but also increasing the performances in terms of accuracy compared with the classic dense topologies.

Due to the small size of the resulting models, it is possible to make them fit within the strict requirements of low-cost embedded systems. This allows us to bring AI features to contexts requiring either complex system architectures or expensive hardware. Overall, this further study on the topic introduces the possibility of adopting sparse neural networks for highly sensitive tasks such as intrusion detection or personnel identification. Following this line, our aim in terms of future steps is to apply this approach to more complex tasks such as human detection and human identification.

The paper is organised as follows: in Sect. 2 the background materials on the theoretical foundations on which our work is based, the experimental setup, and the description of the selected evaluation metrics herein are summarised. Section 3 displays the relevant results obtained from the proposed analyses and

the impact that such an outcome will have on the future design of smaller embedded ML in addressing security issues on lightweight devices.

2 Materials and Methods

This section introduces not only the theoretical background but also the experimental setup, including the dataset used to conduct our simulations.

2.1 Background and Related Works

In this section, the relevant background and related works are provided.

Since our goal is to analyse to what extent the concept of sparsity in Artificial Neural Networks (ANNs) could be pushed forward to define a trade-off between compression and performance with the final goal to develop smarter IoT devices for cheap embedded lightweight CCTVs, it is needed to start our discussion by introducing the Sparse Evolutionary Training (SET) approach firstly defined in the work of Mocanu *et al.* [6] that represents the starting point of the work herein conducted.

The basic idea relies on the intuition that, although starting from a dense ANN, it led to a sparse one at the end of the training. Hence, in the SET algorithm, the ANN is first initialised as a sparse weighted Erdős-Rényi graph [8]. At the end of each epoch, furthermore, null-edges (*i.e.*, intra-layer links having weight equal to zero) are replaced with new non-zero random weights with the aim to both reduce the loss on the training set still keeping the number of connections constant.

In Cavallaro *et al.* [7], we investigated the sparsity potential on the classification problems in the context of disease detection; hence, that work was based on binary classifications.

Herein, instead, we focus on multi-class classification on security application domains rather than medical purposes. We want to verify whether and to what extent it is possible to combine lightweight devices, such as cheap CCTV, with embedded ML by using sparse ANNs. We considered the same initial Keras-based implementation of SET[1] and we adapted it with the aim being able to conduct a multi-class classification of facial images.

The complex networks under scrutiny are the ANNs and we analysed them to achieve fast training of the architectures mentioned above, still keeping the accuracy high. Hence, we want to highlight which is the threshold between compressed ANNs and high accuracy.

2.2 Dataset

To perform our experiments, we used the CelebA dataset[2] [9]. This is a public large-scale face attributes dataset with more than 200.000 celebrity images,

[1] https://github.com/dcmocanu/sparse-evolutionary-artificial-neural-networks/tree/master/SET-MLP-Keras-Weights-Mask.

[2] https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

and we chose it as it contains 40 attribute annotations and, hence, allowed a wide range of customisation. The use of publicly available dataset also allows the replicability of the simulations herein shown. Furthermore, we decided to use one dataset in order to address a specific scenario and focus on the compression performances rather than comparing different possible applications of the compression method itself; nonetheless, future developments of the proposed research will also include this latter aspect.

In terms of data pre-processing, the dataset was split into training, validation, and testing, with 80% of the data used for training and validation purposes. The images were, then, resized according to the desired resolutions which are 96×96, 48×48, and 32×32. We chose, among the available attributes, to consider 8 classes based on the permutations of three factors, that are: age (*i.e.*, young/old), gender (*i.e.*, male/female), and skin tone (*i.e.*, pale/dark) for a two-fold reason: those are the most relevant facial features among the ones available and we also wanted to keep the model as simple as possible.

2.3 Methodology

In this section, a description of the methodology followed after the pre-processing step of the input images described in the previous section is shown.

For the implementation, we employed Keras and Tensorflow APIs to create a training pipeline that allows the generation of ML models at different sparsity levels.

We adopted the sparse training approach, which induces sparsity in the network while training the model itself.

Our analysis is a trade-off among varying architecture, image resolution, and sparsity levels. In detail, we used two inspired CNNs architectures, namely Residual Neural Networks (*i.e.*, ResNet) [10] and AlexNet [11]. We selected them because our previous analysis [7], which was applied to disease detection, unveiled that those two are the most promising architectures to achieve fast and reliable results among the ones evaluated. Similarly to [7], we applied a binary weight mask to induce sparsity at each epoch of the training process.

To detect the lowest, but yet accurate, resolution image, we considered three different levels, namely 96×96, 48×48, and 32×32 contrary to our previous work because of the different nature of the input images. Indeed, in the present study, a higher resolution was required to distinguish facial features.

We recall that CNNs are a type of feed-forward neural network that is able to extract features from input data with convolution structures (*i.e.*, mathematical operations that allow the merging of two sets of information) to filter the information and generate a feature map. They are particularly suitable for the use case as architectures make the implicit assumption that the input is image-like [12].

The focus of our work relies on sparsity-level investigation. We varied the network density from the benchmark (*i.e.*, dense network with 0% sparsity) up to 90%. To the sake of brevity, we reported only the three most relevant sparsity levels, which are 50%, 70%, and 90%.

3 Discussion and Conclusions

In this section, the relevant outcomes are discussed.

We recall that the idea is to develop a compact and reliable ANN to conduct a multi-class classification of low-resolution face images to simulate the scenario of having cheap security cameras with an embedded ANN. The study considered two architectures (*i.e.*, ResNet and AlexNet inspired CNNs), and the sparsity level has been varied up to 90%.

Table 1 shows the results obtained in terms of accuracy and size of the 'tensor flow lite' files (*i.e.*, *tflite*) at the variations of the sparsity level of the networks and the image resolutions.

Table 1. The table shows the accuracy and size outcomes of the simulations conducted on AlexNet inspired and ResNet CNNs architectures at the variation of image resolution (*i.e.*, 32×32, 48×48, and 96×96) and sparsity level (*i.e.*, 50%, 70% and 90% plus the 0% that is the dense network used as benchmark).

Architecture	Resolution (pixel)	Sparsity (%)	Accuracy (%)	Size (Kb)
AlexNet	32×32	0%	60.0%	159 Kb
AlexNet	32×32	50%	62.4%	105 Kb
AlexNet	32×32	70%	63.3%	73 Kb
AlexNet	32×32	90%	55.7%	37 Kb
AlexNet	48×48	0%	67.0%	322 Kb
AlexNet	48×48	50%	68.1%	209 Kb
AlexNet	48×48	70%	67.2%	149 Kb
AlexNet	48×48	90%	59.8%	73 Kb
AlexNet	96×96	0%	82.3%	1198 Kb
AlexNet	96×96	50%	83.9%	786 Kb
AlexNet	96×96	70%	84.3%	551 Kb
AlexNet	96×96	90%	77.7%	226 Kb
ResNet	32×32	0%	75.5%	297 Kb
ResNet	32×32	50%	76.5%	186 Kb
ResNet	32×32	70%	76.1%	131 Kb
ResNet	32×32	90%	61.3%	54 Kb
ResNet	48×48	0%	77.1%	297 Kb
ResNet	48×48	50%	78.0%	186 Kb
ResNet	48×48	70%	78.3%	131 Kb
ResNet	48×48	90%	64.8%	54 Kb
ResNet	96×96	0%	85.3%	297 Kb
ResNet	96×96	50%	85.9%	186 Kb
ResNet	96×96	70%	85.8%	131 Kb
ResNet	96×96	90%	79.3%	54 Kb

As a general overview, the analyses display that both the architectures under scrutiny exhibit higher accuracies for a sparsity level equal to 70%, which achieved even higher performances than the one obtained from the benchmark configuration (*i.e.*, dense network). For larger sparsity, a significant drop in terms of performances has been unveiled as shown in Fig. 1.

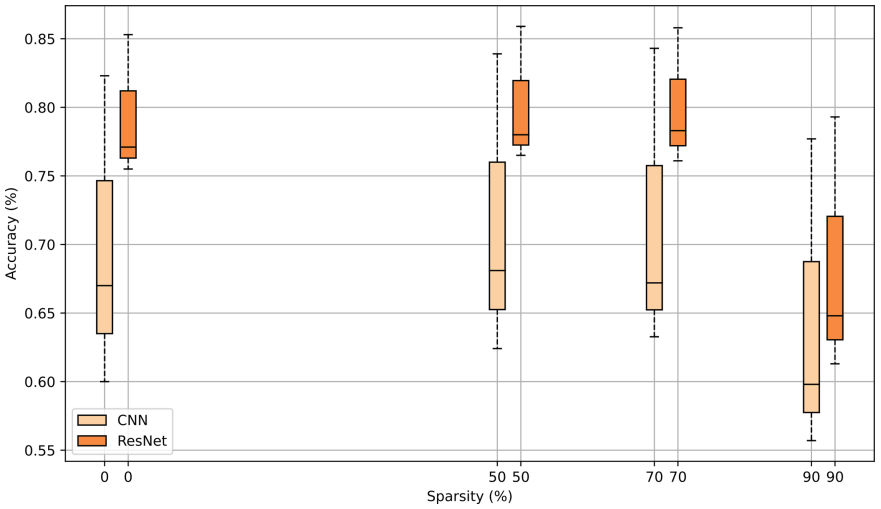


Fig. 1. The figure shows the accuracy percentage trend at the variation of the sparsity level of the two architectures under scrutiny (*i.e.*, AlexNet in light orange and ResNet in dark orange). (Color figure online)

Table 1 also highlights that the resolution of the input image does not affect the size of ResNet models. This feature allows the analysis of more detailed frames and achieves better accuracy overall. Contrary to our previous work on binary classifications of blood cells [7], in which we varied the resolution image from 8×8 up to 32×32, we set herein higher resolutions as it was necessary to identify the higher number of features that characterise a human face.

Hence, from higher (*i.e.*, 96×96) to medium resolutions (*i.e.*, 48×48) with a sparsity of 70% a significant drop of accuracy occurs (from 85.8% to 78.3%). However, it is worth noticing that, when high resolutions are not an option, such as if using cheap devices and having similar hardware constraints, there is no significant performance degradation from medium to low resolutions (from 78.3% to 76.1%).

Since ResNet models maintained the same size, we can consider them as a better solution compared to CNN models in this classification task.

The better performances of ResNet compared with AlexNet inspired CNNs are also visible from Fig. 2. Indeed, the configurations having both lower size and higher accuracy are all ResNets.

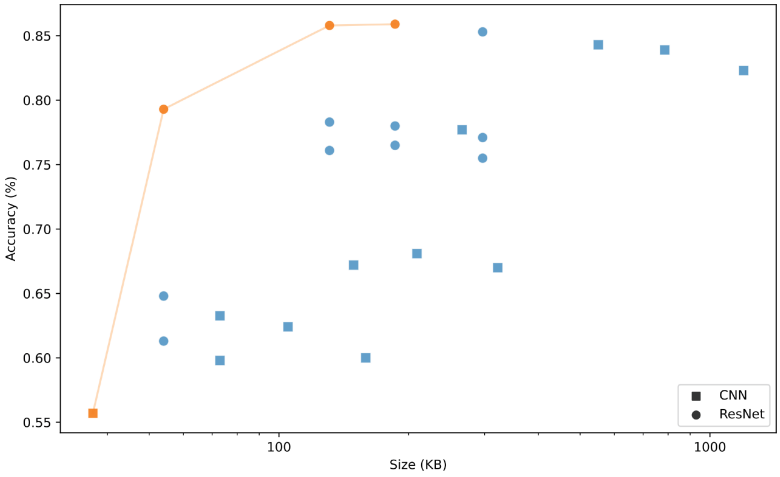


Fig. 2. The figure shows the trade-off between the accuracy's percentage and size of the *tensorflowlite* files (in Kb) of the two architectures under scrutiny (*i.e.*, AlexNet in squares, and ResNet in circles).

In conclusion, the analyses confirmed the goodness of opting for an Erdős-Rényi sparse random graph as ANN instead of having the classic fully-connected layers topology. Furthermore, the paper herein proposed unveiled that for low-resolution images, the best trade-off between accuracy and sparsity level has been achieved with ResNet architectures and a sparsity level of 70% outperforming also the benchmark (*i.e.*, with no sparsity).

The results obtained are also consistent with our previous work on binary classification of disease detection [7] in which we identified not only that ResNet was a more suitable architecture for efficient image classification purposes, which is also consistent with the literature, but also and most importantly that the use of sparse Erdős-Rényi random graphs as initial configuration of ANNs play a crucial role in improving the training performances in terms of achieving high accuracy with strict space constraints. Such performances can be obtained with a sparsity level equal to 70% before encountering the physiological degradation of the network.

As future directions, it would be interesting to apply the intuitions unveiled through our works conducted so far about the investigation of the opportunities that sparsity offers on embedded prototypes with the final goal of developing embedded, smart, and powerful lightweight devices for security object detection purposes on cheap CCTVs.

Acknowledgements. This work was supported by the PRIN 2020 project COMMON-WEARS (grant number I53C21000210001) and by the STEADIER Project (grant number I55F21001900005).

References

1. Brady, C.D.: Apollo guidance and navigation electronics. IEEE Trans. Aerospace (2), 354–362 (1965)
2. Samie, F., Bauer, L., Henkel, J.: Iot technologies for embedded computing: a survey. In: Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, CODES '16, New York, NY, USA. Association for Computing Machinery (2016)
3. Khanna, A., Kaur, S.: Internet of things (IoT), applications and challenges: a comprehensive review. Wirel. Pers. Commun. **114**, 1687–1762 (2020)
4. Rohadi, E., et al. Internet of things: CCTV monitoring by using raspberry pi. In: 2018 International Conference on Applied Science and Technology (iCAST), pp. 454–457. IEEE (2018)
5. Raghubir Singh and Sukhpal Singh Gill: Edge AI: a survey. Internet Things Cyber-Phys. Syst. **3**, 71–92 (2023)
6. Mocanu, D., Mocanu, E., Stone, P., Nguyen, P., Gibescu, M., Liotta, A.: Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nat. Commun. **9**, 06 (2018)
7. Cavallaro, L., Serafin, T., Liotta, A.: Miniaturisation of binary classifiers through sparse neural networks. In: Numerical Computations: Theory and Algorithms. Springer, Heidelberg (2023)
8. Erdös, P., Rényi, A.: On random graphs i. Publicationes Mathematicae Debrecen **6**, 290 (1959)
9. Liu, Z., Luo, P., Wang, X., Tang, X.: Deep learning face attributes in the wild. In: Proceedings of International Conference on Computer Vision (ICCV) (2015)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25** (2012)
12. Teuwen, J., Moriakov, N.: Chapter 20 - convolutional neural networks. In: Zhou, S.K., Rueckert, D., Fichtinger, G. (eds.) Handbook of Medical Image Computing and Computer Assisted Intervention, The Elsevier and MICCAI Society Book Series, pp. 481–501. Academic Press (2020)

Learned Approximate Distance Labels for Graphs

Ikeoluwa Abioye[1], Allison Gunby-Mann[1(✉)], Xu Wang[2], Sarel Cohen[3], and Peter Chin[1]

[1] Dartmouth College, Hanover, NH 03755, USA
{ikeoluwa.f.abioye.23,allisonmann.th,peter.chin}@dartmouth.edu
[2] Boston University, Boston, MA 02215, USA
xuwang@bu.edu
[3] Hasso-Plattner-Institute, Potsdam, Germany
sarel.cohen@hpi.de

Abstract. Distance computation is a fundamental problem in algorithmic graph theory with broad applications across various fields. Distance labeling is the method of assigning a label ℓ to each node in a given graph G such that the distance between any pair of nodes u, v can be efficiently computed (or approximated) using only their labels $\ell(u)$ and $\ell(v)$. Minimizing the size of these labels is of crucial importance for performance. In this paper, we address this challenge by introducing a novel learning-based approach to distance labeling inspired by collaborative filtering. This approach achieves superior performance compared to the theoretical baseline on label size with a trade-off in distance approximation error on special graph classes such as cycles and trees. We also report promising experimental results on general graphs that obtain lower error than cycles and trees.

Keywords: Graph Algorithms · Deep Learning · Shortest Paths · Distance Labels

1 Introduction

Computing and approximating distances between vertices in a graph is a fundamental challenge in algorithmic graph theory, with a myriad of practical applications including navigation, robotics, and social network analysis. Classical algorithms for computing distances (and shortest paths) "from scratch", like Dijkstra's algorithm [10] and A^* algorithm [14], are known for many decades. However, the running time of these algorithms is polynomial in the size of the graph, and can quickly become too slow for practical applications as the graph size grows.

In this work, we explore a theoretical problem of node labeling that is motivated by the need to develop space-efficient data structures for computing distances in a graph. The problem is formally defined below, but roughly speaking, the objective is to assign labels $\ell(\cdot)$ to the nodes of a graph in such a way that the

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 339–350, 2024.
https://doi.org/10.1007/978-3-031-53468-3_29

distance between two nodes u, v can instantly be computed (or approximated to within a multiplicative factor of $1 + \varepsilon$, for small ε) using the labels $\ell(u), \ell(v)$ of these two nodes. The goal is to generate a valid distance labeling which uses as few labels as possible. Distance labeling was investigated from the theoretical perspective for various types of graph classes, e.g., trees [2,5,11], planar graphs [13], sparse graphs [3] and general graphs [6].

To address this issue, recent years have seen many works (e.g., see [1,9] and the references within) aiming to preprocess an input graph into a (time- and space-efficient) data structure that can answer distance queries very quickly, without the need to run expensive computations in real-time. In addition to these traditional algorithmic approaches, modern research has considered the problem of approximate distance labeling, where some error is allowed in the interest of time and space complexity. [15] focuses on road graphs where computing the exact distance is not necessary. [8,15] use machine learning models for approximate distance labels, with models based on computing embeddings and capping their size, and simple multi-layer perceptrons. These had promising results, but improvement can be made on space required for performance.

We propose a structured model to the approximate distance labeling problem that is motivated by collaborative filtering, which is a technique typically used in recommender systems. For graphs, the idea is that nodes that are connected will have similar neighborhoods. When working on large graphs, it is computationally impossible to extract all connectivity information for every node and some filtering is required to combat that overwhelming amount of data. By focusing on the general case of distance labeling in cycles, trees, and finally, arbitrary graphs, we contribute to the theoretical understanding of the problem and show how these theoretical bounds can be pushed experimentally with minor accuracy sacrifices. Our experimental results demonstrate the effectiveness of our approach in comparison to existing methods and provide new insights into the underlying structure of distance labeling in general graphs.

2 Theoretical Analysis

2.1 Problem Definition

Given an undirected arbitrary graph G_i with length i and any two nodes u, v in G_i. Denote the distance between u, v in G_i by $d_{G_i}(u, v)$, the distance labelling problem aims to find a labeling scheme $\ell(\cdot)$ and a function f such that

$$f(\ell(u), \ell(v)) = d_{G_i}(u, v)$$

For Approximate Distance Labeling, denote the number of bits required to store the label of each node by r_l, the target is to find a labeling scheme $\ell(\cdot)$ and a function f such that

$$\text{minimize} \quad |f(\ell(u), \ell(v)) - d_{C_i}(u, v)| \quad \text{subject to} \quad r_l \leq M$$

where M is some pre-defined positive value.

2.2 Combinatorial Distance Labeling

Cycles. Denote the set of cycles in which the maximum length of cycle is n by $\mathcal{C}_n$. The goal is to assign the minimum number of labels to the vertices of all cycles according to the following rules:

- Assume vertices x, y belong to a cycle $C \in \mathcal{C}_n$ and are labelled c_1, c_2.
- Assume vertices u, v belong to a cycle $C' \in \mathcal{C}_n$ (with either $C = C'$ or $C \neq C'$) and are also labelled c_1, c_2.
- The distance between x and y in the cycle C must equal to the distance between u and v in the cycle C'.

We then show the upper bound and lower bound on the number of labels by providing the labelling scheme:

Upper bound on the number of labels: $O(n^{3/2})$. Here is the labeling scheme. Let $C_k = (u_1, u_2, \ldots, u_k)$ be the cycle of length k. Represent the number k in binary and denote its binary representation by $bin(k)$. Let k_1 be the first half of the bits $bin(k)$ and let k_2 be the second half of the bits $bin(k)$. In other words, $bin(k) = k_1 \circ k_2$ where $\circ$ denotes the concatenation operator. Then k_1, k_2 contain each $\frac{\log k}{2}$ bits. We now describe the label of the vertex u_i. If $i < k/2$ then we set the label of u_i to color$(u_i) =< i, k_1, 0 >$. Otherwise we set the label of u_i to $< i, k_2, 1 >$.

Note that the label size is $3/2 \log n + 1$, so we use only $O(n^{3/2})$ labels. Given two labels $c_1 =< i, k_1, b_1 >$ and $c_2 =< j, k_2, b_2 >$ such that $b_1 \leq b_2$ if $b_1 = b_2$. The distance of the corresponding vertices is $|i - j|$, otherwise we can recover $k = k_1 \circ k_2$ and then the distance is $|k - |i - j||$.

Lower bound on the number of labels: $\Omega(n^{4/3})$. We show that there are $\Omega(n^4)$ triples of vertices such that the concatenation of the labels of each triple must be unique. This means that $\chi^3 = \Omega(n^4)$ and thus $\chi = \Omega(n^{4/3})$. We choose the triples only from cycles of lengths $n/2, n/2 + 1, \ldots, n$. We choose Triples $= \{(v_i, v_j, v_\ell) | v_i, v_j, v_\ell \in C_k,\ n/2 \leq k \leq n,\ 1 \leq i \leq k/6,\ k/3 \leq j \leq k/2,\ 2k/3 \leq \ell \leq 5k/6\}$. Then the sum of distances $\text{dist}(v_i, v_j) + \text{dist}(v_j, v_\ell) + \text{dist}(v_\ell, v_i) = k$.

Lemma 1. Each triple $(v_i, v_j, v_\ell) \in$ Triples must be labelled uniquely.

Proof: Let $(v_i, v_j, v_\ell), (v'_i, v'_j, v'_\ell) \in$ Triples be two triples. If the two triples belong to the same cycle, *i.e.*, $v_i, v_j, v_\ell, v'_i, v'_j, v'_\ell \in C_k$ then it is easy to prove that

$$(\ell(v_i), \ell(v_j), \ell(v_\ell)) \neq (\ell(v'_i), \ell(v'_j), \ell(v'_\ell))$$

Assume the two triples belong to different cycles, *i.e.*, $v_i, v_j, v_\ell \in C_k$ and $v'_i, v'_j, v'_\ell \in C_{k'}$ such that $k \neq k'$. Then the two triples must be colored differently as the following equation holds.

$$k = \text{dist}(v_i, v_j) + \text{dist}(v_j, v_\ell) + \text{dist}(v_\ell, v_i)$$

$$k' = \text{dist}(v'_i, v'_j) + \text{dist}(v'_j, v'_\ell) + \text{dist}(v'_\ell, v'_i)$$

$$k \neq k'$$

Thus, we found $\Omega(n^4)$ triples which must be labelled differently, and hence $\chi = \Omega(n^{4/3})$.

Note that the upper bound uses $log_2(n^{3/2}) + 1 = 3/2log_2(n) + 1$ bits and the lower bound shows that at least $log_2(n^{4/3}) = 4/3log_2(n)$ bits are required for exact distances. Our results for cycles are approximate distances, and the number of bits we use approximately match the lower bound.

Trees. We consider distance labeling schemes for trees: given a tree with n nodes, label the nodes with binary strings such that, given the labels of any two nodes, one can determine, by looking only at the labels, the distance in the tree between the two nodes. Alstrup *et al.* showed in [7] that $\frac{1}{4}\log_2(n)$ bits are needed for exact distances and that $\frac{1}{2}\log_2(n)$ bits are sufficient. They also give a $(1+\epsilon)$-stretch labeling schemes using $\Theta(\log n)$ bits for constant $\epsilon > 0$. This result was extended by Freedman *et al.* in [12], who showed that the recent labeling scheme of [7] can be easily modified to obtain an $O(\log_{1/\epsilon} n)$ upper bound they also proved a matching $O(\log_{1/\epsilon} n)$ lower bound. Our method achieves approximately the theoretical bounds of $1 + \epsilon$ approximate distances on trees with $\epsilon = 0.4$.

2.3 Approximate Distance Labeling

Lemma 1. *Let $n > 4$. An approximate distance-labeling scheme for the cycles problem that reports $1 + \varepsilon$-approximate distances with $\varepsilon < \sqrt{2} - 1$ must have a label size of at least $\log_2(n)$ bits.*

Proof. We prove that all the labels in a cycle must be different and thus one must use $\log_2(n)$ bits to represent the labels of the vertices in the largest cycle whose length is n. Assume by contradiction that x,y are vertices in the same cycle that are assigned with the same label and let z be the neighbor of y such that $d(x, z) \geq 2$ (since n $\geq$ 5, it is easy to verify that at least one of the neighbors of y has this property). Then d(z, y)=1 (as y and z are neighbors) and d(z,x)$\geq$2. As x and y have the same label, then the labeling scheme outputs the same distance D for d(z,x) and d(z,y). As the labeling plan is (1+ε) approximate, it must hold that D<=1+ε so that D is a (1+ε)-approximation of d(z,y), and it also must hold that D $\geq \frac{2}{1+\varepsilon}$ so that D is a (1+ε)-approximation of d(z,y). However, these two equations cannot hold when $\varepsilon < \sqrt{2} - 1$, which is a contradiction (Fig. 1).

3 Approach

In this paper, we focus on developing a method that calculates the shortest distance between any two nodes in a graph quickly while simultaneously minimizing the size of the labels employed for distance computation. This dual objective enhances the efficiency of our approach and significantly reduces storage requirements. Our model incorporates an embedding layer and a two-layer feed-forward neural network for this purpose.

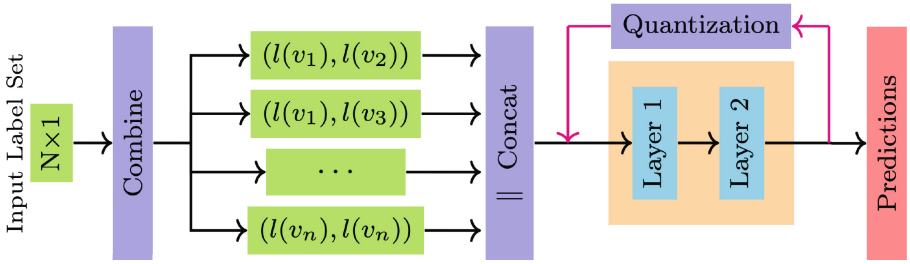


Fig. 1. Architecture of the model

3.1 Training Data

We utilized synthetic datasets for cycles and trees, as well as real-world graphs obtained from [16] for our experiments.

For cycles, we generated a dataset $D_N = \{(u, v, d) : u, v \in C_i, 3 \leq i \leq L\}$ for experiments. Here, $G = \bigcup_i C_i$ represents a union of disjoint cycles, where C_i is a cycle. V_{C_i} is the set of all nodes that belong to C_i, then $V = \bigcup_i V_{C_i}$ is the set of nodes that exist in any cycle within G.

For trees, we followed a similar procedure to generate a dataset $D_N = \{(u, v, d) : u, v \in T_i, 3 \leq i \leq L\}$ for experiments. Define $G = \bigcup_i T_i$ be a union of disjoint random trees, V_{C_i} be the set of all nodes that belong to C_i, then $V = \bigcup_i V_{C_i}$ is the set of nodes that exist in any tree that belongs to G. In both the trees and cycles dataset, each node is assigned a unique integer id.

The real world graphs dataset comprises enzymes graphs obtained from [16] which provides the graph edge lists. We processed the data to ensure that every node in the entire set has a unique id and that there are no gaps between the ids of any two nodes. The dataset $D_N = \{(u, v, d) : u, v \in G_i, 3 \leq i \leq L\}$ was then created from this collection of real-world graphs, where G_i is a graph in the collection. We specifically use the chem-ENZYMES-g1 and chem-ENZYMES-g118 graphs.

3.2 Training

The training phase of our model utilizes a two-layer feed-forward neural network. Our objective is to predict the distance between any pair of nodes from the same cycle, tree, or graph. We employ the embedding layer to generate a vector of size i (the number of bits) as the label for each node. These vectors are then quantized to binary form by taking the sign of each individual element within the vector.

We initialize the feature of the node pairs as the concatenation of their respective quantized labels. After processing this feature through two layers of our model, an output prediction $\hat{Y}$ of the distance between two nodes is produced. Each layer is followed by an activation function. Let n denote the number of

samples; we have $X = \{(l(u), l(v)) : u, v \in C_i \in G\}$, $X \subseteq R^{n\times 2}$. The training phase can be defined formally as:

$$\hat{Y} = \sigma((ReLU(XW_1))W_2)$$

where $W_1 \subseteq R^{2\times d}$, $W_2 \subseteq R^{d\times 1}$ are the weight matrices to be trained. The final activation function σ is modified so that the prediction will be re-ranged to fit the dataset's distance range.

3.3 Label Quantization Strategies

In our work, we employed two distinct quantization strategies: training quantization and post-training quantization.

Training quantization is implemented concurrently with the learning process. Once the labels are generated, we quantize them to further reduce their size. This process takes place during the forward training pass, where we convert the node embeddings into a binary form by taking the sign of each embedding value. The size of each node's embedding is determined by the quantization level, and by manipulating the number of bits i, we can control the trade-off between the label size and the accuracy of node pair distance computation. Despite its intuitive appeal, this approach exhibited suboptimal performance on cycles and graphs, while showing improved results for trees.

On the other hand, post-training quantization is applied after the training phase. In this strategy, we first complete the training without any quantization. Afterward, we quantize the resultant embeddings, reintroduce these quantized labels into the model, and execute a forward pass to derive the predictions. This method demonstrated superior efficacy for cycles and graphs, but was less effective when applied to random trees.

We hypothesize that this disparity in performance is likely related to the prevalence of cycles in the graph datasets, particularly the high frequency of triangles. Thus, it is important selecting the appropriate quantization method based on the characteristics of the graph structure.

4 Experiments

4.1 Dataset

We use synthetic datasets with $N = 2^6$ (or 64). This results in a total of $(64-3)*(3+63)/2 = 2079$ nodes each for both the cycle and tree datasets. To enhance the performance of the model on node pairs with close proximity, we duplicate pairs of cycle nodes with a distance less than 5. We use the entire dataset for training purposes, while reserving 10% of the data as a validation set.

The cycles dataset consists of cycles of length 3 to 2^6.

4.2 Evaluation Metrics

We use the Mean Squared Error(MSE) and Mean Relative Error(MRE) as our metrics. Given the prediction $\hat{Y} = \{\hat{y}_i : 1 \leq i \leq N\}$ and labels $Y = \{y_i : 1 \leq i \leq N\}$, the MSE and MRE are computed as follows:

$$MSE(\hat{Y}, Y) = \frac{1}{N}(\hat{Y} - Y)^T(\hat{Y} - Y) \tag{1}$$

$$MRE(\hat{Y}, Y) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{|y_i|} \tag{2}$$

We define parameter α as a float that ranges from 0.0 to 1.0, the final loss l is computed as the combination of MSE and MRE:

$$l_\alpha(\hat{Y}, Y) = \alpha MSE(\hat{Y}, Y) + (1 - \alpha)MRE(\hat{Y}, Y) \tag{3}$$

We found that an α value of 0.5 produces lower loss values.

5 Results

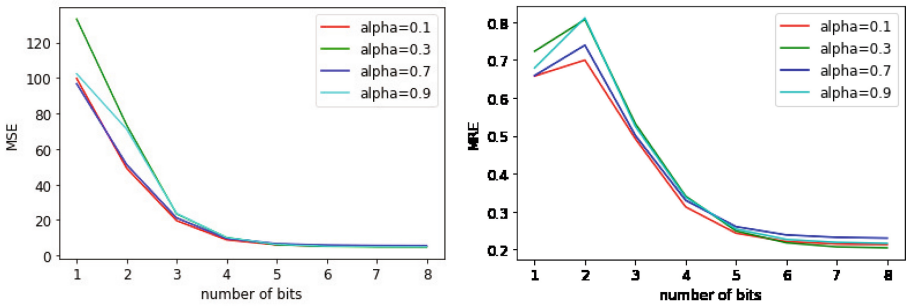
5.1 Cycles and Alpha Values

We evaluate the performance of our model, specifically trained on cycle structures, across various alpha values. This assessment is conducted using the post-training quantization approach.

Table 1. Results with D_6

	$\alpha = 1$		$\alpha = 0$		$\alpha = 0.5$	
	MSE	MRE	MSE	MRE	MSE	MRE
1	149.175964	0.798590	525.602400	1.223840	97.190900	0.648095
2	63.960815	0.622632	90.706116	0.876957	57.236360	0.748088
3	32.998089	0.605301	30.461815	0.582474	21.510620	0.509291
4	12.950354	0.372077	17.285600	0.379353	9.796305	0.325589
5	6.635018	0.263844	13.536446	0.292693	6.938656	0.262967
6	4.4964622	0.227134	12.649083	0.259946	6.095809	0.241920
7	4.551461	0.215136	12.409961	0.249289	5.879529	0.235888
8	4.432967	0.212821	12.357460	0.246148	5.845781	0.234776
9	4.405716	0.211925	12.345035	0.244986	5.829654	0.234216
10	4.399604	0.211756	12.338661	0.244760	5.827143	0.234158

We report the results in Table 1. By comparing the whole table with previous theoretical analysis, we have the following observations:



(a) MSE using different number of bits (b) MRE using different number of bits

Fig. 2. Model's performance on Cycles using different number of bits

– Our approach beat the standard combinatorial baseline. The combinatorial baseline requires $\log_2(n^{4/3})$ bits which is about 9 bits to predict the distance accurately. With our approach, we only need 6 bits to achieve 0.22 MRE. Consider the maximum length of all the cycles, our approach achieves satisfying predictions while using significantly less bits.
– Our approach close the gap toward the lower bound for approximate baseline. The approximate baseline requires at least 6 bits to achieve $(1+\epsilon)$-approximation where $\epsilon > \sqrt{2} - 1$. With our approach, we only need 4 bits to obtain MRE$<$ 0.4, which is a huge improvement.

We also conduct study on the effect of different α. The results are shown in Fig 2. From the figure, we observed that:

– Value of α significantly affects the number of bits needed. When $\alpha = 0.1$ and $\alpha = 0.7$, using two bits, model's performance is obviously better than when $alpha = 0.3$ and $\alpha = 0.9$. In other words, given more strict space requirement, adjusting the value of alpha could significantly improve the performance.
– Value of α has less effect than number of bits. As the number of bits is increasing, models' performance converges. They are quite close when we are using more than six bits.

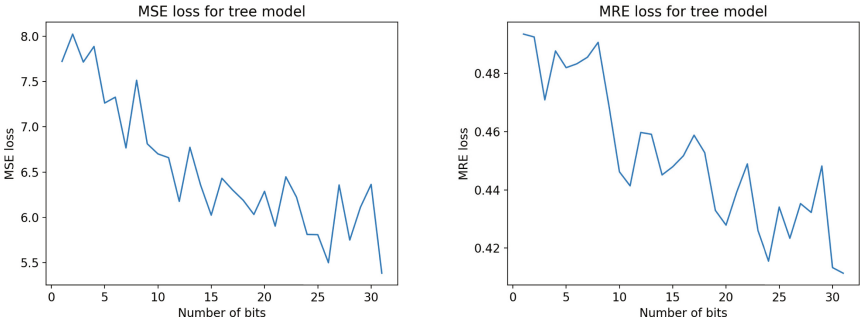
To sum up, our approach demonstrate its effectiveness in learning labels for distance approximation. By changing α, we could obtain descent approximation even using very few bits.

Based on this result, we use an alpha value of 0.5 for the rest of our training.

5.2 Trees

Training Quantization Method. In the case of random tree graphs, our model tends to yield better results when the training quantization strategy is utilized, compared to the post-training quantization method. However, our model's efficacy does not extend as seamlessly to random trees as it does to cycles and

general graphs. When applied to random trees, despite increasing the number of bits to 30, the model records a relatively higher Mean Relative Error (MRE) loss of approximately 0.41. This indicates a need for further optimization or a different approach when dealing with trees. The detailed performance visualizations can be found in Fig. 3.



(a) MSE using different number of bits (b) MRE using different number of bits

Fig. 3. Model's performance on Trees using different number of bits with quantization during training

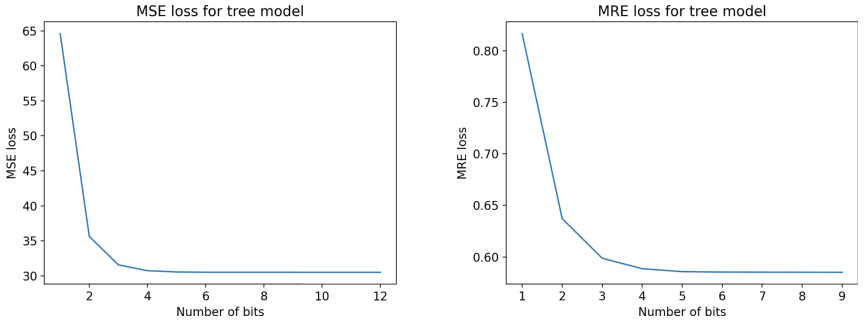
The addition of bits does seem to trend error downwards with the training quantization method, however the effect on error is less dramatic than the results on cycles. Trees require a significantly higher number of bits to achieve an MRE around .45.

Post-Training Quantization Method. Although the post-training quantization strategy has proven to be more effective on cycles and general graphs, it doesn't reach the same level of performance on tree structures. Detailed results can be found in Fig. 4. Further research could explore why this method isn't as effective with trees and how it could be optimized to better handle such data structures.

Our method generates generally worse performance on trees compared to cycles. The performance of the model on trees plateaus around 3 bits with post train quantization, which is an improvement over the theoretical bound of 11, but there is a not insignificant error of .6 MRE/32 MSE associated with this label size. Allowing additional bits does not improve model performance significantly.

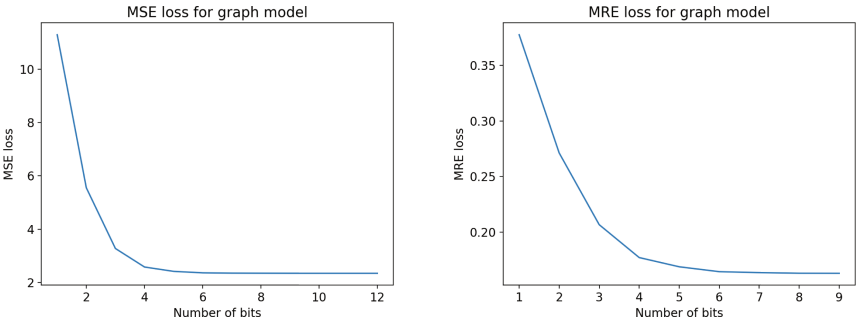
5.3 General Graphs

After fine-tuning our model on both tree and graph datasets, we proceed to evaluate its performance on general graphs. Notably, employing the post-training quantization methodology and setting α to 0.5 yields results superior to those



(a) MSE using different number of bits (b) MRE using different number of bits

Fig. 4. Model's performance on Trees using different number of bits with post-training quantization



(a) MSE using different number of bits (b) MRE using different number of bits

Fig. 5. Model's performance on General graphs using different number of bits

achieved on the cycle dataset. The detailed performance outcomes are depicted in Fig. 5 below.

