Min Zhang · Vincent Ng · Dongyan Zhao
Sujian Li · Hongying Zan (Eds.)

LNAI 11109

Natural Language Processing and Chinese Computing

7th CCF International Conference, NLPCC 2018
Hohhot, China, August 26–30, 2018
Proceedings, Part II

2 Part II

中国计算机学会
CCF

Springer

Min Zhang · Vincent Ng
Dongyan Zhao · Sujian Li
Hongying Zan (Eds.)

Natural Language Processing and Chinese Computing

7th CCF International Conference, NLPCC 2018
Hohhot, China, August 26–30, 2018
Proceedings, Part II

Springer

Editors
Min Zhang
Soochow University
Suzhou
China

Sujian Li
Peking University
Beijing
China

Vincent Ng
The University of Texas at Dallas
Richardson, TX
USA

Hongying Zan
Zhengzhou University
Zhengzhou
China

Dongyan Zhao
Peking University
Beijing
China

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-319-99500-7 ISBN 978-3-319-99501-4 (eBook)
https://doi.org/10.1007/978-3-319-99501-4

Library of Congress Control Number: 2018951640

LNCS Sublibrary: SL7 – Artificial Intelligence

Preface

Welcome to the proceedings of NLPCC 2018, the 7th CCF International Conference on Natural Language Processing and Chinese Computing. Following the highly successful conferences in Beijing (2012), Chongqing (2013), Shenzhen (2014), Nanchang (2015), Kunming (2016), and Dalian (2017), this year's NLPCC was held in Hohhot, the capital and the economic and cultural center of Inner Mongolia. As a leading international conference on natural language processing and Chinese computing organized by CCF-TCCI (Technical Committee of Chinese Information, China Computer Federation), NLPCC 2018 served as a main forum for researchers and practitioners from academia, industry, and government to share their ideas, research results, and experiences, and to promote their research and technical innovations in the fields.

There is nothing more exciting than seeing the continual growth of NLPCC over the years. This year, we received a total of 308 submissions, which represents a 22% increase in the number of submissions compared with NLPCC 2017. Among the 308 submissions, 241 were written in English and 67 were written in Chinese. Following NLPCC's tradition, we welcomed submissions in eight key areas, including NLP Fundamentals (Syntax, Semantics, Discourse), NLP Applications, Text Mining, Machine Translation, Machine Learning for NLP, Information Extraction/Knowledge Graph, Conversational Bot/Question Answering/Information Retrieval, and NLP for Social Network. Unlike previous years, this year we intended to broaden the scope of the program by inviting authors to submit their work to one of five categories: applications/tools, empirical/data-driven approaches, resources and evaluation, theoretical, and survey papers. Different review forms were designed to help reviewers determine the contributions made by papers in different categories. While it is perhaps not surprising to see that more than 88% of the submissions concern empirical/data-driven approaches, it is encouraging to see three resources and evaluation papers and one theoretical paper accepted to the conference. Acceptance decisions were made in an online PC meeting attended by the general chairs, the Program Committee (PC) chairs, and the area chairs. In the end, 70 submissions were accepted as full papers (with 55 papers in English and 15 papers in Chinese) and 31 as posters. Six papers were nominated by the area chairs for the best paper award. An independent best paper award committee was formed to select the best papers from the shortlist. The proceedings include only the English papers accepted; the Chinese papers appear in *ACTA Scientiarum Naturalium Universitatis Pekinensis*.

We were honored to have four internationally renowned keynote speakers — Charles Ling, Joyce Chai, Cristian Danescu-Niculescu-Mizil, and Luo Si — share their views on exciting developments in various areas of NLP, including language communication with robots, conversational dynamics, NLP research at Alibaba, and megatrends in AI.

