



Augmentation and heterogeneous graph neural network for AAI2021-COVID-19 fake news detection

Andrea Stevens Karnyoto¹ · Chengjie Sun¹ · Bingquan Liu¹ · Xiaolong Wang¹

Received: 21 May 2021 / Accepted: 29 December 2021 / Published online: 8 January 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract

Misinformation has become a frightening specter of society, especially fake news that concerning Covid-19. It massively spreads on the Internet, and then induces misunderstandings of information to the national and global communities during the pandemic. Detecting massive misinformation on the Internet is crucial and challenging because humans have struggled against this phenomenon for a long time. Our research concerns detecting fake news related to covid-19 using augmentation [random deletion (RD), random insertion (RI), random swap (RS), synonym replacement (SR)] and several graph neural network [graph convolutional network (GCN), graph attention network (GAT), and GraphSAGE (SAmple and aggreGatE)] model. We constructed nodes and edges in the graph, word-word node, and word-document node to graph neural network. Then, we tested those models in different amounts of sample training data to obtain accuracy for each model and compared them. For our fake news detection task, we found training accuracy steadily increasing for GCN, GAT, and SAGE models from the beginning to the end of the epochs. This result proved that the performance of GNN, whether GCN, GAT, or SAGE gained an entirely insignificant difference precision result.

Keywords COVID-19 fake news detection · Graph neural network · Text augmentations · Text classification

1 Introduction

Misinformation has become a regular history of civilization, especially the rampant circulation of fake news. The spread of fake news on online media is very dangerous [1, 2]. Furthermore, the effects can lead to casualties [3], psychological effects [4, 5], character assassination [5], elections for political parties [6], and state chaos [7]. Fake news concerning Covid-19 spread massively resulted in information misunderstandings to the national and global communities during the pandemic. Detecting this misinformation on the

Internet is a crucial task and challenging because humans are struggling against this phenomenon for long time ago.

Our research concerns detecting fake news that associated to covid-19 using the Constraint @ AAI2021-COVID19 fake news detection in English dataset [8] and Natural Language Processing Approaches. We used the dataset for training and testing are released by the "Constraint shared task organizer" [9], which aims to fight fake news that related to COVID-19 on social media platforms like Facebook, Twitter, Instagram, and other popular news website. The dataset contains 10,700 social media posts, and their labels are categorized into real and fake; all text is written in English. Several previous studies have contributed to this Constraint @ AAI2021—COVID19 fake news detection in English shared task utilized various methods. Such as Azhan et al. [10] applied pre-trained ULMFiT, Kakwani et al. [11] compiled the IndicGLUE benchmark, Baris et al. [12] proposed a modeling framework for those features by using BERT. Considering the challenging task and the number of research studies using this dataset, we decided to get involved in the COVID19 fake news detection shared-task [8] using a different method.

✉ Andrea Stevens Karnyoto
andre@ukitoraja.ac.id

Chengjie Sun
cjsun@insun.hit.edu.cn

Bingquan Liu
liubq@hit.edu.cn

Xiaolong Wang
wangxl@insun.hit.edu.cn

¹ School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

To contribute to the COVID19 fake news detection shared-task, we utilized easy data augmentation (EDA) [13] and graph neural network [14–18]. Data augmentation techniques have been employed in image processing, visual recognition, and text classification projects as it is simple to create and generate new data by straightforward and fast transformations [19]. Augmentation aims to increase the number of training data samples to reduce the overfitting of the model [20, 21]. The main concept of graph neural networks (GNNs) are neural models that can seize the relationship of graphs via messages passing between the nodes of graphs. GNNs have reached sophisticated performance in many research tasks such as predicting protein interface, learning molecular fingerprints, modeling physics systems, and modeling to analyzing disease. Also, it was used for graph classification tasks, link prediction, and many nodes classification [22]. Some research stated that GNN can gain good performance even on a small number of rows in the dataset [23, 24]. Many evolutions of GNNs like graph recurrent network (GRN), graph attention network (GAT), graph convolutional network (GCN) have proved innovative achievements on various deep learning projects [25].

Our contributions are as follows:

1. For the input of graph models, we concatenated of word co-occurrence in the entire corpus (word-word edges) and word occurrence in documents (document-word edges) to create edges among nodes. Then, we utilized the term frequency-inverse document frequency (TF-IDF) of the word inside the document that has resulted from word node and document node edge weight. We tested three graph models to train datasets with and without text augmentation. Three graphs are graph convolutional network, graph attention network, and GraphSAGE (SAmple and aggreGatE). For the augmentation method, we utilized easy data augmentation [random deletion (RD), random insertion (RI), random swap (RS), synonym replacement (SR)]. GraphSAGE with data augmentation gave the highest precision and F1-score in our experiment.
2. In order to prove the resistance of precision and the robustness, we trained our proposed method in various dataset sizes (30, 50, 80, and 100%) of Constraint @ AAI2021—COVID19 fake news detection dataset shared task.

Additionally, as a study concerning graph neural network and text augmentation, our proposed approach improved precision even though the size of dataset training is limited. Also, it proved that our proposed model gains slightly different precision for all datasets that we are tested. We arranged the rest of the paper subsequently. Section 2 formally defines related work in fake news detection, text augmentations, and

graph neural network. Our proposed method (the dataset, pre-processing, the main model, and the explanation of each graph neural network) is described in Sect. 3. And for Sect. 4, we explained the experiment and task. In Sect. 5, we present the result. We put the conclusion in Sect. 6 of this paper.