These results on general, real world graphs are the most promising and experimentally demonstrate that this method has viability for real world applications with large graphs.

6 Limitations

Despite the promising results, our model is not without its limitations. First and foremost, our model exhibits an asymptotic limit in its training loss, implying that past a certain point, increasing the number of bits used for the embedding does not yield further improvements in accuracy. This bottleneck could potentially be addressed by expanding the model size, thereby enabling it to better capture the intricacies of the data structure.

Any application that requires exact distance computation would be unable to benefit from the distance labels produced by our method, as there is some expectation of minor error. Additionally, pre-processing time in the form of training the model is required in advance of distance querying.

7 Computation Specifications

The experimental computations were performed on a server equipped with an Intel(R) Xeon(R) W-2145 CPU with 3.70 GHz and 130 GiB of system memory. The machine was equipped with a single NVIDIA GeForce RTX 2080 Ti GPU, using the driver version 525.105.17 and CUDA Version 12.0.

Each individual experimental run required approximately 30 min. Training was done for 100 epochs for each model. The total compute across all experimental runs included in this paper amounted to less than 1.5 h.

8 Relation to Prior Work

In [5], a distance labeling scheme with labels of length $\frac{log3}{2}n + o(n)$ and constant decoding time is presented. Outperforming previous state-of-the-art, this work focus on general graphs without going deep into specific graph types. [4] narrows down the question on trees. They shows that $\frac{1}{4}log^2 n$ bits are needed and that $\frac{1}{2}log^2 n$ bits are sufficient.

Compared to exact labeling, [15] points out that the exact distance is not always necessary. Focusing on road graphs, they apply a simple multi-layer perceptron(MLP) on distance prediction which use less space than a lookup table. However, the model suffer from a long training time. In [8], both simple MLP and graph neural network models are proposed as the exploration of speeding up path queries. However, the training difficulty and space consumption is significantly increasing as the graph neural network gets more complex.

9 Conclusion

In this paper, we have presented a novel learning-based approach to the Distance Labeling problem, aiming to reduce the size of labels required. Our approach demonstrates substantial improvements over existing combinatorial and approximation baselines, offering increased efficiency and reduced storage needs. Looking forward, an intriguing avenue for future research lies in extending our methodology to accommodate larger and more complex graph structures, including scenarios with node failures. Additionally, we are interested in exploring online cases where cycles are presented sequentially. This direction promises to further expand the applicability and impact of our approach in the field of algorithmic graph theory.

References

1. Akiba, T., Iwata, Y., Yoshida, Y.: Fast exact shortest-path distance queries on large networks by pruned landmark labeling. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, pp. 349–360 (2013)
2. Alstrup, S., Bille, P., Rauhe, T.: Labeling schemes for small distances in trees. SIAM J. Disc. Math. **19**(2), 448–462 (2005)
3. Alstrup, S., Dahlgaard, S., Knudsen, M.B.T., Porat, E.: Sublinear distance labeling. In: 24th Annual European Symposium on Algorithms (ESA 2016), vol. 57, pp. 5:1–5:15 (2016)
4. Alstrup, S., Gørtz, I.L., Halvorsen, E.B., Porat, E.: Distance labeling schemes for trees. CoRR arxiv:1507.04046 (2015)
5. Alstrup, S., Gørtz, I.L., Halvorsen, E.B., Porat, E.: Distance labeling schemes for trees. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, LIPIcs, vol. 55, pp. 132:1–132:16 (2016)
6. Alstrup, S., Kaplan, H., Thorup, M., Zwick, U.: Adjacency labeling schemes and induced-universal graphs. In: Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2015, pp. 625–634 (2015)
7. Alstrup, S., LiGørtz, I., Halvorsen, E.B., Porat, E.: Distance labeling schemes for trees. In: 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy, 11–15 July 2016, LIPIcs, vol. 55, pp. 132:1–132:16 (2016). https://doi.org/10.4230/LIPIcs.ICALP.2016.132
8. Brunner, D.: Distance preserving graph embedding (2021)
9. Chang, L., Yu, J., Qin, L., Cheng, H., Qiao, M.: The exact distance to destination in undirected world. VLDB J. **21** (2012)
10. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
11. Freedman, O., Gawrychowski, P., Nicholson, P.K., Weimann, O.: Optimal distance labeling schemes for trees. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, pp. 185–194 (2017)
12. Freedman, O., Gawrychowski, P., Nicholson, P.K., Weimann, O.: Optimal distance labeling schemes for trees. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, 25–27 July 2017, pp. 185–194. ACM (2017)
13. Gavoille, C., Peleg, D., Pérennes, S., Raz, R.: Distance labeling in graphs. J. Algor. **53**(1), 85–112 (2004)
14. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)
15. Neff, J.: Neural distance oracle for road graphs (2021)
16. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). https://networkrepository.com

Investigating Bias in YouTube Recommendations: Emotion, Morality, and Network Dynamics in China-Uyghur Content

Mert Can Cakmak(✉), Obianuju Okeke, Ugochukwu Onyepunuka, Billy Spann, and Nitin Agarwal

COSMOS Research Center, University of Arkansas - Little Rock, Little Rock, AR, USA
{mccakmak,oiokeke,uponyepunuka,bxspann,nxagarwal}@ualr.edu

Abstract. This research demonstrates how recommendation algorithms can shape public discourse and attitudes on morally charged topics. Our study provides an in-depth analysis of the moral, emotional, and network dimensions of Youtube's recommended videos related to the China-Uyghur issue. We investigated the spread of moral themes and whether the algorithm favors videos with certain emotional feelings. Additionally, we conducted a detailed network analysis to spot the most influential videos and see how the themes change as the recommendations change. We found that as the algorithm recommends more videos, the emotional diversity of the recommendations tends to drift towards positive emotions and away from negative ones. Likewise, there is a decreasing focus on moral dilemmas as one moves through the recommended content. In simple terms, our study shows how YouTube's recommendations may influence viewers' feelings and beliefs. Our network analysis reveals which videos are driving the shift in morality and emotion and how the main discussion points change as more videos are suggested. Through this, we hope to better understand the inherent biases in recommendation engines, especially when they are dealing with emotionally charged and morally complex topics.

Keywords: YouTube's recommendation algorithm · drift analysis · emotion analysis · morality assessment · bias in recommender systems · network analysis

1 Introduction

Recommendation algorithms are frequently associated with biases such as selection bias [14], position bias [7,12], and popularity bias [15,16]. Popular recommendation platforms have, in the past, been associated with patterns that lead users to highly homogeneous content, resulting in certain phenomena such as filter bubbles and echo chambers [11,30]. In such scenarios, users are isolated from diverse content and are instead exposed to a narrower band of information. This can pose the risk of reinforcing specific viewpoints [17,18].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 351–362, 2024.
https://doi.org/10.1007/978-3-031-53468-3_30

This study examines the emotion and morality bias present in YouTube's recommendation algorithm. By analyzing the evolution of emotion and morality across recommended YouTube videos related to the China-Uyghur crisis narrative, we aim to determine if YouTube's recommendation algorithm favors videos with certain emotions over others and to explore the distribution of moral content across YouTube's recommendation algorithm. Also, this research studies the network analysis and aims to find whether echo chambers and topic shifting appear in the narratives by analyzing the eigenvector centrality of videos and tracing the communities.

1.1 Background of Study

In this section, we discuss previous research related to our study, which includes previous works on morality assessment, emotion detection [1,26], network analysis [27,28], and bias analysis in recommender systems [25,30–32]. Recommendation bias has been researched extensively to understand its structure and effects, especially in the areas of radicalization and the spread of misinformation and disinformation [9]. These past works have studied the emergence of homophilic communities within content, such as recommended videos, and they have studied the factors leading to the emergence of these communities. Insights from these studies have been crucial in identifying the emergence of homogeneity, the development of interconnected communities, and the potential bias in recommender systems. Drift is a technique that many researchers have used in studying how content evolves. By studying content evolution, we determine if the content remains the same or changes relative to a standard metric. O' Hare et al. [3] analyzed a sentiment-annotated corpus of textual data to determine topic drift among documents. Liu et al. [2] developed an LDA (Latent Dirichlet Allocation)-based method for topic drift detection in micro-blog posts. Akilaet al. developed a framework to identify the mood of the nation of India by analyzing real-time Twitter posts [20]. Results showed the trends of emotions. These trends were visualized using line graphs and radar maps. Maharani et al. [29] is used eigenvector centrality, a method to identify influential users, to better understand how information spreads on Twitter.

In this research, we apply drift analysis techniques to assess emotion and morality, to determine the pattern of bias in YouTube's recommendation algorithm. By combining both emotion and morality assessment, we take a more holistic approach to understand the nature of and impact on videos recommendations by YouTube's recommendation algorithm. Similarly, by using network analysis and focusing on influential users using eigenvector centrality, we aim to better understand how content spreads and if certain topics dominate the recommendations.

1.2 The China–Uyghur Crisis

The China–Uyghur crisis has garnered significant criticism from various organizations across the globe. According to the Council on Foreign Relations [8], more

than a million Uyghurs—a Muslim, Uyghur-speaking Asian ethnic group—have been detained in China's Xinjiang region since 2017. The United States and the UN Human Rights Office have described these acts as crimes against humanity. Despite reports from international journalists and researchers about the ongoing systems of mass detention throughout the region, which have been backed by satellite images and leaked Chinese government documents, individual testimonies from Chinese officials insist that the rights of Uyghur Muslims have not been violated. They assert that government crackdown measures, such as re-education camps, have been discontinued since 2019 [4]. According to Silverman, content-evoking polarization is propagates faster than non-polarizing content [10]. In effect, since the emotions attached to our seed videos were negative, we also expected to see more negative emotions propagated through videos across recommendation depths. We also worked toward exploring the moral content of these potentially negative emotions to study the moral nature of content distributed by the recommendation algorithm. In addition, by using network analysis and examining the most influential users through eigenvector centrality, we aimed to identify which videos primarily drive the online conversation about the China-Uyghur crisis.

2 Methodology

For this research, we introduce a drift analysis methodology that allowed us to monitor changes in video characteristics and explore the patterns of the recommendation algorithm. We apply the resulting approach to our dataset, which consists of a collection of videos recommended through various methodologies.

2.1 Collection of Data

Video recommendations on YouTube are heavily influenced by the user's watch history, meaning that the algorithm personalizes the videos recommended to a user. To eliminate this personalization bias and control our experiment, we employed the following precautionary steps:

1. Video collection script prevented account login for each watch session.
2. A new browser instance was started for each level or depth of recommendation.
3. Cookies from each previous recommendation depth were cleared to enable a fresh search for videos at the next depth of recommendation.

The data used in this research was composed of YouTube's **'watch-next'** videos which are found in the watch-next panel of the platform. These videos were collected using techniques employed in [9]. To begin our data collection process, we first conducted a series of workshops with subject matter experts to generate a list of relevant keywords related to the China-Uyghur conflict. These keywords were used as search queries on YouTube's search engine to generate the 40 seed videos. Video recommendations were gathered for each seed video using custom-made crawlers over levels or "depths" of recommended videos. Each of the 40

seed videos generated a first depth of recommendations, with subsequent depths serving as parent videos for generating further layers of recommended videos. This process continued until recommendations for four depths were generated, resulting in a total of 15,307 unique videos. In our data collection process, we extracted the video titles and video descriptions. For this research we chose to analyze the videos based on video text data (i.e., titles, descriptions), as these levels of detail will provide information on the content of the videos. The dataset was split by the depth and studied for if and how video characteristics such as emotion and morality changed (or drifted) as the algorithm recommended videos to the user. In our network analysis, we investigated the communities and identified the influencer nodes by looking at each depth. Next, we describe the methodologies used to address the research questions posed in this study.

2.2 Emotion Analysis

We analyzed the emotions embedded in the video text data (i.e., title and description), focusing on seven emotions: anger, disgust, fear, joy, neutral, sadness, and surprise. We used emotional drift to identify emotional bias across various depths of recommendations. The diversity of emotions resulting from the content was illustrated on a line graph, with each point on the depth axis representing a traversed depth of video recommendations. We utilized a fine-tuned version of transfer learning [5], *Emotion-English-DistilRoberta-base* [19], for Natural Language Processing (NLP) tasks to ensure the accuracy of results. Transfer learning aims to increase the accuracy and efficiency of the model training process by preserving information from prior models and applying it to related tasks.

2.3 Morality Assessment

Morality assessment is an effective social computing technique used to extract moral intuition from textual data. For our morality assessment analysis, the extended Moral Foundations Dictionary (eMFD), a dictionary-based tool for extracting moral content from textual corpora, was used. This package was constructed from text annotations generated by a large sample of human coders [24]. Using the Moral Foundations Theory [13], we worked on analyzing the moral content of our narrative and visualized the drift in morality within our dataset across five moral foundations: harm (involving intuitions of sympathy, compassion, and nurturance), cheating (including notions of rights and justice), betrayal (supporting moral obligations of patriotism and "us versus them" thinking), subversion (including concerns about traditions and maintaining social order), and degradation (including moral disgust and spiritual concerns related to the body) [6]. The resulting morality diversity in content was also illustrated on a line graph, with each depth representing a traversed depth of video recommendations.

2.4 Network Analysis

Network analysis gives a perspective through which to view and understand the complex web of recommendation systems and how they influence consumers.

More than just viewing connections, it allows us to identify important patterns, key influencers, and any shifts or drifts that might occur in the network. This kind of visualization is crucial for understanding how different videos relate to each other, and how certain nodes might have a stronger influence on recommendations than others. Eigenvector centrality is our chosen tool to zoom in on these influential nodes. The mathematical details and the definition of this technique are provided below.

For a given graph $G:=(V,E)$ with $|V|$ vertices, let $A = (a_{vt})$ be the adjacency matrix, i.e., $(a_{vt}) = 1$ if vertex v is linked to vertex t, and $(a_{vt}) = 0$ otherwise. The relative centrality score, X_v of vertex v can be defined as:

$$X_v = \frac{1}{\lambda}\Sigma_{t\epsilon M(v)}X_t = \frac{1}{\lambda}\Sigma_{t\epsilon v}a_{vt}X_t \tag{1}$$

where $M(v)$ is the set of neighbors of v and λ is a constant. With a small rearrangement, this can be rewritten in vector notation as the eigenvector equation.

$$Ax = \lambda x \tag{2}$$

3 Results

3.1 Emotion Analysis

The goal of emotional drift analysis is to determine how emotions across the seven categories (anger, surprise, fear, joy, neutral, disgust, and sadness) evolve or drift across recommendation depths. To determine emotional drift, we computed and visualized the predominant emotions at each depth of recommendation, from seed to depth 4, on a line graph. In our analysis, we considered video text data at two different places: video titles and video descriptions. This process allowed us to effectively apply emotion analysis and visualize emotional drift across different levels of video details. The neutral emotion effectively isolates text data with no identifiable emotion embedded. As a result, the neutral emotion in the line graph was not considered in our result analysis. From the graph below in Fig. 1(a), we observe that on emotional drift analysis of the video titles, there was a significant presence of fear, anger, and disgust emotions at the seed level (depth 0), and a reduced presence of joy and surprise emotions. As we moved from seed videos to depth 4 through the recommendations made by the algorithm, we observed an increase in the proportion of joy and surprise emotions. This trend was accompanied by a decrease in the previously heightened fear, anger, and disgust emotions as we approached depth 4. This trend of the emergence of positive emotions and decline of negative emotions across recommendations was also seen in the emotional drift analysis of video descriptions in Fig. 1(b) but with a clear level of distinction. On analyzing video descriptions, we saw a more distinct pattern of emergence and decline; this is most likely due to the higher level of content and information in video descriptions as compared to video titles.

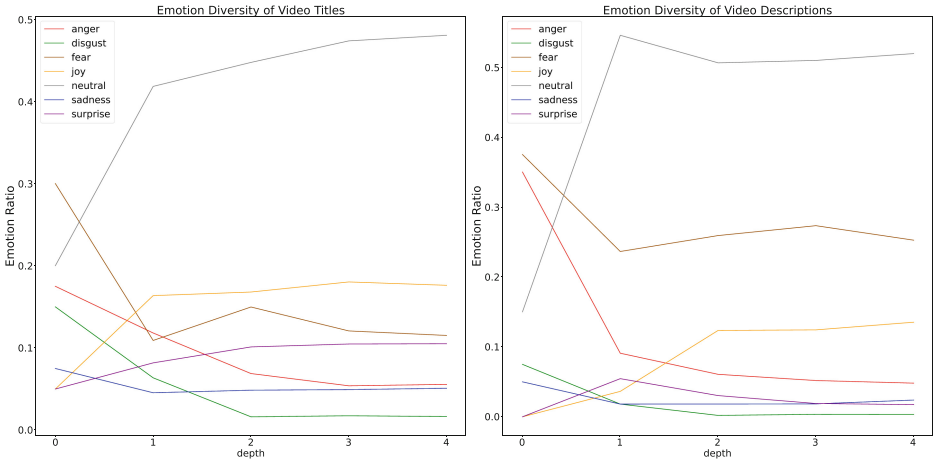


Fig. 1. Line graph showing the distribution of emotions across recommendations of videos using (a) video titles (b) video descriptions.

3.2 Morality Assessment

While emotion analysis is a text analytics tool which has been used to determine the emotional expression in text data [21–23], morality assessment is another form of text analytic technique that has been used to discover ethical reasoning, moral concepts, and decision-making in text data [13]. With the application of morality assessment, we can extract the underlying moral content in text data. This enables us to understand the influence of moral judgments, especially in the areas of information dissemination. The morality drift analysis aimed to gain a more comprehensive understanding of the drift in embedded moral opinions across recommendations and to comprehend the moral implications of the bias in YouTube's recommendation algorithm. As was done in the emotional drift analysis phase, the predominant morality at each depth of recommendation from seed to depth 4 was computed and visualized on a line graph. Given the comprehensive nature of the morality assessment phase, the analysis was conducted across video titles and descriptions. The Morality Vice Assessment computed the distribution of moral vices, such as harm, cheating, betrayal, subversion, and degradation, across recommendations.

Morality Vice Assessment on Video Titles and Description. From Fig. 2(a), which illustrates the morality drift analysis of video titles, we observed a significant presence of all vices (harm, cheating, betrayal, subversion, and degradation) at depth 0, representing our seed videos. Harm appears as the most prevalent moral vice. As we move from the seed videos to depth 4, there is a significant decline in all moral vices, with a plateau observed from depth 3 to 4. This declining trend in the presence of moral vices across recommendations is

also evident in the video descriptions analysis depicted in Fig. 2(b). However, the decline appears to occur at a slower pace compared to that in the video titles.

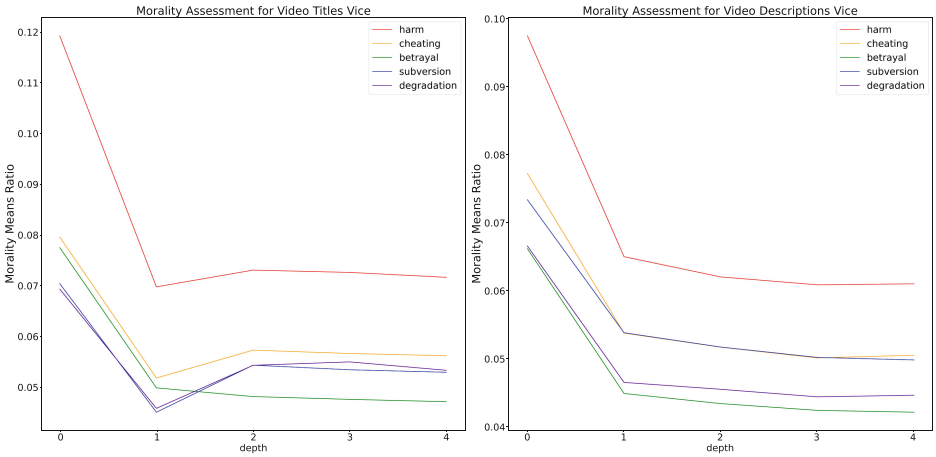


Fig. 2. Line graph showing the distribution of vice morality across recommendations of videos using (a) video titles (b) video descriptions

3.3 Network Analysis

Network analysis was performed on each depth of recommended videos using a recommendation network. For each depth, every video was ranked using its eigenvector centrality measure to determine its influence in the network. Nodes are sized by eigenvector centrality in the graph in Fig. 3. To find the most influential videos, we isolated and analyzed the top five videos with the highest eigenvector centrality score per depth. The mean eigenvector centrality score for the top five videos per depth was found and videos which had an eigenvector centrality score above the resulting mean were filtered out and categorized as 'above-average' influential videos. We infer from the analysis that these 'above-average' influential videos were responsible for driving the recommendations of videos across depths and determined how the conversation across depths evolved.

To ascertain the content divergence of'above-average' videos from our initial seed videos (depth 0), we conducted a content analysis on the most influential videos post-depth 0. This helped us identify latent topics within the recommendations, and each video was subsequently tagged with its respective topic. To visualize the content of these'above-average' influential videos and monitor the progression of dominant topic communities across recommendations, we manually examined the content of the high-influence videos on YouTube (see Table 1.)

In depth 1, the China-Uyghur issue is addressed through the video "Story of the Uyghurs in the City of Kashgar." However, the presence of unrelated

content, like the "Learn Hebrew Alphabet" video, suggests initial diversification in recommendations. In depth 2 and 3, recommendations drift notably from the China-Uyghur topic. The focus shifts to financial transactions and the figure "Ida Dayek". This aligns with the research's observation about changing themes in deeper recommendations. In depth 4, the absence of China-Uyghur-related content continues, indicating further content diversification with topics like space exploration and politics.

To sum up, the persistence of the "Strange transaction at Ministry of Finance" video across depths is intriguing and might signal how recommendation algorithms may persist on certain topics, potentially diverting users away from the original search or interest. The Table 1 acts as empirical evidence for the research's primary claim: YouTube's recommendation system potentially drifts from morally complex and emotionally charged subjects, leading users down diverging paths and potentially shaping their perceptions and beliefs in the process. This makes us realize just how tricky and subtle these recommendation algorithms can be, especially when we're diving into sensitive or emotionally intense topics. Knowing about these shifts helps us all be smarter and more mindful viewers.

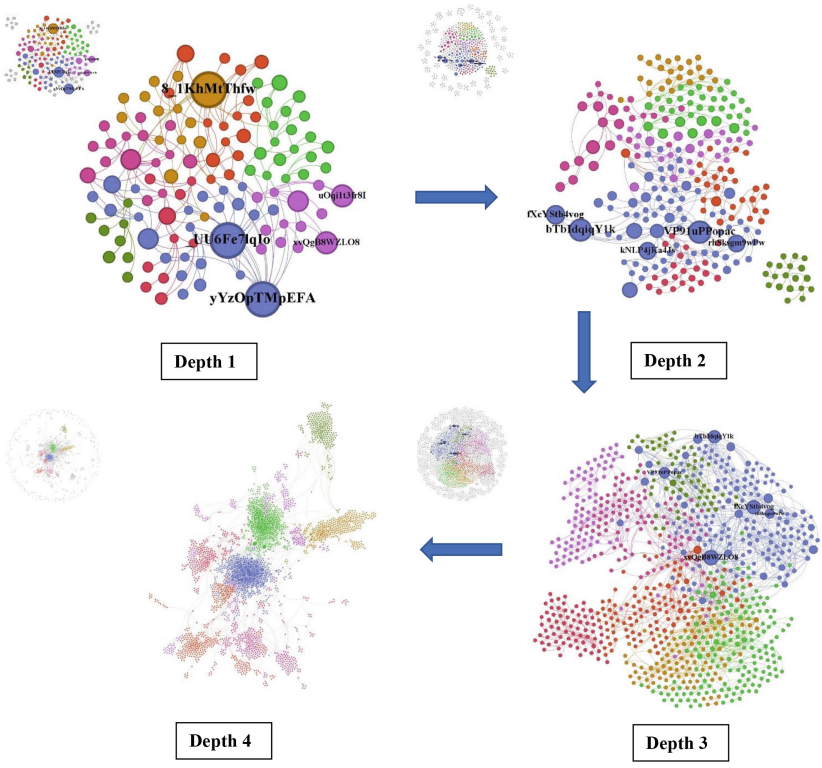


Fig. 3. Recommendation network across depths for China-Uyghur Crisis Dataset

Table 1. Influential videos in each depth of the China-Uyghur narrative and their respective topics.

Depth	VideoID	Eigen value	Topic
1	_UU6Fe7lqIo	1	Learn Hebrew Alphabet
	8_1KhMtThfw	1	Chinese aircraft,ships near Taiwan
	yYzOpTMpEFA	0.988	Story of the Uyghurs in the City of Kashgar
	uOqi1t3fr8I	0.506	Finding middle ground in form of grand coalition
	xvQgB8WZLO8	0.506	Strange transaction at Ministry of Finance
2	VP91uPPopac	1	Hot Debate - Ida Dayek, alternative medicine
	bTbIdqiqY1k	0.993	Phenomenon of Ida Dayak makes eyes widen
	fXcYStb4vog	0.784	Suspicious Transactions Ministry of Finance
	kNLP4jKa4Js	0.686	Explanation of Transactions of 349IDR Trillion
	rhSksgm9wPw	0.686	DPR Discuss Transactions of 349IDR Trillion
3	xvQgB8WZLO8	1	Strange transaction at Ministry of Finance
	fXcYStb4vog	0.902	Suspicious Transactions Ministry of Finance
	bTbIdqiqY1k	0.802	Phenomenon of Ida Dayak makes eyes widen
	VP91uPPopac	0.648	Hot Debate - Ida Dayek, alternative medicine
	rhSksgm9wPw	0.519	DPR Discuss Transactions of 349IDR Trillion
4	xvQgB8WZLO8	1	Strange transaction at Ministry of Finance
	oJDctxlbRi8	0.729	'Grift scandal'-Clarence Thomas US Justice
	dC1-qgR7YO0	0.658	NASA's James Webb Space Telescope
	tq-t1fH0pew	0.638	KPK Chairman Controversy,Docs leak To Wealth
	D-1OmpU5mkU	0.583	Ferdy Sambo Freeing from the Death Penalty

4 Discussion and Conclusion

In this research, we collected videos from four recommendation stages, using relevant seed videos related to the China-Uyghur crisis. On collection of our data, emotional analysis and morality assessment were conducted to determine the nature and impact of YouTube's recommendation algorithm as it relates to the China-Uyghur narrative.

Results from our emotional analysis showed that there was an emergence of positive emotions and a decline of negative emotions as we progressed through recommended videos. This emotion pattern of drift suggests that as the algorithm encounters videos related to the China-Uyghur narrative, the algorithm reduces the recommendation of videos expressing negative emotions while increasing recommendations of videos with positive emotions. On assessing the distribution of morality across recommendations, we see that morality vice assessment of video titles and description showed that all vices were significantly expressed in our seed videos, but as more videos were recommended by the algorithm, these vices continually reduced till they reached minimum levels at depth 4. Our findings suggest that our seed videos had high amounts of negative emotions and vices. As more videos are recommended by the algorithm

from the China-Uyghur seed videos, we see an increase in positive emotions and a decrease in moral vices.

Adding to this, our network analysis, utilizing eigenvector centrality, highlighted how influential videos, over time, changed direction from the China Uyghur topic. This suggests the recommendation system might not just be responding to content but possibly to other factors like popularity. This combined drift, in emotion and content, offers insight into the workings of YouTube's recommendation system, illustrating its tendency to shift users towards broader or more prevalent content themes.

Acknowledgements. This research is funded in part by the U.S. National Science Foundation (OIA-1946391, OIA-1920920, IIS-1636933, ACI-1429160, and IIS-1110868), U.S. Office of the Under Secretary of Defense for Research and Engineering (FA9550-22-1-0332), U.S. Office of Naval Research (N00014-10-1-0091, N00014-14-1-0489, N00014-15-P-1187, N00014-16-1-2016, N00014 -16-1-2412, N00014-17-1-2675, N00014-17-1-2605, N68335-19-C-0359, N00014 -19-1-2336, N68335-20-C-0540, N00014-21-1-2121, N00014-21-1-2765, N00014 -22-1-2318), U.S. Air Force Research Laboratory, U.S. Army Research Office (W911NF-20-1-0262, W911NF-16-1-0189, W911NF-23-1-0011), U.S. Defense Advanced Research Projects Agency (W31P4Q-17-C-0059), Arkansas Research Alliance, the Jerry L. Maulden/Entergy Endowment at the University of Arkansas at Little Rock, and the Australian Department of Defense Strategic Policy Grants Program (SPGP) (award number: 2020-106-094). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations. The researchers gratefully acknowledge the support.

References

1. Shivhare, S.N., Khethawat, S.: Emotion detection from text (2012). https://doi.org/10.48550/arXiv.1205.4944
2. Liu, Q., Huang, H., Feng, C.: Micro-blog post topic drift detection based on LDA model. In: Behavior and Social Computing, Cham, pp. 106–118 (2013)
3. O'Hare, N., et al.: Topic-dependent sentiment analysis of financial blogs. In: Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, New York, NY, USA, pp. 9–16 (2009). https://doi.org/10.1145/1651461.1651464.
4. Suhasini, M., Badugu, S.: Two step approach for emotion detection on twitter data. Int. J. Comput. Appl. **179**, 12–19 (2018). https://doi.org/10.5120/ijca2018917350
5. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
6. Hopp, F.R., Fisher, J.T., Cornell, D., Huskey, R., Weber, R.: The extended Moral Foundations Dictionary (eMFD): development and applications of a crowd-sourced approach to extracting moral intuitions from text. Behav. Res. **53**(1), 232–246 (2021). https://doi.org/10.3758/s13428-020-01433-0
7. Agarwal, A., Zaitsev, I., Wang, X., Li, C., Najork, M., Joachims, T.: Estimating position bias without intrusive interventions. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 474–482 (2019). https://doi.org/10.1145/3289600.3291017.

8. China's Repression of Uyghurs in Xinjiang — Council on Foreign Relations. https://www.cfr.org/backgrounder/china-xinjiang-uyghurs-muslims-repression-genocide-human-rights. Accessed 09 Jan 2023
9. Faddoul, M., Chaslot, G., Farid, H.: A longitudinal analysis of youtube's promotion of conspiracy videos (2020). https://doi.org/10.48550/arXiv.2003.03318.
10. Silverman, C.: This analysis shows how viral fake election news stories outperformed real news on facebook.' BuzzFeed News. https://www.buzzfeednews.com/article/craigsilverman/viral-fake-election-news-outperformed-real-news-on-facebook. Accessed 09 Jan 2023
11. Kitchens, B., Johnson, S.L., Gray, P.: Understanding echo chambers and filter bubbles: the impact of social media on diversification and partisan shifts in news consumption (2020). https://misq.umn.edu/understanding-echo-chambers-and-filter-bubbles-the-impact-of-social-media-on-diversification-and-partisan-shifts-in-news-consumption.html. Accessed 09 Jan 2023
12. Wang, X., Golbandi, N., Bendersky, M., Metzler, D., Najork, M.: Position bias estimation for unbiased learning to rank in personal search. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, in WSDM 2018, pp. 610–618. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3159652.3159732
13. Haidt, J.: The new synthesis in moral psychology. Science **316**(5827), 998–1002 (2007). https://doi.org/10.1126/science.1137651
14. Ovaisi, Z., Ahsan, R., Zhang, Y., Vasilaky, K., Zheleva, E.: Correcting for selection bias in learning-to-rank systems. In: Proceedings of The Web Conference 2020, pp. 1863–1873 (2020). https://doi.org/10.1145/3366423.3380255.
15. Abdollahpouri, H., Burke, R., Mobasher, B.: Controlling popularity bias in learning-to-rank recommendation. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, in RecSys 2017, pp. 42–46. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3109859.3109912
16. Cañamares, R., Castells, P.: Should i follow the crowd? a probabilistic analysis of the effectiveness of popularity in recommender systems. In: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, in SIGIR 2018, pp. 415–424. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3209978.3210014
17. Chaney, A.J.B., Stewart, B.M., Engelhardt, B.E.: How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In: Proceedings of the 12th ACM Conference on Recommender Systems, in RecSys 2018, pp. 224–232. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3240323.3240370
18. Hosanagar, K., Fleder, D., Lee, D., Buja, A.: Will the global village fracture into tribes? recommender systems and their effects on consumer fragmentation. Manage. Sci. **60**(4), 805–823 (2014). https://doi.org/10.1287/mnsc.2013.1808
19. Hartmann, J.: Emotion English DistilRoBERTa-base (2022). https://huggingface.co/j-hartmann/emotion-english-distilroberta-base/
20. Mood of India During Covid-19 - An Interactive Web Portal Based on Emotion Analysis of Twitter Data — Conference Companion Publication of the 2020 on Computer Supported Cooperative Work and Social Computing. https://dl.acm.org/doi/10.1145/3406865.3418567. Accessed 02 June 2023
21. Vo, B.-K.H, Collier, N.I.G.E.L.: Twitter emotion analysis in earthquake situations. Int. J. Comput. Linguist. Appl. **4**(1), 159–173 (2013)

22. Xu, D., Tian, Z., Lai, R., Kong, X., Tan, Z., Shi, W.: Deep learning based emotion analysis of microblog texts. Inf. Fusion **64**, 1–11 (2020). https://doi.org/10.1016/j.inffus.2020.06.002
23. Jamdar, A., Abraham, J., Khanna, K., Dubey, R.: Emotion analysis of songs based on lyrical and audio features. IJAIA **6**(3), 35–50 (2015). https://doi.org/10.5121/ijaia.2015.6304
24. GitHub - medianeuroscience/emfd: The Extended Moral Foundations Dictionary (E-MFD). https://github.com/medianeuroscience/emfd. Accessed 04 June 2023
25. Okeke, O.I., Cakmak, M.C., Spann, B., Agarwal, N.: Examining content and emotion bias in youtube's recommendation algorithm. In the Ninth International Conference on Human and Social Analytics, Barcelona, Spain (2023)
26. Banjo, D. S., Trimmingham, C., Yousefi, N., Agarwal, N.: Multimodal characterization of emotion within multimedia space (2022)
27. Shaik, M., Hussain, M., Stine, Z., Agarwal, N.: Developing situational awareness from blogosphere: an Australian case study (2021)
28. DiCicco, K., Noor, N. B., Yousefi, N., Maleki, M., Spann, B., Agarwal, N.: Toxicity and Networks of COVID-19 discourse communities: a tale of two social media platforms. In: Proceedings (2020). http://ceur-ws.org. ISSN, 1613, 0073
29. Maharani, W., Gozali, A.A.: Degree centrality and eigenvector centrality in twitter. In: 2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA), pp. 1–5. IEEE (2014)
30. Kirdemir, B., Agarwal, N.: Exploring bias and information bubbles in youtube's video recommendation networks. In: Benito, R.M., Cherifi, C., Cherifi, H., Moro, E., Rocha, L.M., Sales-Pardo, M. (eds.) COMPLEX NETWORKS 2021, vol. 1073, pp. 166–177. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-030-93413-2_15
31. Kirdemir, B., Kready, J., Mead, E., Hussain, M.N., Agarwal, N.: Examining video recommendation bias on YouTube. In: Boratto, L., Faralli, S., Marras, M., Stilo, G. (eds.) BIAS 2021, pp. 106–116. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78818-6_10
32. Kirdemir, B., Kready, J., Mead, E., Hussain, M.N., Agarwal, N., Adjeroh, D.: Assessing bias in YouTube's video recommendation algorithm in a cross-lingual and cross-topical context. In Social, Cultural, and Behavioral Modeling: 14th International Conference, SBP-BRiMS 2021, Virtual Event, 6–9 July 2021, Proceedings 14, pp. 71–80. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-78818-6_10

Improving Low-Latency Mono-Channel Speech Enhancement by Compensation Windows in STFT Analysis

Minh N. Bui[1(✉)], Dung N. Tran[2], Kazuhito Koishida[2], Trac D. Tran[1], and Peter Chin[3]

[1] Johns Hopkins University, Baltimore, MD 21211, USA
minhbui@jhu.edu
[2] Microsoft Corporation, Redmond, WA 98052, USA
[3] Dartmouth College, Hanover, NH 03755, USA

Abstract. Speech enhancement is a key component in voice communication technology as it serves as an important pre-processing step for systems such as acoustic echo cancellation, speech separation, speech conversions, etc. A low-latency speech enhancement algorithm is desirable since long latency means delaying the entire system's response. In STFT-based systems, reducing algorithmic latency by using smaller STFT window sizes leads to significant degradation in speech quality. By introducing a simple additional compensation window along with the original short main window in the analysis step of STFT, we preserve signal quality – comparable to that of the original high latency system while reducing the algorithmic latency from 42 ms to 5 ms. Experiments on the full-band VCD dataset and a large full-band Microsoft's internal dataset show the effectiveness of the proposed method.

Keywords: Speech enhancement · algorithmic latency · STFT · compensation window · asymmetric Hann

1 Introduction

Speech enhancement (SE) algorithms recover clean speech from a mixture of speech and noise. It is a key component in audio communication technology stacks, serving as a crucial pre-processing step for down-stream tasks such as automatic speech recognition (ASR) [6,8], acoustic echo cancellation [16], etc. Besides speech quality, low latency is among the most important desirable properties of a SE system, especially in real-time applications such as voice-video teleconferencing or hearing aids [4,5].

Traditional SE algorithms include filter-bank and statistics-based models such as filtering, spectral subtraction, and optimally log-spectral amplitude estimator [14]. These methods, however, are prone to complex noise environments and often inadequate for the current demand of SE. Recent advances in deep

Work performed while Minh N. Bui was an research intern at Microsoft.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 363–373, 2024.
https://doi.org/10.1007/978-3-031-53468-3_31

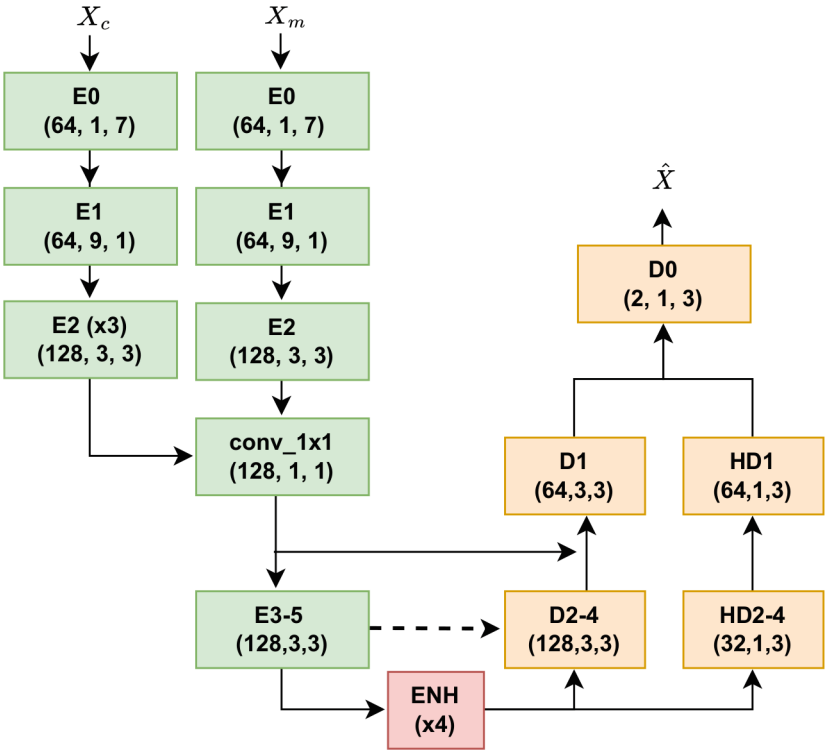


Fig. 1. Overview of our proposed method. E, ENH, D, HD denote the encoder, enhancer, decoder, high-frequency decoder blocks, respectively. The triplet numbers denote number of convolution filters, kernel sizes in temporal axis and frequency axis. All encoder blocks except E0, E1 have stride (1,2). All decoder blocks except D0 has PixelShuffle upsampling with factor (1,2).

learning [10,12,19] have made significant progress in the field and have been preferable to classical SE methods.