We could not have organized NLPCC 2018 without the help of many people:

- We are grateful for the guidance and advice provided by TCCI Chair Ming Zhou, General Chairs Dan Roth and Chengqing Zong, and Organizing Committee Co-chairs Dongyan Zhao, Ruifeng Xu, and Guanglai Gao.
- We would like to thank Chinese Track and Student Workshop Co-chairs Minlie Huang and Jinsong Su, as well as Evaluation Co-chairs Nan Duan and Xiaojun Wan, who undertook the difficult task of selecting the slate of accepted papers from the large pool of high-quality papers.
- We are indebted to the 16 area chairs and the 232 reviewers. This year, we operated under severe time constraints, with only a month between the submission deadline and the notification date. We could not have met the various deadlines during the review process without the hard work of the area chairs and the reviewers.
- We thank ADL/Tutorial Co-chairs Wenliang Chen and Rui Yan for assembling a tutorial program consisting of six tutorials covering a wide range of cutting-edge topics in NLP.
- We thank Sponsorship Co-chairs Kam-Fai Wong and Ming Zhou for securing sponsorship for the conference.
- Publication Co-chairs Sujian Li and Hongying Zan spent a tremendous amount of time ensuring every little detail in the publication process was taken care of and truly deserve a big applause.
- We thank Dan Roth, Xuanjing Huang, Jing Jiang, Yang Liu, and Yue Zhang for agreeing to serve in the best paper award committee.
- Above all, we thank everybody who chose to submit their work to NLPCC 2018. Without their support, we could not have put together a strong conference program.

Enjoy the conference as well as Hohhot' vast green pastures and natural sceneries!

July 2018

Vincent Ng
Min Zhang

Organization

NLPCC 2018 is organized by Technical Committee of Chinese Information of CCF, Inner Mongolia University and the State Key Lab of Digital Publishing Technology.

Organizing Committee

General Chairs

Dan Roth	University of Pennsylvania, USA
Chengqing Zong	Institute of Automation, Chinese Academy of Sciences, China

Program Co-chairs

Min Zhang	Soochow University, China
Vincent Ng	University of Texas at Dallas, USA

Area Chairs

NLP Fundamentals

Nianwen Xue	Brandeis University, USA
Meishan Zhang	Heilongjiang University, China

NLP Applications

Bishan Yang	Carnegie Mellon University, USA
Ruifeng Xu	Harbin Institute of Technology, China

Text Mining

William Yang Wang	University of California, Santa Barbara, USA
Ping Luo	Institute of Computing Technology, Chinese Academy of Sciences, China

Machine Translation

Fei Huang	Facebook, USA
Derek Wong	University of Macau, Macau, SAR China

Machine Learning for NLP

Kai-Wei Chang	University of California, Los Angeles, USA
Xu Sun	Peking University, China

Knowledge Graph/IE

Yun-Nung (Vivian) Chen	National Taiwan University, Taiwan
Wenliang Chen	Soochow University, China

Conversational Bot/QA

Jianfeng Gao	Microsoft AI and Research, USA
Haofen Wang	Gowild.cn, China

NLP for Social Network

Wei Gao	Victoria University of Wellington, New Zealand
Bo Wang	Tianjin University, China

Organization Co-chairs

Guanglai Gao	Inner Mongolia University, China
Ruifeng Xu	Harbin University of Technology, China
Dongyan Zhao	Peking University, China

ADL/Tutorial Chairs

Wenliang Chen	Soochow University, China
Rui Yan	Peking University, China

Student Workshop Chairs

Minlie Huang	Tsinghua University, China
Jinsong Su	Xiamen University, China

Sponsorship Co-chairs

Kam-Fai Wong	The Chinese University of Hong Kong, SAR China
Ming Zhou	Microsoft Research Asia, China

Publication Chairs

Sujian Li	Peking University, China
Hongying Zan	Zhengzhou University, China

Publicity Chairs

Wanxiang Che	Harbin Institute of Technology, China
Qi Zhang	Fudan University, China
Yangsen Zhang	Beijing University of Information Science and Technology, China

Evaluation Chairs

Nan Duan	Microsoft Research Asia
Xiaojun Wan	Peking University, China

Program Committee

Wasi Ahmad	University of California, Los Angeles, USA
Xiang Ao	Institute of Computing Technology, Chinese Academy of Sciences, China
Deng Cai	Shanghai Jiao Tong University, China