2 Related work

2.1 Fake news detection

The current state of misleading information prevention and fake news detection research reveals that most experiments used the textual level for their detection-based. It is extensively known in Natural Language Processing (NLP) tools [26]. Some fake news detection studies use Machine Learning [27–29], and others use Deep Learning [30, 31]. Generally, those techniques can be classified as Social Context-based learning and News Content-based learning. News content-based methods deal with the different writing styles of published news articles, focusing on extracting several texts that can be categorized as misleading information related to word order in a text and the writing style. Social context-based approaches deal with the latent connection words between the writer and news article. We can apply social engagements as a noteworthy feature for detecting fake news (to get the semantic connection between writers and their news articles) [32]. In the field of fake news detection research, many datasets have been utilized, such as PolitiFact [33, 34], Fake news Kaggle [27, 35], the fake news challenge (FNC-1) [36, 37], and Constraint@AAAI2021—COVID19 fake news detection [9–12]. Ahmad et al. [27] used Machine Learning Ensemble Methods that consisting of Random Forest (RF), Support Vector Machine, and Logistic Regression to develop fake news detection research. Monti et al. [30] proposed a method that works on graph-structured data. They claimed that their approach is a novel class of deep learning techniques that exploit deep geometric learning to learn fake news-specific propagation patterns. Konkobo et al. [33] created a model to distill expression of users' comments and opinions. Their work evaluated users' credibility and built a small network using the CredRank algorithm toward a piece of given news.

2.2 Text augmentation

Augmentation has helped numerous classification tasks. The data structure should be added and modified to increase the number of row samples and the possibility of incomplete samples availability. More diversity patterns fed into the model during training will improve model generalizes. As a result, the system will give better prediction when performed

by using new examples. Many websites offer automatic paraphrasing nowadays, but the most reliable way to augment sentences is by employing experts who have language skills/native speaker even though it is costly. Textual data augmentation requires language modeling rules written by experts in linguistics, but this task is still challenging due to human resources limitations and low-resource languages. We observed that many studies had been done to handle augmentation problems in text classification, for example, Word2vec-based augmentation [38], WordNet-based [13], Semantic-enriched Representation [39], and many more.

In this paper, we used “Easy data augmentation techniques for boosting performance on text classification tasks [13]” to complete our experiments. EDA using WordNet to generated random insertions and all synonyms for synonym replacements. It uses 300-dimensional word embeddings trained using GloVe. First, induced some noise to helps prevent overfitting by producing augmented data that is alike to original data. Second, this approach introduces new words as a substitute when synonym replacement and random insertion operations are activated, approving models to induce vocabulary in the test set that was not in the training set. EDA provides several features that result in text augmentations: (1) random deletion (RD): base on probability p , randomly remove individual words in the text. (2) Random insertion (RI): replace random words with random synonyms that nearest meaning other than those that are stopwords. Put that synonym into any position. Repeat it for n times. (3) Random swap (RS): swap two words are chosen randomly in the text, and it will change the position. Repeat it for n times. (4) Synonym replacement (SR): randomly pick n words from the text other than is stopwords. Then change those words with a random synonym. Example result of EDA (Table 1):

2.3 Graph neural network

Recently, graph classifications have been solved a significant task and achieved excellent performance for many sophisticated applications. Many researchers from various organizations have developed graph neural networks effectively, relying on graph embeddings to preserving global structure knowledge, and it is based on relational structure.

A new research area that uses graph embeddings and graph neural networks has moved to broad field of research [40, 41]. Even they expand their research further. For example, Zhang et al. [42] facilitated regular convolution operations computing subgraphs in the quantum walk to build subgraph convolutional network (SCN). Zhang et al. [43], based on Weisfeiler–Lehman (WL) algorithm, built the SortPooling layer-based to get vertex information globally and locally for recent deep graph convolutional neural network (DGCNN) model as a result. On text classification, some modern studies investigated graph neural networks [15, 44–47]. However, they also surveyed a text or a sentence as a graph of word nodes [45–47] or made articles citation relation to build the graph for the not-routinely-available [15].

Our work is based on the following method:

GCN Gori et al. [24] and Scarselli et al. [48] created recurrent neural networks based on graphs as neural networks relation. Later, Li et al. (2016) assembled the original graph neural network framework into recurrent neural network training in sophisticated practices. Duvenaud et al. [49] by using graphs and methods for graph-level analysis to made a convolution-like propagation rule. Kipf et al. [15] developed convolutional neural networks based on the spectral graph; this idea first time is introduced by Bruna et al. [50], and then it was extended by Defferrard et al. [45] using fast localized convolutions. Last, Liang Yao et al. [14], try to build a single text graph on word co-occurrence in the corpus-based; they implemented GCN for text classification.

GAT Graph attention network can be categorized as a partial part of MoNet [30]. It is also made edges vector sharing based on neural network relation. It is implicative of the formulation of relational systems [51] and VAIN [52], wherein relationships between agents or objects are aggregated pairwise by applying a shared work system. As another attention model, this approach can be related to Denil et al. [54] and Duan et al. [53], which employ a nearest node attention operation. Finally, Veličković et al. [16] introduced the recent neural network designs that work on graph-structured data, which are called graph attention networks (GATs).

GraphsSAGE This approach has been successfully implemented for a wide range of purposes. GraphsSAGE (SAmple and aggreGatE) conceptually related to node embedding approaches [55–59], supervised learning over graphs [23,

Table 1 Example sentence are resulted by easy data augmentation

Operation	Sentence
None	Graph classifications have been solved a significant task and achieved excellent performance
Synonym replacement (SR)	Graph classifications have been solved a significant job and achieved magnificent performance
Random insertion (RI)	Graph classifications have been solved a significant task and achieved excellent performance sophisticated
Random swap (RS)	Graph performance have been solved a significant task and achieved excellent classifications
Random deletion (RD)	Graph classifications have been solved a task and achieved excellent performance

24], and graph convolutional networks [45, 49, 50]. GraphSAGE [17] to train a model that produces embeddings uses leverage feature information for node embedding approaches toward unseen nodes. For supervised learning, GraphSAGE generating useful representations for individual nodes. Furthermore, for the graph convolution network, an approach can be seen as an enlargement of the GCN framework to the inductive environment.

3 Proposed method

3.1 Dataset statistics

The Constraint @ AAI2021—COVID19 fake news detection in English dataset [8] provided the shared task dataset, which contains 10,700 humans interpreted from media articles and posts are acquired from multiple platforms. It is divided into data training (6420 rows), validation (2140 rows), and test (2140 rows). The unique words in the training dataset are 30,046, most length tokens are 1481, and balance data distribution for Real and Fake labels. The dataset contains the post ID, tweet, and label fields (Table 2).