Deep learning-based SE methods can be categorized into temporal-based and spectral-based methods. The former operates directly on waveforms whereas the latter takes a transformation of the original waveform (typically short-time Fourier Transform - STFT) as input. Spectral-based methods require significantly more time samples to obtain a balanced trade-off in time-frequency tiling. As a result, spectral-based approaches produce better prediction accuracy at the cost of higher latency compared to temporal-based approaches. Recent spectral-based methods have latency in the regime of 40 ms [7,20,21], much higher than the typical value on temporal methods. Our work focuses on improving the latency of spectral-based methods while preserving their superior predictive performance.

In spectral-based methods, algorithmic latency relies heavily on STFT windows design (see Sect. 2.1). To modify these windows, certain perfect reconstruction (PR) constraints [1] must be satisfied. Recent works have successfully

reduced the algorithmic latency by modifying synthesis window and analysis window focused on satisfying these constraints [15,18].

This work proposes a compensation windowing scheme to reduce the latency of SE systems while preserving their prediction accuracy. To the best of our knowledge, this is the first to reduce the latency of spectral-based methods in supervised full-band, e.g., 48 kHz sampling rate, SE tasks. Furthermore, unlike previous works that focus on designing complicated PR window pairs, we keep the same main processing window pair short and simply add an additional longer compensation window along with a proper deep neural network (DNN) architecture. Experimental results on the VoiceBank-DEMAND (VCD) [13] and a Microsoft internal dataset show that our proposed approach achieves better SE quality with lower latency compared with state-of-the-art methods.

2 Proposed Method

2.1 Spectral Methods: Pipeline and Algorithmic Latency

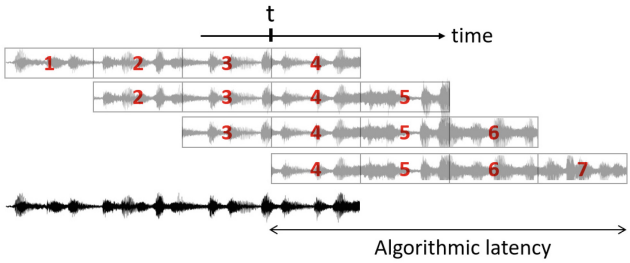
We use STFT/iSTFT as the analysis/synthesis transformation. The pipeline includes analysis step, enhancement step and synthesis step. In analysis, an STFT operation transforms an input signal $\mathbf{x} \in \mathbb{R}^d$ to a T-F spectrogram $X \in \mathbb{R}^{T \times F \times 2}$. In enhancement, a preprocessed X is then fed into a deep neural network $\mathcal{T}$ to produce the enhanced spectrogram $\hat{X} = \mathcal{T}(X) \in \mathbb{R}^{T' \times F \times 2}$ (either via means of predicting complex spectral mask or via direct spectral values). Finally, in synthesis, iSTFT recovers the enhanced 1D signal $\hat{x}$ from $\hat{X}$. In the synthesis process, neighboring frames will produce overlapping 1D signals, and the signals will be summed over overlapping regions to form the final signals. This algorithmic latency induced by the synthesis operation is equal to the duration of the STFT window used in the iSTFT and is illustrated in Fig. 2.

2.2 Compensation Windows in Analysis

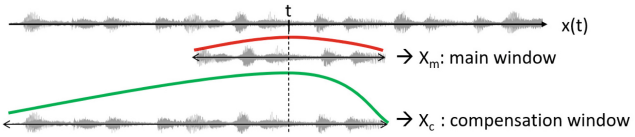
A straightforward approach to reduce the algorithmic latency is to shorten the STFT window. However, this leads to a lower frequency resolution in the T-F spectrogram and hampers the prediction accuracy. To overcome this shortcoming, along with the main window in a typical single-analysis-single-synthesis-window setup [15,18], we introduce a simple compensation window to compensate for the loss of frequency resolution in the analysis step. In other words, we utilize multiple windows to extract information temporally in the analysis phase but still only employ one single conventional window for synthesizing the output.

More specifically, our compensation window does not require any extra look ahead compared to the main window. Formally, the main STFT and compensation STFT can be computed as follows:

$$X_m(t) = \mathrm{DFT}(x[t - F_m/2 : t + F_m/2] * w_m),$$
$$X_c(t) = \mathrm{DFT}(x[t - F_c/2 - \Delta : t + F_c/2 - \Delta] * w_c),$$



(a) Algorithmic latency induced by overlap-add



(b) Main and compensation window setup in analysis STFT

Fig. 2. Example of algorithmic latency for STFT with window size of 4 ms and hop size of 1 ms and illustration of main and compensation window. (a). At time t, the information for the beginning of the 4^{th} block is available (gray signal) but to produce the corresponding output wave (black signal), the algorithm will need to wait until receiving complete the 7^{th} block for overlap-add. (b) Compensation window (green) looks further into the past when compared to the main window (red) depicted with their Hann windowing functions.

where $\{X_m, F_m, w_m\}$ and $\{X_c, F_c, w_c\}$ are the STFT frame, the window size and the windowing function of the main window and the compensation window, respectively. Also, $\Delta = \frac{F_c - F_m}{2}$. Intuitively, X_c is the transformation of the right-shifted version, by an amount of Δ, of x. The compensation hop size is always set equal to the main hop size. The set (X_m, X_c) is used as the input of the enhancement algorithm.

The benefits of our proposal are threefold. First, as the main analysis-synthesis window pair is kept intact, PR is always guaranteed without complicated window designing schemes as in previous works [18] [15]. Second, the compensation window thus can ignore PR constraints, allowing downstream feature learning to focus more on learning useful representations for the prediction task. Finally, the transformation for the compensation window is versatile: one can replace STFT with equivalent transformations.

2.3 Choice of Windowing Function

Following previous works [18] on designing an asymmetric pair of analysis-synthesis windows, we use the asymmetric Hann window (the green curve in Fig. 2) as a compensation STFT windowing function. Mathematically, this asymmetric Hann window is parameterized by F_c and F_w and can be expressed as

$$w_c[n] = \begin{cases} \sqrt{H_{2(F_c-F_m)}}[n] & \text{if } n \in [0, F_c - F_m), \\ \sqrt{H_{2F_m}}[n - (F_c - 2F_m)] & \text{if } n \in [F_c - F_m, F_c), \\ 0 & \text{otherwise.} \end{cases}$$

Our experimental results show that the asymmetric Hann window function produces better results than the regular Hann window. Since PR is satisfied as we keep the regular pair of analysis/synthetic windows, our choice for the compensation window is agnostic of windowing functions or transformation, thus any windowing designs (Square, Turkey, etc.) can be applied. Therefore, we choose the asymmetric Hann window and focus on studying the compensation scheme. We refer the readers to previous work [15,18] for a detailed study of different window types.

2.4 Multi Encoder Deep Neural Network for Low Latency Deep Noise Suppression

We use a UNet-based model with causal convolution and the network directly regresses enhanced spectrogram given two input spectrograms: one from the main window and another from the compensation window. The DNN has three main components: encoder, and enhancer, and decoder. The encoder comprises two different encoder heads corresponding to the main window and the compensation window. The outputs of these heads are fused by 1×1 convolution before being fed into several shared encoder blocks. The enhancer includes 4 enhancement blocks, each comprises 4 sequential ResNet blocks. The decoder consists of two separate branches, both of which can process the same output of the enhancer to produce the lower and higher frequency parts of the output, respectively. The output of the decoders is concatenated to form an enhanced spectrogram. Each encoder-decoder block includes a convolution layer, a LeakyRELU layer with negative slope of 0.2 and a Batch Norm layer. Such architecture is depicted in Fig. 1.

Let $\phi(.) : \mathbb{C}^{T \times (\frac{F}{2}+1)} \rightarrow \mathbb{C}^{T \times (\frac{F}{2}+1)}$ be a preprocessing function defined as $\phi(X) = (|X|^{\frac{1}{3}}/|X|)X$. Let $X_c \in \mathbb{R}^{T \times F_c/4 \times 2}, X_m \in \mathbb{R}^{T \times F_m/4 \times 2}$ be the processed spectrograms of the main and compensation window after applying ϕ, removing half of higher frequency bins and concatenate real and imaginary parts to form a third dimension. The network produces $\hat{X} \in \mathbb{R}^{T' \times F_m/2 \times 2}$ spectrogram.

For the loss function L, we employ the consistency projection loss as in [17] [2]. Let $X, \hat{X}$ be the ground truth and enhanced spectrogram (unprocessed by ϕ). Let $\gamma_{w,l}(.)$ be the STFT transformation parameterized by window size w and hop size l. Then,

$$\begin{aligned} L(X, \hat{X}) &= L_{spectral} + L_{magnitude} \\ &= |||\phi(\gamma_{F_L,H_L}(\gamma^{-1}_{F_m,H_m}(\hat{X}))) - \phi(\gamma_{F_L,H_L}(\gamma^{-1}_{F_m,H_m}(X)))||| \\ &+ |||\phi(\gamma_{F_L,H_L}(\gamma^{-1}_{F_m,H_m}(\hat{X})))| - |\phi(\gamma_{F_L,H_L}(\gamma^{-1}_{F_m,H_m}(X)))|||. \end{aligned}$$

3 Experiments

3.1 Datasets and Metrics

We train and test the networks using two datasets including VCD and the Microsoft dataset derived from the DNS challenge [4] dataset. Both are full-band 48 kHz. Both are speech datasets that are often used for benchmarking noise suppression systems.

The VCD dataset includes 11572 noisy-clean pairs for training and 824 pairs for testing. We pre-process the audio samples by zero-padding at the beginning of the signals so that the minimum duration is 4 s.

The Microsoft dataset is derived from [4] by data augmentation. This dataset includes 1000 h, 360,000 noisy-clean pairs for training. Each noisy or clean sample is 10 s long. The test set includes 560 noisy-clean multilingual pairs for testing, each last 17–21 s.

For evaluation, we ignore the first 3 s of each audio (since convergence time is 2.2 s) and evaluate the rest. Since PESQ score only operates on signals under 16 kHz [9], we downsample the enhanced outputs to 16 kHz and compute the PESQ score for each output given the corresponding target signal. We also employ STOI [11], and DNSMOS [3] includes background (BAK), signal (SIG) and overal (OVL) scores.

3.2 Training and Model Configurations

We use step-wise Adam Optimizer with an initial learning rate of $1e^{-4}$, which decays by 10 at epochs 10 and 30 for a total of 80 epochs for the Microsoft dataset and epochs 50 and 150 for a total of 200 epochs for VCD. We loop through the entire dataset 4 times per epoch to ensure enough coverage of the training data. The training time for each experiment is 15 h and 70 h for VCD and Microsoft data, respectively, on 3 NVIDIA TITAN RTX GPUs.

Since we have multiple different input shapes throughout different latency, the number of encoders and decoders also needs to change accordingly. We maintain the number of encoders/decoders so that the frequency dimension of the last encoder's output is 8. The model is trained on 16-bit precision. We set up $T = 226, T' = 66, F_L = 2048, H_L = 480$. F_m, H_m and F_c, H_c are the sizes/hop sizes of the main and compensation windows, respectively. When computing L_s, we only process the first 256 frequency bins and ignore the rest as they provide little information for speech.

3.3 Comparison of Different Windowing Strategies Under Different Latency

We measure the speech enhancement quality under different latency and analysis window setups. We consider four different latency values: 42 ms, 21 ms, 10 ms, and 5 ms. In all experiments, the compensation window size is 2048. Table 1 and Table 5 show the results for the VCD and Microsoft datasets. The

results support our aforementioned insights and claims. First and most important, even if shrinking the window size reduces the latency and sacrifices speech enhancement quality (e.g. line 6 compares to line 1), the usage of compensation window makes the speech quality still comparable to those produced by high latency setting (e.g. line 5, 8, 11 compares to line 1). Second, using the compensation window only significantly outperforms the baseline that uses the main short window. Third, utilizing an asymmetric Hann window for compensation yields better prediction than the regular Hann window. Finally, combining both the main window and the compensation window produces the best prediction quality (line 5, line 8). We also observe that in the 5 ms case, using both the main window and compensation window does not give best results in all metrics (line 10,11), we believe the reason is when the main window becomes too small, it may not offer significant useful information, hence a better design is needed for under 5 ms systems.

Table 1. Performance on VCD test set. W, H, M, C, Lat, (R) denotes main window size, hop size, main window used, compensation window used, latency, and regular Hann window (symmetric), respectively.

#	W	H	M	C	STOI	PESQ	BAK	SIG	OVL	Lat (ms)
1	2048	512	✓		0.9640	2.9251	3.8769	3.3360	2.9866	42
2	1024	512	✓		0.9628	2.9129	3.8774	3.3518	3.0007	21
3	1024	512	✓	✓(R)	0.9633	2.9915	3.8824	3.3453	2.9965	21
4	1024	512		✓	**0.9643**	2.8759	3.8837	**3.3584**	3.0094	21
5	1024	512	✓	✓	0.9632	**2.9945**	**3.8926**	3.3568	**3.0110**	21
6	512	256	✓		0.9620	2.8823	3.8588	3.3444	2.9872	10
7	512	256		✓	0.9633	2.9267	3.8765	3.3564	3.0055	10
8	512	256	✓	✓	**0.9637**	**2.9543**	**3.8798**	**3.3579**	**3.0076**	10
9	256	128	✓		0.9573	2.7233	3.8623	3.3138	2.9633	5
10	256	128		✓	0.9621	**2.8581**	3.8737	**3.3562**	**3.0041**	5
11	256	128	✓	✓	**0.9628**	2.8263	**3.8744**	3.3554	3.0035	5

4 Ablation Study

We conduct ablation study to better understand the impact of the compensation window. Table 2 shows the SE quality for several compensation window sizes. We keep the baseline main window size as 512 and vary the compensation window sizes from 512 to 4096 and observe that the 2048 window size achieves the best performance. Note that a too long window (4096) degrades the signal quality, as it includes unrelated information from the past and also increases network complexity.

Table 3 shows how increasing the number of compensation windows in addition to the main window influences the enhancement performance. For the two

Table 2. Comparison of different compensation window length on VCD dataset. $\|C\|$ denotes the length of compensation windows.

#	W	H	M	$\|C\|$	STOI	PESQ	BAK	SIG	OVL	Lat (ms)
1	512	256	✓		0.9620	2.8823	3.8588	3.3444	2.9872	10
2	512	256	✓	512	0.9588	2.8296	3.8709	3.3197	2.9734	10
3	512	256	✓	1024	0.9595	2.8291	3.8584	3.3246	2.9703	10
4	512	256	✓	2048	**0.9637**	**2.9543**	**3.8798**	**3.3579**	**3.0076**	10
5	512	256	✓	4096	0.9591	2.8064	3.8567	3.3177	2.9640	10

Table 3. Comparison on number of compensation windows. $\#C$ denotes number of compensation windows on VCD dataset. $\|C\|$ denotes the length of compensation windows.

#	W	H	#C	$\|C\|$	STOI	PESQ	BAK	SIG	OVL	Lat (ms)
1	512	256	0		0.9620	2.8823	3.8588	3.3444	2.9872	10
2	512	256	1	2048	0.9637	2.9543	3.8798	3.3579	3.0076	10
3	512	256	2	1024, 2048	**0.9640**	**2.9641**	**3.8878**	**3.3569**	**3.0102**	10

Table 4. Comparison of different fusion types on VCD dataset.

#	W	H	Type	STOI	PESQ	BAK	SIG	OVL	Lat (ms)
1	512	256	Proposed	**0.9637**	**2.9543**	**3.8798**	**3.3579**	**3.0076**	10
2	512	256	1 Encoder	0.9617	2.8785	3.8791	3.3542	3.0047	10
3	512	256	2 Encoder	0.9627	2.9543	3.8663	3.3466	2.9925	10

compensation window configuration, we simply add another encoder branch corresponding to that window. We observe that using more windows improves the performance. This suggests that at the cost of increasing network complexity, increasing different number of signal representations in encoder improves the denoising quality and hence the compensation windows provide useful information.

In Table 4, we justify the network design choice to fuse the main and compensation window by comparing the proposed design to the alternatives: namely, to use only one encoder to process both features from the main window and compensation window after concatenated together (line 2) and use two encoders but fuse the features at an earlier stage (line 3). Through this experiment, we conjecture that by leaving a separate and strong enough feature representation for itself, the compensation window/branch is able to focus on the representation it needs to excel in the SE task.

Table 5. Performance on Microsoft test set.

#	Window size	Hop size	Main window	Compensation window	PESQ	Lat (ms)
0	2048	512	✓		2.4132	42
1	2048	1024	✓		2.3539	42
2	1024	512	✓		2.3242	21
3	1024	512		✓	2.3555	21
4	1024	512	✓	✓	**2.3570**	21
5	512	256	✓		2.1906	10
6	512	256		✓	2.2153	10
7	512	256	✓	✓	**2.2250**	10
8	256	128	✓		2.1222	5
9	256	128		✓	2.1614	5
10	256	128	✓	✓	**2.1676**	5

Lastly, we test the performance of the proposed duel window strategies on a larger dataset in Table 5. Due to certain time constraints, we are able to evaluate PESQ scores on our experiments. The result confirms the proposed approach consistently improves speech quality in low latency settings (line 4,7,10).

5 Conclusions

In this work, we propose a simple approach to enhance the speech enhancement quality for a low-latency speech SE system. Through various experiments on two different datasets, we observe that by simply adding an additional compensation window along with the main window analysis helps improve the speech quality while lowering the latency, possibly pushing it down to 5 ms. Future possible directions include exploring specific window/filter designs for this compensation window and taking advantage of other forms of feature representation in this compensation window.

References

1. Allen, J.: Short term spectral analysis, synthesis, and modification by discrete Fourier transform. IEEE Trans. Acoust. Speech Signal Process. **25**(3), 235–238 (1977). https://doi.org/10.1109/TASSP.1977.1162950
2. Braun, S., Gamper, H., Reddy, C.K.A., Tashev, I.: Towards efficient models for real-time deep noise suppression (2021). https://doi.org/10.48550/ARXIV.2101.09249, https://arxiv.org/abs/2101.09249
3. Dubey, H., et al.: Deep speech enhancement challenge at ICASSP 2023. In: ICASSP (2023)
4. Dubey, H., et al.: ICASSP 2022 deep noise suppression challenge. In: ICASSP (2022)

5. Graetzer, S., et al.: Clarity-2021 challenges: machine learning challenges for advancing hearing aid processing. In: Interspeech (2021)
6. Li, C.Y., Vu, N.T.: Improving speech recognition on noisy speech via speech enhancement with multi-discriminators CycleGAN. In: 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 830–836 (2021). https://api.semanticscholar.org/CorpusID:245123920
7. Li, Q., Gao, F., Guan, H., Ma, K.: Real-time monaural speech enhancement with short-time discrete cosine transform (2021). https://doi.org/10.48550/ARXIV.2102.04629, https://arxiv.org/abs/2102.04629
8. Pandey, A., Liu, C., Wang, Y., Saraf, Y.: Dual application of speech enhancement for automatic speech recognition. In: IEEE Spoken Language Technology Workshop, SLT 2021, Shenzhen, China, 19-22 January 2021, pp. 223–228. IEEE (2021). https://doi.org/10.1109/SLT48900.2021.9383624, https://doi.org/10.1109/SLT48900.2021.9383624
9. Rix, A.W., Beerends, J.G., Hollier, M., Hekstra, A.P.: Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In: 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221), vol. 2, pp. 749–752 (2001). https://api.semanticscholar.org/CorpusID:5325454
10. Schröter, H., Escalante, A.N., Rosenkranz, T., Maier, A.K.: DeepFilternet: a low complexity speech enhancement framework for full-band audio based on deep filtering. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7407–7411 (2021). https://api.semanticscholar.org/CorpusID:238634774
11. Taal, C., Hendriks, R., Heusdens, R., Jensen, J.: A short-time objective intelligibility measure for time-frequency weighted noisy speech, pp. 4214 – 4217 (2010). https://doi.org/10.1109/ICASSP.2010.5495701
12. Taherian, H., Eskimez, S.E., Yoshioka, T., Wang, H., Chen, Z., Huang, X.: One model to enhance them all: array geometry agnostic multi-channel personalized speech enhancement. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 271–275 (2021). https://api.semanticscholar.org/CorpusID:239049883
13. Valentini-Botinhao, C.: Noisy speech database for training speech enhancement algorithms and TTS models (2017)
14. Vihari, S., Murthy, A., Soni, P., Naik, D.: Comparison of speech enhancement algorithms. Procedia Comput. Sci. **89**, 666–676 (2016). https://doi.org/10.1016/j.procs.2016.06.032
15. Wang, Z.Q., Wichern, G., Watanabe, S., Roux, J.L.: STFT-domain neural speech enhancement with very low algorithmic latency. IEEE/ACM Trans. Audio Speech Lang. Process. **31**, 397–410 (2022). https://api.semanticscholar.org/CorpusID:248300088
16. Westhausen, N.L., Meyer, B.T.: Acoustic Echo Cancellation with the Dual-Signal Transformation LSTM Network. In: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7138–7142 (2021). https://doi.org/10.1109/ICASSP39728.2021.9413510
17. Wisdom, S., Hershey, J.R., Wilson, K.W., Thorpe, J., Chinen, M., Patton, B., Saurous, R.A.: Differentiable consistency constraints for improved deep speech enhancement. CoRR **abs/1811.08521** (2018). http://arxiv.org/abs/1811.08521
18. Wood, S.U.N., Rouat, J.: Unsupervised low latency speech enhancement with RT-GCC-NMF. IEEE Journal of Selected Topics in Signal Processing **13**(2), 332–346 (2019). https://doi.org/10.1109/jstsp.2019.2909193

19. Zhang, G., Yu, L., Wang, C., Wei, J.: Multi-scale temporal frequency convolutional network with axial attention for speech enhancement. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 9122–9126 (2022). https://doi.org/10.1109/ICASSP43922.2022.9746610
20. Zhang, Z., Zhang, L., Zhuang, X., Qian, Y., Li, H., Wang, M.: FB-MSTCN: a full-band single-channel speech enhancement method based on multi-scale temporal convolutional network (2022). https://doi.org/10.48550/ARXIV.2203.07684, https://arxiv.org/abs/2203.07684
21. Zhao, S., Ma, B., Watcharasupat, K.N., Gan, W.S.: FRCRN: boosting feature representation using frequency recurrence for monaural speech enhancement (2022). https://doi.org/10.48550/ARXIV.2206.07293, https://arxiv.org/abs/2206.07293

Network Embedding

Filtering Communities in Word Co-Occurrence Networks to Foster the Emergence of Meaning

Anna Béranger, Nicolas Dugué(✉), Simon Guillot, and Thibault Prouteau

Université du Mans, Laboratoire d'Informatique de l'Université du Mans (LIUM), Avenue Olivier Messiaen, 72000 Le Mans, France
{nicolas.dugue,thibault.prouteau}@univ-lemans.fr

Abstract. With `SINr`, we introduced a way to design graph and word embeddings based on community detection. Contrary to deep learning approaches, this approach does not require much compute and was proven to be at the state-of-the-art for interpretability in the context of word embeddings. In this paper, we investigate how filtering communities detected on word co-occurrence networks can improve performances of the approach. Community detection algorithms tend to uncover communities whose size follows a power-law distribution. Naturally, the number of activations per dimensions in `SINr` follows a power-law: a few dimensions are activated by many words, and many dimensions are activated by a few words. By filtering this distribution, removing part of its head and tail, we show improvement on intrinsic evaluation of the embedding while dividing their dimensionality by five. In addition, we show that these results are stable through several runs, thus defining a subset of distinctive features to describe a given corpus.

Keywords: Word co-occurrence networks · community detection · word embedding · linguistics · interpretability

1 Introduction

In the field of Natural Language Processing (NLP), one of the main challenges is to represent the meaning of words into vectors, these vectors then being used as input to classification systems in order to solve various tasks such as part-of-speech tagging, named entity recognition, machine translation, etc. Vectors that represent words are commonly designated as word embeddings: the meaning of words is embedded in a small latent space with dense vectors. Approaches to train such vectors are based on the distributional hypothesis. Harris defines this hypothesis, writing that "linguistic items with similar distributions have similar meanings". To train word embedding, one thus need to estimate these distributions using word co-occurrences from large corpora. The seminal approaches to train word embeddings are actually based on word co-occurrences matrix factorization [16,19,20]. Other popular approaches such as `Word2vec` [18] use neural networks to build lexical representations, thus approximating matrix factorization methods [15]. Finally, transformer-based approaches [17] and large language

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 377–388, 2024.
https://doi.org/10.1007/978-3-031-53468-3_32

models based on these architectures [14] have demonstrated impressive performances, being able to contextualize word representations according to words' occurrences.

`SINr` (Sparse Interpretable Node Representations), the approach we introduced in Prouteau et al. [21] is based on those recent progresses in NLP. However, it pursues an alternative path. Instead of focusing on performance, the method aims to train interpretable word vectors frugally. To do so, it leverages the distributional hypothesis in a rather direct manner, using word co-occurrence networks. Communities are then detected and considered as dimensions of the latent space. The word vector is finally extracted using the connectivity of its representing nodes to the communities extracted, in line with Harris' claim: we consider that words with similar distributions of links through communities have similar meanings. The model has proven its low compute requirement, it is indeed based on the Louvain algorithm [5]. Furthermore, it was shown that dimensions of this model are mostly interpretable, the model being on-par with `SPINE` [23], a competing state-of-the-art approach for interpretability [22]. Indeed, dimensions are not abstract as in conventional approaches: they are the communities uncovered, tangible groups of words [22]. Even though `SINr` does not focus strictly on performance, it is still an important goal. On the similarity task, the approach is on-par with `SPINE`, but slightly under-performs when compared to `Word2vec`.

In this paper, we show how to foster community filtering to significantly improve performances of `SINr`, allowing to catch up with `Word2vec` while preserving its interpretability and low compute properties, even lowering its memory footprint. We first describe `SINr` in more details Sect. 2. In Sect. 3, we analyze dimensions' activations and how they seem related to the distribution of community sizes. Using this analysis, we propose our community filtering method based on dimension activation to improve the model. Then, in Sect. 4, we detail the textual corpora used to train the word embedding approaches considered, and the similarity task that we use to evaluate model performances. We finally detail the results by showing that filtering communities by removing part of the head and tail of the dimensions activations distribution allows a significant improvement in results while reducing models' memory footprint.

2 `SINr`: Interpretable Word Vectors Based on Communities

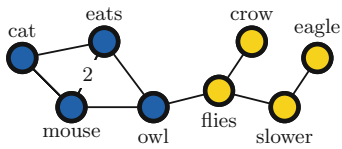
In this section, we first detail the `SINr` approach as we introduced it in [21]. As far as we know, it is the first graph-based approach to train word embeddings, but like other approaches, it is based on the distributional hypothesis, adjusting Harris' formula by claiming that words with similar distributions of links through communities have similar meanings.

We start with an undirected weighted network, the word co-occurrence network, where words are vertices, and edges between nodes represent co-occurrences of words inside a sentence, and at a distance at most of w. Textual corpora were preprocessed to keep only meaningful words (see Fig. 1a). Weights

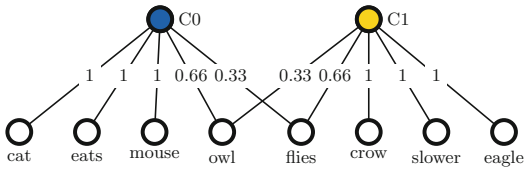
are associated to edges and represent the number of co-occurrences. We then filter our word co-occurrence network by setting to 0 the edges weights whose co-occurrence is not significant, according to the PMI (the ratio of the probability of nodes u and v co-occurring together divided by their probability of occurrence). This is very similar to the construction of a co-occurrence matrix, showing that `SINr` is related to the matrix factorization approaches such as [16,19]. In matrix factorization, the next step is to factorize the matrix to reduce dimensionality and get dense vectors. In `SINr`, we use community detection with the Louvain algorithm to group words together and get the interpretable dimensions of our latent space (see Fig. 1b). We use the multiscale γ parameter [13] and set it to 60 in this paper: it allows uncovering small, thus consistent communities. We eventually calculate the distribution of the weighted degrees of each node through communities to get word embeddings, such as described in Fig. 1c,1d. This distribution is computed according to the Node Recall introduced in [9] and defined as follows. Given a vertex u, a partition of the vertices $\mathcal{C} = \{C_0, \ldots, C_j\}$, and C_i the i_{th} community so that $1 \leq i \leq j$, the node recall of u considering the i^{th} community is : $\text{NR}_i(u) = \frac{d_{C_i}(u)}{d(u)}$ with $d_{C_i}(u) = \sum_{v \in C_i} W_{uv}$. To refine these vectors, we show in [10] that we can keep, for each vector, the 50 highest values, improving interpretability and lightening up the model.

Corpus before preprocessing: *My cat eats mice. Your owl eats mice. This owl flies. A crow flies slower than an eagle.*
After preprocessing: *cat eats mouse. owl eats mouse. owl flies. crow flies slower eagle.*

(a) Text corpus.



(b) A graph $G = (V, E, W)$ partitioned in two communities.



(c) Bipartite projection of G into graph $G\prime = (\top, \bot, E, W)$ along the communities. Weight on the edges is based on NR, the proportion of the weighted degree of each node related to the community.

	C0	*C1*
cat	1	0
eats	1	0
mouse	1	0
owl	0.66	0.33
flies	0.33	0.66
crow	0	1
eagle	0	1
slower	0	1

(d) Adjacency matrix of $G\prime$, each row is a `SINr` embedding.

Fig. 1. `SINr`: words are represented based on the communities they are linked to.

3 SINr-filtered: Sampling Communities Using Activations

First, community sizes tend to follow a power-law distribution [8]. We use the γ multiscale resolution parameter [13] to uncover small consistent communities, but there are still a few communities bigger than the rest as shown in Fig. 2. In this figure, we applied SINr to two corpora, BNC and UkWac which are described in details in Sect. 4.

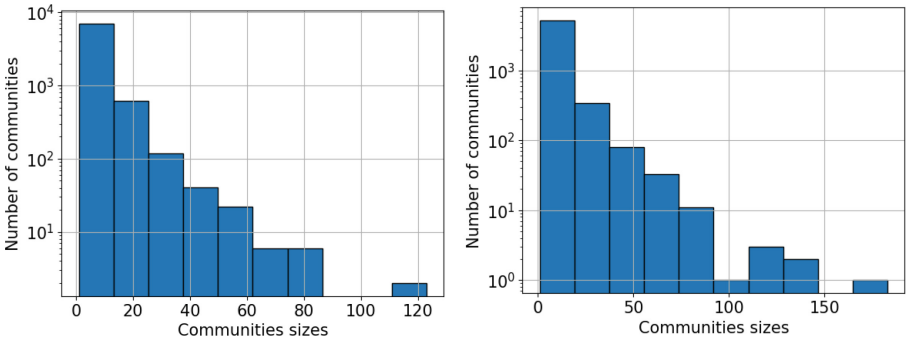


Fig. 2. Distribution of community sizes on BNC (left) and UkWac (right) corpora. The ordinate axis is in logarithmic scale.

Furthermore, linguists have studied the distribution of words occurrences and co-occurrences in corpora. It has been shown that it follows Zipf's law [3], which is consistent with a power-law [1], commonly observed in complex networks. It means that some words would co-occur far more than others with the rest of the vocabulary, and a lot of them may co-occur very few with the rest of the vocabulary. Because communities are made of words following this power-law, and because communities sizes also follow a power-law, some communities may be much more connected with the rest of the graph than others, and some of them may be quite isolated. Let us recall that dimensions of our SINr model correspond to these communities. Thus, if we say that a dimension is activated by a word if this dimension's value is greater than 0 for this word vector, we expect the activation of dimensions by the word vectors to follow a power-law.

As we can see in Fig. 3, the number of activations (non-zero values) follows a distribution that looks like a power-law. This is critical to understand how the model works and how we can improve its performances while reducing its memory footprint.

As stated by Subramanian et al. [23], having dimensions activated by a large part of the vocabulary is not compatible with interpretability, which is a focus of our work: a dimension is interpretable if the set of words that activate it is consistent. Such heavily activated dimensions may be based on communities gathering very frequent words that appear in very different contexts, thus being

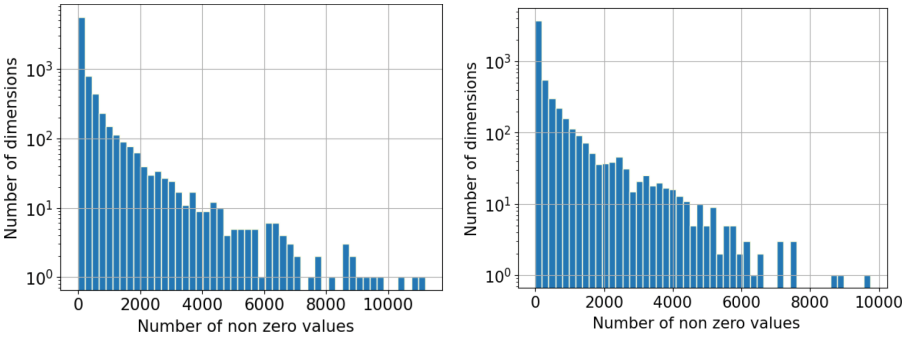


Fig. 3. Distribution of the number of activations per dimension on `BNC` (left) and `UkWac` (right) corpora. The ordinate axis is in logarithmic scale.

not consistent topically. We thus propose to remove these dimensions from the model, and we will see in Sect. 4 that it improves model performances.

We can also see that many dimensions are actually activated by only a few words. While these dimensions may be useful for very specific topics, they may not be useful for most of the vocabulary. They may also be noisy dimensions that penalize performances. We also propose to remove these dimensions from the models, and we will demonstrate in Sect. 4 that it notably reduces the memory footprint of the model (dividing the number of dimensions by 5) while preserving performances.

4 Experiments and Results

4.1 Experimental Setup

Task and Baseline. We consider an intrinsic task to evaluate the performance of embeddings: the similarity task. In this task, we consider five datasets to evaluate our models: `WS353` [2], `MEN` [6], `SCWS` [12] and `SimLex` [11] split in `SimLex999` and `SimLex665`. Each dataset is made up of pairs of words associated with a similarity score that is a mean of ratings given by humans. The first three datasets comprise pairs of words, both representing word *similarity* (approximately synonymy, or at least substitutability, like "cat" and "feline") and word *relatedness* (much broader, encompasses pairs like "cup" and "coffee"). However, datasets differ regarding the parts of speech they include: `WS353` only includes nouns, while `MEN` and `SCWS` include nouns, verbs and adjectives. Lastly, `SCWS` is designed to evaluate contextual representations, scores representing word similarity in different contexts: "bank" can be scored high with both "river" and "money" depending on the phrastic context presented to annotators. `SimLex`, on the other hand, is a dataset specialized on word *similarity*, which is more restrictive regarding the substitutability criterion, and allows for hyperonym-hyponym pairs to be closer than co-hyponyms. For example, in this dataset "father" and "parent" are rated at 7.07 on a scale of 10 regarding similarity, while "dad" and "mother"

are at 3.55. Furthermore, SimLex does not rely on frequency information from a reference corpus to select its word pairs, it thus includes rarer words than WS353, MEN or SCWS. This dataset is of particular interest due to its difficulty, and its split with regard to parts of speech : SimLex999 being the whole dataset with noun-noun, adjective-adjective and verb-verb pairs, and SimLex665 being the noun subset. This split allows us to determine the validity of our modeling on different word categories, which probably follow different distributions of contexts.

Similarities in embedding spaces using cosine similarity are supposed to be correlated with human similarities, as shown in Fig. 4. Correlation is computed with Spearman's definition: the closer to 1, the better. In order to assess the performances of our model, we also consider Word2vec, one of the most popular approaches to train word embeddings. We do not consider more recent state-of-the-art approaches that allow to get better performances because our approach focuses on interpretability and low compute. Eventually, it may be used in more complex architectures, such as transformers,

w_1	w_2	human rating		cosine_sim(w_1, w_2)
tiger	cat	7.35		0.73
plane	car	6.31	Spearman Correlation ×	0.65
drink	mother	2.85		0.20
forest	graveyard	1.85		0.12

Fig. 4. Example of word similarity rating from the MEN dataset and cosine similarity between vectors.

Corpora. We perform our evaluation on two text corpora:

- the written part of BNC [7], a collection of journalistic, academic and fictional texts totaling 100 million tokens;
- UkWac [4], a cleaned crawled corpus from the .uk internet domain with over 2 billion tokens.

Both of these corpora are classic for the similarity tasks. Their size, their content, and the way they are constituted is very different. By reproducing the experiments on such different corpora, we aim to show that the results may be generalized to any collection.

Preprocessing and Models Parameters. The text corpora are preprocessed with spaCy to improve the quality of cooccurrence information and reduce the vocabulary to be covered by the models. The text is tokenized and lemmatized, named entities are chunked, words shorter than three characters, punctuation and numerical characters are deleted. The minimum frequency to represent a type is set at 20 for BNC and 50 for UkWac. All models use a cooccurrence window of 5 words to the left and to the right of a target within sentence boundaries.

Furthermore, as stated in Sect. 2, the multiscale resolution parameter γ is set at 60 for Louvain's community detection in SINr.

SINr-Filtered Approach. We first compute our SINr approach as described in Sect. 2. Then, by considering the distribution of activations per dimension, we explore the removal of the few very activated dimensions, and of the dimensions forming the long tail of this distribution. We explore the similarity performances regarding these removals and, in particular, the threshold used for these removals. The choice of thresholds is guided by performances on the word similarity evaluation task (see Figs. 5, 6).