Jinhua Du	Dublin City University, Ireland
Matthias Eck	Facebook, USA
Derek F. Wong	University of Macau, Macau, SAR China
Chuang Fan	Harbin Institute of Technology Shenzhen Graduate School, China
Chunli Fan	Guilin University of Electronic Technology, China
Yang Feng	Institute of Computing Technology, Chinese Academy of Sciences, China
Yansong Feng	Peking University, China
Guohong Fu	Heilongjiang University, China
Michel Galley	Microsoft Research, USA
Jianfeng Gao	Microsoft Research, Redmond, USA
Qin Gao	Google LLC, USA
Wei Gao	Victoria University of Wellington, New Zealand
Niyu Ge	IBM Research, USA
Lin Gui	Aston University, UK
Jiafeng Guo	Institute of Computing Technology, CAS, China
Jiang Guo	Massachusetts Institute of Technology, USA
Weiwei Guo	LinkedIn, USA
Yupeng Gu	Northeastern University, USA
Xianpei Han	Institute of Software, Chinese Academy of Sciences, China
Tianyong Hao	Guangdong University of Foreign Studies, China
Ji He	University of Washington, USA
Yanqing He	Institute of Scientific and Technical Information of China, China
Yifan He	Alibaba Inc., USA
Yulan He	Aston University, UK
Zhongjun He	Baidu Inc., China
Yu Hong	Soochow University, China
Dongyan Huang	Institute for Infocomm Research, Singapore
Fei Huang	Alibaba DAMO Research Lab, USA
Guoping Huang	Tencent AI Lab, China
Jiangping Huang	Chongqing University of Posts and Telecommunications, China
Ruihong Huang	Texas AM University, USA
Shujian Huang	Nanjing University, China
Ting-Hao Huang	Carnegie Mellon University, USA
Xiaojiang Huang	Microsoft, China
Xuanjing Huang	Fudan University, China
Zhiting Hu	Carnegie Mellon University, USA
Junyi Jessy Li	University of Texas at Austin, USA
Jingtian Jiang	Microsoft AI Research, USA
Shengyi Jiang	Guangdong University of Foreign Studies, China
Wenbin Jiang	Baidu Inc., China
Yuxiang Jia	Zhengzhou University, China

Yuan-Fang Li	Monash University, Australia
Zhenghua Li	Soochow University, China
Ping Luo	Institute of Computing Technology, CAS, China
Weihua Luo	Alibaba Group, China
Wencan Luo	Google, USA
Zhunchen Luo	PLA Academy of Military Science, China
Qi Lu	Soochow University, China
Wei Lu	Singapore University of Technology and Design, Singapore
Chen Lyu	Guangdong University of Foreign Studies, China
Yajuan Lyu	Baidu Company, China
Cunli Mao	Kunming University of Science and Technology, China
Xian-Ling Mao	Beijing Institute of Technology, China
Shuming Ma	Peking University, China
Yanjun Ma	Baidu, China
Yue Ma	Université Paris Sud, France
Fandong Meng	Tencent, China
Haitao Mi	Alipay US, USA
Lili Mou	AdeptMind Research, Canada
Baolin Peng	The Chinese University of Hong Kong, SAR China
Haoruo Peng	UIUC, USA
Nanyun Peng	University of Southern California, USA
Longhua Qian	Soochow University, China
Tao Qian	Wuhan University, China
Guilin Qi	Southeast University, China
Yanxia Qin	Harbin Institute of Technology, China
Likun Qiu	Ludong University, China
Xipeng Qiu	Fudan University, China
Weiguang Qu	Nanjing Normal University, China
Feiliang Ren	Northerstern University, China
Yafeng Ren	Guangdong University of Foreign Studies, China
Huawei Shen	Institute of Computing Technology, Chinese Academy of Sciences, China
Wei Shen	Nankai University, China
Xiaodong Shi	Xiamen University, China
Wei Song	Capital Normal University, China
Aixin Sun	Nanyang Technological University, Singapore
Chengjie Sun	Harbin Institute of Technology, China
Weiwei Sun	Peking University, China
Yu Su	University of California Santa Barbara, USA
Duyu Tang	Microsoft Research Asia, China
Zhi Tang	Peking University, China
Zhiyang Teng	Singapore University of Technology and Design, Singapore
Jin Ting	Hainan University, China
Ming-Feng Tsai	National Chengchi University, Taiwan, China