Table 3 shows post tweets containing URL, Mention, Retweet, Hashtag, HTML special entities, and Number.

3.2 Data preprocessing

First, we executed our tweet preprocessing and text preprocessing to remove useless punctuation marks for text

Table 2 Data distribution for constraint @ AAI2021—COVID19 fake news detection

Data	Real	Fake	Total	Unique Word
Train	3360	3060	6420	30,046
Validation	1120	1120	2140	13,697
Testing	1120	1120	2140	14,121

Table 3 Some post fake and real

Label	Post
Real	This #FourthOfJuly weekend if you choose to spend time outdoors at an event or gathering stay 6 ft apart & wear a cloth face cover to slow the spread of #COVID19. Learn more at https://t.co/c4F0aouMLd . https://t.co/u5tTl3m572
Real	We launched the #COVID19 Solidarity Response Fund which has so far mobilized \$225 + M from more than 563,000 individuals companies & philanthropies. In addition we mobilized \$1 + billion from Member States & other generous to support countries-@DrTedros https://t.co/xgPkPdv0r
Fake	@realDonaldTrump has shifted his focus at different moments in the #CoronavirusOutbreak. We updated our running timeline of his response to the virus. https://t.co/pgXjssaRCB Reply to us with any recent Trump moments you think belong on this running list. https://t.co/g4WYcppDSO
Fake	RT @EllenCutch: Coronavirus misinformation is moving offline. A reddit user posted this flyer to the site and told us it had been delive...

classification. We kept symbols '@' and '#' due to those have specific semantics in tweets. Second, we transformed the text into lowercase and replaced URLs, mentions, and emojis into unique tokens. Third, we used the Python emoji library to displace the emoji with a short textual description: redheart:,thumbsup:, etc. Furthermore, we converted hashtags into words ("#COVID" → "COVID").

3.3 Augmentation

We are using Easy Data Augmentation Techniques, which be made up of four robust yet straightforward operations: random deletion (RD), random insertion (RI), random swap (RS), synonym replacement (SR) [13]. In our case, generating original sentences into more than four augmented sentences was not helpful since models tend to generalize properly when a massive number of samples are available. According to easy data augmentation [13] paper, the following are recommendations usage parameters for EDA tools in Table 4.

The Alpha (α) parameter roughly means "percent of words in sentence changed by each augmentation." N_{aug} is the number of augmented sentences that were generated using EDA per original sentence. Since our N_{train} is more than 5000 rows, we used parameters $\alpha = 0.1$ and $N_{aug} = 4$ to produce additional rows for SR, RI, RS, and RD.

3.4 Graph convolutional networks (GCN)

A graph convolutional network (GCN) is a multilayer neural network based on the properties of their neighborhoods to

Table 4 Recommended usage parameters

N_{train}	α	n_{aug}
500	0.05	16
2000	0.05	8
5000	0.1	4
More	0.1	4

induce embedding vectors of nodes and operates directly on a graph. The base formulation is $G=(V, \mathcal{E})$, where \mathcal{E} is sets of edges and V with $|V|=n$ is set of nodes. With edges $(v_i, v_j) \in \mathcal{E}$, N nodes $v_i \in V$, a degree matrix $D_{ii}=\sum_j A_{ij}$ and an adjacency matrix $A \in \mathbb{R}^{N \times N}$ (binary or weighted).

Figure 1 shows each document node connected to many word nodes by edges. Also, word nodes are connected to many document nodes by edges. Document nodes start with "O," and others nodes are described as word nodes. Document-word edges are drawn in bold black edges, and magenta edges are word-word edges. X is a matrix of node feature vectors (words in sentence). $R(x)$ is representation (embedding) of X . Dotted lines are document classes (two classes are shown as described in a different color).

As shown in Fig. 1, a significant and heterogeneous Text GCN [14] contains document nodes and word nodes. It makes global word co-occurrence can be clearly modeled, and graph convolution can be easily adjusted. The sum of the number of unique words (vocabulary size) in a corpus and the number of documents (corpus size) yields the number of nodes in the text graph $|V|$. A one-hot vector as the input from every word or document is interpreted as feature matrix X for graph convolution network. The concatenate of word co-occurrence in the whole corpus (word-word edges), and word occurrence in documents (document-word edges) created edges among nodes. The term frequency-inverse document frequency (TF-IDF) of the word in the document has resulted from word node and document node edge weight.

Graph convolution neural network was the introduction in GCN, and the formula is defined by dgl.ai [18] as follows:

$$h_i^{(l+1)} = \sigma \left(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{1}{c_{ji}} h_j^{(l)} W^{(l)} \right)$$

where $\mathcal{N}(i)$ is the set of neighbors of node i , c_{ji} is the product of the square root of node degree (i.e., $c_{ji} = \sqrt{\mathcal{N}(i)}\sqrt{\mathcal{N}(j)}$). $\sigma(\cdot)$ is an activation function; such as the $\text{ReLU}(\cdot) = \max(0; \cdot)$. It will apply an activation function to the updated node features. $h_j^{(l)}$ is the current input feature. $h_i^{(l+1)}$ is the current output feature. $H^{(l)} \in \mathbb{R}^{N \times M}$ is the matrix of activation in the l th layer; and $H^{(0)} = X$. C_{ji} is the way to apply normalizer. There are three types of normalizers are ‘right,’ ‘none,’ and ‘both’. If the normalizer is ‘right,’ divide the aggregated messages by each node in degrees, which is equivalent to averaging the received message. If the normalizer is ‘none,’ no normalization is applied. In our case we are using ‘both’. $b^{(l)}$ is bias, and function will add a learnable bias to the output. $W^{(l)}$ is a layer-specific trainable weigh matrix. If a weight tensor on each edge is provided, the weighted graph convolution is defined as:

$$h_i^{(l+1)} = \sigma(b^{(l)} + \sum_{j \in \mathcal{N}(i)} \frac{e_{ji}}{c_{ji}} h_j^{(l)} W^{(l)})$$

where e_{ji} is the scalar weight on edge from node j to node i . Also, e_{ji} is NOT variable which is equivalent to the weighted graph convolutional network formulation.