4.2 Results

Baseline. In order to assess the performance of SINr-filtered, we also provide results for Word2vec, a reference approach which is not interpretable, and SINr, the original approach without filters on the communities. Results are averaged over 10 runs for the whole section. Table 1 show that SINr-filtered is always better than the original version, and that it catches up with Word2vec, even outperforming this baseline on MEN and WS353.

Table 1. Summary of the results of competing models and of SINr and its filtered version, SINr-filtered, introduced in this paper.

	MEN		WS353		SCWS		SimLex999		SimLex665	
	BNC	UkWac	BNC	UkWac	BNC	UkWac	BNC	UkWac	BNC	UkWac
W2V	**.73**	**.75**	.64	.66	**.61**	**.64**	**.28**	**.34**	**.34**	**.37**
SINr	.67	.70	.63	.68	.56	.56	.20	.23	.28	.30
SINr-filtered	.72	**.75**	**.65**	**.70**	.58	.59	.25	.25	.30	.33

We then show the effects of filtering using the distribution of activations on the SINr performance regarding the similarity evaluation.

Filtering the Distribution's Head. As one can see in Fig. 5, model performances varies a lot with regard to the filter threshold. On the right, when the threshold is set to 12 000, there is actually no filtering. Filtering more and more, moving the threshold to the left until the best threshold 4000, allows to gradually increase performances of models for both UkWac and BNC. Between 4000 and 2000, results are rather plateauing. After 2000, significant information is removed, leading to a decrease in performance. Using the 4000 threshold allows catching up with Word2vec's performances, our reference. Indeed, the 4000 filter allows a gain of 5 points in performance for the MEN dataset (from 0.67 to 0.72 for BNC and from 0.70 to 0.75 for UkWac), and a slight gain of 2 points for the WS353 dataset (from 0.63 to 0.65 for BNC, from 0.68 to 0.70 on UkWac) and the SCWS dataset (from 0.56 to

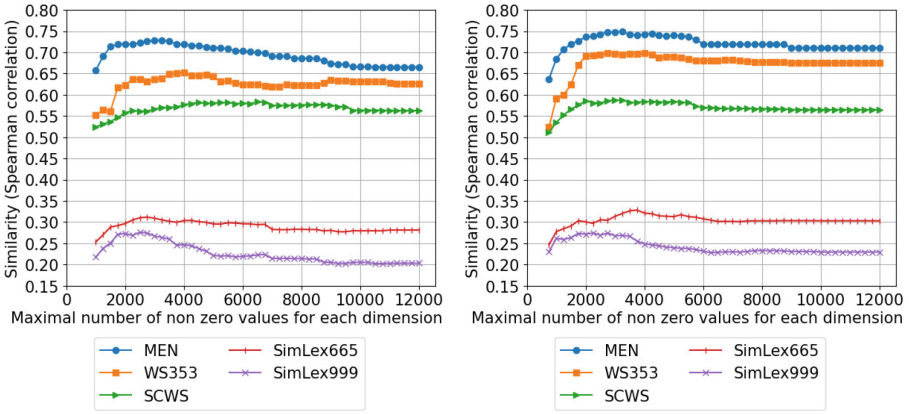


Fig. 5. Similarity on `BNC` (left) and `UkWac` (right) corpora. Dimensions **often** activated are removed according to the threshold in abscissa.

0.58 for `BNC`, from 0.56 to 0.59 for `UkWac`). Such gains are statistically significant, and they are particularly interesting because they result from a simplification of the model, even if only 95 (resp. 90) dimensions are removed in average on the `BNC` (resp. `UkWac`) model using this filter. The `SimLex` dataset is much harder than the three others, for `SINr` but also for the reference model `Word2vec`. However, as one can see, filtering allows significant gain for `SimLex` also, especially for `SimLex999` on `BNC` (from 0.20 to 0.25), and the results are better between 2000 and 4000 thresholds like for the other datasets.

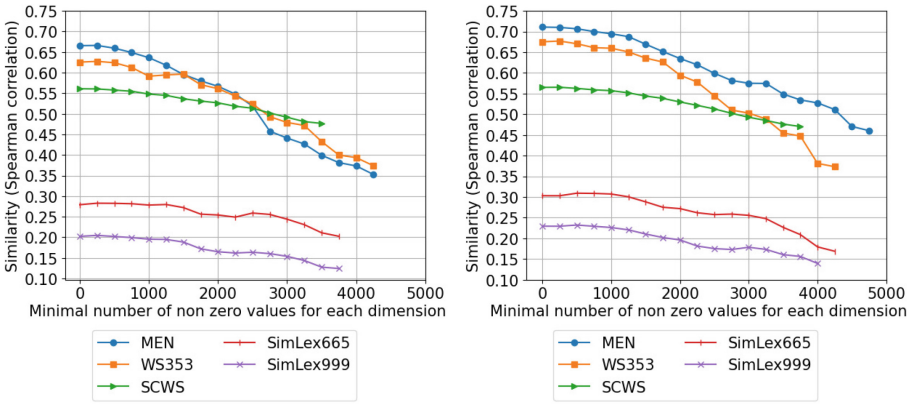


Fig. 6. Similarity on `BNC` (left) and `UkWac` (right) corpora. Dimensions **scarcely** activated are removed according to the threshold in abscissa.

Filtering the Distribution's Long Tail. The effect of filtering the long tail of the distribution of activations is quite different, as one can see in Fig. 6. At left, no filter is applied, and increasing the filter does not lead to any gain in performances. Still, it is interesting to notice that filtering dimensions with less than 500 activations does not lead to any significant loss in information, on the five similarity datasets used for evaluation. Indeed, it actually divides by 5 the number of dimensions of the model, reducing its number of dimensions from roughly 6600 to 1200 on average for `BNC`, and from 5700 to 1100 for `UkWac`, thus allowing to drastically improve its memory footprint!

Is Filtering Dimensions the Same as Filtering Communities? As one can see in Fig. 7, filters (more than 500 and less than 4000) applied using the number of dimensions have an expected effect on community distributions. Here, we assume that when a dimension is removed from the model, its community is also removed. We can see that removing dimensions with more than 4000 activations mostly removes big communities. This is especially the case for `UkWac` where the larger communities, those accounting for more than 150 words, are removed. For `BNC`, removed communities are not the largest, their size ranges from 60 to 80 words mostly. Similarly, removing dimensions with less than 500 activations tend to remove small communities. Still, most of the smallest communities are kept in the model while some larger are removed. These observations show that, even if the number of activations of a dimension and the size of its community are related, using the distribution of activations is different from using the distribution of the community sizes, showing the relevance of our approach.

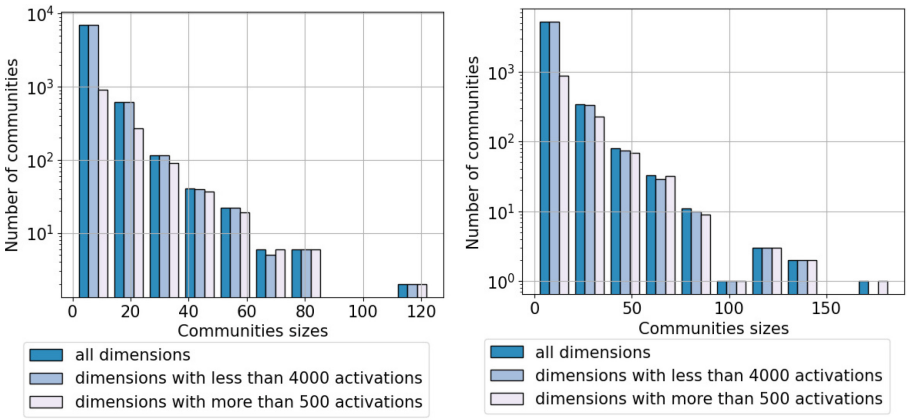


Fig. 7. Filtering effects on the community sizes distribution for `BNC` (left) and `UkWac` (right).

Singling out the Subset of Distinctive Contexts. One may wonder if from one run of `SINr` to another, by filtering dimensions with more than 500 and less than 4000 activations, the vocabulary that forms communities that are kept is the same. It is surprising to notice that it is mostly the case: roughly 80% of the vocabulary kept is actually the same over ten runs when considering `BNC` and `UkWac` separately. However, this set is not the same from one corpus to another, only 35% of the vocabulary kept is actually common to `BNC` and `UkWac`. It seems to mean that these respective subsets of the vocabulary are essential to describe the meaning of words in these respective corpora. Those results, combined to the similarity improvements, point towards the notion that our filtering approach discriminates a subset of the dimensions that is the best fit to describe a given corpus. However, the evaluation results solely give insight on the subset of the lexicon covered by the similarity datasets, a subset that is heavily biased toward nouns, and especially frequent concrete nouns.

5 Conclusion

`SINr` is a graph-based approach to train word embedding which requires low compute and whose results are interpretable. In this paper, we show that we can significantly improve model performances and reduce its memory footprint by filtering its dimensions. Indeed, filtering the most activated dimensions allows gaining a few points on the similarity task for each dataset considered, showing that these dimensions are actually the bearer of noise into the model. This gain allows `SINr` performing on-par with `Word2vec`. Furthermore, filtering-out dimensions that are the least activated allows dividing the number of dimensions by 5 while preserving performances. We show that these filters relying on activations of the dimensions are somehow correlated with community sizes, but not completely, showing their relevance. Finally, we demonstrate that the vocabulary of communities that correspond to dimensions that are not filtered remains the same from one run of `SINr` to the other. We plan to experiment on other corpora but also on downstream tasks to confirm the ability of these results to generalize in a variety of contexts. Furthermore, it would be particularly interesting to test the ability of these filtered embeddings to model the meaning of very specialized vocabulary, to evaluate if removing dimensions affects the representation of these words.

Acknowledgments. The work has been funded by the ANR project DIGING (ANR-21-CE23-0010).

References

1. Adamic, L.: Unzipping Zipf's law. Nature **474**(7350), 164–165 (2011)
2. Agirre,E., Alfonseca,E., Hall,K., Kravalova, J., Paşca, M.: A study on similarity and relatedness using distributional and WordNet-based approaches. In: NAACL, pp. 19–27, (2009)

3. Baroni, M.: 39 distributions in text. In: Corpus Linguistics: An international handbook, vol. 2, pp. 803–822. Mouton de Gruyter (2005)
4. Baroni, M., Bernardini, S., Ferraresi, A., Zanchetta, E.: The wacky wide web: a collection of very large linguistically processed web-crawled corpora. Lang. Resour. Eval. **43**(3), 209–226 (2009)
5. Blondel, V.D., Guillaume,J.-L., Lambiotte,R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech.: Theory Exp., p. P10008 (2008)
6. Bruni, E., Tran, N.K., Baroni, M.: Multimodal distributional semantics. J. Artif. Int. Res. **49**(1), 1–47 (2014)
7. BNC Consortium. British national corpus, XML edition (2007). Oxford Text Archive
8. Dao, V.L., Bothorel, C., Lenca, P.: Community structure: a comparative evaluation of community detection methods. Netw. Sci. **8**(1), 1–41 (2020)
9. Dugué, Nicolas, Lamirel, Jean-Charles., Perez, Anthony: Bringing a feature selection metric from machine learning to complex networks. In: Aiello, Luca Maria, Cherifi, Chantal, Cherifi, Hocine, Lambiotte, Renaud, Lió, Pietro, Rocha, Luis M.. (eds.) COMPLEX NETWORKS 2018. SCI, vol. 813, pp. 107–118. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05414-4_9
10. Guillot, S., Prouteau, T., Dugué, N.: Sparser is better: one step closer to word embedding interpretability. In: International Workshop on Computational Semantics (2023)
11. Hill, F., Reichart, R., Korhonen, A.: SimLex-999: evaluating semantic models with (genuine) similarity estimation. CoRR, abs/1408.3456 (2014)
12. Huang, E., Socher, R., Manning, C., Ng, A.: Improving word representations via global context and multiple word prototypes. In: ACL, pp. 873–882 (2012)
13. Lambiotte, R.: Multi-scale modularity and dynamics in complex networks. In: Mukherjee, A., Choudhury, M., Peruani, F., Ganguly, N., Mitra, B. (eds.) Dynamics On and Of Complex Networks, vol. 2, pp. 125–141. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-6729-8_7
14. Launay, J., et al.: PAGnol: an extra-large french generative model. arXiv preprint arXiv:2110.08554 (2021)
15. Levy, O., Goldberg, Y.: Neural Word Embedding as Implicit Matrix Factorization. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q., (eds.) Advances in Neural Information Processing Systems, vol. 27. Curran Associates, Inc. (2014)
16. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. ACL **3**, 211–225 (2015)
17. Louis Martin, et al.: CamemBERT: a tasty French language model. arXiv:1911.03894 (2019)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR (2013)
19. Murphy, B., Talukdar, P., Mitchell, T.: Learning effective and interpretable semantic models using non-negative sparse embedding. In: COLING, pp. 1933–1950 (2012)
20. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: EMNLP, pp. 1532–1543 (2014)
21. Prouteau, T., et al.: SINr: fast computing of sparse interpretable node representations is not a sin! In: Abreu, P.H., Rodrigues, P.P., Fernández, A., Gama, J. (eds.) IDA 2021. LNCS, vol. 12695, pp. 325–337. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-74251-5_26

22. Prouteau, T., Dugué, N., Camelin, N., Meignier, S.: Are embedding spaces interpretable? Results of an intrusion detection evaluation on a large French corpus. In: LREC (2022)
23. Subramanian, A., Pruthi, D., Jhamtani, H., Berg-Kirkpatrick, T., Hovy, E.: Spine: SParse interpretable neural embeddings. In: AAAI (2018)

DFI-DGCF: A Graph-Based Recommendation Approach for Drug-Food Interactions

Sofia Bourhim(✉)

ENSIAS, Mohammed V University, Rabat, Morocco
sofia.bourhim@um5s.net.ma

Abstract. Drug discovery focuses on understanding different types of interactions from drug-drug interactions (DDIs) to drug-food interactions (DFIs). The main purpose of DFI is to discover how a particular food affects drug absorption, side effects and its effectiveness. The study of drug-food interactions (DFIs) can provide valuable insight into optimizing patient care, adjusting dosages, and improving patient safety.

In this work, we propose a novel workflow where we aim to use a community-based recommender system to infer and identify novel DFIs while incorporating the concept of community profiling and leveraging the power of Graph Neural Networks. We conduct experiments on the DrugBank dataset and use FooDB dataset to learn more about food constituents. Our experiments reveal significant improvements over a number of the latest approaches designed for DFI identification. The findings substantiate that the utilization of multiple graphs to leverage diverse forms of relationships holds the potential to yield better recommendations by extracting complex relationships through the community structure.

Keywords: Drug discovery · Drug-Food Interactions · Graph Neural Networks · Recommender systems · Community profiling

1 Introduction

Drug-food interaction (DFI) refers to the alteration of a drug's pharmacological effect by a food component. DFIs can occur through a variety of mechanisms, including changes in the absorption, distribution, metabolism, and excretion of the drug. DFIs can exert a substantial influence on the safety and effectiveness of drug therapy. In certain instances, DFIs may even result in severe adverse drug reactions. It can occur through a variety of mechanisms that include changes in absorption where food components can interfere with the absorption of drugs from the gut, leading to lower blood levels of the drug. For instance, grapefruit juice can inhibit the absorption of the cholesterol-lowering drug atorvastatin. It also includes the change in distribution where the food components can bind to drugs in the bloodstream, preventing them from reaching their target tissues. For example, calcium can bind to the antibiotic tetracycline, making it less effective against bacteria.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 389–399, 2024.
https://doi.org/10.1007/978-3-031-53468-3_33

Over the past few years, there has been a surge in interest towards developing predictive models for DFIs. These models serve the purpose of detecting potential DFIs, evaluating the associated risks, and devising effective strategies to mitigate such risks. There are a variety of models for predicting DFIs including: In silico models where it uses computer simulations to predict the interactions between drugs and food components, Statistical models, and deep learning models.

Lately, deep learning models applied to DFIs showed interesting results and can learn complex relationships between drug molecules and food components, which allows them to predict new drug-food interactions that would not be detected by other methods. Traditionally, the identification of DFIs has been a difficult task due to the limited availability of data and the complexity of the interactions between drugs and food. However, recent advances in machine learning have made it possible to develop more effective methods for identifying DFIs.

Graphs are a powerful data structure for modeling biomedical systems. They can be used to represent the relationships between different entities, such as drugs, food compounds, and proteins. This makes them well-suited for modeling DFIs, which are interactions between drugs and food compounds. In recent years, there has been a growing interest in using graph embedding methods to predict DFIs. Graph embedding methods learn a low-dimensional representation of each node in a graph while maximally preserving the structural information of the graph. This allows the relationships between nodes to be preserved, even when the nodes are represented in a low-dimensional space. One of the most popular graph embedding methods for DFI prediction is DeepDDI [3]. DeepDDI uses a deep neural network to learn the representations of drugs and food compounds. The model is trained on a dataset of known DFIs, and it can then be used to predict new DFIs.

Contribution. In this paper, we propose a novel workflow for prediction drug-food interactions using a graph-based approach. It entails the creation of the networks and the construction of a full profile of a community' based on the community profiling. In summary, our contributions can be summarized as follows:

- To address the lack of DFI resources, we build a DFI dataset using the DrugBank and FooDB datasets.
- We adapted our previous work on DGCF recsys model to extract better embeddings for drugs and food by capturing different signals in the graph: behavioral similarity and proximity similarity.
- The evaluation results demonstrate that this novel workflow performs better for the DFI prediction task than the current approaches.

Paper Organization. The paper starts by delving into Sect. 2 where it introduces the novel workflow for recommending novel Drug-Food interactions. The results of applying a graph-based community approach are shown and discussed in Sect. 3. Finally, we review possible directions for future investigation.

2 Methodology

We propose a workflow for inferring novel DFIs, set as a link prediction problem. The workflow consist of three main steps : (1) Data Preparation, (2) DFI Network Construction, (3) DGCF Model Adaptation, and (4) DFIs prediction.

2.1 Data Preparation

We first collect DFI data from DrugBank database [1], which is a curated database that provides detailed information about drugs, their chemical structures, pharmacological properties, and drug targets. DrugBank contains data on a vast number of drugs, including both approved and experimental compounds.

We have downloaded the DrugBank dataset in XML format, which contains comprehensive information on Drug-Food Interactions (DFIs). Through parsing the XML document, we have extracted approximately 3000 sentences that describe various aspects of DFIs. Subsequently, we have manually identified and extracted DFIs from these sentences, while also defining the nature of the relationship between the food constituent and the drug, i.e., whether it is an"increase" or "decrease" relationship.

Furthermore, we observe that the food constituent contains both the constituent and the food. Given that drug-food interactions primarily occur between drugs and food components, it becomes necessary to distinguish between the food and its constituent. To address this, we enhanced the data using the FooDB database [2] to obtain constituents of food.

FooDB provides detailed information about various food items, including their composition, nutritional content, flavor profiles, potential bio-active compounds, and even the mapping of food and its components. By leveraging this database, we are able to obtain a clear understanding of the constituents present in different foods, enabling a more accurate dataset of drug-food interactions. In addition, we introduced a weight value that quantifies the impact of the interaction between the food and the drug. This weight value ranges between 0, indicating no interaction between the drug and food pair, and 1, representing a complete and significant interaction. By assigning this weight, we can effectively capture and measure the extent of the interaction between the drug and food in a more precise manner. This weight value was extracted using an adjustment of DeepDDI [3] framework applied to DFIs.

As a result, we have a total of 2159 drugs, 274 food constituent, and $358,995$ DFIs.

2.2 DFI Network Construction

We, then extracted the DFI network, which is a bipartite graph containing two types of nodes (drugs and food constituents) and the edge interaction that represent either the increasing or the decrease relationship.

Furthermore, we construct a drug-drug graph by computing the Jaccard similarity on the drug-food interaction matrix, for both the decrease and the increase relationship.

2.3 DGCF Model Adaptation

In order the extract the embeddings for both drug and food, we used DGCF framework [4] which is an extension of a classical graph approach for recommender systems GCF [12,13]. The DGCF model has three layers: the Community Encoding layer (CE), the Bipartite Graph Convolutional Networks encoder (EB-GCN), and the Information Fusion layer (IF). This layer effectively captures the community profile of drugs, which is a measure of their similarity to one another. The EB-GCN layer, on the other hand, generates representations of drugs and foods and captures the collaborative signal in the drug-food interaction bipartite graph. By extracting the interactions between drugs and foods, this layer enables the prediction of potential drug-food interactions. Finally, the IF layer aggregates the embeddings from the CE and EB-GCN layers, thereby fusing information from different perspectives to create a more comprehensive representation of drugs and foods.

DGCF is a promising approach for identifying novel drug-food interactions. It captures both the community profile of drugs and the collaborative signal in drug-food interactions. This makes it a more comprehensive approach than traditional methods that only consider one or the other. The DGCF Framework is depicted in Fig. 1.

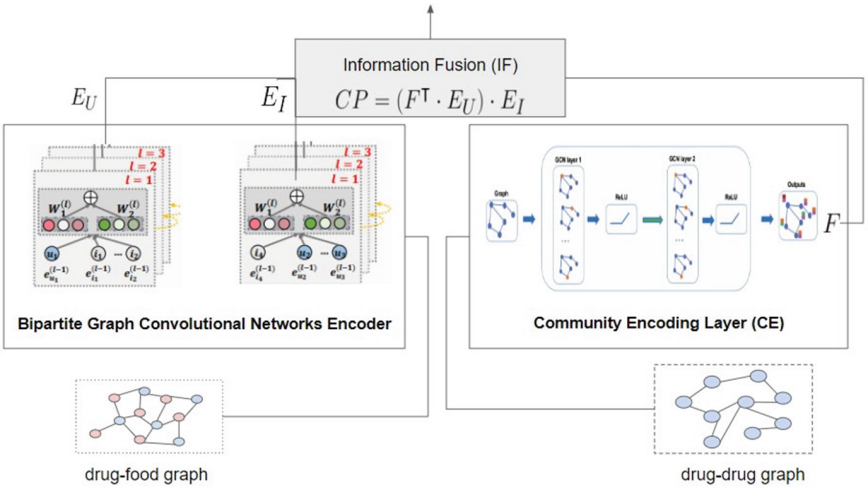


Fig. 1. The general architecture of DGCF.

Community Encoding Layer (CE). To identify community profiles, the initial step involves detecting communities. This is achieved through the utilization of a Community Encoding layer (CE), which computes the sub-communities of the homophily network based on the drug-drug graph. The CE layer combines

GCN (Graph Convolutional Network) and the Bernoulli-Poisson model to generate an affiliation matrix.

The detection of overlapping communities is treated as a probabilistic inference task. It entails deducing the unobserved associations of drugs with communities from the drug-drug network G through the use of a GCN architecture. The representation of drug affiliations with communities is denoted by F, while the binary adjacency matrix of the undirected and unweighted graph G is indicated by A, where N represents the count of drug nodes.

$$F := GCN_\theta(A) = ReLU(\hat{A}ReLU(\hat{A}W^{(1)})W^{(2)}) \tag{1}$$

In this context, the normalized adjacency matrix of graph G is represented by $\hat{A} = D^{-1/2}\tilde{A}D^{-1/2}$, where D denotes the diagonal degree matrix of A and $\tilde{A} = A + I_N$ is the adjacency matrix with self-loops. The weights $W^{(1)}$ and $W^{(2)}$ are optimized using the GCN architecture. To ensure non-negativity of the affiliation matrix F, the ReLU non-linearity is applied to the output layer.

The aim is to reduce the negative log-likelihood by identifying the optimal parameters (weights) denoted by θ in the GCN model:

$$\theta := argmin_\theta \mathcal{L}(GCN_\theta(A)) \tag{2}$$

The negative log-likelihood function, denoted as $\mathcal{L}(F)$, is obtained through the derivation of the Bernoulli-Poisson model:

$$\theta := argmin_\theta \mathcal{L}(GCN_\theta(A)) \tag{3}$$

where $\mathcal{L}$ represents the negative log-likelihood of the Bernoulli-Poisson model.

$$\mathcal{L}(F) = -\sum_{(u,v)\notin E} log(1 - exp(-F_u F_v^T)) + \sum_{(u,v)\notin E} F_u F_v^T \tag{4}$$

The row vector of community affiliation F for node u and node v are denoted as F_u and F_v, respectively. The set of edges that connect nodes in the graph is represented by E.

In order to optimize the F matrix, we update the parameters of the neural network architecture by minimizing the negative log-likelihood. Our encoding layer employs a 2-layer GCN with a hidden size of 128, and the final layer outputs the number of communities to be detected. To prevent overfitting, we incorporate batch normalization and dropout with a ratio of 0.5. Furthermore, we leverage the unique relationships conveyed by the two graphs by merging the outputs of the CE and EB-GCN layers, which correspond to the drug-food and drug-drug graphs, respectively.

Bipartite Graph Convolutional Networks Encoder (EB-GCN). The EB-GCN layer tackles the issue of data sparsity in collaborative filtering by generating additional embeddings for drugs and food. The embedding vectors for drug u and food i are represented as e_u and e_i, respectively, where d denotes

the embedding size. In this encoding layer, we input the drug-food bipartite graph, which consists of nodes belonging to drugs and nodes belonging to food. The main objective is to capture collaborative signals from various types of interactions in the network and learn the final representations for both drugs and food. To achieve this, we leverage the power of GNN algorithms applied to the bipartite graph. The EB-GCN layer exploits the high-order connectivity present in drug-food interactions (DFIs). It utilizes the message-passing mechanism of GNNs to encode drug and food nodes by iteratively aggregating information from neighboring drugs. The high-order propagation is achieved through stacking multiple embedding layers. Each layer involves the construction and aggregation of messages. The construction of the message for a drug-food pair (u, i) is defined as $m_{u \leftarrow i}$:

$$m_{u \leftarrow i} = h(e_i, e_u, p_{ui}) \tag{5}$$

The function denoted by $h(\cdot)$ encodes the message by utilizing the drug embedding e_u, the food embedding e_i, and the coefficient p_{ui} as inputs. The coefficient p_{ui} governs the decay factor for each propagation and edge (u, i).

The message encoding function h is defined as follows:

$$m_{u \leftarrow i} = \frac{1}{\sqrt{|N_u||N_i|}}(W_1 e_i + W_2(e_i \odot e_u)) \tag{6}$$

The trainable weights W_1 and $W_2 \in \mathbb{R}^{d' \times d}$ are utilized to extract propagated information, where d' denotes the transformation size. The coefficient p_{ui} is determined as the graph Laplacian norm $1/\sqrt{|N_u||N_i|}$, with N_u and N_i representing the first-hop neighbors of drug u and food i, respectively.

Following the construction and obtaining of messages, we aggregate the propagated messages from drug u's neighborhood to enhance its representation. The aggregation function is defined as:

$$e_u^{(1)} = \text{LeakyReLU}(m_{u \leftarrow u} + \sum_{i \in N_u} m_{u \leftarrow i}) \tag{7}$$

The first element in the aggregation function represents the information retained by drug u, while the second element aggregates all the information gathered from its neighborhood. Similarly, we propagate information from adjacent nodes to derive the embedding representation $e_i^{(1)}$ for food i. To learn the model parameters, we optimize the pairwise Bayesian Personalized Ranking (BPR) loss, widely used in recommender systems. The loss function is defined as:

$$\text{Loss} = \sum_{(u,i,j) \in N} -\ln(\sigma(\hat{y}ui - \hat{y}uj)) + \lambda||\theta|| \tag{8}$$

Here, N denotes the set of pairwise training data with observed and unobserved interactions. The EB-GCN layer generates two embeddings, one for drugs and one for food.

2.4 DFIs Prediction

The drug embedding E_U, food embedding E_I, and community affiliation matrix F are obtained from the CE and EB-GCN layers. These layers produce outputs $E_U \in \mathbb{R}^{n \times d}$, $E_I \in \mathbb{R}^{k \times d}$, and $F \in \mathbb{R}^{n \times c}$, where n, k, c, and d represent the number of drugs, food constituents, communities, and embedding size, respectively. The information fusion (IF) layer combines these embeddings and defines the fusion method.

To create the community profile CP, we utilize the outputs from the encoding layers. The community profile is computed as follows:

$$CP = (F^{\intercal} \cdot E_U) \cdot E_I \tag{9}$$

To select the most relevant communities, we choose the top two affiliations for each drug. The fusion formula captures the similarity between the drug u and the food constituent i, taking into account the profile of the communities to which the drug belongs.

3 Evaluation and Results

Baselines. In order to effectively assess the effectiveness of our model, we explore and compare the model to various graph learning representations methods.

- GraRep [6]: is a method for learning representations of nodes in a graph that captures global structural information. The approach utilizes higher-order proximity to capture co-occurrence patterns between nodes, resulting in a more comprehensive understanding of the graph's topology. To achieve this, GraRep constructs a sequence of k-step transition matrices, where each matrix represents the probability of transitioning from one vertex to another in k steps. These matrices are then used to train a skip-gram model, which learns to predict the k-step neighbors of each vertex. The final representation of each vertex is a concatenation of the k learned representations.
- LINE [7]: aims to preserve both local and global network properties in the learned embeddings by optimizing an objective function based on first-order and second-order proximities. The approach employs the notion of node proximity to capture the local context of a node as well as its broader structural role within the network. By leveraging both first-order proximity (i.e., node co-occurrence) and second-order proximity (i.e., shared neighborhood), LINE generates low-dimensional representations that effectively capture the network's structure.
- HOPE [10]: Higher-Order Proximity preserved Embedding method focuses on learning embeddings by preserving the asymmetric transitivity information. It capturer the higher-order proximity patterns in the graph, specifically the asymmetric transitivity property, which refers to the relationship between nodes A and C through an intermediary node B. By considering

the paths of different lengths in the graph, HOPE constructs a higher-order proximity matrix that captures the co-occurrence patterns between nodes. It then utilizes singular value decomposition (SVD) to generate low-dimensional embeddings that preserve the asymmetric transitivity information.
- SDNE [8]: uses deep autoencoder architecture, it consists of of an encoder network that maps nodes into low-dimensional latent representations and a decoder network that reconstructs the original graph structure from the latent space.
- GAE [9]: employs a variational approach, introducing a probabilistic encoder that models the latent space as a probability distribution. This enables GAE to capture the uncertainty in the latent representations and generate diverse node embeddings. By optimizing the reconstruction loss and incorporating the Kullback-Leibler divergence as a regularization term, GAE learns meaningful and smooth representations that preserve the graph's structural properties.
- DeepDDI [3]: is a deep learning framework for drug-drug interaction prediction. It is composed of three main components: a feature extractor, a deep neural network, and a post-processing module. The feature extractor extracts features from the drug molecules, such as their molecular structure and physicochemical properties. The deep neural network learns the relationships between the features and the drug-drug interactions. The post-processing module filters out predictions with low confidence and generates human-readable descriptions of the predicted interactions.
- DFinder [11]: starts by constructing a heterogeneous graph that incorporates information about drugs, food items, and their interactions and then employs a deep learning-based model to learn low-dimensional embeddings for both drugs and food items, capturing their semantic and relational properties.

Evaluation Metrics. We assessed the accuracy of new drug-food interactions by utilizing both the recall and the precision evaluation metrics.

- Recall: measures the completeness of positive predictions. It is calculated by dividing the number of true positive predictions by the sum of true positive and false negative predictions. In essence, it represents the proportion of all positive DFIs interactions that were accurately identified.

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{10}$$

- Precision: is a metric that quantifies the accuracy of positive predictions. It is calculated by dividing the number of true positive predictions by the sum of true positive and false positive predictions. In essence, precision represents the proportion of predicted DFIs that are deemed relevant.

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{11}$$

Parameter Settings. Our DGCF model was developed utilizing Pytorch. The model boasts an embedding size of 64 and was trained using the Adam optimizer with default parameters. The Xavier initializer was employed to initialize the model parameters. The learning rate, $L2$ normalization coefficient, and dropout ratio were set to 10^{-3}, 10^{-5}, and 0.5, respectively. Additionally, an early stopping mechanism was implemented to prevent overfitting of the model.

Comparison with Baselines. The DGCF model performed better than the older approaches on both recall and precision. However, it performed similarly to DFinder on recall, but not on precision. DFinder has a recall of 87.33%, while DGCF has a recall of 87.40%. This means that both models are very similar in terms of their ability to correctly classify positive instances. However, DGCF has a precision of 88.01%, which is higher than DFinder's precision of 87.11%. This means that DGCF is more accurate in its predictions of positive instances. This is an important property for drug discovery, where it is important to avoid false positives. In other words, DGCF is better at distinguishing between positive and negative instances. This means that DGCF is more likely to correctly classify a positive instance as positive, and it is also less likely to incorrectly classify a negative instance as positive.

Overall, the DGCF model is a promising new approach for drug discovery. It is more accurate than the older approaches, and it is particularly good at avoiding false positives. This makes it a valuable tool for identifying potential drug-food interactions (Table 1).

Table 1. The overall performance comparison for DFIs.

Method	Recall	Precision
GraRep [6]	0.2895	0.618
LINE [7]	0.2684	0.6
HOPE [10]	0.2921	0.5967
SDNE [8]	0.0001	0.0001
GAE [9]	0.229	0.4943
DeepDDI [3]	0.817	0.648
DFinder [11]	0.8733	0.8711
DGCF	**0.8740**	**0.8801**

To assess the relevance of incorporating a community encoding (CE) layer within the DL architecture, we conducted a comparative analysis of our findings against various deep learning models, with a primary focus on the NGCF approach. NGCF, a deep learning-based method, calculates recommendations using a user-item bipartite graph, generating embeddings for both item and user nodes and using them for predictions. In contrast, our DGCF approach

takes a step further by integrating information from the EB-GCN and CE layers. While NGCF primarily captures similarity signals based on user behavior towards items, DGCF captures a broader range of signals, particularly the community behavioral signal, which reflects the influence of sub-communities. The results presented in Table 2 demonstrate how the inclusion of the community detection step enhances overall performance and validates our hypothesis regarding the advantages of incorporating contextual and topological information.

Table 2. The overall comparison of quality metrics in detecting communities for the DFI dataset.

Metrics	Ground truth communities	Predicted communities
Coverage	0.712	0.970
Conductance	0.402	0.211
Density	0.213	0.503
Clustering coefficient	0.040	0.604

DGCF highlights a limitation in using a GCN layer for bipartite graphs, as it primarily captures signals related to similarity that extend beyond the inherent properties of sub-groups within the graph. In essence, it overlooks the specific characteristics of local communities during the learning process. Therefore, the inclusion of the CE layer becomes essential, as it plays a pivotal role in extracting more comprehensive information regarding the interactions between drugs and food components. This significance arises from the observation that drugs and food components sharing the same properties in the same community are more prone to interacting with one another.

4 Conclusion

In this paper, we proposed a novel workflow that uses community profile concept to infer and identify novel drug-food interactions. The workflow was evaluated on the DrugBank dataset, and we showed that the DGCF model was able to predict new drug-food interactions than the baseline models. We first prepare the DFI dataset, then we create the DFI networks, and then apply the DGCF model to predict the novel drug-food interactions.

We believe that the proposed workflow is a promising approach for identifying novel drug-food interactions, and we plan to enhance the workflow in future work by adding more features for the drugs and the food components. We also plan to evaluate the workflow on a larger dataset of drug-food interactions.

References

1. Wishart, D., et al.: DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Research. **46**, D1074–D1082 (2018). https://doi.org/10.1093/nar/gkx1037
2. Wishart, D., et al.: FoodDB: a comprehensive food database for dietary studies, research, and education. Nucleic Acids Res. **37**, D618–D623 (2009). https://doi.org/10.1093/nar/gkn815
3. Ryu, J., Kim, H., Lee, S.: Deep learning improves prediction of drug-drug and drug-food interactions. Proc. Natl. Acad. Sci. **115**, E4304–E4311 (2018)
4. Bourhim, S., Benhiba, L., Idrissi, M.: A community-driven deep collaborative approach for recommender systems. IEEE Access. **10**, 131144–131152 (2022)
5. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings Of The Twenty-Fifth Conference On Uncertainty In Artificial Intelligence (2012)
6. Cao, S., Lu, W., Xu, Q.: GraRep: learning graph representations with global structural information. In: Proceedings Of The 24th ACM International On Conference On Information And Knowledge Management, pp. 891–900 (2015)
7. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: Proceedings Of The 24th International Conference On World Wide Web, pp. 1067–1077 (2015)
8. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings Of The 22nd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, pp. 1225–1234 (2016)
9. Kipf, T., Welling, M.: Variational graph auto-encoders. In: International Conference On Learning Representations (2017)
10. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings Of The 22nd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining, pp. 1105–1114 (2016)
11. Wang, T., et al.: DFinder: a novel end-to-end graph embedding-based method to identify drug-food interactions. Bioinformatics. **39**, btac837 (2023)
12. Bourhim, S., Benhiba, L., Idrissi, M.: Towards a Novel Graph-based collaborative filtering approach for recommendation systems. In: Proceedings Of The 12th International Conference On Intelligent Systems: Theories And Applications, pp. 1–6 (2018)
13. Bourhim, S., Benhiba, L., Idrissi, M.: Investigating algorithmic variations of an RS Graph-based collaborative filtering approach. In: Proceedings Of The ArabWIC 6th Annual International Conference Research Track, pp. 1–6 (2019)

L2G2G: A Scalable Local-to-Global Network Embedding with Graph Autoencoders

Ruikang Ouyang[1], Andrew Elliott[3,4], Stratis Limnios[3(✉)], Mihai Cucuringu[2,3], and Gesine Reinert[2,3]

[1] Department of Engineering, University of Cambridge, Cambridge, UK
[2] Department of Statistics, University of Oxford, Oxford, UK
[3] The Alan Turing Institute, London, UK
slimnios@turing.ac.uk
[4] School of Mathematics and Statistics, University of Glasgow, Glasgow, UK

Abstract. For analysing real-world networks, graph representation learning is a popular tool. These methods, such as a graph autoencoder (GAE), typically rely on low-dimensional representations, also called *embeddings*, which are obtained through minimising a loss function; these embeddings are used with a decoder for downstream tasks such as node classification and edge prediction. While GAEs tend to be fairly accurate, they suffer from scalability issues. For improved speed, a Local2Global approach, which combines graph patch embeddings based on eigenvector synchronisation, was shown to be fast and achieve good accuracy. Here we propose L2G2G, a Local2Global method which improves GAE accuracy without sacrificing scalability. This improvement is achieved by dynamically synchronising the latent node representations, while training the GAEs. It also benefits from the decoder computing an only local patch loss. Hence, aligning the local embeddings in each epoch utilises more information from the graph than a single post-training alignment does, while maintaining scalability. We illustrate on synthetic benchmarks, as well as real-world examples, that L2G2G achieves higher accuracy than the standard Local2Global approach and scales efficiently on the larger data sets. We find that for large and dense networks, it even outperforms the slow, but assumed more accurate, GAEs.

Keywords: Graph Autoencoder · Local2Global · Node Embedding · Group Synchronisation

1 Introduction

Graph representation learning has been a core component in graph based real world applications, for an introduction see [13]. As graphs have become ubiquitous in a wide array of applications, low-dimensional representations are needed to tackle the curse of dimensionality inherited by the graph structure. In practice, low-dimensional node embeddings are used as efficient representations to address tasks such as graph clustering [25], node classification [2], and link prediction [27], or to protect private data in federated learning settings [14,20].