Yuen-Hsien Tseng	National Taiwan Normal University, Taiwan, China
Zhaopeng Tu	Tencent AI Lab, China
Bin Wang	Institute of Information Engineering, Chinese Academy of Sciences, China
Chuan Wang	Google Inc., USA
Di Wang	Carnegie Mellon University, USA
Haofen Wang	Shenzhen Gowild Robotics Co. Ltd., China
Kun Wang	Alibaba, China
Longyue Wang	Dublin City University, Ireland
Quan Wang	Institute of Information Engineering, Chinese Academy of Sciences, China
Xiaojie Wang	Beijing University of Posts and Telecommunications, China
Zhiguo Wang	IBM Watson Research Center, USA
Zhongqing Wang	Soochow University, China
Zhongyu Wei	Fudan University, China
Hua Wu	Baidu, China
Yunfang Wu	Peking University, China
Tong Xiao	Northestern University, China
Yanghua Xiao	Fudan University, China
Rui Xia	Nanjing University of Science and Technology, China
Wayne Xin Zhao	RUC, China
Chenyan Xiong	Carnegie Mellon University, USA
Deyi Xiong	Soochow University, China
Shufeng Xiong	Wuhan University, China
Jun Xu	Institute of Computing Technology, CAS, China
Kun Xu	IBM T.J. Watson Research Center, USA
Endong Xun	Bejing Language and Cultural University, China
Jie Yang	Singapore University of Technology and Design, Singapore
Liang Yang	Dalian University of Technology, China
Liner Yang	Tsinghua University, China
Liu Yang	University of Massachusetts Amherst, USA
Yating Yang	The Xinjing Technical Institute of Physics and Chemistry, CAS, China
Zi Yang	Google, USA
Oi Yee Kwong	The Chinese University of Hong Kong, SAR China
Peifeng Yin	IBM Almaden Research Center, USA
Wenpeng Yin	University of Pennsylvania, USA
Bei Yu	Syracuse University, USA
Dong Yu	Beijing Language and Culture University, China
Junjie Yu	Soochow University, China
Mo Yu	IBM Research, USA
Zhengtao Yu	Kunming University of Science and Technology, China
Xiangrong Zeng	Institute of Automation, Chinese Academy of Sciences, China

Ying Zeng	Peking University, China
Feifei Zhai	Sogou Inc., China
Chengzhi Zhang	Nanjing University of Science and Technology, China
Dongdong Zhang	Microsoft Research Asia, China
Fan Zhang	University of Pittsburgh, USA
Fuzheng Zhang	MSRA, China
Min Zhang	Tsinghua University, China
Peng Zhang	Tianjin University, China
Qi Zhang	Fudan University, China
Weinan Zhang	Shanghai Jiao Tong University, China
Xiaodong Zhang	Peking University, China
Xiaowang Zhang	Tianjin University, China
Yongfeng Zhang	Rutgers University, USA
Yue Zhang	Singapore University of Technology and Design, Singapore
Hai Zhao	Shanghai Jiao Tong University, China
Jieyu Zhao	University of California, Los Angeles, USA
Sendong Zhao	Harbin Institute of Technology, China
Tiejun Zhao	Harbin Institute of Technology, China
Guoqing Zheng	Carnegie Mellon University, USA
Deyu Zhou	Southeast University, China
Guangyou Zhou	Central China Normal University, China
Hao Zhou	Bytedance AI Lab, China
Junsheng Zhou	Nanjing Normal University, China
Ming Zhou	Microsoft Research Asia, China
Muhua Zhu	Alibaba Inc., China