3.5 Graph attention networks

The input to our GAN layer is a set of node features, $h = \{\vec{h}_1, \vec{h}_2, \dots, \vec{h}_N\}$, $\vec{h}_i \in \mathbb{R}^F$, where N is the number of nodes, and F is the number of features in each node. We used the same method to transform features into nodes as shown in Fig. 1 in the word document graph box. The output of first layer is set of node feature, $h^i = \{\vec{h}'_1, \vec{h}'_2, \dots, \vec{h}'_N\}$, $\vec{h}'_i \in \mathbb{R}^{F'}$.

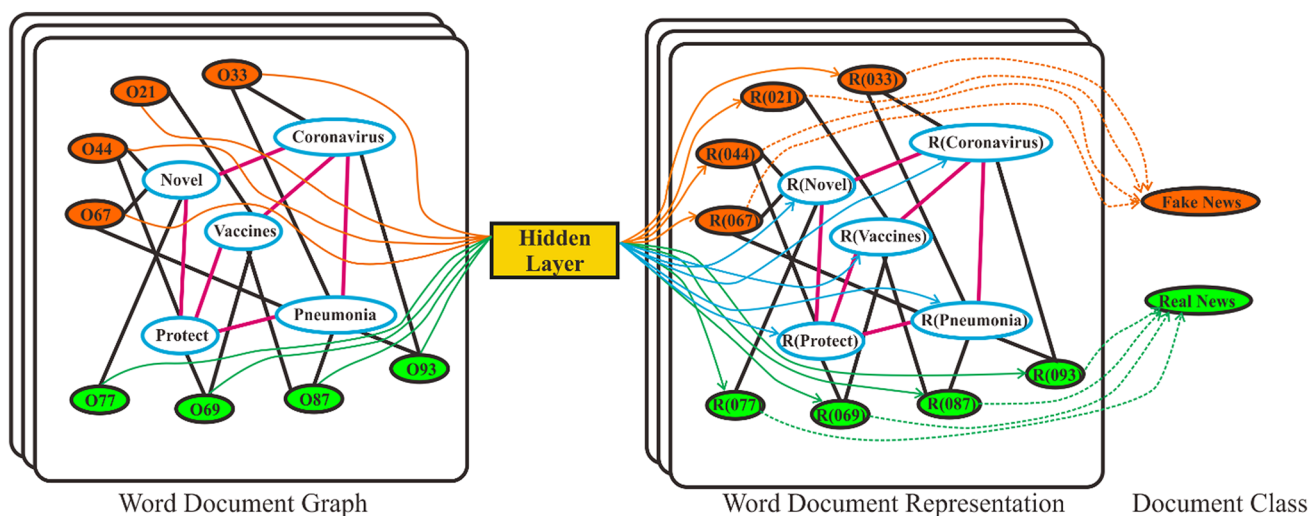


Fig. 1 Diagram of text GCN

According to dgl.ai [18], we use the following equation to get the value of each edge:

$$e_{ij}^l = \text{LeakyReLU}(\vec{a}^T |Wh_i| |Wh_j|)$$

W is weight matrix that is produced from shared linear transformation, $W \in \mathbb{R}^{F' \times F}$, it is applied to every node. \vec{a}^T is self-attention on the nodes, it is an attentional mechanism, $\vec{a}^T : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$. e_{ij}^l is an edges values as attention coefficients. *LeakyReLU* is Leaky Rectified Linear Unit. It is a type of activation function based on a ReLU, but it has a small slope for negative values instead of a flat slope.

$$\alpha_{ij}^l = \text{softmax}_i(e_{ij}^l)$$

where α_{ij}^l is the attention score between node i and node j . *Softmax* is normalized exponential function as logistic function to get regression values from e_{ij}^l .

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{i,j} h_j^{(l)} W^{(l)}\right)$$

where, $h_i^{(l)}$ is the current input feature. $h_i^{(l+1)}$ is the current output feature.

3.6 GraphSAGE

GraphSAGE uses forward propagation for embedding generation. The GraphSAGE approach begins by considering the model has already been trained, and the weight matrices and aggregator function parameters are set. For each node, this algorithm aggregates information from the node's neighbors iteratively, the node's neighbors-neighbors, and so on. In this paper, we used the same method to transform features into nodes as shown in Fig. 1 in the word document graph box. According to dgl.ai [18], explanation of GraphSAGE in next following equation:

$$h_{N(i)}^{(l+1)} = \text{aggregate}(\{h_j^l, \forall j \in \mathcal{N}(i)\})$$

$$h_i^{(l+1)} = \sigma(W \cdot \text{concat}(h_i^l, h_{N(i)}^{(l+1)}))$$

$$h_i^{(l+1)} = \text{norm}(h_i^l)$$

where, $h_i^{(l)}$ is the current input feature. $h_i^{(l+1)}$ is the current output feature. $\sigma(\cdot)$ is an activation function. Aggregation step (1) depends on the representations generated at the previous iteration. After aggregating the neighboring feature vectors, GraphSAGE then concatenates the node's current representation $h_{N(i)}^{(l+1)}$, with the aggregated neighborhood vector h_i^l , and this concatenated vector is supplied through a fully connected layer with nonlinear activation function σ ,

which implements the representations to be applied at the next step. If a weight tensor on each edge is provided, the aggregation becomes:

$$h_{N(i)}^{(l+1)} = \text{aggregate}(\{e_{ij} h_j^l, \forall j \in \mathcal{N}(i)\})$$

where e_{ij} is the scaler weight on the edge from node j to node i . e_{ij} is broadcastable with h_j^l .

4 Experiment and task

4.1 Experiment setup

The experiments ran on Intel Core (TM) i7 8700, 6 core 3.20 GHz Processor, 16 GB RAM, Nvidia GeForce GTX 1050 Ti GPU 4 GB. Base program and training use Python 3.7.8 and TensorFlow Version: 2.1.0. We used TfidfVectorizer from NLTK 3.5 and Cosine Similarity from Scipy 1.4.1 for preprocessing and get the distance between words. We used dgl.ai as the main tools for build the graph. Several important hyper-parameters determine this architecture: training model learning-rate=0.01, epoch 15, batch-size 16.