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 400–412, 2024.
https://doi.org/10.1007/978-3-031-53468-3_34

Graph Autoencoders (GAEs) [19,23] emerged as a powerful Graph Neural Network (GNN) [5] tool to produce such node representations. GAEs adapt autoencoders and variational autoencoders [1,19] to graph structure data using a Graph Convolutional Neural Network (GCN) [28] as the encoder and for node embeddings. Although a GAE can achieve high accuracy in graph reconstruction, it suffers from a high computational cost. Several solutions for reducing computational workload have been proposed. Linear models, like PPRGo [4] and SGC [8], remove the non-linearities between layers. Layer-wise sampling methods such as GraphSage [12], FastGCN [6] and LADIES [29] sample a subset of neighbors of the nodes in each layer, while subgraph-sampling based methods such as GraphSaint [26] and Cluster-GCN [9] carry out message passing only through a sampled subgraph.

In this paper, we use FastGAE [22] as a starting point, which computes approximate reconstruction losses by evaluating their values only from suitably selected random subgraphs of the original graph. While FastGAE reduces the computational cost of a traditional GAE, its overall performance can be substantially inferior to a GAE when the sample used to approximate the loss is not large enough. For improved performance, the general Local2Global (L2G) framework by [15] leverages the eigenvector synchronization of [10,11], to align independently created embeddings in order to produce a globally consistent structure (here we employ GAEs for embeddings and denote the resulting method by GAE+L2G). However, this architecture is data inefficient and suffers from a loss of performance in downstream tasks since it learns multiple separate GAEs. Moreover aggregating the local embeddings after the training process might lead to a loss of useful information learned during training.

Instead, we propose the Local to GAE to Global (L2G2G) model, which optimizes the Local2Global aligned embeddings directly, reducing the amount of information loss and allowing us to train a single global modified GAE. This structure leverages the scalable approach of FastGAE by only considering small sections of the graph when updating the weights. Figure 1 shows the model pipeline. Our main contributions are:

1. We introduce L2G2G as a new fast network embedding method.
2. We provide a theoretical complexity analysis for GAE+L2G and an experimental comparison of the runtimes showing that the runtime sacrifice for performance in L2G2G is minimal.
3. We test L2G2G and the baseline methods on real and synthetic data sets, demonstrating that L2G2G can boost the performance of GAE+L2G, especially on medium scale data sets, while achieving comparable training speed.

The paper is structured as follows. Section 2 introduces notation and discusses GAEs and the Local2Global framework by [15], including GAE+L2G. Section 3 presents our method, L2G2G, as well as a time complexity analysis, comparing it to GAE, FastGAE,and GAE+L2G. Section 4 provides experimental results on synthetic and real data sets, on networks of up to about $700,000$ nodes. In Sect. 5 we discuss the results and indicate directions for future work. The code is available at https://github.com/tonyauyeung/Local2GAE2Global.

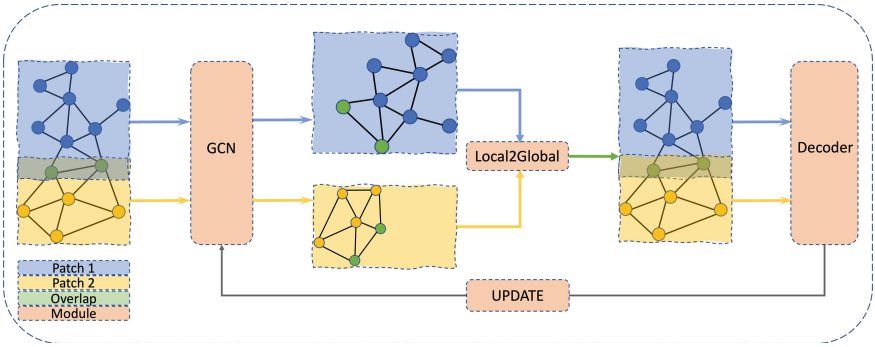


Fig. 1. L2G2L pipeline for two patches. The two patches are in blue and yellow, the overlapping nodes between them in green. Separate node embeddings for each patch are obtained via a single GCN. The decoder aligns the embeddings using the Local2Global synchronisation algorithm to yield a global embedding and then uses a standard sigmoid function. The GCN is then iteratively optimised using the training loss.

2 Preliminaries

Notations: An undirected attributed graph $G = (V, E, X)$ consists of a set of nodes V of size N, a set of unweighted, undirected edges E of size M, and a $N \times F$ matrix X of real-valued node attributes (features). The edge set is also represented by the $N \times N$ adjacency matrix A. Moreover, based on the L2G framework, we define a patch P to be a subgraph of G which is induced by a subset of the node set V; hence a patch P_i with the feature matrix corresponding to its nodes is denoted as $(V^{(i)}, E^{(i)}, X^{(i)})$ Node embeddings are denoted as a $N \times e$ matrix Z, where e is the embedding size and σ is the sigmoid function.

Graph Autoencoders (GAEs): Given a graph $G = (V, E, X)$, a GCN is used to obtain an $N \times e$ embedding matrix $Z = GCN(X, A)$, and a sparse approximation of the adjacency matrix through $\hat{A} = \sigma(ZZ^T)$. The GAE obtains the embedding through minimising the cross-entropy reconstruction loss, $L_{GAE}(\hat{A}, A) = Loss(\hat{A}, A) := -\sum_{i,j=1}^{N} A_{ij} log \hat{A}_{ij}$ with respect to the parameters of the GCN; this minimisation is also called *training*. Here a recursive method called *message passing* is used. The *decoder* then computes an inner product between each pair of node embeddings in the graph as proxy for the edge probabilities. Even though GAEs outperform traditional node embedding methods, such as spectral clustering [24] and DeepWalk [21], they usually scale poorly to large graphs. This is due to having to visit all the neighbors of a node recursively during the message passing phase in the encoding GCN, and the decoder scaling as $O(N^2)$ in complexity. We highlight two approaches for improving scalability:

1. FastGAE [22]: This model addresses the scalability issues by reconstructing the adjacency matrix of a sampled subgraph. This is achieved by evaluating an approximate reconstruction loss ($Loss_{approx}$) for every subgraph and

aggregating them in one loss to be optimized by the model. This sampling procedure reduces the computation complexity of decoding during each training epoch from $O(N^2)$ to $O(N_S^2)$, where N_S is the number of nodes in the subgraph.

2. Local2Global (L2G) [15]: This framework is a generic method to align embeddings computed on different parts of the graph (potentially on different machines and by different entities with different privacy constraints) into a single global embedding, regardless of the embedding method, as follows. Suppose that $P_1, ..., P_k$ are k patches, which pairwise overlap on at least d nodes and at most l nodes. It is assumed that the graph union of all patches gives the initial graph G. The pattern of overlapping patches is captured in a so-called *patch graph*, denoted $G_P = (V_P, E_P)$, whose node set $V_P = \{P_1, \ldots, P_k\}$ denote the patches. An edge between two nodes in G_P indicates that there is an overlap of at least d nodes in the initial graph G between those two patches. Then, for each patch P_i a node embedding matrix Z_i is computed using an embedding method of choice. When the embedding is obtained through a GAE, we refer to the method as GAE+L2G. Local2Global then leverages the overlap of the patches to compute an optimal alignment based on a set of affine transforms which synchronizes all the local patch embeddings into a single and globally consistent embedding, as follows. First, we estimate rotation matrices $\hat{S}_j \in \mathbb{R}^{F \times F}, j = 1, \ldots, k$, one for each patch. With $M_{ij} = \sum_{u \in P_i \cap P_j} X_u^{(i)} (X_u^{(i)})^T$ we first estimate the rotations between each pair of overlapping patches $(P_i, P_j) \in E_p$ by $R_{ij} = M_{ij}(M_{ij}^T M_{ij})^{-1/2}$. Next we build $\tilde{R}_{ij} = w_{ij} R_{ij} / \sum_j |V(P_i) \cap V(P_j)|$ to approximately solve the eigen problem $S = \tilde{R}S$, obtaining $\hat{S} = [\hat{S}_1, ...\hat{S}_k]$. We also find a translation matrix $\hat{T} = [\hat{T}_1, \ldots, \hat{T}_k]$ by solving $\hat{T} = \underset{T \in \mathbb{R}^{k \times F}}{\arg min} ||BT - C||_2^2$, where $B \in \{-1, 1\}^{|E_p| \times k}$ is the incidence matrix of the patch graph with entries $B_{(P_i,P_j),t} = \delta_{it} - \delta_{jt}$, δ is the Kronecker Delta, and $C \in \mathbb{R}^{|E_p| \times F}$ has entries $C_{(P_i,P_j)} = \sum_{t \in P_i \cap P_j} \left(\hat{Z}_t^{(i)} - \hat{Z}_t^{(j)}\right) / |P_i \cap P_j|$. This solution yields the estimated coordinates of all the nodes up to a global rigid transformation. Next, we apply the appropriate rotation transform to each patch individually, $\hat{Z}^{(j)} = Z^{(j)} \hat{S}_j^T$, then apply the corresponding translation to each patch (hence performing translation synchronisation), and finally average in order to obtain the final aligned node embedding $\bar{Z}_i = \sum_j (\hat{Z}_i^{(j)} + \hat{T}_j) / |\{j \ : \ i \in P_j\}|$.

3 Methodology

Local-2-GAE-2-Global Combining Local2Global and GAEs produces a scalable GAE extension for node embeddings using autoencoders; using separate GAEs for each of the patches allows specialization to the unique structure in each of the patches. Our Local-2-GAE-2-Global (L2G2G) framework leverages the same divide-and-conquer technique Local2Global capitalises on, but is designed and adapted to the traditional GAE pipeline to benefit from its accuracy. The core

idea of L2G2G is to evaluate embeddings locally on the patches but synchronizing the patch embeddings using the L2G framework while training a GAE. This leads to k GCNs encoding the k patches: $Z_i = GCN(X^{(i)}, A^{(i)})$, for $i = 1, \ldots, k$. To account for the dynamic update during training and adapt to the local optimization scheme, we modify the GAE decoder to adjust the embeddings using the Local2Global framework; hence the patch-wise decoder in L2G2G estimates the edge probabilities between nodes in patches i and j by $\sigma((S_i Z_i + T_i)^T (S_j Z_j + T_j))$, where $S_i = S_i(Z)$ and $T_i = T_i(Z)$ are the Local2Global transformations of each of the patch embeddings.

In contrast to GAE+L2G, L2G2G synchronizes the embeddings before the decoder step and also performs synchronizations during the model training, thus taking full advantage of the patch graph structure during training. The cross-entropy losses of each patch are aggregated to give a global loss function:

$$L_{L2G2G} = \sum_{j=1}^{k} N_j L_{GAE}\left(\hat{A}^{(j)}, A^{(j)}\right)/N.$$

Similarly to the FastGAE algorithm, L2G2G reduces computation by only considering local structure. However, rather than training the network using only the local information, L2G2G aggregates the local embeddings to reconstruct the global information, thus boosting performance.

A schematic diagram of L2G2G is shown in Fig. 1, and pseudo-code for L2G2G is given in algorithm 1. As computing the alignment step can be costly, assuming that the Local2Global alignment would not change too quickly during training, we update the rotation and translation matrices only every 10 epochs.

Algorithm 1. Local-2-GAE-2-Global (L2G2G): An overview

Require: $P_1, ..., P_k$, where $P_j = (X^{(j)}, A^{(j)})$
  for e in $[1, ..., T]$ **do**
    for j in $[1, ...k]$ **do**
      $Z_j \leftarrow GCN(X^{(j)}, A^{(j)})$
    end for
    $\hat{Z}_1, ..., \hat{Z}_k \leftarrow Sync(Z_1, ..., Z_k)$
    $L \leftarrow 0$
    for j in $[1, ..., k]$ **do**
      $\hat{A}_j \leftarrow \sigma(\hat{Z}_j \hat{Z}_j^T)$
      $L \leftarrow L + \frac{N_j}{N} L_{GAE}(\hat{A}^{(j)}, A^{(j)})$
    end for
    Optimize encoder using L
  end for

Complexity Analysis Following the computations in [7,15], we derive the complexity of GAE, FastGAE, GAE+L2G and L2G2G. We assume that the number of nodes, edges and features satisfy $N, M, F \geq 1$, and, following [7], that the

dimensions of the hidden layers in the GCN are all F. Then, the complexity of a L-layer GCN scales like $O(LNF^2+LMF)$ and that of the inner product decoder scales like $O(N^2F)$. maybe add something here about full decoder? Thus, for as shown in [7], for T epochs the time complexity of the decoder and the encoder of a GAE scales like $O(T(LNF^2 + LMF + N^2F))$. In contrast, as stated in [22], the complexity of per-epoch of FastGAE with a $\sqrt{N}$ down-sampling size is $O(LNF^2 + LMF + NF)$, and hence for T epochs the FastGAE complexity scales like $O(T(LNF^2 + LMF + NF))$.

To simplify the complexity analysis of both Local2Global approaches we assume that the overlap size of two overlapping patches in the patch graph is fixed to $d \sim F$. Following [15], finding the rotation matrix S scales like $O(|E_p|dF^2) = O(|E_p|F^3)$. The translation problem can be solved by a t-iteration solver with a complexity per iteration of $O(|E_p|F)$, where t is fixed. To align the local embeddings, one has to perform matrix multiplications, which requires $O(N_jF^2)$ computations, where N_j is the number of nodes in the j^{th} patch. The complexity of finding the rotation matrix ($O(|E_p|F^3)$) dominates the complexity of the computing the translation ($O(|E_p|F)$). Thus, the complexity of the L2G algorithm with k patches is $O\big(|E_p|F^3 + F^2\sum_{j=1}^{k} N_j\big)$.

The GAE+L2G algorithm uses a GAE for every patch, and for the j^{th} patch, for T training epochs the GAE scales like $O(T(LN_jF^2 + LM_jF + N_j^2F))$, with M_j number of edges in the j^{th} patch. Summing over all patches and ignoring the overlap between patches as lower order term, so that $\sum_j N_j = O(N)$, $\sum_j N_j^2 \approx N^2/k$, and $\sum_j M_j \approx M$, the GAE+L2G algorithm scales like $O\left(TF\left(LNF + LM + N/k\right) + kF^3\right)$. For the complexity of L2G2G, as L2G2G aligns the local embeddings in each epoch rather than after training, we replace $kF^3 + NF^2$ with $T(kF^3 + NF^2)$, and thus the algorithm scales like $O\Big(T\Big(LNF^2 + LMF + \frac{N^2}{k}F + kF^3\Big)\Big)$. In the PyTorch implementation of FastGAE, the reconstruction error is approximated by creating the induced subgraph from sampling $\lfloor\sqrt{N}\rfloor$ proportional to degree, with an expected number of at least $O(M/N)$ edges between them. Then, the computation of the decoder is (at least) $O(M/N)$ instead of $O(N^2)$. Table 1 summarises the complexity results.

Table 1. Complexity comparison in the general and in the sparse case.

Model	General Time Complexity	PyTorch implementation
GAE	$O\left(TF(LNF + LM + N^2)\right)$	$O\left(TF(LNF + LM + M)\right)$
FastGAE	$\geq O\left(TF(LNF + LM + N)\right)$	$O\left(TF(LNF + LM + M/N)\right)$
GAE+L2G	$O\left(TF(L(NF + M) + \frac{N^2}{k}) + kF^3\right)$	$O\left(TF(LNF + LM + M) + kF^3\right)$
L2G2G	$O\left(TF(L(NF + M) + \frac{N^2}{k} + kF^3)\right)$	$O\left(TF(LNF + LM + M + kF^3)\right)$

Thus, in the standard case, increasing the number of patches k reduces the complexity of the computation of the GAE decoders. In the PyTorch implementation, if k scales linearly with N, the expression is linear in N. In contrast, when

the number of nodes N is not very large, the number of features F becomes more prominent, so that the training speed may not necessarily increase with increasing number of patches. Table 1 shows that L2G2G sacrifices $O\left(TkF^3\right)$ training time to obtain better performance; with an increase in the number of patches, the training speed gap between L2G2G and GAE+L2G increases linearly.

4 Experimental Evaluation

Datasets To measure the performance of our method, we compare the ability of L2G2G to learn node embeddings for graph reconstruction against the following benchmark datasets Cora ML, Cora [3], Reddit [26] and Yelp [26].

In addition, we tested the performance of L2G2G on four synthetic data sets, generated using a Stochastic Block Model (SBM) which assigns nodes to blocks; edges are placed independently between nodes in a block with probability p_{in} and between blocks with probability p_{out} [17]. We encode the block membership as node features; with L blocks, v being in block l is encoded as unit vector $e_l \in \{0,1\}^L$. To test the performance across multiple scales we fix the number of blocks at 100, and vary the block size, p_{in} and p_{out}, as follows:

1. 'SBM-Small' with block sizes 10^2 and $(p_{in}, p_{out}) = (0.02, 10^{-4})$,
2. 'SBM-Large-Sparse' with block sizes 10^3 and $(p_{in}, p_{out}) = (10^{-3}, 10^{-4})$,
3. 'SBM-Large' with blocks of sizes 10^3 and $(p_{in}, p_{out}) = (0.02, 10^{-4})$,
4. 'SBM-Large-Dense' with block sizes 10^3 and $(p_{in}, p_{out}) = (0.1, 0.002)$.

Table 2 gives some summary statistics of these real and synthetic data sets.

Table 2. Network data statistics: N = no. nodes, M = no. edges, F =no. features

	Stochastic block model				Real Data			
	Small	Large-Sparse	Large	Large-Dense	Cora ML	Cora	Reddit	Yelp
N	10,000	100,000	100,000	100,000	2,995	19,793	232,965	716,847
M	104,485	99,231	1,493,135	14,897,099	16,316	126,842	23,213,838	13,954,819
F	100	100	100	100	2,879	8,710	602	300

Experimental setup and Results To assess whether L2G2G is a scalable alternative for the use of a GAE without having to sacrifice accuracy in downstream tasks, we compare it against the standard GAE [19], GAE+L2G [15] and FastGAE [22]. We train the models on each data set for 200 epochs, with learning rate 0.001 and the Adam optimizer [18], and two layers in the GCN. The dimension of the first hidden layer is 32 and the dimension of the last layer is 16. We run each experiment 10 times with different random seeds for each model on each data set. All the experiments were conducted on a V100 GPU. We then compare the methods using the Area Under the Curve (AUC) and the Average Precision (AP). Following [15], we test our algorithm with fixed patch size 10.

Table 3 shows that L2G2G outperforms both FastGAE and GAE+L2G on most experiments. Having established the theoretical training speed gain of L2G2G, these results illustrate that L2G2G can perform better than the GAE+L2G, as well as achieve comparable training speed. do we need to clarify with new pytorch stuff? Furthermore, in contrast to FastGAE we observe that the performance of L2G2G and of the GAE are very close to each other on the medium and large-scale data sets, indicating that L2G2G does not lose much performance compared to the much slower but assumed more accurate classic GAE. Furthermore, L2G2G even outperforms the GAE when the data set is large and dense, such as SBM-Large-dense and Reddit.

Table 3. Experiments on different data sets with patch size 10. Bold: the best among the fast methods, underlined: the model outperforms the GAE.

Average Performance On Different Datasets (AUC in %)				
	GAE	FastGAE	GAE+L2G	L2G2G
Cora ml	95.95 ± 0.42	83.90 ± 1.10	90.25 ± 0.19	**92.58 ± 0.35**
SBM-small	95.32 ± 0.18	76.34 ± 0.57	93.84 ± 0.14	**95.39 ± 0.21**
Cora	96.07 ± 0.09	81.78 ± 0.76	90.59 ± 0.11	**94.96 ± 0.26**
SBM-Large-sparse	94.88 ± 0.23	80.89 ± 0.84	94.73 ± 0.07	**95.02 ± 0.23**
SBM-Large	86.84 ± 0.11	70.90 ± 1.29	84.60 ± 0.10	**86.62 ± 0.25**
SBM-Large-dense	64.07 ± 0.12	65.20 ± 0.94	65.45 ± 0.05	**65.88 ± 0.03**
Reddit	88.69 ± 0.57	79.99 ± 1.31	**88.50 ± 0.23**	88.37 ± 0.39
Yelp	86.55 ± 0.28	73.79 ± 6.54	**85.82 ± 0.17**	84.01 ± 0.11
Average Performance On Different Datasets (AP in %)				
	GAE	FastGAE	GAE+L2G	L2G2G
Cora ml	95.37 ± 0.57	83.90 ± 1.10	90.57 ± 0.19	**92.41 ± 0.39**
SBM-small	95.23 ± 0.12	76.34 ± 0.57	95.22 ± 0.11	**95.71 ± 0.24**
Cora	95.76 ± 0.14	81.78 ± 0.76	90.50 ± 0.15	**94.67 ± 0.29**
SBM-Large-sparse	95.26 ± 0.24	80.89 ± 0.84	**95.88 ± 0.07**	95.44 ± 0.26
SBM-Large	89.64 ± 0.21	70.90 ± 1.29	87.35 ± 0.11	**89.34 ± 0.33**
SBM-Large-dense	67.64 ± 0.30	65.20 ± 0.94	71.25 ± 0.06	**72.08 ± 0.05**
Reddit	88.16 ± 0.60	79.99 ± 1.31	88.40 ± 0.18	**88.57 ± 0.40**
Yelp	86.73 ± 0.29	73.79 ± 6.54	**85.26 ± 0.12**	83.56 ± 0.11

Figure 2 shows a comparison of the training time of the models, as well as the changes of training speed as the data set size increases (on the log scale). It is worth mentioning that we are not accounting for the run time of the graph clustering. The results show that the training speed of L2G2G and GAE+L2G are very close on both small and large scale datasets. Although the gap between the training speed of L2G2G and that of GAE+L2G increases for large-scale data sets, L2G2G still achieves high training speed, and is even not much slower

that FastGAE while achieving much better performance. In almost all cases, L2G2G is faster than the standard GAE, except for the two smaller datasets. Its training time is around an order of magnitude smaller per epoch for the larger models. As an aside, GAEs suffer from memory issues as they need to store very large matrices during the decoding step.

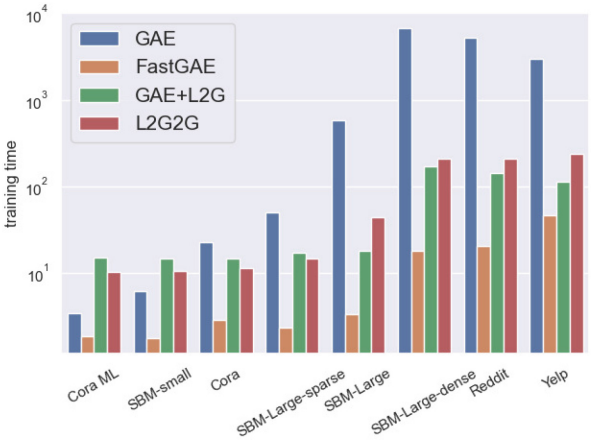


Fig. 2. Training time of the baseline models(GAE, FastGAE and GAE+L2G) and L2G2G on benchmark data sets (excluding partitioning time). Note that the y-axis is on a log-scale, and thus the faster methods are at least an order of magnitude faster.

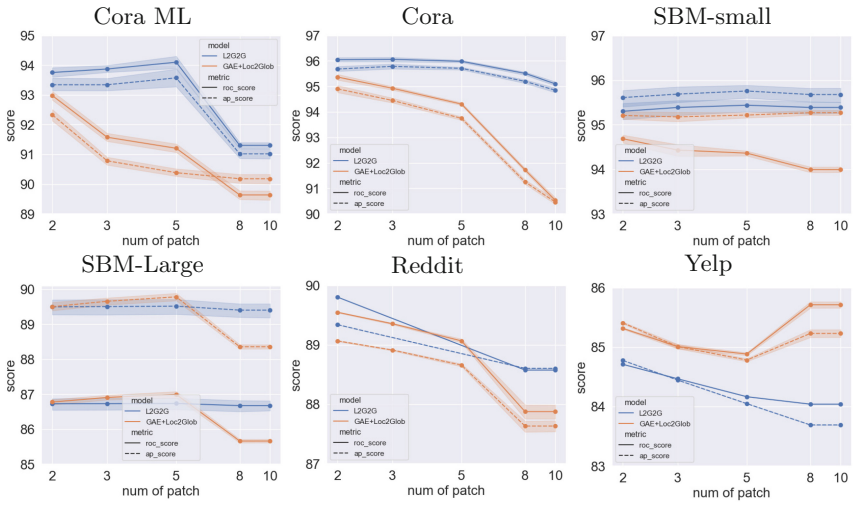


Fig. 3. Lineplots of the ROC score and accuracy of L2G2G and GAE+L2G, trained on each dataset, with different patch sizes. For each subplot, the blue lines represent the metrics for L2G2G, while the orange ones represent those for GAE+L2G. The shadows in each subplot indicate the standard deviations of each metric.

Ablation Study Here we vary the number of patches, ranging from 2 to 10. Figure 3 shows the performance changes with different number of patches for each model on each data set. When the patch size increases, the performance of L2G2G decreases less than GAE+L2G. This shows that updating the node embeddings dynamically during the training and keeping the local information with the agglomerating loss actually brings stability to L2G2G.

Moreover, we have explored the behaviour of training time for L2G2G when patch size increases from 2 to 30, on both a small (Cora) and a large (Yelp) dataset. Figure 4 shows that on the small-scale data set Cora, the gap in training speed between L2G2G and GAE+L2G remains almost unchanged, while on Yelp, the gap between L2G2G and GAE+L2G becomes smaller. However, the construction of the overlapping patches in the Local2Global library can create patches that are much larger than N/k, potentially resulting in a large number of nodes in each patch. Hence, the training time in our tests increases with the number of patches.

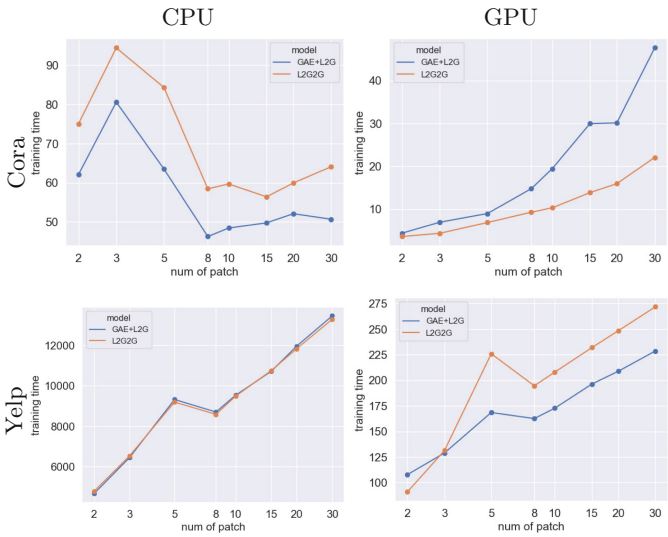


Fig. 4. Training time (excluding partitioning) of L2G2G and GAE+L2G on Cora (**Top**) and Yelp (**Bottom**), while varying patch size with CPU results presented on the **left** and GPU results presented on the **right**. The x axis is shown in log scale.

Since all the computations in Local2Global library built by [15] are carried out on the CPU, the GPU training can be slowed down by the memory swap between CPU and GPU. Thus, to further explore the behaviour of our algorithm when the number of patches increases, we ran the test on both CPU and GPU. The results are given by Fig. 4. This plot illustrates that the GPU training time of L2G2G increases moderately with increasing patch size, mimicking the behaviour of GAE+L2G. In contrast, the CPU training time for the smaller data

set (Cora) decreases with increasing patch size. The larger but much sparser Yelp data set may not lend itself naturally to a partition into overlapping patches. Summarising, L2G2G performs better than the baseline models across most settings, while sacrificing a tolerable amount of training speed.

5 Conclusion and Future Work

In this paper, we have introduced L2G2G, a fast yet accurate method for obtaining node embeddings for large-scale networks. In our experiments, L2G2G outperforms FastGAE and GAE+L2G, while the amount of training speed sacrificed is tolerable We also find that L2G2G is not as sensitive to patch size change as GAE+L2G.

Future work will investigate embedding the synchronization step in the network instead of performing the Local2Global algorithm to align the local embeddings. This change would potentially avoid matrix inversion, speeding up the calculations. We shall also investigate the performance on stochastic block models with more heterogeneity. To improve accuracy, one could add a small number of between–patch losses into the L2G2G loss function, to account for edges which do not fall within a patch. The additional complexity of this change would be relatively limited when restricting the number of between–patches included. Additionally, the Local2Global library from [16] is implemented on CPU, losing speed due to moving memory between the CPU and the GPU.

References

1. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: Guyon, I., Dror, G., Lemaire, V., Taylor, G., Silver, D., (eds.) Proceedings of ICML Workshop on Unsupervised and Transfer Learning, Proceedings of Machine Learning Research, vol. 27, pp. 37–49. PMLR, Bellevue, Washington, USA (2012)
2. Bayer, A., Chowdhury, A., Segarra, S.: Label propagation across graphs: node classification using graph neural tangent kernels. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5483–5487. IEEE (2022)
3. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: unsupervised inductive learning via ranking. In: International Conference on Learning Representations (2018). https://openreview.net/forum?id=r1ZdKJ-0W
4. Bojchevski, A., et al.: Scaling graph neural networks with approximate PageRank. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM (2020)
5. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
6. Chen, J., Ma, T., Xiao, C.: FastGCN: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
7. Chen, M., Wei, Z., Ding, B., Li, Y., Yuan, Y., Du, X., Wen, J.: Scalable graph neural networks via bidirectional propagation. CoRR **abs/2010.15421** (2020). https://arxiv.org/abs/2010.15421

8. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: International Conference on Machine Learning, pp. 1725–1735. PMLR (2020)
9. Chiang, W.L., Liu, X., Si, S., Li, Y., Bengio, S., Hsieh, C.J.: Cluster-GCN. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. ACM (2019)
10. Cucuringu, M., Lipman, Y., Singer, A.: Sensor network localization by eigenvector synchronization over the Euclidean group. ACM Trans. Sen. Netw. **8**(3), 1–42 (2012)
11. Cucuringu, M., Singer, A., Cowburn, D.: Eigenvector synchronization, graph rigidity and the molecule problem. Inf. Infer. **1**(1), 21–67 (2012)
12. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
13. Hamilton, W.L.: Graph Representation Learning. Morgan & Claypool Publishers (2020)
14. He, C., et al.: FedGraphNN: a federated learning system and benchmark for graph neural networks. CoRR **abs/2104.07145** (2021). https://arxiv.org/abs/2104.07145
15. Jeub, L.G., Colavizza, G., Dong, X., Bazzi, M., Cucuringu, M.: Local2Global: a distributed approach for scaling representation learning on graphs. Mach. Learn. **112**(5), 1663–1692 (2023)
16. Jeub, L.G.S.: Local2Global github package. Github (2021). https://github.com/LJeub/Local2Global
17. Karrer, B., Newman, M.E.J.: Stochastic blockmodels and community structure in networks. Phys. Rev. E **83**(1), 016107 (2011)
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (Poster) (2015)
19. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
20. Pan, Q., Zhu, Y.: FedWalk: communication efficient federated unsupervised node embedding with differential privacy. arXiv preprint arXiv:2205.15896 (2022)
21. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM (2014)
22. Salha, G., Hennequin, R., Remy, J.B., Moussallam, M., Vazirgiannis, M.: FastGAE: scalable graph autoencoders with stochastic subgraph decoding. Neural Netw. **142**, 1–19 (2021)
23. Simonovsky, M., Komodakis, N.: GraphVAE: towards generation of small graphs using variational autoencoders. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11139, pp. 412–422. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01418-6_41
24. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Discov. **23**(3), 447–478 (2011)
25. Tsitsulin, A., Palowitch, J., Perozzi, B., Müller, E.: Graph clustering with graph neural networks. arXiv preprint arXiv:2006.16904 (2020)
26. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: GraphSAINT: graph sampling based inductive learning method. In: International Conference on Learning Representations (2020)
27. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. In: Advances in Neural Information Processing Systems, vol. 31 (2018)

28. Zhang, S., Tong, H., Xu, J., Maciejewski, R.: Graph convolutional networks: a comprehensive review. Comput. Soc. Netw. **6**(1), 1–23 (2019)
29. Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., Gu, Q.: Layer-dependent importance sampling for training deep and large graph convolutional networks. In: Advances in neural information processing systems, vol. 32 (2019)

A Comparative Study of Knowledge Graph-to-Text Generation Architectures in the Context of Conversational Agents

Hussam Ghanem[1,2(✉)] and Christophe Cruz[1]

[1] ICB, UMR 6306, CNRS, Unversité de Bourgogne, Dijon, France
hussam.ghanem@gmail.com
[2] Davi The Humanizers, Puteaux, France

Abstract. This work delves into the dynamic landscape of Knowledge Graph-to-text generation, where structured knowledge graphs are transformed into coherent natural language text. Three key architectural paradigms are explored: Graph Neural Networks (GNNs), Graph Transformers (GTs), and linearization with sequence-to-sequence models. We discuss the advantages and limitations of these architectures, and we do some experiments on these architectures. Performance evaluations on WebNLG V.2 demonstrate the superiority of sequence-to-sequence Transformer-based models, especially when enriched with structural information from the graph. Despite being unsupervised, the CycleGT model also outperforms GNNs and GTs. However, practical constraints, such as computational efficiency and model validity, make sequence-to-sequence models the preferred choice for real-time conversational agents. Future research directions include enhancing the efficiency of GNNs and GTs, addressing scalability issues, handling multimodal knowledge graphs, improving interpretability, and devising data labeling strategies for domain-specific models. Cross-lingual and multilingual extensions can further broaden the applicability of these models in diverse linguistic contexts. In conclusion, the choice of architecture should align with specific task requirements and application constraints, and the field offers promising prospects for continued innovation and refinement.

Keywords: Conversational Agents · Knowledge Graphs · Natural Language Generation · Graph Neural Networks · Graph Transformers · Sequence-to-Sequence Models

1 Introduction

Conversational agents, commonly known as chatbots, have emerged as sophisticated computer programs that emulate human-like conversations with users [55]. These agents find applications across various platforms, such as messaging services, mobile apps, and websites, enabling instantaneous customer support and handling routine tasks like answering queries and assisting with bookings. The ones incorporating knowledge graphs (KG) [35] and [21] have emerged prominently within the broad spectrum of

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 413–426, 2024.
https://doi.org/10.1007/978-3-031-53468-3_35

architectural options. These conversational agents, rooted in KG, leverage the organized information within a knowledge graph to craft responses that closely mimic human language. These agents can access and employ structured information during conversations by utilizing the richly interconnected representation of entities and their relationships within the knowledge graph, resulting in more accurate and comprehensive user interactions. Incorporating knowledge graphs in conversational agents significantly augments their capabilities, making interactions more informative and beneficial.

In natural language processing and knowledge representation, the survey on graph-to-text generation architectures assumes a vital role. This review guides researchers, practitioners, and decision-makers through this rapidly evolving landscape by presenting a comprehensive overview of current techniques. Its insights into emerging trends, approach strengths, and limitations facilitate the design of effective systems for transforming structured data into coherent human-readable narratives. Furthermore, the survey encourages collaboration, knowledge sharing, and innovation within the field, thus advancing graph-to-text generation and bridging the gap between structured knowledge and natural language expression.

Recently, neural approaches have demonstrated remarkable performance, surpassing traditional methods in achieving linguistic coherence. However, challenges persist in maintaining semantic consistency, particularly with long texts [41]. The inherent complexity of neural approaches also poses limitations, as they need more parameterization and control over the structure of the generated text. Consequently, while current neural approaches tend to lag template-based methods regarding semantic consistency [42], they outperform them significantly in terms of linguistic coherence. This distinction can be attributed to the ability of large language models (LLM) to capture specific syntactic and semantic properties of the language. Despite the advantages neural approaches offer regarding linguistic consistency, their performance in maintaining semantic consistency is still a work in progress.

Graph-to-text (G2T) generation is a natural language processing (NLP) task that involves transforming structured data from a graph format into human-readable text. This task converts a knowledge graph (structured data representation where entities are nodes and relationships between entities are edges) into coherent sentences or paragraphs in a natural language. Generating text from graphs necessitates sophisticated methods in graph processing for extracting pertinent information and in natural language generation to produce coherent and contextually fitting text. This is a demanding yet valuable endeavor within the larger framework of content creation and communication driven by data.

In the natural language generation (NLG), two criteria [13] are used to assess the quality of the produced answers. The first criterion is semantic consistency (Semantic Fidelity), which quantifies the fidelity of the data produced against the input data. The most common indicators are 1/ Hallucination: It is manifested by the presence of information (facts) in the generated text that is not present in the input data; 2/ Omission: It is exemplified by the omission of one of the pieces of information (facts) in the generated text; 3/ Redundancy: This is manifested by the repetition of information in the generated text; 4/ Accuracy: The lack of accuracy is manifested by the modification of information such as the inversion of the subject and the direct object complement in the generated

text; 5/ Ordering: It occurs when the sequence of information is different from the input data. The second criterion is linguistic coherence (Output Fluency) to evaluate the fluidity of the text and the linguistic constructions of the generated text, the segmentation of the text into different sentences, the use of anaphoric pronouns to reference entities and to have linguistically correct Sentences.

Our objective is to delve into the intricacies of deep neural network architectures that harness the power of graphs, aiming to gain a comprehensive understanding of their inherent strengths and limitations for optimal utilization in the context of conversational agents.

This paper follows a structured progression to delve into the knowledge graph-to-text (KG2T) generation landscape. Beginning with an in-depth review of advanced KG2T approaches in Sect. 2, the paper examines the architectures and innovations in Sect. 3. Section 4 explores the empirical aspects, encompassing model performance assessment, datasets, metrics, and experiments. Section 5 then encapsulates the findings and discussions, presenting the culmination of results. Finally, the paper concludes in Sect. 6 by critically evaluating the implications of the discussed techniques within the context of conversational agents.

2 Background

The goal of KG-to-text generation is to create comprehensible sentences in natural language based on knowledge graphs (KGs), all while upholding semantic coherence with the KG triplets (Fig. 1). The term "knowledge graph" has been in existence since 1972, but its current definition can be attributed to Google's introduction of its Knowledge Graph in 2012 [5]. This marked the beginning of a trend, with numerous companies like Airbnb, Amazon, eBay, Facebook, IBM, LinkedIn, Microsoft, and Uber also making similar announcements. This collective push has led to widely adopting knowledge graphs across various industries [21]. Consequently, academic research in this domain has experienced a notable upsurge in recent years, resulting in many scholarly publications focused on knowledge graphs [21]. These graphs employ a data model based on graphs to efficiently manage, integrate, and extract valuable insights from large and diverse datasets [38].

The problem formulation is the following. Given the input KG G, which is composed of $\{(\mathrm{h}_1, r_1, t_1), \ldots, (\mathrm{h}_n, r_n, t_n) | \mathrm{h}_*, t_* \in \mathrm{E}, r_* \in \mathrm{R}\}$, where E denotes the entity set and R represents the relation set, The objective of the KG-to-text generation task is to produce a coherent and logically sound sequence of text $T =< \mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_n >, \mathrm{t}_k \in \mathrm{V}$ where V denotes the vocabulary composed of n output tokens.

Unlike the conventional text generation task (Seq2Seq), generating text from a knowledge graph adds the extra challenge of ensuring the accuracy of the words within the generated sentences. The current methods can be classified into three distinct categories, as illustrated in (Fig. 1). We will later delve into these categories in more comprehensive detail in the Sect. 3:

1. **Linearization** with Sequence-to-Sequence (Seq2Seq): This involves converting the graph G into a sequence $\mathrm{G}_{\mathrm{linear}} = <\mathrm{g}_1, \mathrm{g}_2, \ldots, \mathrm{g}_\mathrm{m}>$ consisting of m input tokens for the sequence-to-sequence model.

2. **Graph Neural Networks** (GNNs) [47]: These models encode the topological structures of a graph and learn entity representations by aggregating the features of entities and their neighbors. GNNs are not standalone; they require a decoder to complete the encoder-decoder architecture.
3. **Graph Transformer** (GT): This is an enhanced version of the original transformer [52] model adapted to handle graph data.

Graph-to-Text (G2T) leverages graph embedding techniques and Pre-trained Language Models (PLMs). Graph embeddings and PLMs are essential for distinct reasons, playing crucial roles in G2T tasks. Graph embeddings allow us to capture subtle relationships between entities and properties in a numerical format, facilitating the manipulation of this data and the creation of generative models. Generating text and establishing alignments between source entities/relationships and target tokens is challenging for standard language models due to limited parallel graph-text data availability. The following two sections deal with these topics.

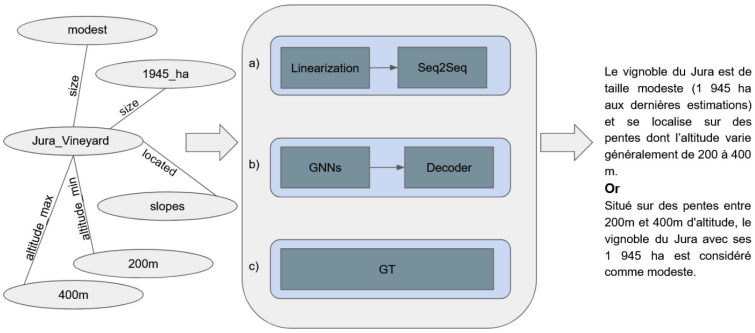


Fig. 1. The architecture of KG-to-text generation with the three categories of representation: a) Linearization + Seq2Seq, b) GNNs with decoder (e.g., LSTM), and c) Graph Transformer (GT)

2.1 Graph Embeddings

Graph embeddings in Graph Neural Networks (GNNs) refer to low-dimensional vec Graph embeddings in Graph Neural Networks (GNNs) involve creating concise node representations through iterative information exchange [9]. These embeddings are crucial for GNN performance. Translational Models in Knowledge Graph (KG) embeddings like TransE [3] and TransH [56] use distance-based scoring functions. Message Passing Neural Network (MPNN) [16] pioneered differentiable architectures for graphs [11], updating node states through message passing. Methods like ERNIE [57] and KnowBERT [40] employ techniques such as TransE [3] and Tucker [1] for link prediction, offering a baseline for knowledge graph encoding.

2.2 Mainstream Architectures

Research focuses on developing pre-trained language models tailored for various domains. The typical method involves converting input graphs into text sequences and

fine-tuning pre-trained seq2seq models. These models leverage encoder-decoder architectures, with the encoder capturing input sequence context and transmitting it to the decoder for generating output sequences. Encoders can employ diverse structures like RNNs, LSTMs, GRUs, Transformers, etc.

Noteworthy models like BART [26], T5 [44], and GPT [33], rooted in KG-to-text datasets, excel due to self-supervised pre-training on extensive unlabeled text corpora. They outperform complex models in KG-to-text tasks [45], especially after fine-tuning. Modifications include replacing traditional sequential encoders with structure-aware graph encoders such as GCNs and graph-state LSTMs [24, 50], enhancing encoding of input structural data. However, a challenge arises in aligning structured graph data with linear text during decoding. Researchers address this by incorporating sophisticated encoder structures, often utilizing GNNs and GTs to retain structural information from linearized graph inputs [10, 15, 36]. [59] propose a central "planner" component which organizes and generates coherent text from input KG, determining text order, style, and structure. By bridging the gap between structured KG data and natural language text, the planner contributes to fluent and coherent text generation.

Cycle training, encompassing both graph-to-text (G2T) and text-to-graph (T2G) conversions, encounters a hurdle due to limited training data availability, particularly in comparison to tasks like machine translation. To counter this, cycle training techniques simultaneously learn conversions between G2T and T2G utilizing non-parallel graph-to-text data [17, 18, 31], allowing for direct output-input comparison during evaluation. However, traditional cycle-consistent training faces challenges with many-to-one or surjective mappings, leading to errors. Innovative solutions include employing conditional variational autoencoders (CVAEs) [18] to transform surjective mappings into implicit bijections, enhancing diversity and minimizing errors. Another strategy involves Deep ReAder-Writer (DRAW) networks [31], incorporating Reader, Writer, and Reviewer modules. The Reader predicts new links in knowledge graphs (KGs) by considering multi-hop neighborhoods, augmenting KGs, while the Writer encodes KGs using graph neural networks (GNNs) to generate paragraphs. The Reviewer module refines paragraph quality, addressing issues like word repetition and enhancing output diversity. These advancements pave the way for more effective and accurate KG-to-text generation.

3 Knowledge Graph-to-Text Generation Architectures

This section introduces three critical components in the KG-to-text generation task: Graph Linearization (GL), Graph Neural Networks (GNNs), and Graph Transformers (GTs). These elements collectively harness structured data within knowledge graphs. We'll discuss converting complex graphs into linear forms, GNNs' ability to capture relationships, and GTs' specialized handling of graph-based data. This analysis sheds light on their roles and significance across applications.

3.1 Graph Linearization

Graph linearization involves transforming intricate graph-based structures into linear or sequential formats. This rearrangement orders nodes and relationships, creating a

sequence suitable for algorithms, tasks, or models requiring linear input, such as many methods in NLP. One approach involves linearizing the knowledge graph (KG) [15, 45] and using PLMs like GPT, BART, or T5 for seq2seq generation. PLMs can generalize for downstream NLG tasks [27], but most were trained on text data [26, 43], needing more structured input.

Heuristic search algorithms like breadth-first search (BFS) or predefined rules are standard for graph linearization. However, these methods often need more attention to structural information during KG encoding, as they don't explicitly consider relationships between input entities.

Researchers such as [19, 36, 59] introduce different neural planners to compute the input triples order before linearization. [36] employ data-driven scoring for precise plans, considering recommendations and user rules. Others use GCN-based neural planners [59] (Graph Convolution Network), reordering graph nodes for sequential content plans encoded by LSTM-based sequential encoders. Plan-and-pretrain techniques introduced by [19] leverage text planners based on relational graph convolutional networks (R-GCN) [59] and pretrained T5 Seq2Seq models. [23] proposed a joint graph-text learning framework called JointGT.

To address the issue when the input is a sequence of RDF triplets, [10] introduces an encoder model called GTR-LSTM. This model maintains the structure of RDF triplets within a small knowledge graph, enabling it to capture relationships both within individual triplets (intra-triple relations) and between interconnected triplets (inter-triple relations). This approach improves sentence generation accuracy. Unlike TreeLSTM [51], which lacks cycle handling, GTR-LSTM handles cycles using a combination of topological sorting and breadth-first traversal. An attention model is employed to gather comprehensive global information from the knowledge graph. Additionally, unlike GraphLSTM [29], which only supports specific entity relations, all relations are incorporated into the calculation of hidden states [10].

In efforts to preserve the graph's topology, despite striving for maximum retention using seq2seq methods, the Transformer-based seq2seq models come with significant costs, particularly during the pretraining phase. Additionally, the computational expense of linearization can become substantial when dealing with extensive knowledge graphs. Therefore, to more effectively maintain the graph's topology, the introduction of Graph Neural Networks (GNNs) has been proposed, and their details will be explored in the following section.

3.2 Graph Neural Networks (GNNs)

Various approaches employ different versions of Graph Neural Network (GNN) architectures for processing graph-structured data. GNNs, including Graph Convolutional Networks (GCNs) [24], extended forms like Syn-GCNs [34] and DCGCNs [20], Graph Attention Networks (GATs) [54], and Gated Graph Neural Networks (GGNNs) [6, 7, 28], are well-suited for modeling entity relationships within knowledge graphs to generate text. GNNs have demonstrated promise in knowledge graph-to-text generation by effectively representing relationships between entities in a knowledge graph.

GCNs, Syn-GCNs, and DCGCNs encode entity relationships into low-dimensional representations. GATs dynamically assign weights to entities and relationships to influence text generation. GGNNs utilize gating mechanisms to control information flow among graph nodes, aiding context incorporation. Some studies combine GNNs with reinforcement learning to optimize text generation based on the knowledge graph [6].

Overall, using GNNs for knowledge graph-to-text generation is an active area of research, with many recent studies exploring different architectures and training methods. The results suggest that GNNs can effectively capture the relationships between entities in a knowledge graph and generate high-quality text based on that information. The limitation of KG-to-Text generation with GNNs is that GNNs can be computationally expensive and may need help handling large knowledge graphs. Additionally, their performance may degrade for graphs with complex relationships or structures. Despite these limitations, GNNs remain a promising direction for knowledge graph-to-text generation.

3.3 Graphs Transformers (GTs)

Recent works have proposed to adapt the Transformer architecture to benefit from the power of models based on Transformer to model tree or graph-type data structures as with GNNs and to overcome the limitations of local neighborhood aggregation while avoiding strict structural inductive biases. As Graph Transformers are equipped with self-attention mechanisms, they can capture global context information by applying these mechanisms to the graph nodes.

According to [4], GT differs from GNNs because it allows direct modeling of dependencies between any pair of nodes regardless of their distance in the input graph. An undesirable consequence is that it treats any graph as a fully connected graph, significantly reducing the explicit structure of the graph. To maintain a structure-aware view of the graph, their proposed model introduces an explicit relationship encoding and integrates it into the pairwise attention score computation as a dynamic parameter.

From the GNNs pipeline, if we make several parallel heads of neighborhood aggregation and replace the sum on the neighbors by the attention mechanism, e.g., a weighted sum, we would get the Graph Attention Network (GAT). Adding normalization and MLP feed-forward, we have a Graph Transformer [22]. For the same reasons as Graph Transformer [32] presents the K-BERT model, they introduce four components to augment the Transformer architecture and to be able to handle a graph as input. The first knowledge layer component takes a sentence and a set of triplets as input and outputs the sentence tree by expanding the sentence entities with their corresponding triplets. They also add the Seeing Layer component to preserve the original sentence structure and model the relationships of the triples by building a Visibility Matrix. Another component is the Mask-Transformer, where they modify the self-attention layer to consider the visibility matrix when calculating attention.

The use of Graph Transformers for Knowledge graph text generation has gained popularity in recent years [25, 48] due to their ability to effectively handle graph structures and capture the relationships between nodes in the graph. Additionally, Graph Transformers can handle large graphs and can model long-range dependencies between nodes in the graph. Despite the advantages, the training of Graph Transformers can be computationally expensive, and the interpretability of the model still needs to be improved.

Overall, using Graph Transformers for Knowledge graph-to-text generation is a promising area of research and can lead to significant improvements in the text generation from knowledge graphs.

4 Experiments on Some Models and Datasets

4.1 Datasets and Metrics

We turn our attention to the datasets utilized in the implementation of various models. During the training and validation phases, the utilized datasets align with those employed by each specific model [12, 15, 37, 49] (Table 1). However, when transitioning to the testing and evaluation phase, a significant portion of the models is subjected to assessment using the Webnlg v.2 dataset [49], thereby providing a standardized and consistent benchmark for performance evaluation. Our evaluation approach needs to be revised in light of the current challenges surrounding the availability of a Webnlg v.2 dataset tailored for the CycleGT [17] model and P2 [19] model. Instead, we conduct the evaluation of these models on the test dataset corresponding to the same version as the utilized training and validation datasets. These approaches are necessitated by the absence of a dedicated Webnlg v.2 dataset compatible with these models at this juncture.

Table 1. Datasets used in our experiments

Dataset	Train	Validation	Test
WebNLG V.2 2018 [49]	34.3k	4.3k	4.2k
WebNLG 2017 [15]	18.1k	0.87k	1.8k
WebNLG 2020 [12]	35.4k	4.4k	5.1k
DART [37]	62.6k	6.9k	12.5k

In our conducted experiments, we employed a set of ngram-based metrics, namely Rouge [30], Meteor [2], Bleu [39], and Cider [53], to assess the effectiveness of our developed model. The selection of these four distinct metrics aims to analyze various aspects of the evaluated text comprehensively. These metrics encompass precision (Bleu), recall (Rouge), and F-score (Bleu and Rouge), as well as the consideration of term frequency-inverse document frequency (TF-Idf) aspects (Cider). This approach comprehensively evaluates our model's performance across multiple linguistic dimensions.

4.2 Experiments

We empirically assessed several approaches executed on our local machines and evaluated them using the same test dataset. Our evaluation encompassed various types of systems, including graph linearization [19, 23, 45], a model employing Graph Neural Networks (GNNs) [46], and another using Graph Transformers (GTs) [48]. Additionally, we implemented an unsupervised cycling model [17]. While preserving most of the

original characteristics of these models, we occasionally adjusted hyperparameters for resource limitations or faster training.

Graph Linearization Approaches. Here, we present models that use PLMs as seq2seq models to do KG2T generation.

In the "JointGT" approach [23], T5 and BART served as foundational models. These models underwent a pretraining phase on the KGTEXT dataset [8] and were further refined through tasks involving text and graph reconstruction as well as alignment prediction. This comprehensive process resulted in the creation two pretrained models known as JointGT_T5 and JointGT_BART. In our implementation, we chose to work with JointGT_T5, maintaining the original fine-tuning hyperparameters while making a batch size adjustment to 16 during both training and prediction phases to accommodate computational limitations.

The adaptation of BART and T5 for graph-to-text generation by adding <H>, <R>, and <T> tokens to the models' vocabulary [45] led to vocabulary extension for KG datasets and fine-tuning on WebNLG 2017 and augmented dataset DART [37]. We maintained their implementation with slight modifications, increasing the batch size to 32 for faster fine-tuning.

[19] employed the T5-Large model and the planner module using DGL, Pytorch, and Transformers. Most hyperparameters were retained, with the batch size adjusted to 1 due to resource limitations.

GNNs Approaches. [46] implemented their models using PyTorch Geometric (PyG) and OpenNMT-py, employing byte pair encoding (BPE) for entity word segmentation. Our implementation maintained their hyperparameters, utilizing cge-lw (Parallel Graph Encoder - layer-wise) as the graph encoder where global and local node representations are concatenated layer-wise.

GTs Approaches. Graformer [48] introduced additional preprocessing steps, including training a BPE vocabulary and incorporating specialized tags for entities and relations. Their training implemented an Epoch curriculum strategy. Our implementation mirrored their hyperparameters.

Unsupervised Cycling Approaches The unsupervised CycleGT model [17] introduced a non-parallel training and validation dataset by segregating text and graph data. In our implementation, we adhered to CycleGT's GitHub repository specifications.

Overall, our experiments provide comprehensive insights into the behaviors and performance of these diverse approaches within the context of graph-to-text generation, and we discuss the results (Table 1) in the next section.

5 Results and Discussion

Each architecture has advantages and disadvantages, and the choice of architecture will depend on the specific requirements of the actual task.

In light of these elements, and with the constraint of the data labialization for specific domains of KG2T generation, we choose to go further with seq2seq Transformer based models (PLMs) in our Knowledge Graph-to-Text Generation.

We see in Table 2 that all models using seq2seq Transformer-based models have better results than GNNs and GTs models, and with a considerable margin when these models have some additional processes before linearization to keep some information about the structure of the graph like in JointGT [23] and P2 [19]. Even the CycleGT model has better results than GNNs and GTs models, an unsupervised model that uses a linearization phase on the graph.

As mentioned before, we evaluate our models on WebNLG V.2 [49], which has 1600 test instance (source KGs), except P2 and CycleGT, which are evaluated on the enriched version of WebNLG V.2 [49] and which has 1860 test instance.

6 Conclusion and Perspectives

In this paper, we have explored and compared three distinct architectures for Knowledge Graph-to-text generation: Graph Neural Networks (GNNs), Graph Transformers (GTs), and linearization with seq2seq models, primarily Pre-trained Language Models (PLMs). Each of these architectures has its unique advantages and limitations, making them suitable for different scenarios and use cases.

GNNs offer a flexible and scalable approach to model graph structures and relationships, but they may need help with efficiency when handling large and complex knowledge graphs. GTs, on the other hand, provide a specialized solution for graph-based tasks, offering a more direct and efficient way to process graph structures. However, they may require extensive training data and computational resources.

Linearization with seq2seq models, especially those based on PLMs, simplifies the process by converting knowledge graphs into linear sequences and generating text from them. Despite its simplicity, this approach can lose some structural information during linearization. However, as our experiments show, seq2seq Transformer-based models consistently outperformed GNNs and GTs, especially when models incorporate additional processes to retain graph structure information like JointGT and P2.

In the context of conversational agents, where factors like response time and correctness are critical, the inference time and resource requirements of GNNs and GTs can be limiting. Hence, for practical deployment, seq2seq Transformer-based models stand out as a more feasible choice, given their superior performance and efficiency.

Looking ahead, the field of Knowledge Graph-to-text generation presents several avenues for advancement. Firstly, there's a pressing need to enhance the computational efficiency of Graph Neural Networks (GNNs) and Graph Transformers (GTs) to make them more suitable for real-time applications. This involves optimizing their architectures, parallelizing computations, and harnessing hardware accelerators. Additionally, as the scale and complexity of knowledge graphs continue to grow, developing strategies to effectively handle large graphs while maintaining performance is a significant challenge that warrants further exploration.

Moreover, extending these approaches to accommodate multimodal knowledge graphs, which integrate textual, visual, and other data types, could open up new horizons for comprehensive information retrieval and generation, especially with the use of Meta-transformers multi-modal approach [58]. Furthermore, ensuring the interpretability of GNNs and GTs is crucial for building trust in generated text, particularly in domains

like healthcare and law. Moreover, addressing the scarcity of labeled data for specific knowledge domains through innovative data labeling and augmentation techniques can enhance the training of domain-specific models. Lastly, the advancement of these models to handle multiple languages and facilitate cross-lingual knowledge transfer holds promise for their broader applicability in diverse linguistic contexts.

In conclusion, the choice of architecture for Knowledge Graph-to-text generation should be guided by the specific requirements of the task and the constraints of the application. While GNNs and GTs offer valuable approaches for particular scenarios, the efficiency and performance of seq2seq Transformer-based models make them a compelling choice for many real-world applications. Future research should address the challenges and opportunities presented by these diverse architectures to advance the field further.

Table 2. The performance of the implemented models on WebNLG V.2 [49]

		Metric	BLEU				METEOR	ROUGE_L	CIDEr
		Model	1	2	3	4			
Graph Linearization	Supervised	PLMs-G2T (webnlg) [45]	79.18	67.66	57.31	48.54	40.28	64.91	3.34
		PLMs-G2T (DART) [45]	83.45	72.77	63.22	55.03	43.20	68.95	3.82
		P2_35 [19]	86.91	76.30	66.49	57.95	44.36	70.31	**4.98**
		JointGT [23]	**90.35**	**81.82**	**73.60**	**66.14**	**47.32**	**75.96**	4.59
	Unsupervised	CycleGT [17]	74.79	62.88	53.68	46.16	36.21	68.26	3.20
GNNs		KG2T_G&L [46]	60.93	49.36	40.99	34.42	28.11	49.04	2.02
GTs		Graformer [48]	70.67	59.72	51.12	44.22	33.82	58.13	2.79

Acknowledgement. The authors thank the French company DAVI (Davi The Humanizers, Puteaux, France) for their support and the French government for the plan France Relance funding.

References

1. Balažević, I., Allen, C., Hospedales, T.M.: Tucker: tensor factorization for knowledge graph completion. arXiv preprint arXiv:1901.09590 (2019)
2. Banerjee, S., Lavie, A.: METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization (2005)
3. Bordes, A., et al.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, vol. 26 (2013)

4. Cai, D., Lam, W.: Graph transformer for graph-to-sequence learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 05 (2020)
5. Chaudhri, V., et al.: Knowledge graphs: introduction, history and perspectives. AI Mag. **43**(1), 17–29 (2022)
6. Chen, Y., Wu, L., Zaki, M.J.: Reinforcement learning based graph-to-sequence model for natural question generation. arXiv preprint arXiv:1908.04942 (2019)
7. Chen, Y., Wu, L., Zaki, M.J.: Toward subgraph-guided knowledge graph question generation with graph neural networks. IEEE Trans. Neural Netw. Learn. Syst., 1–12 (2023)
8. Chen, W., et al.: KGPT: knowledge-grounded pre-training for data-to-text generation. arXiv preprint arXiv:2010.02307 (2020)
9. Dai, Y., et al.: A survey on knowledge graph embedding: approaches, applications and benchmarks. Electronics **9**(5), 750 (2020)
10. Distiawan, B., et al.: GTR-LSTM: a triple encoder for sentence generation from RDF data. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (2018)
11. Duvenaud, D.K., et al.: Convolutional networks on graphs for learning molecular fingerprints. In: Advances in Neural Information Processing Systems, vol. 28 (2015)
12. Ferreira, T.C., et al.: Enriching the WebNLG corpus. In: Proceedings of the 11th International Conference on Natural Language Generation (2018)
13. Ferreira, T.C., et al.: Neural data-to-text generation: a comparison between pipeline and end-to-end architectures. arXiv preprint arXiv:1908.09022 (2019)
14. Fu, Z., et al.: Partially-aligned data-to-text generation with distant supervision. arXiv preprint arXiv:2010.01268 (2020)
15. Gardent, C., et al.: The WebNLG challenge: generating text from RDF data. In: Proceedings of the 10th International Conference on Natural Language Generation (2017)
16. Gilmer, J., et al.: Neural message passing for quantum chemistry. In: International Conference on Machine Learning. PMLR (2017)
17. Guo, Q., et al.: CycleGT: unsupervised graph-to-text and text-to-graph generation via cycle training. arXiv preprint arXiv:2006.04702 (2020)
18. Guo, Q., et al.: Fork or fail: cycle-consistent training with many-to-one mappings. In: International Conference on Artificial Intelligence and Statistics. PMLR (2021)
19. Guo, Q., et al.: P2: a plan-and-pretrain approach for knowledge graph-to-text generation: a plan-and-pretrain approach for knowledge graph-to-text generation. In: Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+) (2020)
20. Guo, Z., et al.: Densely connected graph convolutional networks for graph-to-sequence learning. Trans. Assoc. Comput. Linguist. **7**, 297–312 (2019)
21. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. (Csur) **54**(4), 1–37 (2021)
22. Joshi, C.: Transformers are graph neural networks. The Gradient 7 (2020)
23. Ke, P., et al.: JointGT: graph-text joint representation learning for text generation from knowledge graphs. arXiv preprint arXiv:2106.10502 (2021)
24. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
25. Koncel-Kedziorski, R., et al.: Text generation from knowledge graphs with graph transformers. arXiv preprint arXiv:1904.02342 (2019)
26. Lewis, M., et al.: Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461 (2019)
27. Li, J., et al.: Pretrained language models for text generation: A survey. arXiv preprint arXiv:2201.05273 (2022)
28. Li, Y., et al.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)

29. Liang, X., Shen, X., Feng, J., Lin, L., Yan, S.: Semantic object parsing with graph LSTM. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016, Part I. LNCS, vol. 9905, pp. 125–143. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_8
30. Lin, C.-Y.: Rouge: a package for automatic evaluation of summaries. Text summarization branches out (2004)
31. Liu, L., et al.: How to train your agent to read and write. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 15 (2021)
32. Liu, W., et al.: K-BERT: enabling language representation with knowledge graph. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 03 (2020)
33. Mager, M., et al.: GPT-too: a language-model-first approach for AMR-to-text generation. arXiv preprint arXiv:2005.09123 (2020)
34. Marcheggiani, D., Frolov, A., Titov, I.: A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. arXiv preprint arXiv:1701.02593 (2017)
35. Moon, S., et al.: OpenDialKG: explainable conversational reasoning with attention-based walks over knowledge graphs. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (2019)
36. Moryossef, A., Goldberg, Y., Dagan, I.: Step-by-step: separating planning from realization in neural data-to-text generation. arXiv preprint arXiv:1904.03396 (2019)
37. Nan, L., et al.: DART: Open-domain structured data record to text generation. arXiv preprint arXiv:2007.02871 (2020)
38. Noy, N., et al.: Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. Queue **17**(2), 48–75 (2019)
39. Papineni, K., et al.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (2002)
40. Peters, M.E., et al.: Knowledge enhanced contextual word representations. arXiv preprint arXiv:1909.04164 (2019)
41. Puduppully, R., Dong, L., Lapata, M.: Data-to-text generation with content selection and planning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33. no. 01 (2019)
42. Puzikov, Y., Gurevych, I.: E2E NLG challenge: neural models vs. templates. In: Proceedings of the 11th International Conference on Natural Language Generation (2018)
43. Radford, A., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
44. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(1), 5485–5551 (2020)
45. Ribeiro, L.F.R., et al.: Investigating pretrained language models for graph-to-text generation. arXiv preprint arXiv:2007.08426 (2020)
46. Ribeiro, L.F.R., et al.: Modeling global and local node contexts for text generation from knowledge graphs. Trans. Assoc. Comput. Linguist. **8**, 589–604 (2020)
47. Scarselli, F., et al.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)
48. Schmitt, M., et al.: Modeling graph structure via relative position for text generation from knowledge graphs. arXiv preprint arXiv:2006.09242 (2020)
49. Shimorina, A., Gardent, C.: Handling rare items in data-to-text generation. In: Proceedings of the 11th International Conference on Natural Language Generation (2018)
50. Song, L., et al.: A graph-to-sequence model for AMR-to-text generation. arXiv preprint arXiv:1805.02473 (2018)
51. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 (2015)

52. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
53. Vedantam, R., Lawrence Zitnick, C., Parikh, D.: Cider: consensus-based image description evaluation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
54. Veličković, P., et al.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
55. Wahde, M., Virgolin, M.: Conversational agents: theory and applications. In: Handbook on Computer Learning and Intelligence: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation, pp. 497–544 (2022)
56. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, pp. 1112–1119. AAAI Press (2014)
57. Zhang, Z., et al.: ERNIE: enhanced language representation with informative entities. arXiv preprint arXiv:1905.07129 (2019)
58. Zhang, Y., et al.: Meta-transformer: a unified framework for multimodal learning. arXiv preprint arXiv:2307.10802 (2023)
59. Zhao, C., Walker, M., Chaturvedi, S.: Bridging the structural gap between encoding and decoding for data-to-text generation. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (2020)

Network Embedding Based on DepDist Contraction

Emanuel Dopater, Eliska Ochodkova(✉), and Milos Kudelka

VSB - Technical University of Ostrava, 17. listopadu 2172/15, 708 00 Ostrava-Poruba, Czech Republic
{emanuel.dopater.st,eliska.ochodkova,milos.kudelka}@vsb.cz

Abstract. Networks provide a very understandable representation of the data in which we get information about the relationships between pairs of nodes. For this representation, we can use one of the powerful analytical tools, 2D visualization. In visualization, networks have an alternative vector representation to which a wide range of machine learning methods can be applied. More generally, networks can be transformed to a low-dimensional space by network embedding methods. In this paper, we present a new embedding method that uses a non-symmetric dependency to find the distance between nodes and applies an iterative procedure to find a satisfactory distribution of nodes in space. Using experiments with small networks and dimension 2, we show the effectiveness of this method and discuss its properties.

Keywords: network embedding · non-symmetric dependency · dependency distance

1 Introduction

Finding an efficient representation of network data is crucial for efficient network data processing, because traditional network data representations are computationally intensive, have low parallelizability, or, e.g., cannot use machine learning methods [2]. Therefore, attention is now being paid to the development of new methods for network embedding, which means transforming the original network space into a low-dimensional vector space. The fundamental problem is to learn a mapping function between these two spaces. In the network embedding space, the relationships among the nodes is captured by the distances (or similarities) between nodes in the vector space, and the structural characteristics of a node are encoded into its embedding vector.

Network embedding supports network processing and analysis, such as community (cluster) detection, network visualization, or link prediction [17] and it allow us to make general machine learning techniques applicable to networks. For the embedding space there are two requirements for network embedding. First, the original network can be reconstructed from the learned embedding space. If there is an edge between two nodes, their distance in the embedding space must be relatively small, as this preserves the relationships in the network well.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 427–439, 2024.
https://doi.org/10.1007/978-3-031-53468-3_36

Second, the learned embedding space can effectively support network inference, such as predicting unseen links or identifying important nodes [2].

In this paper, we present a new network embedding method based on calculating the distance between pairs of network nodes based on their structurally non-symmetric relationship. We describe an iterative procedure that, using the distance defined in this way, quickly reveals the community structure. We perform experiments with four well-known small networks and show the results of the application of our method to dimension 2.

2 Related Work

Different models can be used to transform networks from the original network space to the embedding one, working with different types of information or addressing different goals. Commonly used models include matrix factorization, where, e.g., Singular Value Decomposition is used due to its optimality for the low-rank approximation of the adjacency matrix [15] and non-negative matrix factorization is often used due to its advantages as an additive model [19].

Furthermore, random walk models, analogous to Word2Vector, are used to generate random paths through the network. If a node is considered as a word, the random path can be considered as a sentence, and the neighborhood of the node can be identified using a measure of cooccurrence as in the case of Word2Vector [12]. For example, the Node2Vec embedding method [6], similar in design principle to the DeepWalk method [14], can be considered. However, Node2Vec improves the random walk generation in DeepWalk and mirrors the depth and breadth sampling properties to enhance the network embedding effect.

Finally, let us mention deep neural networks and their variants because they are a suitable choice if we are looking for an efficient model for learning nonlinear functions. Representative methods include SDNE [17] or, e.g., SiNE [18].

One of the important applications for network embedding is visualization of a network in two-dimensional space. We can find a comparison of visualization results with different embedding approaches in [10]. Classes of graph drawing algorithms, including multi-level and dimensionality reduction-based techniques, are described in detail in a review [5]. In network analysis fields, interpretation and understanding of network structure may be based on calculating local or global measures. Visual representation of network structure can help detect, understand, and identify unexpected patterns or outliers in networks.

The layout and arrangement of nodes affect how the user perceives relationships in the network. There is no one best way; the layout of a network depends on which network features are important to us. These may be, for example, specific measures of centrality or important properties of nodes or edges. Criteria for evaluating a network layout include the algorithm's computational complexity, the network's size, the algorithm's ability to follow certain layout rules or aesthetics, clustering, etc.

Many of the methods used are based on the force-directed network paradigm, a paradigm of modeling the network as a physical object system where nodes

attract and repel according to some force. Other network drawing algorithms are methods using multilevel and dimensionality reduction-based techniques.

There are two approaches to force-directed layouts: those based on spring embedding and those that solve optimization problems. A very often used method of this type is the method of Fruchterman and Reingold [4], the connected nodes attract each other while all other nodes, modeled as electrical charges, repel each other.

The second approach considers the layout problem as an optimization problem that minimizes an energy function designed concerning the properties of the network being visualized. Important energy-based techniques are Noack's LinLog [13] and ForceAtlas [8] layouts. Noack's edge repulsion model removes the bias of the node model towards attraction by ensuring that nodes that are strongly attracting are also strongly repelling, similarly for nodes with weak attraction. Therefore, nodes with a high degree are less likely to be clustered in the center of the network, and it is able to show any underlying clustering structure in the network. ForceAtlas is strongly associated with Noack's LinLog. Its advantage is that all nodes are subject to at least some repulsive force, and poorly connected nodes are thus approximated by well-connected nodes, reducing visual clutter. The forces in the algorithm vary between Noack's edge repulsion model and the Fruchterman and Reingold distributions.

Multilevel algorithms are one of the options that can be used to streamline force-directed techniques. Their idea is to find a sequence of coarser representations of the network, optimize the drawing in the coarsest representation, and propagate this distribution back to the original network. The coarser representations are created by composing connected nodes whose edges become the union of the edges of all the nodes [7].

Other options for drawing networks are dimension reduction techniques, including multidimensional scaling, linear dimension reduction [1], or spectral graph drawing approaches. The challenge is to preserve the information in a high-dimensional space and capture it in a lower-dimensional representation. Most dimension reduction techniques used for network layout use the graph-theoretical distance between nodes, [3], as the information to be preserved.

As mentioned above, the most common use cases of node embedding are visualization, clustering, and link prediction. The problem of visualizing networks in 2D, with its long history, and network drawing algorithms are probably the most well-known embedding techniques commonly used to visualize networks in 2D space. Data-driven network layouts, such as spring embedding, are unsupervised methods of arranging nodes based on their connectivity and are de facto dimensionality reduction techniques. Despite the great potential layouts are rarely the basis of systematic network visualization.

Therefore, node embedding offers a powerful new paradigm for network visualization: because nodes are mapped to real-valued vectors, researchers can easily leverage general techniques for visualizing high-dimensional data. For example, node embedding can be combined with well-known techniques such as t-SNE

[11] to create 2D network visualizations [16] that can be useful for revealing communities and other hidden structures.

3 Non-symmetric Structural Dependency

Structural dependency (hereafter referred to as *dependency*) is a non-symmetric relationship between pairs of nodes that applies to both weighted and unweighted networks [9]. In our experiments, we work with both types of networks; however, these are always undirected. For this paper, we formulate the dependency in a slightly different way. First, let us establish a way to determine the weight $w(A, B, X)$ of the relation between two nodes of the network A, B given their common neighbor X. Let $w(A, X)$ be the weight of the edge between nodes A, X and similarly $w(B, X)$ be the weight of the edge between nodes B, X. Then:

$$w(A, B, X) = w(B, A, X) = \frac{w(A,X) \cdot w(B,X)}{w(A,X) + w(B,X)}. \tag{1}$$

The weight defined in this way is half of the harmonic mean, i.e. if the values $w(A, X)$ and $w(B, X)$ are balanced, then the weight $w(A, B, X)$ is around half of $w(A, X)$ and $w(B, X)$ respectively. If, on the other hand, they are not balanced and at least one of the weights $w(A, X), w(B, X)$ is close to zero, then the weight $w(A, B, X)$ is also close to zero.

Now, let us define the strength of the relation between the nodes A, B. If a pair of nodes A, B has multiple common neighbors X_i, then the strength of the relationship between them (the dependency of one on the other) is affected not only by the weights of the edge between these nodes but also by the weights $w(A, B, X_i)$. Therefore, let us define the dependency $D(A, B)$ of a node A on a node B as follows:

$$D(A, B) = \frac{w(A,B) + \sum_{X_i \in \Gamma(A,B)} w(A,B,X_i)}{\sum_{X_j \in N(A)} w(A,X_j)}, \tag{2}$$

where $\Gamma(A, B)$ is the set of common neighbors of nodes A, B and $N(A)$ is the neighborhood (set of all neighbors) of node A.

If there is an edge between nodes A, B, then $w(A, B)$ is the weight of this edge; otherwise, $w(A, B) = 0$. Thus the dependency is non-zero if and only if the node A, B have an edge or at least one common neighbor. A dependency defined in this way is non-symmetric, so $D(A, B) = D(B, A)$ generally does not hold. While the value of the numerator is the same in both directions of the dependency, the value of the denominator may be different. Therefore, it may be true that the dependencies between the nodes of A, B may be substantially different.

Informally speaking, dependency is high if a node is significantly connected to its neighbor through common neighbors compared to the rest of its neighbors. This property provides information about the network's community structure since nodes in a community should have stronger dependencies with each other than with nodes outside the community.

3.1 Distance Based on Non-symmetric Dependency

The unanswered question is, what distance should the two nodes of the network be if we want to start from the exact dependencies Two situations can arise: (1) nodes have zero dependencies and thus have neither an edge nor a common neighbor, and (2) nodes have non-zero, potentially non-symmetric dependencies. In the first case, we have no straightforward information to determine the distance. In the second case, we have to convert the dependencies into the Euclidean space that is, by definition, symmetric. For further considerations, let us start with a simple interpretation of dependency, which can be described as a relation that attracts two nodes together. To express this relation, let us define the mutual dependency coefficient $q_S(A, B)$ as the product of the partial dependencies of the nodes A, B with their arithmetic mean, i.e.:

$$q_S(A, B) = D(A, B) \cdot D(B, A) \cdot \frac{D(A,B)+D(B,A)}{2} \tag{3}$$

The coefficient q takes into account both dependencies and, thanks to the average, information about their balance. The coefficient q_s can be further used to determine the symmetric distance between nodes A, B. An alternative is to work with the non-symmetric distance and leave the determination of the symmetric distance to the iterative procedure described in Sect. 4. In this case, the assumed distance between the nodes may be non-symmetric at the input. For this case, let us define the coefficient $q_N(A, B)$:

$$q_N(A, B) = D(A, B)^2 \cdot D(B, A) \tag{4}$$

The values of $q_S(A, B), q_N(A, B)$ are from the interval $[0, 1]$ and severely penalize situations where at least one of the dependencies is very low. Our experiments show that using both alternatives of the q coefficient provides the same result; however, the non-symmetric version converges more quickly to a stable result. Therefore, $q(A, B) = q_N(A, B)$ will hold for the following.

Now we can define the maximum distance $maxDepDist$ between pairs of network nodes, from which we derive the distance of node A from node B with non-zero dependencies as follows:

$$DepDist(A, B) = (1 - q(A, B)) \cdot maxDepDist. \tag{5}$$

Note that we cannot determine this distance based on non-symmetric dependency (*DepDist* for short) between absolutely independent nodes. In the following, we show that we can still use such an incompletely formulated distance for network embedding.

4 DepDist Contraction

As mentioned above, the essence of network embedding is to find a network representation in low-dimensional space in which the relationships between network nodes are highly preserved. We next present an iterative procedure based on a

straightforward use of DepDist that provides such a representation. We refer to this procedure as *DepDist Contraction*[1], and this is because its essence is to bring pairs of nodes closer together so that the result is close to the distance based on their mutual dependencies.

Even though we are concerned with network embedding and the presented procedure is independent of the chosen dimension, we focus our experiments only on dimension 2. This allows us to visualize the DepDist Contraction result and at least visually assess it.

Remark In the following, we will use the term node A to mean both a node of the network and a point representing this node in n-dimensional space.