4.2 Description of task

To gain comprehensive results and prove our proposed method got good precision, we tested models for various training dataset sizes. We took 30, 50, 80, and 100 percent of the total dataset rows, then those are referred to Train-30, Train-50, Train-80, and Train-100, respectively. Moreover, for datasets that use augmentation, we defined them as Train-30 + Aug, Train-50 + Aug, Train-80 + Aug, and Train-100 + Aug. Next, we did preprocess and augmentation as explains in the proposed method. In this process, the dataset is processed to produced more rows due to the addition of random deletion (RD), random insertion (RI), random swap (RS), and synonym replacement (SR). Then, we ran the remove-word process and build-graph to construct nodes and edges in the graph, word-word node, and word-document node. Furthermore, we tested each model on the testing dataset, and the results can be seen in the next chapter.

5 Result and analysis

This section presents the experiment results and analysis from the proposed method and experiments and tasks that we detailed in previous chapters. We arrange the result and analysis base on experiments order and it contain discussion for each table and figure.

5.1 Statistic and precision result

Table 5 shows the dataset statistic after preprocessing and augmentation process. The training dataset rows number with augmentation is always five times more than training data without augmentation because we used the original text and added four texts that are generated using SR, RI, RS, and RD. The increasing number of tokens after augmentation shows that our augmentation method successfully generates more new vocabulary in the text. The max number of tokens column indicates that the augmentation makes the text longer on average.

We utilized precision and F1-score as metric to compare the reliability of each model. Precision is important because it represents closeness between label prediction and actual label that the model can guess correctly. And the F1-score is important as it represents the mean of precision and accuracy. As displayed in Table 6, we used non-graph model network (LSTM and CNN) as baseline, and the three graph

methods (GCN, GAT, GraphSAGE). Graph group with Train-100 + Aug dataset obtained the highest precision, recall, and f1-score due to more diverse vocabulary than other datasets as shown in Table 5. In contrast to LSTM and CNN, the two non-graph networks got the highest results on Train-80 + Aug. We assume that this is due to training structure model differences even though GNN and the non-graph networks use TfidfVectorizer as primary process input. The worse precision was obtained using Train-30, Train-30, and Train-80 for GCN, GAT, and GraphSAGE, respectively. We assume Train-80 in the GraphSAGE group got worse result because of the epoch iterations for this condition is still underfitting in 15 epochs.

5.2 Effect of augmentation operation on precision

We tested each augmentation operation on models for comprehensive results. We took Train-100 + Aug dataset and then tested selected operation one by one. After that we

Table 5 Dataset statistic after preprocessing and augmentation

	Total train rows and validation	Total tokens	Sentence statistics		
			Min number of token	Max number of token	Average number of token
Without augmentation					
Train-30	2569	4709	2	58	12.55
Train-50	4281	6421	2	248	12.94
Train-80	6848	8988	2	259	13.34
Train-100	8558	10,689	2	263	13.48
With augmentation					
Train-30 + Aug	12,845	14,985	2	63	14.16
Train-50 + Aug	21,405	23,545	2	317	14.28
Train-80 + Aug	34,240	36,380	2	314	14.38
Train-100 + Aug	42,790	44,930	2	317	14.39

Table 6 Precision and F1-score of testing results

	Non-graph network				Graph neural network					
	LSTM		CNN		GCN		GAT		GraphSAGE	
	Prec	F1	Prec	F1	Prec	F1	Prec	F1	Prec	F1
Train-30	0.8401	0.8672	0.8017	0.7997	0.8947	0.8946	0.8925	0.8922	0.9015	0.9017
Train-50	0.893	0.8764	0.8397	0.8372	0.9012	0.9006	0.9012	0.9012	0.9096	0.9096
Train-80	0.8875	0.8916	0.9019	0.9619	0.9136	0.9133	0.9117	0.9114	0.8944	0.8919
Train-100	0.8957	0.8893	0.8909	0.8906	0.9137	0.9133	0.9058	0.9059	0.9204	0.9204
Train-30 + Aug	0.8681	0.8679	0.8793	0.8789	0.8975	0.8963	0.8969	0.8964	0.9057	0.9053
Train-50 + Aug	0.8868	0.8867	0.9068	0.9068	0.9053	0.9043	0.9025	0.9015	0.9129	0.9123
Train-80 + Aug	0.9150	0.9148	0.9130	0.9130	0.9160	0.9151	0.9152	0.9142	0.9090	0.9065
Train-100 + Aug	0.9141	0.9141	0.9073	0.9073	0.9183	0.9175	0.9183	0.9175	0.9279	0.9279

Bold values indicate the highest performance for each model

Table 7 Comparison between operation

Operation	Precision				
	LSTM	CNN	GCN	GAT	SAGE
Random deletion (RD)	0.9118	0.9091	0.9142	0.9148	0.9249
Random insertion (RI)	0.9143	0.9121	0.9171	0.9176	0.9277
Random swap (RS)	0.9128	0.9108	0.9160	0.9162	0.9257
Synonym replacement (SR)	0.9098	0.9075	0.9125	0.9129	0.9224
RD+RI+RS+SR	0.9150	0.9130	0.9183	0.9183	0.9279

discussed which operation has the most impact on testing precision.

Table 7 shows that using the whole operation yield the best results. When only one operator was activated, random insertion (RI) got the highest and closest precision to the precision of the entire operation was activated. This result is due to RI adding new vocabulary in the text without eliminating other existing vocabulary. The lowest precision was obtained by random deletion (RD) because the drawback of one of the words in the text will affects the quality of the training process.

5.3 Model framework analysis

Table 6 shows graphSAGE gained the highest precision in almost every testing group result. This condition happened because the main structure of graphSAGE is a general inductive framework. The graph model can start their node simultaneously in one complete text and go deep to the most important nodes. The graphSAGE mechanism works by generating embedding using samples and aggregators from neighboring nodes for the beginning process. In our case, this mechanism creeps every salient word in the text, with our TF-IDF approach making it easy to find a node sample. In other words, unlike standard convolution in GCN, graphSAGE also uses multiple aggregators. The aggregator function can be formed from the Mean aggregator, LSTM aggregator, or pooling aggregator. This kind of graph does not use all their neighbors but utilizes

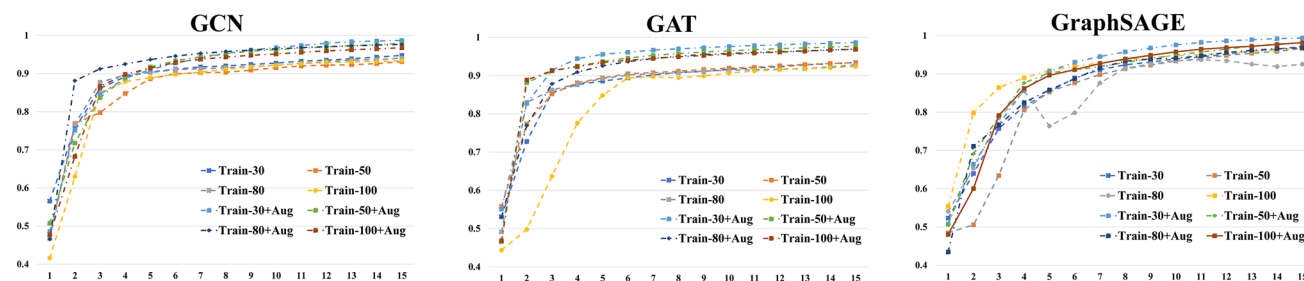
fixed-size by uniform sampling. The sampling method uses task-dependent heuristics and applies uniform sampling and node as one of the sampling methods. GraphSAGE also replaces complete Laplacian graphs with learnable aggregations, allowing graphSAGE to select or skip hidden nodes or select the most valuable nodes. Finally, so it looks like graphSAGE uses a random neighbor sampling method to alleviate receptive field expansion. This method is different from GAT and GCN, which utilize all of their neighbors, making GAT and GCN more time-consuming and unsuitable for massive Graph structures. For our case, graphSAGE is more relevant and robust.

Table 6 also shows the most model that gains the lowest precision so often is GAT. Although the attention mechanism has been effectively applied in sequence-based applications such as machine translation, machine reading, and so on, the attention mechanism for GNN yielded the lowest result in our experiments. Compared to standard GCN, which treats all neighbors node equally, the attention mechanism sets different attention scores and identifies the most important neighbor. We assume that GAT got the lowest score because GAT has a more profound ability to rearrange the vocabulary of its neighbor and gives less precise value to each word during the learning process, and it created some mistakes in weighted.

5.4 Comparison between same models for training accuracies

This sub section shows the result and comparison for experiments of all datasets from three models.

As shown in Fig. 2, the accuracies are pretty similar for all models at the end of the epoch. Train-30 + Aug Epoch got high precision in every model, and this figure shows that Train-30 + Aug raised the stability in accuracy escalation during the training dataset time. The worse accuracies are Train + 50, Train + 50, Train + 50, and Train + 80 for GCN, GAT, and GraphSAGE. All dataset accuracy results that produce the worst outcomes are datasets without augmentation. Meanwhile, datasets which gain highest

**Fig. 2** GCN, GAT, and GraphSAGE training accuracies

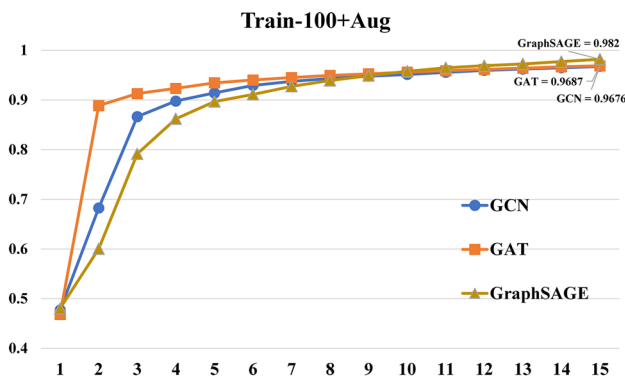


Fig. 3 Comparison of training accuracy for GCN, GAT, and SAGE toward Train-100 + Aug dataset

Table 8 Effect of sentences length toward classification accuracies

	Sentence length (words)			
	≤ 15	16–30	31–45	≥ 46
GCN	0.9182	0.9226	0.8723	0.6000
GAT	0.9174	0.9237	0.8723	0.6000
SAGE	0.9265	0.9351	0.8936	0.6000

Bold values indicate the highest accuracy for each sentence length group

accuracy are datasets with augmentation, even though those datasets do not use 100 percent rows.

5.5 Comparison between different the number of samples in different models

Different from Table 6, which shows datasets that got the highest precision, recall, and f1-score for testing dataset. Furthermore, we compare and discuss GCN, GAT, and GraphSAGE toward Train-100 + Aug dataset for training dataset.

According to Fig. 3, we found a steady accuracy increase for GCN, GAT, and GraphSAGE models in our fake news detection task start from the beginning to the end of epochs. As we can see, the difference in accuracy between each result is also very close one to another, it is only about 0.0096. This result proved that the performance of GNN, whether GCN, GAT, or GraphSAGE produced an entirely insignificant difference. According to Table 6, we found the same precision result for GCN and GAT, but the highest is the GraphSAGE model. We hypothesize these small gaps precision between models due to exceptionally trained datasets, which producing a helpful data encoding.