4.1 Algorithm

The first step of the algorithm to find the representation of the network in n-dimensional space is to randomly distribute the points representing each network node into a cube of dimension n with edge length a. Next, we set the value of $maxDepDist$ to be much smaller than the edge length a (so there is enough space for contraction). We then iterate so that in one iteration, each node A moves in space to some node B (we will return to the selection of node B later). For the move, step length corresponds to the distance between A and B and their coefficient $q(A, B)$. The iterating terminates, as we show later, either after a fixed number of steps or after the contraction stabilizes.

4.2 One Step of Iteration

Let us consider a node A, a node B selected for it, their coefficient $q(A, B)$, and their expected $DepDist(A, B)$. By one iteration step, we mean moving node A to node B so that their distance approaches $DepDist(A, B)$. Let $\hat{u}$ be a unit vector in the direction of the vector $B - A$. The new position A' of node A is:

$$A' = A + acc(A, B) \cdot (\parallel B - A \parallel - DepDist(A, B)) \cdot \hat{u}. \tag{6}$$

The function $acc(q(A, B))$ changes the effect of the coefficient $q(A, B)$ on the length of the move. The function is designed to increase the move length significantly when the distance between nodes A, B is too large (above some defined threshold), i.e., when $\parallel B - A \parallel$ is significantly larger than $DepDist(A, B)$. On the other hand, the move length decreases with decreasing distance of nodes, which gradually stabilizes node positions in space when node positions change negligibly. Therefore, we define a maximum threshold distance for acceleration $maxAccDist > maxDepDist$. Next, for each pair of nodes A, B, we determine the threshold distance for acceleration:

$$accDist(A, B) = (1 - q(A, B)) \cdot maxAccDist. \tag{7}$$

[1] Non-parallel Python implementation used for the experiments in this paper is at https://github.com/emanueldopater/DepDistContraction/tree/conference.

Based on this threshold distance for acceleration, we then define the acceleration coefficient $accCoef$ to be equal to one for $accDist(A, B) = \| B - A \|$:

$$accCoef(A, B) = 0.5 + 0.5 \cdot \frac{\|B-A\|}{accDist(A,B)}. \tag{8}$$

The acc function is then defined as:

$$acc(A, B) = q(A, B)^{\frac{1}{accCoef(A,B)}}. \tag{9}$$

Thus, in general, pairs of nodes that are weakly dependent on each other are farther apart than strongly dependent nodes, slowly converging to the expected distances $DepDist(A, B)$ and $DepDist(B, A)$, respectively. Thus, for example, two high-degree nodes that share a common edge but have very few common neighbors compared to their other neighbors will hardly change position during an iteration. This contrasts with, for example, nodes that are part of a large and almost disjoint clique, which have strong dependencies and thus small distances to neighbors that they move to very quickly.

More interesting is the situation when node A is strongly dependent on node B, but the reverse is not true, i.e., when, for example, node B is a hub and node A has degree 1. Using the non-symmetric alternative $q_N(A, B)$, node A will approach node B very fast, and node B will slowly move towards node A.

4.3 Selecting Node to Move

In one iteration, for each node A, a different node B is selected, to which node A is moved according to the procedure described above; it is, therefore, necessary to determine how node B can be selected. As described above, non-zero dependency can only be calculated for nodes with a common edge or at least one common neighbor; therefore, this assumption limits the selection. For DepDist Contraction, we use a strategy based on the assumption that the fewer neighbors a node has, the less information we have about its neighborhood, and we should "look further." For node A, we therefore set the probability of random selection of its neighbor B to be related to the degree of node A:

$$p(A) = 1 - \frac{1}{1+k(A)}, \tag{10}$$

where $k(A)$ is the degree of node A; with complementary probability $1 - p(A)$, a neighbor of the neighbors of A is then chosen at random. Thus, if a node has a very high degree, its neighbor is chosen with near certainty, and conversely, if a node has degree $k(A) = 1$, then its neighbor is chosen with probability 0.5.

4.4 When to Stop Iterating

The algorithm depends on only three parameters: (1) the edge length a of the n-dimensional cube into which the nodes of the network are randomly distributed at the beginning, (2) the maximum expected distance $maxDepDist$ of the nodes in the embedding from which the distances between each pair of nodes are

derived, and (3) the maximum distance $maxAccDist$ for the acceleration of the move from which the distance above which the move accelerates and below which it decreases is derived for each pair of nodes. Thus, from the perspective of the whole network, it is a contraction that results in a distribution of nodes in a small part of the input n-dimensional cube in which the nodes almost stop moving. Our experiments show that, regardless of the size of the network, after $20 - 50$ iterations, the community structure emerges (strongly dependent node groups), and after $200 - 500$ iterations, the distribution changes very little; groups of strongly dependent nodes move (relatively) away from each other, and the distribution stabilizes. Thus, the number of iterations needed is not much affected by the network size because the algorithm efficiently separates locally strongly connected sub-structures from the rest of the network. The strength of the DepDist Contraction algorithm is, therefore, most evident when applied to networks with significant community structure, and its discovery in embedding is only a side effect of the DepDist.

Figure 1 visualizes the distribution of nodes after 50 iterations of the four networks we used in our experiments; it is a 2D embedding, which is complemented by the edges between the nodes and the sizes of the nodes corresponding to their degree for better clarity. As can be seen, even after a relatively small number of iterations, the community structure of the networks is obvious.

4.5 Scalability

Calculating the dependency of one node on another is similar to calculating the clustering coefficient and has time complexity $O(k^2)$, where k is the average degree of the network (we can calculate the dependencies in both directions simultaneously). However, the computations for each pair of nodes are independent and can be computed in parallel as needed. Within a single iteration, storing the current node positions at the beginning and computing the dependencies including moving the nodes to their new positions in parallel is possible. When the iteration ends, the current positions are swapped with the new ones. Thus, during the algorithm, we work with two states of the network (current and new node positions); therefore, the spatial complexity is $O(N)$, where N is the number of nodes in the network.

To estimate the time complexity, we assume a sparse network for which the relationship between the number of edges and nodes is $O(N)$. If we want to optimize the computational complexity, we must continuously compute the dependencies during the iterative procedure (i.e., only when necessary) and store them for reuse. Thus, the estimate of the time complexity of computing all dependencies is based on the dependency computation complexity and the total number of node pairs for which the dependency must be computed. Given that we compute dependencies for neighbors and neighbors of neighbors based on random selection, we can estimate the time complexity of computing all required dependencies to be $O(Nk^4)$; however, this is the worst case, where we assume computing dependencies for all neighbors and neighbors of neighbors for all nodes in the network. Moreover, for sparse networks in the case of stored dependencies,

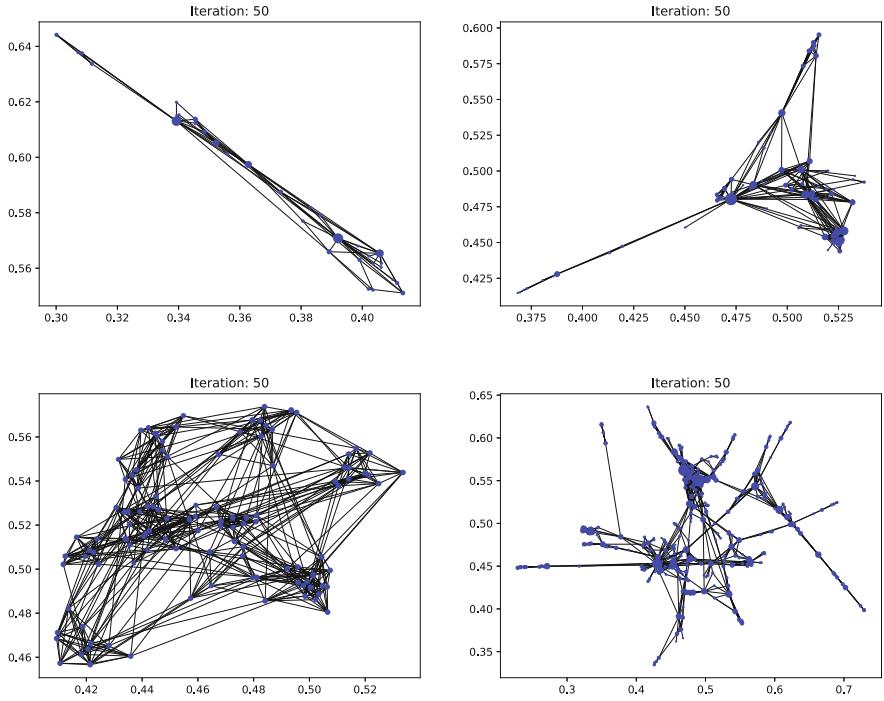


Fig. 1. 2D embedding for karate, lesmis, football, netscience (giant component) networks after 50 iterations.

the spatial complexity changes to $O(Nk^2)$. Here again, this is the worst case that does not occur in practice since dependencies with neighbors of neighbors are computed only rarely for nodes with a higher degree (see Sect. 4.3). In general, for sparse networks, we can expect a time and space complexity of $O(Nk^3)$ and $O(Nk)$, respectively.

Random selection around the selected node depends on the representation of the network. If we use an adjacency list, then neighbor selection has complexity $O(1)$. Therefore, for the total complexity, we only need to consider the number of iterations r; the estimate of the total time complexity is then $O(rN + Nk^3)$ for sparse networks.

5 Experiment

To present the effectiveness of the DepDist Contraction algorithm, we used four small networks; for these small networks, the quality of the embedding can be visually assessed in the form of a visualized network layout. We chose well-known

networks from Mark E.J. Newman[2]: *Zachary's karate club (karate), Les Miserables (lesmis), American College football (football), giant component of Coauthorships in network science (netscience).* In Table 1, we can see that each network has different properties (number of nodes and edges, average, minimum and maximum degree, average clustering coefficient, Louvain modularity).

Table 1. Properties of the four experimental networks.

Network	N	M	k	min_k	max_k	CC	Q
karate	34	78	4.588	1	17	0.588	0.415
lesmis	77	254	6.597	1	36	0.736	0.551
footbal	115	613	10.661	7	12	0.403	0.604
netscience	379	914	4.823	1	34	0.798	0.845

5.1 Results

In the experiment, we used dimension $n = 2$ for all four networks, the side size of the square for the random initial nodes distribution $a = 1$, i.e., a square with a diagonal $[0, 0], [1, 1]$, the maximum expected dependency distance $maxDepDist = 0.002$, and the maximum acceleration distance $maxAccDist = 0.01$. The result of applying the DepDist Contraction algorithm is shown for 50 and 500 iterations in Figs. 1 and 2; the difference between 200 and 500 iterations is visually negligible, and there is virtually no further moving. Figure 3 shows the changes in the positions of the network nodes expressed in terms of mean squared error (MSE) between two consecutive iterations.

Even though our goal is embedding (i.e., in this case, transforming the network to a vector representation of dimension 2), the result is comparable to layout-oriented algorithms, which usually use balancing based on attractive and repulsive forces between pairs of nodes. However, compared to the force-directed layout in Fig. 4, one significant difference can be seen in the karate layout. Namely, the dependency is much more related to the connectivity of the nodes to the neighborhood than to the edge weights. Therefore, the distance between pairs of nodes is small only when both dependencies are high. On the other hand, if at least one dependency decreases to zero, the distance increases, regardless of the edge weights. In Fig. 2, this property in the karate network highlights (1) the separation of the three groups of nodes in the middle and at the boundaries and (2) the relatively large distances between nodes weakly connected to their neighborhood.

The figures show how embedding is affected by other properties of the individual networks. Zachary's karate club contains no larger cliques except triangles; Les Miserables, on the other hand, contains well-separated cliques, near-clique

[2] http://www-personal.umich.edu/~mejn/netdata/.

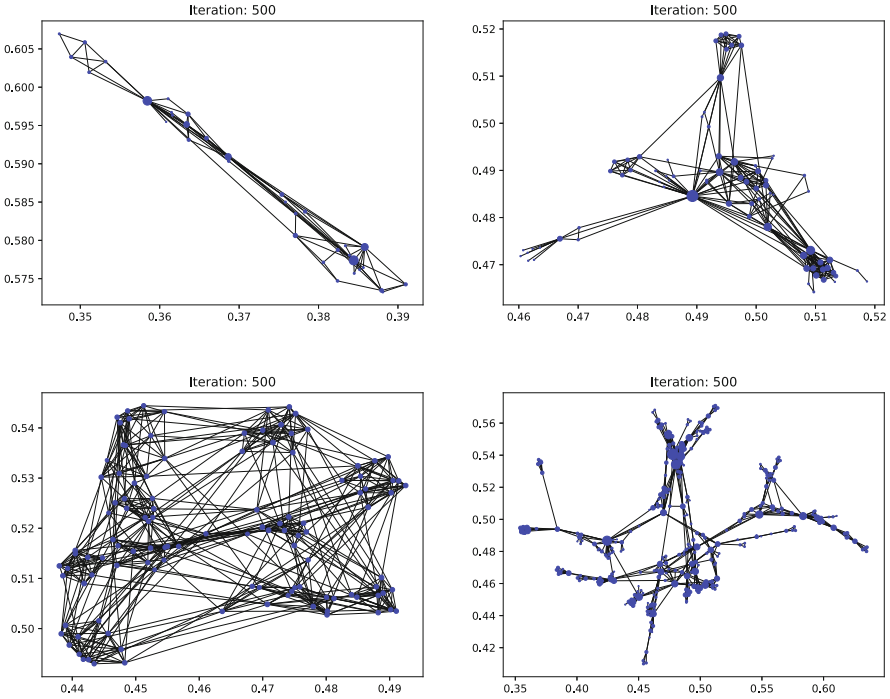


Fig. 2. 2D embedding for karate, lesmis, football, netscience (giant component) networks after 500 iterations.

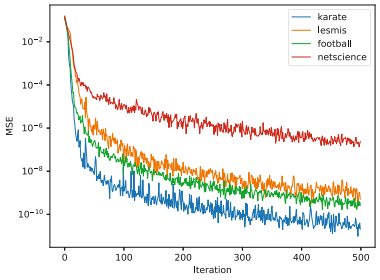


Fig. 3. Evolution of MSE between two consecutive iterations for karate, lesmis, footbal, netscience (giant component) networks.is significantly larger.

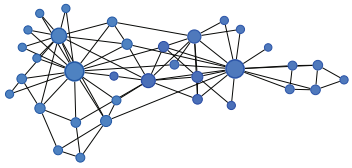


Fig. 4. Force-directed layout of karate club network.

and star-like sub-structures; American college football contains several near-clique structures and no hubs; Coauthorships in network science contains many small distinctly separated cliques clustered around variously sized hubs.

6 Conclusion and Future Work

The DepDist Contraction algorithm is very simple and as seen in experiments with small networks, it gives surprisingly good results. In our experiments, we also worked with networks with thousands to tens of thousands of nodes. However, the presentation of these experiments and their evaluation is beyond the scope of this paper.

Three problems emerged that we will address in the future. The first is that for both large and small networks, the initial random distribution of nodes sometimes has a negative effect on the result. It will be necessary to find a way to quantify this effect on the resulting embedding and reduce this effect. At the same time, we need to compare our approach with other network embedding algorithms. The second problem is computing dependencies in large networks with a high average degree (e.g., 100+). Despite the use of parallelism, the computation is time-consuming in these cases. Here, it will be necessary to use an estimate instead of an exact dependency calculation, which can be done using sampling around the pair of nodes. The third task is to estimate the number of iterations needed to stabilize the embedding. Experiments even with larger networks show that at most low hundreds of iterations are sufficient for stabilization. The estimation can be based on the assumption that stabilization is characterized by the fact that the distances between nodes almost stop changing; therefore, we assume that we will be able to describe the relationship between a sufficiently stable embedding, its parameters (dimension n, cube edge length a, $maxDepDist$) and the size of the network N.

Funding. This work is supported by SGS, VSB-Technical University of Ostrava, under grant no. SP2023/076.

References

1. Çivril, A., Magdon-Ismail, M., Bocek-Rivele, E.: SSDE: fast graph drawing using sampled spectral distance embedding. In: Kaufmann, M., Wagner, D. (eds.) Graph Drawing, pp. 30–41. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70904-6_5
2. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. IEEE Trans. Knowl. Data Eng. **31**(5), 833–852 (2018)
3. Freeman, L.C.: Graphic techniques for exploring social network data. Models Methods Soc. Netw. Anal. **28**, 248–269 (2005)
4. Fruchterman, T.M., Reingold, E.M.: Graph drawing by force-directed placement. Softw. Pract. Exp. **21**(11), 1129–1164 (1991)
5. Gibson, H., Faith, J., Vickers, P.: A survey of two-dimensional graph layout techniques for information visualisation. Inf. Vis. **12**(3–4), 324–357 (2013)

6. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
7. Hu, Y.: Efficient, high-quality force-directed graph drawing. Math. J. **10**(1), 37–71 (2005)
8. Jacomy, M., Venturini, T., Heymann, S., Bastian, M.: Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. PloS One **9**(6), e98679 (2014)
9. Kudelka, M., Ochodkova, E., Zehnalova, S., Plesnik, J.: Ego-zones: non-symmetric dependencies reveal network groups with large and dense overlaps. Appl. Netw. Sci. **4**(1), 1–49 (2019)
10. Liao, L., He, X., Zhang, H., Chua, T.S.: Attributed social network embedding. IEEE Trans. Knowl. Data Eng. **30**(12), 2257–2270 (2018)
11. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. J. Mach. Learn. Res. **9**(11) (2008)
12. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. Adv. Neural Inf. Process. Syst. **26** (2013)
13. Noack, A.: Energy models for graph clustering. J. Graph Algorithms Appl. **11**(2), 453–480 (2007)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
15. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 459–467 (2018)
16. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077 (2015)
17. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234 (2016)
18. Wang, S., Tang, J., Aggarwal, C., Chang, Y., Liu, H.: Signed network embedding in social media. In: Proceedings of the 2017 SIAM International Conference on Data Mining, pp. 327–335. SIAM (2017)
19. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)

Evaluating Network Embeddings Through the Lens of Community Structure

Jason Barbour[1,2(✉)], Stephany Rajeh[3,4(✉)], Sara Najem[2,5(✉)], and Hocine Cherifi[6(✉)]

[1] Graduate Program in Computational Science, American University of Beirut, Beirut, Lebanon
jgb21@mail.aub.edu
[2] Department of Physics, American University of Beirut, Beirut, Lebanon
sn62@aub.edu.lb
[3] LIP6 CNRS, Sorbonne University, Paris, France
stephanyrajeh@gmail.com
[4] Efrei Research Lab, EFREI Paris, Villejuif, France
[5] Center for Advanced Mathematical Sciences, American University of Beirut, Beirut, Lebanon
[6] ICB UMR 6303 CNRS, University of Burgundy, Dijon, France
hocine.cherifi@gmail.com

Abstract. Network embedding, a technique that transforms the nodes and edges of a network into low-dimensional vector representations while preserving relevant structural and semantic information, has gained prominence in recent years. Community structure is one of the most prevalent features of networks, and ensuring its preservation is crucial to represent the network in a lower-dimensional space accurately. While the core objective of network embedding is to bring related nodes in the original network close together in a lower-dimensional space, common classification metrics overlook community structure preservation. This work addresses the need for a comprehensive analysis of network embedding algorithms at the community level. On a set of synthetic networks that span strong to weak community structure strengths, we showcase the variability in the performance of network embedding techniques across mesoscopic metrics. Additionally, we highlight that the mesoscopic metrics are not highly correlated with the classification metrics. The community structure can further diminish the correlation as its strength weakens.

Keywords: Network Embedding · Community Structure · Evaluation Metrics

1 Introduction

Networks often exhibit a modular structure, where nodes cluster into communities with shared characteristics or functions [1]. Understanding these community structures is crucial for various applications, from recommendation systems to the optimal spread of information and disease control [2–10]. With network sizes increasingly increasing, generating lower order representation, known as network

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 440–451, 2024.
https://doi.org/10.1007/978-3-031-53468-3_37

embedding, has gained significant attention in recent years [11]. This technique transforms networks into low-dimensional vector representations.

While certain techniques are designed to explicitly maintain or enhance the community structure through the embedding process, others may not consider community structure preservation a primary objective. Nonetheless, one of the fundamental goals of all network embedding techniques is to project the similarity of the nodes of the original network onto the lower-dimensional space.

Further, network embedding techniques are commonly evaluated through classification metrics [11]. Nonetheless, these metrics are agnostic about the community structure: they do not indicate whether it is well preserved after the embedding process. In other words, they offer information about the overall quality of results but do not reveal the fine-grained details of community structure within a network.

Consequently, there is a need for a comprehensive comparative analysis of network embedding algorithms from a modular perspective. This paper analyzes the performance of the most prominent network embedding algorithms on controlled synthetic networks.

The rest of the paper is organized as follows. Section 2 overviews the fundamental concepts of network embedding and introduces the mesoscopic evaluation metrics. Section 3 presents the synthetic modular network generation. It details the experimental setup and evaluation metrics for comparing these algorithms and the voting model used in ranking these algorithms. Section 4 presents the results of our comparative analysis and discusses the findings and their implications. Finally, Sect. 5 concludes the paper.

2 Background

The landscape of network embedding algorithms is notably diverse. To ensure that we encompass a spectrum of approaches, we include ten widely recognized methods that span random walks, matrix factorization, and deep learning [12]. This diversity should allow us to understand the challenges and opportunities of different network embedding strategies. We briefly describe these algorithms, highlighting their specificity. It is worth highlighting that the DeepWalk, Node2Vec, Walklets, M-NMF, and M-GAE algorithms explicitly indicate maintaining the community structure as they acquire node representations.

2.1 Random-Walk-Based Methods

- **DeepWalk:** applies techniques from deep learning to learn node embeddings by treating network walks as sentences and using Skip-gram models to capture the context of nodes. It uses random walks to explore the network and capture local neighborhood information [13].
- **Node2Vec:** extends DeepWalk by introducing a biased random walk strategy that explores both breadth-first and depth-first neighborhood structures. It allows for fine-tuning exploration behavior, making it more versatile for different network structures [14].

- **Diff2Vec:** incorporates differential information from network snapshots over time to capture evolving node embeddings. It is particularly suitable for dynamic networks where nodes and their connections change over time [15].
- **Walklets:** is a variation of DeepWalk that leverages multiple lengths of random walks to capture local and global network structures. It creates embeddings by considering various context sizes, effectively capturing hierarchical relationships [16].

2.2 Matrix Factorization-Based Methods

- **Modularity-Normalized Matrix Factorization (M-NMF):** M-NMF optimizes a modularity-based objective function to learn embeddings that preserve community structure. It directly incorporates community information into the embedding process, emphasizing preserving community properties [17].
- **Laplacian Eigenmaps (LEM):** is a spectral embedding method that utilizes the eigenvalues and eigenvectors of the Laplacian matrix to map nodes to a low-dimensional space. It emphasizes preserving the pairwise distances between nodes, which can highlight the underlying geometric structure of the network [18].
- **Randomized Network Embedding (RandNE):** employs randomized matrix factorization to generate node embeddings while preserving global and local network properties. It introduces randomness in the factorization process, which can lead to more diverse embeddings [19].
- **Boosted Network Embedding (BoostNE):** leverages ensemble learning techniques to combine multiple embeddings generated by different methods, enhancing the overall performance. It focuses on improving embedding quality through ensemble techniques and is versatile in incorporating various base embedding methods [20].
- **Network Matrix Factorization (NetMF):** factorizes the network's adjacency matrix to obtain embeddings that capture higher-order proximity patterns. It emphasizes capturing different types of proximities in the network, such as Katz similarity, which goes beyond traditional pairwise node relationships [21].

2.3 Deep Learning-Based Method

- **Modularity-Aware Graph Autoencoder (M-GAE):** utilizes autoencoder architectures to learn embeddings by reconstructing the adjacency matrix or other network-related properties. It introduces a novel regularizer inspired by modularity and can adapt to various reconstruction objectives, allowing it to capture different aspects of network structure [22].

2.4 Evaluation Metrics

The classification metrics used in the literature to evaluate the quality of the embedding algorithms are mainly grounded in information theory and heavily

used by the machine learning community, which include the adjusted mutual information score (AMI), normalized mutual information score (NMI), adjusted random score (ARI), Micro-F1 score, and Macro-F1 score, which are not necessarily community-aware evaluators [23]. Here we propose a complementary set of metrics to assess the quality of these algorithms. These below-described mesoscopic metrics are used to evaluate the quality of the embedding techniques in preserving the community structure. The latter, denoted by C, is defined as follows: for an undirected unweighted graph $G(V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges, $C = \{c_1, c_2, ..., c_q, ..., c_{|C|}\}$ where c_q is q-th community, m_{c_q} and n_{c_q} are the total number of links and nodes inside community c_q, respectively, and $\mid C \mid$ is the total number of communities. A node i in G has a total degree of $k_i^{tot} = k_i^{intra} + k_i^{inter}$ where k_i^{intra} denotes its intra-community links and k_i^{inter} denotes its inter-community links. The mesoscopic metrics are calculated for each community in the network and then averaged over all the communities. We denote each evaluation metric for each community c_q as $f(c_q)$. In the context of this study, we employ a set of nine mesoscopic metrics, where seven are defined within our work. The internal degree and community size distributions are inherently self-explanatory through their nomenclature:

- **Internal distance:** is the average shortest distance of nodes inside a given community $f(c_q)$:

$$f(c_q) = \sum_{i,j \in c} \frac{d(i,j)}{n_{c_q}(n_{c_q} - 1)}$$

  where $d(i, j)$ is the shortest path from node i to node j.
- **Internal density:** is the edge density inside a given community c_q:

$$f(c_q) = \frac{2m_{c_q}}{n_{c_q}(n_{c_q} - 1)}$$

- **Maximum-out degree fraction (Max-ODF):** is based on the inter-community links of a node that possesses the highest inter-community links in its community c_q:

$$f(c_q) = max_{(i \in c_q)} \frac{k_i^{inter}}{k_i^{tot}}$$

- **Average-out degree fraction (Average-ODF):** is based on the inter-community links of all the nodes in the community c_q they belong to:

$$f(c_q) = \frac{1}{n_{c_q}} \sum_{i \in c_q} \frac{k_i^{inter}}{k_i^{tot}}$$

- **Flake-Out degree fraction (Flake-ODF):** is based on the percentage of nodes in community c_q that have more inter-community links than intra-community links:

$$f(c_q) = \sum_{i \in c_q} \frac{|f_i|}{n_{c_q}}$$

where f_i is the fraction of nodes having $k_i^{inter} \geq k_i^{intra}$.

– **Embeddedness:** quantifies the intra-community links of a node. It is the opposite of Average-ODF. It reaches a value of 1 if the nodes in a given community c_q only have intra-community links (i.e., all neighbors are in the same community):

$$f(c_q) = \frac{1}{n_{c_q}} \sum_{i \in c_q} \frac{k_i^{intra}}{k_i^{tot}}$$

– **Hub dominance:** is based on the intra-community links of a node that has the highest intra-community links in its community c_q:

$$f(c_q) = max_{(i \in c_q)} \frac{k_i^{intra}}{n_{c_q}(n_{c_q} - 1)}$$

3 Experimental Setup and Evaluation

This section describes the fundamental steps of the experimental setup employed to evaluate the network embedding algorithms' efficacy in maintaining the community structure.

3.1 Synthetic Network Generation

The ABCD generator offers the ability to change multiple parameters of the network generation process, namely: number of nodes (N), power-law exponent for degree distribution (τ_1), minimum degree (d_{min}), maximum degree (d_{max}), power-law exponent for community size distribution (τ_2), minimum community size (c_{min}), maximum community size (c_{max}) and the mixing parameter (μ) [24]. In what follows, we use it with the below choice for the parameters:

$$\begin{cases} N = 10000 & \tau_1 = 2.7 \\ d_{min} = 3 & d_{max} = 150 \\ \tau_2 = 2.7 & c_{min} = 10 \\ c_{max} = 1000 & \mu = [0.1 \rightarrow 0.7] \end{cases} \tag{1}$$

For our experimental setup, we focus on μ as a control parameter and vary it from 0.1 to 0.7 in increments of 0.1 and thus generate seven corresponding graphs. μ controls the number of edges between clusters where 0 represents the absence of inter-clusters edges and 1 is the other limit of high connectivity where the clusters become highly mixed. ABCD generator provides labeling for each node indicating which community it belongs to, which will be used as the ground truth for the rest of this paper.

3.2 Experimental Setup

Seven main steps are described below to evaluate the network embedding algorithms. A bird's-eye view of the experimental setup is illustrated in Fig. 1.

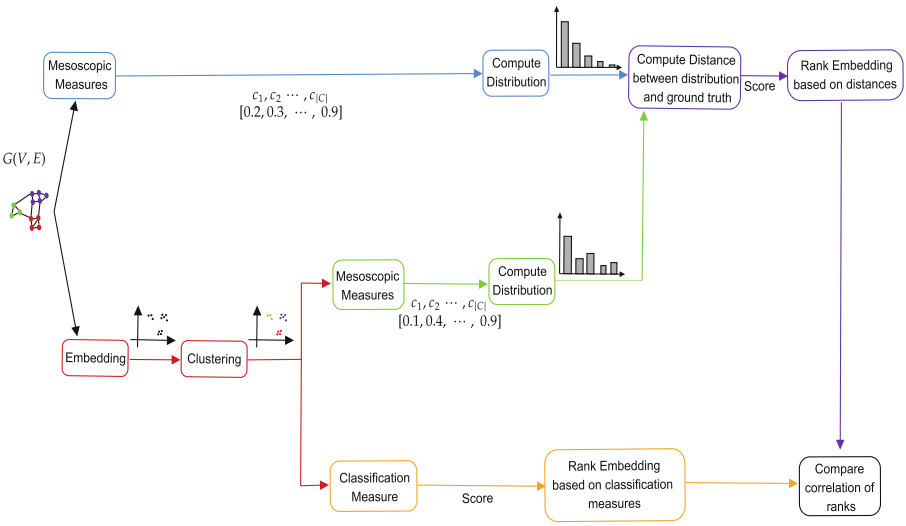


Fig. 1. Flowchart of the experimental setup to evaluate the performance of the network embedding algorithms. Mesoscopic metrics are calculated individually for each of the network's communities.

- **Embedding the graphs:** We use ten different embedding algorithms for each generated network to embed the network into a 128-dimensional space. The algorithms under investigation are: Deepwalk, Node2Vec, Diff2Vec, Walklets, M-NMF, Laplacian Eigenmaps, RandNE, BoostNE, NetMF, and M-GAE[1].
- **Clustering the embeddings:** Once we have the embeddings, we use the K-means clustering algorithm to group them into clusters whose number is defined by the ABCD as it as the number of clusters.
- **Compute the distance between the distributions:** We compute the Kullback-Leibler (KL) divergence [25] between all the above-introduced mesoscopic measures between the ground truth and after the embedding[2], including the internal degree distribution, community size distribution, internal

[1] The embedding algorithms were run on the American University of Beirut (AUB)'s high-performance computers using Intel Xeon E5-2665 CPUs in parallel. The runtime of these algorithms is reported in https://github.com/JasonBarbour-2002/ExploringNetworkEmbeddings in Fig. 1 of the supplementary material.

[2] Since most methods rely on a stochastic process, we run each method 30 times and take the average and standard deviation of the score for each measure.

distance, internal density, Max-ODF, Average-ODF, Flake-ODF, embeddedness, and hub dominance for all communities.
- **Compute the classification metrics:** After clustering, classification metrics including AMI, NMI, ARI, Micro-F1 score, and Macro-F1 score are computed.
- **Ranking the embedding algorithms based on the mesoscopic metrics:** Following the KL-divergence computation between the ground-truth distribution and the distribution of the embedding algorithms after the embedding process, the embedding algorithms are ranked from the most performing (i.e., lowest distance) and least performing (i.e., highest distance)
- **Ranking the embedding algorithms based on the classification metrics:** Following the computation of the classification metrics, the embedding algorithms are ranked from the most performing (i.e., highest magnitude) and least performing (i.e., lowest magnitude).
- **Comparing the correlation of the ranks:** When the ranks of the embedding techniques for all metrics are obtained, the correlation between the mesoscopic and classification metrics is computed.

3.3 Voting Model

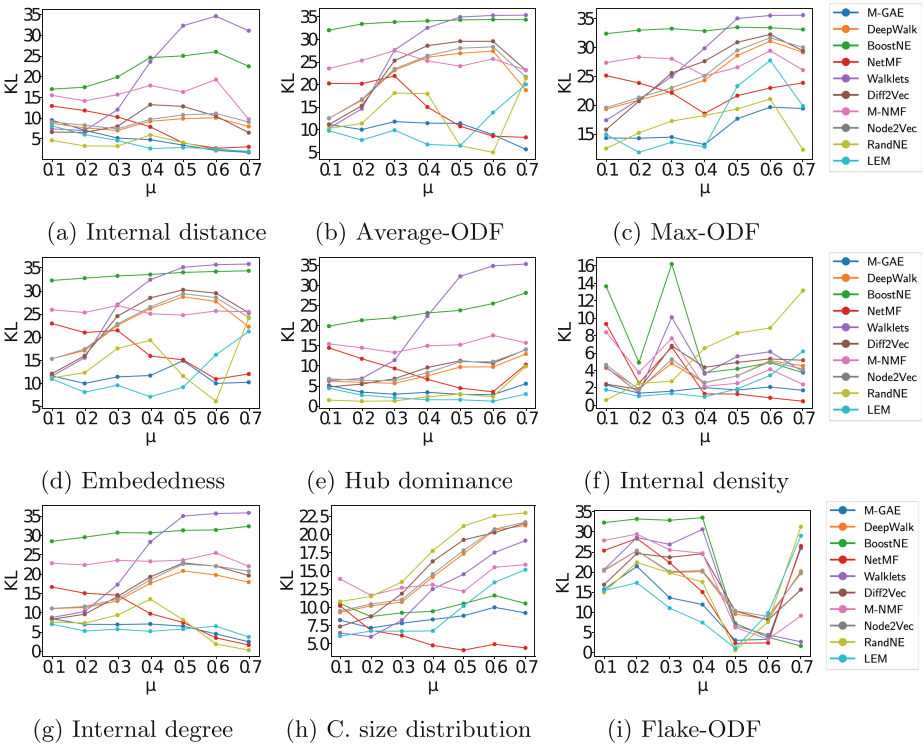
Given the metrics used as performance evaluators, a ranking scheme to report on the overall quality of the algorithms is needed. For this purpose, we used Schulze's voting model [26]. First, we compute the KL-divergence score of the distributions of the mesoscopic metrics versus the ground truth ones, as well as the classification metrics of the network embedding algorithms, which will be the voters, while the candidates are the ranks of each algorithm for each metric. Subsequently, the voting model orders them by comparing candidates in head-to-head matchups, calculating how many voters prefer one candidate. It then finds the candidate that outperforms all others by the largest margin, aiming to identify a broadly preferred winner. In the end, the final ranks of the network embedding algorithms represent a consensus between all the metrics.

4 Results and Discussion

In what follows, we present the performance of each embedding algorithm for the above-listed mesoscopic measures by evaluating the KL divergence between the ground truth distribution of the mesoscopic measure in question and the recovered one.

In Fig. 2a, we follow the KL-divergence of the distribution of internal distance. Almost all algorithms consistently have a low KL divergence as μ grows. Beyond $\mu = 0.4$, the performances of Waklets and BoostNE drop significantly as they exhibit an increase in their respective KL-divergences.

As for the Average-ODF, BoostNE stands out with the lowest performance up to $\mu = 0.4$. This is also accompanied by a drop in the performance of Waklets as shown in Fig. 2b.



(a) Internal distance (b) Average-ODF (c) Max-ODF

(d) Embededness (e) Hub dominance (f) Internal density

(g) Internal degree (h) C. size distribution (i) Flake-ODF

Fig. 2. KL-divergence between the ground truth distributions of the mesoscopic metrics and those recovered by the different algorithms.

Concerning Max-ODF, embeddedness and internal degree, over the full range of μ, BoostNE is outperformed by the other algorithms as shown in Figs. 2c, 2d, and 2g. Interestingly, it is joined by Walklets after the $\mu = 0.4$ value. Hub dominance in Fig. 2e follows this trend. However, in this case, the performance of Walklets decreases considerably more than the value of BoostNE.

For internal density in Fig. 2f, BoostNE is performing the worst up to $\mu = 0.4$, which becomes similar to the rest of the algorithms. At that point, we see a big decrease in the performance of RandNE.

As for Flake-ODF, all the algorithms seem to be following the same trend where the value of the KL is high for low values of μ, then after the value of $\mu = 0.4$, we see a sharp decline.

In the case of community size distribution, all algorithms' performance decreases around $\mu = 0.4$ as shown in Fig. 2h, except for NetMF which increases.

The results of the comparison of the ranks are rendered to a heatmap describing the correlation between the rankings of the classification metrics and the mesoscopic ones for the values of $\mu = 0.1$ and 0.7 are shown in Figs. 3 and 4, respectively. We note that the performance of the different algorithms changes

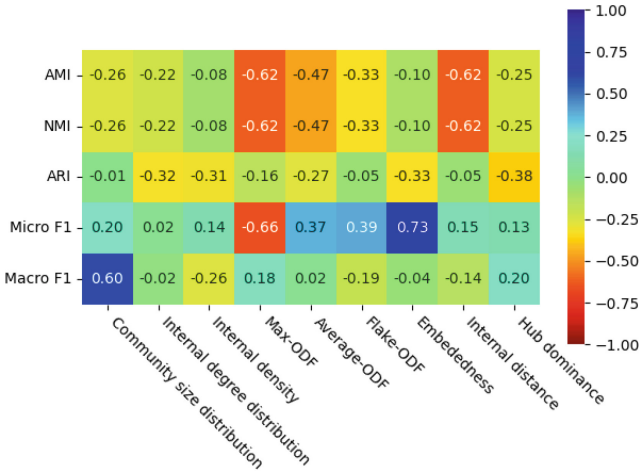


Fig. 3. Correlation between the ranking of the algorithms based on the classification metrics and the mesoscopic metrics for $\mu = 0.1$.

with the mixing parameter μ. Particularly, beyond the value of $\mu = 0.4$, some algorithms exhibit a significant drop in performance, which we report in Table 1. In the latter table, we also report the ranking of the algorithms using the classification metrics.

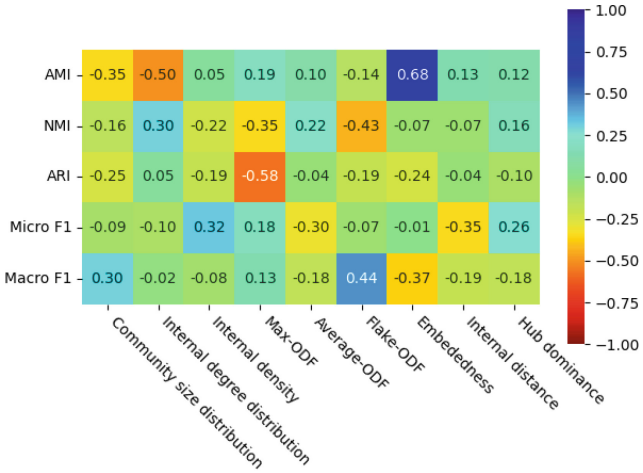


Fig. 4. Correlation between the ranking of the algorithms based on the classification metrics and the mesoscopic metrics for $\mu = 0.7$.

Based on the mesoscopic metrics, denoted as Meso in the table, LEM ranks first while M-GAE ranks second for $\mu \leq 0.4$, which is in total agreement with

the performance assessment based on the classification metrics, denoted by CL in the table. On the other hand, for Meso, when $\mu > 0.4$, M-GAE takes the lead, while NetMF ranks second. The latter remains true when the classification metrics. From the above it is obvious that the performance quality is μ-dependent, revealing the effect of mixing. Though Table 1 seems to suggest that there is total agreement between the measures, Figs. 3 and 4 shows the mismatch between the mesoscopic and the classification metrics in which we report uncorrelation and decorrelation between the measures.

Table 1. The ranking of the embedding algorithms based on Schulze's method for mesoscopic and classification metrics.

$\mu Value$	Metric	Ranks									
		1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	7^{th}	8^{th}	9^{th}	10^{th}
$0.1 \leq \mu \leq 0.4$	Meso	LEM	M-GAE	RandNE	NetMF	DeepWalk	Node2Vec	Diff2Vec	Walklets	M-NMF	BoostNE
	CL	LEM	M-GAE	RandNE	Walklets	NetMF	M-NMF Node2Vec		DeepWalk	Diff2Vec	BoostNE
$0.5 \leq \mu \leq 0.7$	Meso	M-GAE	NetMF	RandNE LEM		M-NMF	DeepWalk	Diff2Vec	Node2Vec	BoostNE	Walklets
	CL	M-GAE	NetMF	M-NMF	BoostNE	LEM	DeepWalk	Node2Vec RandNE		Diff2Vec	Walklets
Total	Meso	LEM	M-GAE	RandNE	NetMF	DeepWalk	Node2Vec	Diff2Vec	M-NMF	Walklets	BoostNE
	CL	M-GAE	LEM	NetMF	M-NMF	RandNE	Node2Vec	DeepWalk	BoostNE	Walklets	Diff2Vec

It is also worth noting that RandNE, LEM, and NetMF are matrix factorization methods that rely on a distance optimization scheme, while M-GAE is a deep learning-based algorithm. This seems to suggest that the metrics have inherent biases. More precisely, LEM, which relies on evaluating the Laplacian, is expected to rank first for most mesoscopic measures. The reason is the Laplacian's properties, the difference between the degree and the adjacency matrices. In that, it retains information about each node's degree and neighborhood. Moreover, the number of degenerate eigenvectors corresponding to the eigenvalue 0 of the Laplacian represents the number of communities, equivalent to a distance optimization problem [27]. In contrast, the mutual information measures tend to rank M-GAE first. That could be explained by the fact that M-GAE is optimizing on modularity which is related to the mutual information between two nodes. Therefore, it seems like it is a self-looping metric that is optimizing for itself.

In accordance with the outcomes presented here, upon crossing $\mu = 0.4$, M-GAE exhibits superior performance while it is second when $\mu \leq 0.4$ in both classification and mesoscopic metrics. Additionally, LEM ranks first when $\mu \leq 0.4$, ranks fifth with $\mu > 0.4$; NetMF, which ranks fourth and fifth with mesoscopic metrics and classification metrics, respectively, now ranks second when $\mu > 0.4$.

To summarize, LEM demonstrates outstanding performance within a robust community structure, excelling in community-aware and classification metrics. However, as the community structure strength diminishes, its effectiveness prominently declines with classification metrics. The opposite behavior is seen with NetMF. In contrast, M-GAE maintains outperformance across both community-aware and classification metrics regardless of the community structure strength, by ranking either first or second.

5 Conclusion

Preserving network community structure is crucial, and network embedding techniques offer a significant potential. However, the evaluation metrics commonly used in the literature fail to capture this preservation effectively. This study highlights the need for a comprehensive comparison of network embedding algorithms from a modular perspective. Our work is limited to evaluating the effect of the mixing parameter on the embedding quality. Our study specifically aims to determine the adequacy of classification metrics employed in the literature to comprehend the effectiveness of network embeddings. Results reveal that these metrics do not comprehensively reflect the network's community structure, exhibiting a low correlation with community-aware metrics. Furthermore, the efficacy of certain embedding techniques, such as LEM, M-GAE, and NetMF, is influenced by the strength of the community structure. These findings underscore the need for a more attentive approach in evaluating embedding techniques tailored to the specific application.

Acknowledgment. S.N and J.B would like to acknowledge support from the Center for Advanced Mathematical Science (CAMS) at the American University of Beirut (AUB).

References

1. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS **99**(12), 7821–7826 (2002)
2. Cherifi, H., Palla, G., Szymanski, B.K., Lu, X.: On community structure in complex networks: challenges and opportunities. Appl. Netw. Sci. **4**(1), 1–35 (2019)
3. Orman, K., Labatut, V., Cherifi, H.: An empirical study of the relation between community structure and transitivity. In: Menezes, R., Evsukoff, A., González, M. (eds.) Complex Networks. Studies in Computational Intelligence, vol. 424, pp. 99–110. Springer, Cham (2013). https://doi.org/10.1007/978-3-642-30287-9_11
4. Gupta, N., Singh, A., Cheri, H.: Community-based immunization strategies for epidemic control. In: 2015 7th International Conference on Communication Systems and Networks (COMSNETS), pp. 1–6. IEEE (2015)
5. Chakraborty, D., Singh, A., Cherifi, H.: Immunization strategies based on the overlapping nodes in networks with community structure. In: Nguyen, H., Snasel, V. (eds.) Computational Social Networks: 5th International Conference, CSoNet 2016, Ho Chi Minh City, Vietnam, 2–4 August 2016, Proceedings 5, pp. 62–73. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42345-6_6
6. Kumar, M., Singh, A., Cheri, H.: An efficient immunization strategy using overlapping nodes and its neighborhoods. In: Companion Proceedings of the Web Conference 2018, pp. 1269–1275 (2018)
7. Ghalmane, Z., Cheri, C., Cheri, H., El Hassouni, M.: Extracting backbones in weighted modular complex networks. Sci. Rep. **10**(1), 15539 (2020)
8. Rajeh, S., Savonnet, M., Leclercq, E., Cheri, H.: Interplay between hierarchy and centrality in complex networks. IEEE Access **8**, 129717–129742 (2020)
9. Rajeh, S., Savonnet, M., Leclercq, E., Cheri, H.: Characterizing the interactions between classical and community-aware centrality measures in complex networks. Sci. Rep. **11**(1), 10088 (2021)

10. Rajeh, S., Savonnet, M., Leclercq, E., Cheri, H.: Comparative evaluation of community-aware centrality measures. Qual. Quant. **57**(2), 1273–1302 (2023)
11. Hou, M., Ren, J., Zhang, D., Kong, X., Zhang, D., Xia, F.: Network embedding: taxonomies, frameworks and applications. Comput. Sci. Rev. **38**, 100296 (2020)
12. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: a survey. Knowl.-Based Syst. **151**, 78–94 (2018)
13. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
14. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
15. Rozemberczki, B., Sarkar, R.: Fast sequence-based embedding with diffusion graphs. In: Cornelius, S., Coronges, K., Goncalves, B., Sinatra, R., Vespignani, A. (eds.) Complex Networks IX: Proceedings of the 9th Conference on Complex Networks CompleNet 2018 9, pp. 99–107. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73198-8_9
16. Perozzi, B., Kulkarni, V., Chen, H., Skiena, S.: Don't walk, skip! Online learning of multi-scale network embeddings. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, pp. 258–265 (2017)
17. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
18. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, vol. 14 (2001)
19. Zhang, Z., Cui, P., Li, H., Wang, X., Zhu, W.: Billion-scale network embedding with iterative random projection. In: ICDM, pp. 787–796. IEEE (2018)
20. Li, J., Wu, L., Guo, R., Liu, C., Liu, H.: Multi-level network embedding with boosted low-rank matrix approximation. In: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, pp. 49–56 (2019)
21. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: unifying DeepWalk, LINE, PTE, and node2vec. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 459–467 (2018)
22. Salha-Galvan, G., Lutzeyer, J.F., Dasoulas, G., Hennequin, R., Vazirgiannis, M.: Modularity-aware graph autoencoders for joint community detection and link prediction. Neural Networks **153**, 474–495 (2022)
23. Xuan Vinh, N., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: is a correction for chance necessary? In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 1073–1080 (2009)
24. Kamiński, B., Prałat, P., Théberge, F.: Artificial benchmark for community detection (ABCD)-fast random graph model with community structure. Netw. Sci. **9**(2), 153–178 (2021)
25. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
26. Schulze, M.: A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. Soc. Choice Welfare **36**, 267–303 (2011)
27. Strang, G.: Linear algebra and learning from data. SIAM (2019)

Deep Distance Sensitivity Oracles

Davin Jeong[1(✉)], Allison Gunby-Mann[2], Sarel Cohen[3], Maximilian Katzmann[4], Chau Pham[5], Arnav Bhakta[6], Tobias Friedrich[3], and Peter Chin[2]

[1] Harvard University, Cambridge, MA 02138, USA
djeong@college.harvard.edu
[2] Dartmouth College, Hanover, NH 03755, USA
allisonmann.th@dartmouth.edu, peter.chin@dartmouth.edu
[3] Hasso Plattner Institute, Potsdam, Germany
[4] Karlsruhe Institute of Technology, Karlsruhe, Germany
[5] Boston University, Boston, MA 02215, USA
[6] Yale University, New Haven, CT 06520, USA

Abstract. Shortest path computation is one of the most fundamental and well-studied problems in algorithmic graph theory, though it becomes more complex when graph components are susceptible to failure. This research utilizes a *Distance Sensitivity Oracle (DSO)* for efficiently querying replacement paths in graphs with potential failures to avoid inefficiently recomputing them after every outage with traditional techniques. By leveraging technologies such as node2vec, graph attention networks, and multi-layer perceptrons, the study pioneers a method to identify pivot nodes that lead to replacement paths closely resembling optimal solutions with deep learning. Tests on real-world network demonstrate replacement paths that are longer by merely a few percentages compared to the optimal solution.

Keywords: GNNs · Graph Algorithms · Combinatorial Optimization · Shortest Paths · Distance Sensitivity Oracles · Learned Data-Structures

1 Introduction

The shortest path problem is frequently encountered in the real-world. In road networks, users want to know how long it will take to get from one place to another [17]. In biological networks, consisting of genes and their products, the shortest paths are used to find clusters and identify core pathways [23]. In social networks, the number of connections between users can be used for friend recommendation [26]. In web search, relevant web pages can be ranked by their distances from queried terms [27].

For graphs in the real world, often consisting of millions of nodes, special data structures called *Distance Oracles* (DO) are used to store information about distances of an input graph $G = (V, E)$ with n vertices and m edges. Without storing the entire graph, they can quickly retrieve important distance information to answer the shortest path queries. These shift the computational burden to the preprocessing step, so that queries can be answered quickly.

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 452–463, 2024.
https://doi.org/10.1007/978-3-031-53468-3_38

However, in addition to being large in size, real-world networks are also frequently susceptible to failures. For example, in road networks, a construction, a traffic accident, or an event might temporarily block nodes. In social networks, users might temporarily deactivate their accounts, resulting in a null node. And on the internet, web servers may be temporarily down due to mechanical failures or malicious attacks [4]. In these instances, we desire a method that can continue answering shortest path queries without stalling or having to recompute shortest paths on the entire graph again.

Distance Sensitivity Oracles (DSO) are a type of DO that can respond to queries of the form (s, t, f), requesting the shortest path between nodes s and t when a vertex f fails and is thus unavailable. Desirable DSOs should provide reasonable trade-offs among space consumption, query time, and *MRE* (i.e., quality of the estimated distance). In this paper, we consider the simplest case, in which there is only one failed node.

1.1 Contributions

We have tested our method on a variety of real-world networks and achieved state-of-the-art performance on all of them, our accuracy even outperforms models for shortest paths without node failures. Our contributions mainly lie in three aspects:

- In our theoretical analysis, we first present a simple proof for the existence of an underlying combinatorial structure for replacement paths: specifically, the existence of pivot nodes.
- We observe that one can use deep learning to find pivot nodes in distance sensitivity oracles. In fact, to the best of our knowledge, we are the first to use deep learning to build a distance sensitivity oracle.
- We empirically evaluate our method and compare it with related works to demonstrate near-exact accuracy across a diverse range of real-world networks.

1.2 Related Work

Given we are the first to propose a deep learning approach to DSOs, we describe previous works in both DSOs and deep learning in this section.

Distance-Sensitivity Oracles. The problem of constructing a DSO is well-studied in the theoretical computer science community. Demetrescu *et al.* [13] showed that given a graph $G = (V, E)$, there is a DSO which occupies $O(n^2 \log n)$ space, and can answer a query in constant time. The preprocessing of this DSO, that is the time it takes to construct this DSO, is $O(mn^2 + n^3 \log n)$.

Several theoretical results attempted to improve the preprocessing time required by the DSO. Bernstein and Karger [3] improved this time bound to

$\tilde{O}(mn)$[1]. Note that the All-Pairs Shortest Paths (APSP) problem, which only asks the distances between each pair of vertices u, v, is conjectured to require $mn^{1-o(1)}$ time to solve [20]. Since we can solve the APSP problem by using a DSO, by querying it with (s, t, emptyset) for every s, t, the preprocessing time $\tilde{O}(mn)$ is theoretically asymptotically optimal in this sense, up to a polylogarithmic factor (note that, in practice, such polylogarithmic factors may be very large). Several additional results improved upon the theoretical preprocessing time by using fast matrix multiplication [8,16,22].

With respect to the size of the oracle, Duan and Zhang [14] improved the space complexity of [13] to $O(n^2)$, which is from a theoretical perspective asymptotically optimal for dense graphs (*i.e.*, $m = \Theta(n^2)$). To do so, Duan and Zhang store multiple data-structures, which is reasonable for a theoretical work, however from a practical perspective the hidden constant is large. Therefore, it may also be interesting to consider DSOs with smaller space, at the cost of an approximate answer.

Here are several DSOs that provide tradeoffs between the size of DSO and the stretch (the length reported divided by the actual length):

- The DSO described in [2], for every parameter $\epsilon > 0$ and integer $k \geq 1$ has stretch $(2k-1)(1+\epsilon)$ and size $O(k^5 n^{1+1/k} \log^3 n/\epsilon^4)$.
- The DSO described in [9], for every integer parameter $k \geq 1$ has stretch $(16k-4)$ and size $O(kn^{1+1/k} \log n)$.

Note that even though the size of the above two DSOs for $k \geq 2$ is asymptotically smaller than $O(n^2)$, the stretch guarantee is at least 3 in [2] and at least 28 in [9], which is far from the optimum and may not be practical in many applications.

In this work, we construct the first DSO that is built using deep learning. Our method uses deep learning to find pivot nodes (as described in Sect. 3), utilizing a combintorial structural property we observe in Sect. 2, computing near optimal paths as shown in Sect. 5.

Shortest Paths Using Deep Learning. Previous works towards answering shortest path queries typically employ a two-stage solution: 1) representation learning and 2) distance prediction.

In general, graph embeddings are used to generate low-dimensional representations of nodes and edges which preserve properties of the original graph [6]. Methods include matrix factorization, deep learning with and without random walks, edge reconstruction, graph kernels, and generative models [6]. By either optimizing embeddings for their specific task or using general embedding techniques like node2vec, these graph embeddings may then be combined with existing techniques to tackle tasks such as node classification, node clustering, and link detection [11,30].

[1] For a non-negative function $f = f(n)$, we use $\tilde{O}(f)$ to denote $O(f \cdot \mathsf{polylog}(n))$.

Among the first to apply graph embeddings to the shortest paths problem was Orion [29]. Inspired by the successes of virtual coordinate systems, a landmark labelling approach was employed, where positions of all nodes were chosen based on their relative distances to a fixed number of landmarks. Using the Simplex Downhill algorithm, representations were found in a Euclidean coordinate space, allowing constant time distance calculations and producing mean relative error (MRE) between 15% - 20% [29]. Other existing coordinate systems have also been used. Building off of network routing schemes in hyperbolic spaces, Rigel used a hyperbolic graph coordinate system to reduce the MRE to 9% and found that the hyperbolic space performed empirically better across distortion metrics than Euclidean and spherical coordinate systems [12,31]. In road networks, geographical coordinates have been utilized with a multi-layer perceptron to predict distances between locations with 9% MRE [18].

In addition to these coordinate systems, general graph embedding techniques have recently been employed to handle shortest path queries to great success. In 2018, researchers from the University of Passau proposed node2vec-Sg[24]. To find the shortest path between nodes s and t, their Node2vec and Poincare embeddings were combined through various binary operations and fed into a feed-forward neural network, which was trained only on the distances between l landmark nodes $l << n$ and the rest of the graph. The model which took concatenated Node2vec embeddings performed the best, with an MRE between 3% to 7% .

Researchers have also demonstrated the accuracy of graph embeddings learned alongside distance predictors, to produce representations more specific to the shortest path task. Vdist2vec directly learned vertex embeddings by passing the gradient from the distance predictor back to a $N \times k$ matrix, achieving an MRE between 1% to 7% [21]. Huang et al. computed shortest path distances on road networks using a hierarchical embedding model and achieved an MRE of 0.7% [17]. Most recently, ndist2vec built upon the landmark learning, graph embedding, and neural network aspects of all of these approaches, reporting an MRE of 3.4% with a dataset on the order of $O(n)$.

Current works for estimating the shortest path lengths between two nodes are limited by the representations they learn. They rely on datasets which, even using schemas like landmark labelling or hierarchical, are proportional to n, the number of nodes in the network[10,17]. This presents a significant bottleneck for larger graphs. Taking these lessons to the deep learning DSO task, we present a model in Sect. 3, which extracts signal more efficiently thus requiring training samples without sacrificing accuracy.

2 Theoretical Analysis

In this section we consider a combinatorial structural property of replacement paths: such a path is a concatenation of a few original shortest paths. As we previously described, later on our deep learning algorithm builds on this lemma.

Our research is motivated by the following lemma from Afek *et al.*

Lemma 1. *[1] After k edge failures in an unweighted graph, each new shortest path is the concatenation of at most $k+1$ original shortest paths.*

In other words, the replacement path can be defined using so called pivot nodes that specify at which nodes in the graph the shortest paths may be stitched together. In this work we are interested in the failure of a single node, which is equivalent to the failure of its incident edges. The number of concatenations (and with that the number of corresponding pivots) required to obtain the replacement path then depends on the degree of the failed node. While in real-world networks the average degree is often rather small, finding suitable pivots remains a hard task. To overcome this problem, we consider an approximate setting, where we allow for a slack in the quality of the obtained paths (they may be longer than a shortest replacement path) but where only one pivot node is used. From the theoretical perspective, the following lemma is a special case of Lemma 1 for the case of a single edge failure.

Lemma 2. *After an edge failure in an unweighted undirected graph, each new shortest path is the concatenation of at most two original shortest paths.*

Given (s, t, f), let $P(s, t, f)$ be a shortest path from s to t in $G - \{f\}$. According to Lemma 2 it follows that $P(s, t, f)$ is a concatenation of two original shortest paths, or in other words, there exists a pivot vertex v such that $P(s, t, f)$ is the concatenation of the two shortest paths $P(s, v)$ and $P(v, t)$, where $P(s, v)$ is a shortest path from s to v in G and $P(v, t)$ is a shortest path from v to t in G. Motivated by this lemma, we will assume that it is sufficient to approximate the replacement path of a node failure using a single pivot node. As mentioned above, in the remainder of this paper we show that it is possible to use deep learning to find such pivot nodes.

3 Method

By the above argumentation, we reduce the problem of finding the shortest replacement path to finding pivot candidates. In the following section, we describe how we use a graph convolutional network to encode relevant graph information and a multi-layer perceptron to select pivot nodes.

3.1 Training Data

To generate our training data, we first generated training triplets (s, t, f), representing start, target, and failed nodes. s and t were rejected if not connected or if direct neighbors. For each input triplet, we simulated the node failure, computed replacement paths, and determined the pivot nodes.

$min(10^5, n^2)$ input triplets were randomly sampled for a graph with n nodes (Sect. 1). An 80/10/10 split was used for training, validation, and testing sets. Though previous works using landmark labelling selected $l(n - l)$ samples to improve the quality of their candidates, we found that capping our dataset by a constant value was sufficient [17,24]. This value can be modified based on the input graph if desired.

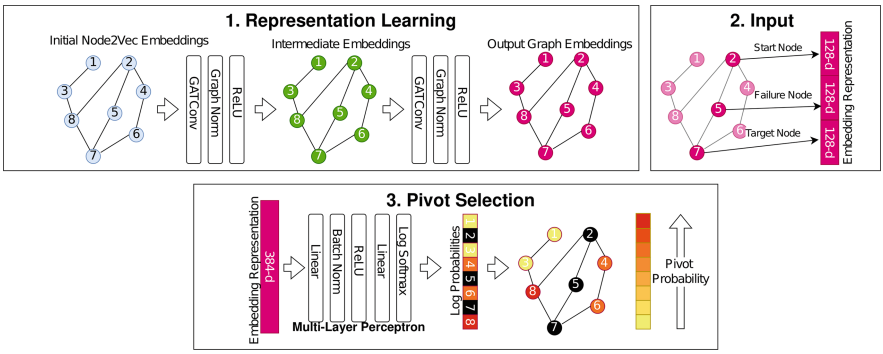


Fig. 1. The overall neural network architecture. During back-propagation, the gradient is passed through the MLP and to the relevant GAT parameters, so that the representations learned to encode relevant task-specific attributes.

3.2 Representation Learning with Graph Convolutional Networks

In order to learn representations from not only the shortest path data but also the network structure, we leverage the Graph Attention (GAT) layer. This attention-based convolutional layer computes each nodes' embeddings by implicitly learning different importance to different nodes within a neighborhood [28]. While they have been used successfully for tasks like biological link prediction and text classification, to the best of our knowledge, we are the first to apply the GAT mechanism to shortest path problems.

Like previous deep learning for shortest paths works, we initialize our node embeddings using node2vec [15,24]. We use 16 iterations, a walk length of 10, num walks of 80, $p = 1$, $q = 1$, and window size of 10. Next, we use two GAT layers to refine our embeddings towards our task. We introduce non-linearities using a ReLU layer, and we speed up convergence using GraphNorm layers, a normalization technique that was recently found to be more effective than similar techniques for training graph neural networks [7]. We also use dropout (0.1) to deal with overfitting.

3.3 Multi-layer Perceptron

Multiple pivot nodes are possible, so we frame the task of pivot selection as a multi-layer classification task. As shown in Fig. 1, after the GAT component of our model, we select the three embeddings corresponding the start, target, and failing node triplet. These are flattened into a 384-sized vector, which serves as the input for our feedforward neural network.

The multilayer perceptron consists of two linear layers and an output layer. We use ReLU layers as an activation function between the first two layers to introduce non-linearities. We also use BatchNorm layers to enable faster convergence and introduce regularization. The output layer is a log-softmax layer (commonly used for multi-label classification) that normalizes the vector values.

Our final output is a n-sized vector, representing the log likelihoods of each node in the network being a pivot node.

3.4 Summary

In total, our neural network consists of a representation learning module, based both on characteristics of the network and our DSO task, and a pivot selection module, a simple MLP (Fig. 1). Since multiple pivot nodes are possible, the model was trained using BCELoss and the node with the highest likelihood was selected as the output. We trained our neural network for 32 epochs using the Adam optimizer [19]. Our learning rate is 1e-3, our betas are 0.9 and 0.999, and our epsilon value is 1e-8.

4 Experiments

As mentioned previously, to the best of our knowledge, we are the first to propose a deep learning approach to the DSO problem. Thus, we will evaluate our proposed method against the state-of-the-art deep learning approaches to the shortest paths problem: namely, ndist2vec and node2vec-Sg [10,24].

We ran the authors' implementations for all comparison models with their recommended hyperparameter settings. All embeddings were 128-dimensional, in line with previous shortest paths works [15,21,24].

All experiments were implemented in Python and run on a Quadro RTX 8000 and an Intel(R) Xeon(R) Silver 4214R CPU @ 2.40GHz.

4.1 Datasets

Table 1. Statistics of real-world networks.

Network Name	Nodes	Density	Average Degree	Dataset Size
chem-ENZYMES-g118	95	2.71E-02	2.547	9.03E+03
chem-ENZYMES-g296	125	1.82E-02	2.256	1.56E+04
infect-dublin	410	3.30E-02	13.488	1.00E+05
bio-celegans	453	1.98E-02	8.940	1.00E+05
bn-mouse-kasthuri-graph-v4	987	3.16E-03	3.112	1.00E+05
can-1072	1,072	1.18E-02	12.608	1.00E+05
scc_retweet	1,150	9.98E-02	114.713	1.00E+05
power-bcspwr09	1,723	2.78E-03	4.779	1.00E+05
inf-openflights	2,905	3.71E-03	10.771	1.00E+05
inf-power	4,941	5.40E-04	2.669	1.00E+05
ca-Erdos992	4,991	5.97E-04	2.977	1.00E+05
power-bcspwr10	5,300	9.66E-04	5.121	1.00E+05
bio-grid-yeast	6,008	8.70E-03	52.245	1.00E+05
soc-gplus	23,613	1.41E-04	3.319	1.00E+05
ia-email-EU	32,430	1.03E-04	3.355	1.00E+05
ia-wiki-Talk	92,117	8.50E-05	7.833	1.00E+05
dbpedia-occupation	127,569	3.08E-05	3.934	1.00E+05
tech-RL-caida	190,914	3.33E-05	6.365	1.00E+05

We test our method on a variety of undirected, unweighted networks from the Network Repository [25], which covers a diverse set of areas, such as road networks, biological networks, and communication networks, all representing potential input for real-world applications. For a list of all considered networks, we refer to Table 1.

The training schema of each comparison model was used, but the testing dataset was standardized for the purposes of a fair comparison. Specifically, $min(10^5, n^2) * 0.10$ node pairs (a, b) were randomly sampled without replacement from each network, such that $a \neq b$.

4.2 Evaluation

In line with previous works computing shortest paths with deep learning, we evaluate our method using the **Mean Relative Error (MRE)** metric. Let $\hat{d}_{i,j}$ denote the predicted distance and $d_{i,j}$ denote the actual distance. The Relative Error is then given by

$$RE = \frac{|\hat{d}_{i,j} - d_{i,j}|}{d_{i,j}}$$

We note that, for evaluations of DSOs, $\hat{d}_{i,j}$ and $d_{i,j}$ denote the predicted and actual distances on the graph after a node failure.

We also provide the representation factor, denoting the ratio of the MRE obtained with random pivots and the one obtained using our method, as a metric for evaluating the quality of our representations.

5 Results

We aimed to evaluate our deep learning approach across the following questions:

1. How much longer than the optimal paths are our replacement paths?
2. How does our deep learning model compare with previous state-of-the-art shortest paths works?
3. Is the resulting performance an achievement of our approach or merely an artifact of the structure of the input graph?

The motivation behind the first question is obvious. Computed replacement paths are only suitable if they are not much longer than the actual shortest paths in the graph. Table 2 lists the MRE values obtained on all considered networks. Except for **bio-grid-yeast**, **ia-wiki-Talk**, **tech-RL-caida**, and **dbpedia-occupation** all MRE values are below 1%, with a mean value of 1.82% and a median value of 0.32%. This means that the computed replacement paths were almost as short as the optimal replacement paths. Interestingly, the networks with the highest MRE values were not those with the highest density or

Table 2. Results of models across several real-life networks.

Network Name	MRE (Ndist2vec)	MRE (Rizi)	MRE (Random Pivots)	Representation Factor	MRE (Our Method)
chem-ENZYMES-g118	100.00%	14.19%	215.96%	811.89	0.27%
chem-ENZYMES-g296	100.00%	7.85%	257.69%	530.22	0.49%
infect-dublin	38.33%	13.58%	191.02%	1,091.55	0.18%
bio-celegans	15.45%	9.84%	176.28%	4,299.51	0.04%
bn-mouse-kasthuri-graph-v4	17.15%	16.91%	204.20%	5,105.11	0.04%
can-1072	36.43%	11.23%	211.58%	573.40	0.37%
scc_retweet	100.00%	10.78%	167.64%	3,287.11	0.05%
power-bcspwr09	42.75%	20.45%	237.46%	1,493.46	0.16%
inf-openflights	100.00%	16.32%	202.05%	280.63	0.72%
inf-power	59.08%	31.67%	228.67%	1,013.10	0.23%
ca-Erdos992	100.00%	18.93%	206.26%	630.77	0.33%
power-bcspwr10	37.02%	31.95%	232.83%	739.13	0.32%
bio-grid-yeast	14.67%	15.95%	173.80%	27.80	6.25%
soc-gplus	100.00%	55.99%	200.20%	654.61	0.31%
ia-email-EU	13.31%	119.61%	202.93%	216.11	0.94%
ia-wiki-Talk	20.83%	28.19%	203.29%	26.50	7.67%
dbpedia-occupation		28.41%	205.07%	31.56	6.50%
tech-RL-caida	Did not finish	51.92%	206.02%	26.04	7.91%

average degree. For instance, our model obtained an MRE less than 1% **soc-gplus** and **ia-email-EU** but struggled with the comparably smaller **bio-grid-yeast** network at 6.25%. These insights suggest that other factors related to the networks' structures affected our models' performance

Given the lack of DSOs using deep learning, the second question hopes to understand how our model performs compared to methods finding the shortest paths without node failures. Across all networks, we were able to match or outperform the state-of-the-art shortest paths works, often by several degrees of magnitude (Table 2). We did so with less training cases (numerically and proportionally) and no special selection process. In doing so, we demonstrated that deep learning can be used to effectively find the shortest replacement paths as well.

We'd like to note that we used the authors' implementation of ndist2vec and confirmed its performance on the road network datasets presented in the original paper [10]. Nonetheless, the model performed significantly worse on our real-world networks and did not finish after a week of computation for the two largest networks. We have two potential explanations. First, the authors suggest that their landmark-labelling approach will not scale well to sparse, large networks, many of which were used in our experiments. Second, the model often became stuck on local optima during training, producing a MRE value of 100% corresponding to a constant ouput of 0, demonstrating a reliance on initial values. For a fair comparison, we depict the best MRE values after two runs in Table 2.

Finally, we aimed to determine whether our model performed well because of the model or because of the inherent structure of the networks. For example, consider a graph that is almost a clique (almost all vertices are pairwise connected). Then, all paths are short and after a failure (having almost no impact on the graph structure) most replacement paths are short as well. In this setting

almost any node can serve as a suitable pivot, yielding a replacement path with a small stretch and one would expect that even randomly chosen pivots would yield good results. While the networks considered in our experiments are not as dense (see Table 2), other graph properties like a small diameter may make finding good pivots easier.

Table 2 lists the MRE values we get when considering random pivots, which we obtained by replacing the output of our pipeline with random noise. As can be clearly seen, the MRE is much larger in this setting. For most networks the MRE is larger than 200%, meaning the found paths are more than 3 times longer than the shortest replacement paths. In order to compare our method with the random approach, Table 2 also lists the representation factor. Except for **bio-grid-yeast**, **ia-wiki-Talk**, and **tech-RL-caida** this factor is always larger than 200, meaning on most networks our approach is over 200 times better than the random method, clearly indicating that the close to optimal performance is due to the quality of our approach and not an artifact of properties of the considered inputs.

6 Conclusion

We have shown that distance sensitivity oracles with close to optimal performance can be obtained by utilizing the power of deep learning. Our method builds on a combinatorial property that allows for finding replacement paths based on pivot vertices. On a variety of real-world networks in the presence of failures, we can reliably find suitable pivots where the lengths of the corresponding replacement paths are very close to those of optimal paths. Moreover, our experiments suggest that these results are not artifacts of the inherent structure of the inputs, but are instead based on the fact that the different building blocks of our pipeline successfully capture the relevant structural information about the input graph.

As a consequence, it would be interesting to apply this method to related tasks where similar structural information needs to be captured. One such example is *local routing*, where the goal is to find short paths in a graph without the use of a central data structure by greedily routing to nearby embeddings. Prior work has shown that close to optimal greedy routing can be performed when embedding networks into hyperbolic space [5]. However, the resulting embeddings were susceptible to numerical inaccuracies, and network failures decreased routing performance a lot. It would thus be interesting to see whether our approach can be extended to the greedy routing setting as well, in order to overcome the previously observed issues.

Additionally, our approach has currently not been tested on larger networks containing millions of nodes. By calculating the APSP information using a distance oracle and using an improved node2vec implementation, we plan to test our networks' scalability in the future.

References

1. Afek, Y., Bremler-Barr, A., Kaplan, H., Cohen, E., Merritt, M.: Restoration by path concatenation: fast recovery of MPLS paths. Dis. Comput. **15**(4), 273–283 (2002). https://doi.org/10.1007/s00446-002-0080-6
2. Baswana, S., Khanna, N.: Approximate shortest paths avoiding a failed vertex: near optimal data structures for undirected unweighted graphs. Algorithmica **66**, 18–50 (2013).https://doi.org/10.1007/s00453-012-9621-y
3. Bernstein, A., Karger, D.R.: A nearly optimal oracle for avoiding failed vertices and edges. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, 31 May - 2 June 2009, pp. 101–110. ACM (2009). https://doi.org/10.1145/1536414.1536431
4. Billand, P., Bravard, C., Iyengar, S.S., Kumar, R., Sarangi, S.: Network connectivity under node failure. Econ. Lett. **149**, 164–167 (2016)
5. Bläsius, T., Friedrich, T., Katzmann, M., Krohmer, A.: Hyperbolic embeddings for near-optimal greedy routing. ACM J. Exp. Algorithmics **25** (2020). https://doi.org/10.1145/3381751. https://doi.org/10.1145/3381751
6. Cai, H., Zheng, V.W., Chang, K.C.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
7. Cai, T., Luo, S., Xu, K., He, D., Liu, T.y., Wang, L.: Graphnorm: a principled approach to accelerating graph neural network training. In: International Conference on Machine Learning, pp. 1204–1215. PMLR (2021)
8. Chechik, S., Cohen, S.: Distance sensitivity oracles with subcubic preprocessing time and fast query time. In: Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, 22-26 June 2020, pp. 1375–1388. ACM (2020). https://doi.org/10.1145/3357713.3384253
9. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: f-sensitivity distance oracles and routing schemes. Algorithmica **63**, 861–882 (2012). https://doi.org/10.1007/s00453-011-9543-0
10. Chen, X., et al.: Ndist2vec: node with landmark and new distance to vector method for predicting shortest path distance along road networks. ISPRS Int. J. Geo Inf. **11**(10), 514 (2022)
11. Crichton, G., Guo, Y., Pyysalo, S., Korhonen, A.: Neural networks for link prediction in realistic biomedical graphs: a multi-dimensional evaluation of graph embedding-based approaches. BMC Bioinform. **19**(1), 1–11 (2018)
12. Cvetkovski, A., Crovella, M.: Hyperbolic embedding and routing for dynamic graphs. In: IEEE INFOCOM 2009, pp. 1647–1655. IEEE (2009)
13. Demetrescu, C., Thorup, M., Chowdhury, R.A., Ramachandran, V.: Oracles for distances avoiding a failed node or link. SIAM J. Comput. **37**(5), 1299–1318 (2008). https://doi.org/10.1137/S0097539705429847
14. Duan, R., Zhang, T.: Improved distance sensitivity oracles via tree partitioning. In: WADS 2017. LNCS, vol. 10389, pp. 349–360. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62127-2_30
15. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
16. Gu, Y., Ren, H.: Constructing a Distance Sensitivity Oracle in $O(n^2.5794M)$ Time. In: Bansal, N., Merelli, E., Worrell, J. (eds.) 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021), Leibniz International Proceedings in Informatics (LIPIcs), vol. 198, pp. 76:1–76:20.

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ICALP.2021.76. https://drops.dagstuhl.de/opus/volltexte/2021/14145
17. Huang, S., Wang, Y., Zhao, T., Li, G.: A learning-based method for computing shortest path distances on road networks. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 360–371. IEEE (2021)
18. Jindal, I., Chen, X., Nokleby, M., Ye, J., et al.: A unified neural network approach for estimating travel time and distance for a taxi trip. arXiv preprint arXiv:1710.04350 (2017)
19. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Lincoln, A., Williams, V.V., Williams, R.R.: Tight hardness for shortest cycles and paths in sparse graphs. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, 7-10 January 2018, pp. 1236–1252. SIAM (2018). https://doi.org/10.1137/1.9781611975031.80
21. Qi, J., Wang, W., Zhang, R., Zhao, Z.: A learning based approach to predict shortest-path distances. In: EDBT, pp. 367–370 (2020)
22. Ren, H.: Improved distance sensitivity oracles with subcubic preprocessing time. J. Comput. Syst. Sci. **123**, 159–170 (2022). https://doi.org/10.1016/j.jcss.2021.08.005
23. Ren, Y., Ay, A., Kahveci, T.: Shortest path counting in probabilistic biological networks. BMC Bioinform. **19**(1), 1–19 (2018)
24. Rizi, F.S., Schloetterer, J., Granitzer, M.: Shortest path distance approximation using deep learning techniques. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 1007–1014. IEEE (2018)
25. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). https://networkrepository.com
26. Tian, X., Song, Y., Wang, X., Gong, X.: Shortest path based potential common friend recommendation in social networks. In: 2012 Second International Conference on Cloud and Green Computing, pp. 541–548. IEEE (2012)
27. Ukkonen, A., Castillo, C., Donato, D., Gionis, A.: Searching the wikipedia with contextual information. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 1351–1352 (2008)
28. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
29. Wang, H.S., Zhu, X., Peh, L.S., Malik, S.: Orion: a power-performance simulator for interconnection networks. In: 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35), Proceedings., pp. 294–305. IEEE (2002)
30. Zhang, X., Mu, J., Liu, H., Zhang, X.: Graphnet: graph clustering with deep neural networks. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3800–3804. IEEE (2021)
31. Zhao, X., Sala, A., Zheng, H., Zhao, B.Y.: Efficient shortest paths on massive social graphs. In: 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp. 77–86. IEEE (2011)

Correction to: Leveraging the Power of Signatures for the Construction of Topological Complexes for the Analysis of Multivariate Complex Dynamics

Stéphane Chrétien, Ben Gao, Astrid Thébault Guiochon, and Rémi Vaucher

Correction to:
Chapter 24 in: H. Cherifi et al. (Eds.): *Complex Networks & Their Applications XII*, SCI 1141,
https://doi.org/10.1007/978-3-031-53468-3_24

In the original version of this chapter, the Chapter author name Guiochon, A.T has been corrected as "Astrid Thébault Guiochon" the first name is Astrid and the last name "Thébault Guiochon," and the citation should read as "Thébault Guiochon, A.".

The updated version of this chapter can be found at
https://doi.org/10.1007/978-3-031-53468-3_24

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, p. C1, 2024.
https://doi.org/10.1007/978-3-031-53468-3_39

Author Index

H. Cherifi et al. (Eds.): COMPLEX NETWORKS 2023, SCI 1141, pp. 465–466, 2024.
https://doi.org/10.1007/978-3-031-53468-3