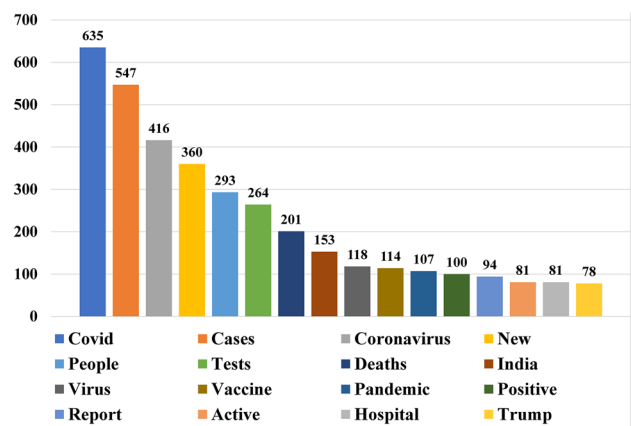


Fig. 4 The most common term was detected by our models in The Constraint @ AAI2021—COVID19 fake news detection dataset

5.6 Effect of sentence length

In our testing dataset, there are 1210 rows (56.54%) contain 15 words or less, 878 rows (41.02%) have 16–30 words, 47 rows (2.19%) contain 31–45 words and five rows (0.23%) contain more than 45.

Table 8 shows the effect of sentence length for each model towards Train-100 + Aug dataset. The sentences that contain 16–30 words got better accuracy for most models. All models have the same accuracy trend and a small range between them, which shows that ability of each model's detection is not significantly different. Moreover, sentences that contain over 45 words have the lowest accuracy due to the uneven distribution of data.

5.7 Most common terms in fake news our model can detect

We also explored the most frequent words in our fake news dataset that our model can detect. However, we found the terms frequently used in fake news are similar to those that appeared most often in the entire dataset.

Figure 4 shows the most frequently used words in the dataset that our model can detect. The context of every line in the dataset may contain more than one most common word. Because the dataset is related to covid-19, we only present the essential terms related to covid-19. The words "Covid," "Cases," and "Coronavirus" are the most frequently used words in this fake news dataset, but in fact those words are common to appear in the news media and attract readers the most. After the "India" term, the next frequently used words do not significantly differ in number.

6 Conclusion

Detecting this misinformation on the Internet is a crucial task and challenging because humans have struggled against this phenomenon for a long time. People also faced difficulty detecting fake news based on context. To deal with the COVID19 fake news detection shared-task, we used easy data augmentation (EDA) and Graph neural network (graph convolutional network (GCN), graph attention network (GAT), and GraphSAGE (SAmple and aggreGatE)). We took 30, 50, 80, and 100 percent of the total dataset rows and then augmented them. The Train-100 + Aug dataset obtained the highest precision, recall, and f1-score due to a more diverse vocabulary than other datasets. The worse precision was obtained using train-30, train-30, and train-80 for GCN, GAT, and GraphSAGE. The precision is slightly different. For our case, graphSAGE is more relevant and robust. GraphSAGE replaces complete Laplacian graphs with learnable aggregations, allowing graphSAGE to select or skip hidden nodes or select the most valuable nodes. It looks like graphSAGE uses a random neighbor sampling method to alleviate receptive field expansion. It is different with GAT and GCN, which utilize all of their neighbors; GraphSAGE is less time-consuming. The sentences that contain 16–30 words got better accuracy for most models. The words "Covid," "Cases," and "Coronavirus" are the most frequently used words in this fake news dataset.

Funding NLP Lab.

Declarations

Conflict of interest The authors (Andrea Stevens Karnyoto, Chengjie Sun, Bingquan Liu, Xiaolong Wang) of the paper (Title: Augmentation and Heterogeneous Graph Neural Network for AAAI2021-COVID19 Fake News Detection) declare that there is no conflict of interests.

References

- Shu K, Mahudeswaran D, Wang S, Lee D, Liu H (2020) Fakeneedsnet: a data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data* 8(3):171–188
- Riedel B, Augenstein I, Spithourakis GP, Riedel S (2017) A simple but tough-to-beat baseline for the fake news challenge stance detection task. arXiv preprint <http://arxiv.org/abs/1707.03264>
- Apuke OD, Omar B (2020) Fake news proliferation in Nigeria: consequences, motivations, and prevention through awareness strategies. *Humanit Soc Sci Rev* 8(2):318–327
- Ozbay FA, Alatas B (2020) Fake news detection within online social media using supervised artificial intelligence algorithms. *Physica A* 540:123174
- Pulido CM, Ruiz-Eugenio L, Redondo-Sama G, Villarejo-Carballedo B (2020) A new application of social impact in social media for overcoming fake news in health. *Int J Environ Res Public Health* 17(7):2430
- Maldonado MA (2019) Understanding fake news: technology, affects, and the politics of the untruth. *Historia y Comunicación Social* 24(2):533
- Waisbord S (2018) Truth is what happens to news: on journalism, fake news, and post-truth. *J Stud* 19(13):1866–1878
- Constraint-shared-task-2021. Available: <https://constraint-shared-task-2021.github.io/> (current April 2021)
- Akhtar MS, Chakraborty T (2021). Overview of constraint 2021 shared tasks: detecting English covid-19 fake news and Hindi hostile posts. In: Combating online hostile posts in regional languages during emergency situation: first international workshop, CONSTRAINT 2021, collocated with AAAI 2021, virtual event, February 8, 2021, Revised Selected Papers (p. 42). Springer Nature
- Azhan M, Ahmad M (2021) LaDiff ULMFiT: a layer differentiated training approach for ULMFiT. arXiv preprint <http://arxiv.org/abs/2101.04965>
- Kakwani D, Kunchukuttan A, Golla S, Gokul NC, Bhattacharyya A, Khapra MM, Kumar P (2020) iNLPSuite: monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In: Proceedings of the 2020 conference on empirical methods in natural language processing: findings, pp 4948–4961
- Baris I, Boukhers Z (2021) ECOL: early detection of COVID lies using content, prior knowledge and source information. arXiv preprint <http://arxiv.org/abs/2101.05499>
- Wei J, Zou K (2019) Eda: easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint <http://arxiv.org/abs/1901.11196>
- Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, no 01, pp 7370–7377
- Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv preprint <http://arxiv.org/abs/1609.02907>
- Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv preprint <http://arxiv.org/abs/1710.10903>
- Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. arXiv preprint <http://arxiv.org/abs/1706.02216>
- Wang M, Zheng D, Ye Z, Gan Q, Li M, Song X, Zhang Z et al (2019) Deep graph library: a graph-centric, highly-performant package for graph neural networks. arXiv preprint <http://arxiv.org/abs/1909.01315>
- Giridhara PKB, Mishra C, Venkataramana RKM, Bukhari SS, Dengel A (2019) A study of various text augmentation techniques for relation classification in free text. *ICPRAM* 3:5
- Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE, pp 248–255
- Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst* 25:1097–1105
- Xu K, Hu W, Leskovec J, Jegelka S (2018) How powerful are graph neural networks? arXiv preprint <http://arxiv.org/abs/1810.00826>
- Dai H, Dai B, Song L (2016) Discriminative embeddings of latent variable models for structured data. In: International conference on machine learning. PMLR, pp 2702–2711
- Gori M, Monfardini G, Scarselli F (2005) A new model for learning in graph domains. In: Proceedings. 2005 IEEE international joint conference on neural networks, 2005, vol 2. IEEE, pp 729–734

25. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Sun M et al (2020) Graph neural networks: a review of methods and applications. *AI Open* 1:57–81
26. Granskogen T (2018) Automatic detection of fake news in social media using contextual information. Master's thesis, NTNU
27. Ahmad I, Yousaf M, Yousaf S, Ahmad MO (2020) Fake news detection using machine learning ensemble methods. *Complexity* 2020:8885861
28. Gilda S (2017) Notice of violation of IEEE publication principles: evaluating machine learning algorithms for fake news detection. In: 2017 IEEE 15th student conference on research and development (SCOREd). IEEE, pp 110–115
29. Aphiwongsophon S, Chongstitvatana P (2018) Detecting fake news with machine learning method. In: 2018 15th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON). IEEE, pp 528–531
30. Monti F, Frasca F, Eynard D, Mannion D, Bronstein MM (2019) Fake news detection on social media using geometric deep learning. arXiv preprint <http://arxiv.org/abs/1902.06673>
31. Sahoo SR, Gupta BB (2021) Multiple features based approach for automatic fake news detection on social networks using deep learning. *Appl Soft Comput* 100:106983
32. Kaliyar RK, Goswami A, Narang P (2021) FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. *Multimedia Tools Appl* 80(8):11765–11788
33. Konkobo PM, Zhang R, Huang S, Minoungou TT, Ouedraogo JA, Li L (2020) A deep learning model for early detection of fake news on social media. In: 2020 7th international conference on behavioural and social computing (BESC). IEEE, pp 1–6
34. Oriola O (2021) Exploring N-gram, word embedding and topic models for content-based fake news detection in FakeNewsNet evaluation. *Int J Comput Appl* 975:8887
35. Shakeel D, Jain N (2021) Fake news detection and fact verification using knowledge graphs and machine learning
36. Xu J, Zadorozhny V, Zhang D, Grant J (2020) FaNDS: fake news detection system using energy flow. arXiv preprint <http://arxiv.org/abs/2010.02097>
37. Hassan FM, Lee M (2020) Multi-stage news-stance classification based on lexical and neural features. In: Conference on complex, intelligent, and software intensive systems. Springer, Cham, pp 218–228
38. Yan B, Janowicz K, Mai G, Gao S (2017) From itdl to place2vec: reasoning about place type similarity and relatedness by learning embeddings from augmented spatial contexts. In: Proceedings of the 25th ACM SIGSPATIAL international conference on advances in geographic information systems, pp 1–10
39. Lu X, Zheng B, Velivelli A, Zhai C (2006) Enhancing text categorization with semantic-enriched representation and training data augmentation. *J Am Med Inform Assoc* 13(5):526–535
40. Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Pascanu R et al (2018) Relational inductive biases, deep learning, and graph networks. arXiv preprint <http://arxiv.org/abs/1806.01261>
41. Cai H, Zheng VW, Chang KCC (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. *IEEE Trans Knowl Data Eng* 30(9):1616–1637
42. Zhang Z, Chen D, Wang J, Bai L, Hancock ER (2019) Quantum-based subgraph convolutional neural networks. *Pattern Recogn* 88:38–49
43. Zhang M, Cui Z, Neumann M, Chen Y (2018) An end-to-end deep learning architecture for graph classification. In: Proceedings of the AAAI conference on artificial intelligence, vol 32, no 1
44. Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. arXiv preprint <http://arxiv.org/abs/1506.05163>
45. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. arXiv preprint <http://arxiv.org/abs/1606.09375>
46. Peng H, Li J, He Y, Liu Y, Bao M, Wang L, Yang Q et al (2018) Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In: Proceedings of the 2018 world wide web conference, pp 1063–1072
47. Zhang Y, Liu Q, Song L (2018) Sentence-state lstm for text representation. arXiv preprint <http://arxiv.org/abs/1805.02474>
48. Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G (2008) The graph neural network model. *IEEE Trans Neural Netw* 20(1):61–80
49. Duvenaud D, Maclaurin D, Aguilera-Iparraguirre J, Gómez-Bombarelli R, Hirzel T, Aspuru-Guzik A, Adams RP (2015) Convolutional networks on graphs for learning molecular fingerprints. arXiv preprint <http://arxiv.org/abs/1509.09292>
50. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. arXiv preprint <http://arxiv.org/abs/1312.6203>
51. Santoro A, Raposo D, Barrett DG, Malinowski M, Pascanu R, Battaglia P, Lillicrap T (2017) A simple neural network module for relational reasoning. arXiv preprint <http://arxiv.org/abs/1706.01427>
52. Hoshen Y (2017) Vain: attentional multi-agent predictive modeling. arXiv preprint <http://arxiv.org/abs/1706.06122>
53. Duan Y, Andrychowicz M, Stadie BC, Ho J, Schneider J, Sutskever I, Zaremba W et al (2017) One-shot imitation learning. arXiv preprint <http://arxiv.org/abs/1703.07326>
54. Denil M, Colmenarejo SG, Cabi S, Saxton D, de Freitas N (2017) Programmable agents. arXiv preprint <http://arxiv.org/abs/1706.06383>
55. Cao S, Lu W, Xu Q (2015) 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
56. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
57. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
58. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077
59. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1225–1234

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.