Organizers

Organized by

China Computer Federation, China

Supported by

Asian Federation of Natural Language Processing

Hosted by

Inner Mongolia University

State Key Lab of Digital Publishing Technology

In Cooperation with:

Lecture Notes in Computer Science

Springer

ACTA Scientiarum Naturalium Universitatis Pekinensis

Sponsoring Institutions

Primary Sponsors

AI Strong

JINGDONG

Diamond Sponsors

Tencent LINGO Lab

ZHINENGYIDIAN

Sogou

AISPEECH

LIULISHUO

China Mobile

Alibaba Group

GTCOM

Platinum Sponsors

Microsoft

Baidu

Leyan Tech

Laiye

Huawei

GRID SUM

LENOVO

AITC

Unisound

XIAOMI

CVTE

ByteDance

WISERS

Golden Sponsors

NiuParser

SoftBank

Genelife

Contents – Part II

Contents – Part I

NLP for Social Network

A Fusion Model of Multi-data Sources for User Profiling in Social Media

Liming Zhang[1], Sihui Fu[1], Shengyi Jiang[1,2(✉)], Rui Bao[1], and Yunfeng Zeng[1]

[1] School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou, China
zhangliming134@foxmail.com, jiangshengyi@163.com
[2] Engineering Research Center for Cyberspace Content Security of Guangdong Province, Guangzhou, China

Abstract. User profiling in social media plays an important role in different applications. Most of the existing approaches for user profiling are based on user-generated messages, which is not sufficient for inferring user attributes. With the continuous accumulation of data in social media, integrating multi-data sources has become the inexorable trend for precise user profiling. In this paper, we take advantage of text messages, user metadata, followee information and network representations. In order to integrate seamlessly multi-data sources, we propose a novel fusion model that effectively captures the complementarity and diversity of different sources. In addition, we address the problem of friendship-based network from previous studies and introduce celebrity ties which enrich the social network and boost the connectivity of different users. Experimental results show that our method outperforms several state-of-the-art methods on a real-world dataset.

Keywords: User profiling · Social media · Multi-data sources
Fusion model

1 Introduction

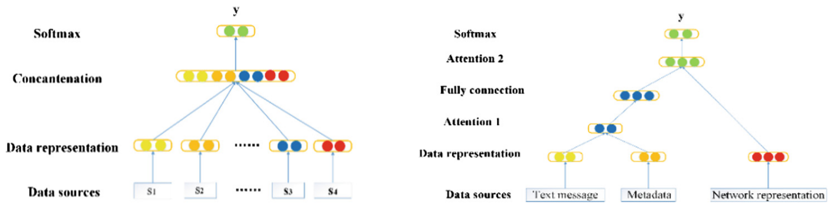
User profiling, which aims at effectively extracting user attributes from massive data information, is essentially valuable in various scientific research and business applications, such as recommendation system [1], search engine optimization [2], political position detection [3] and social network analysis [4]. Many attempts have utilized automated analysis model for user profiling tasks, such as user gender [5], age [6], geolocation [7], occupation [8], hobbies [9], personality [10] and influence [11].

With the rapid development of social media, like Twitter and Facebook, user profiling in social media obtains increasing attention. Traditional approaches [12–15] are mainly based on user-generated messages, such as tweets and micro-blogs, from which they construct a series of sophisticated features as the input of machine learning algorithms. Nevertheless, user-generated messages are usually short and full of noisy information. In addition to user's posts, recently, there have been several attempts to utilize other data sources in social media. Among them, social relationships network

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 3–15, 2018.
https://doi.org/10.1007/978-3-319-99501-4_1

[16, 17] takes the most important role. Intuitively, people who become friends are likely to share the similar attributes. However, one of the existing problems in the friendship-based network is that if the training model does not include the user's friends in the network, the model may fail to predict the user's attributes.

Apart from text messages and network information, social media provides other data sources, such as users' nicknames, self-introductions, and personalized labels, which are also helpful for the identification of user attributes. Nowadays, more and more data sources are generated in social media, integrating multi-data sources has thus become the inexorable trend for precise user profiling. One naïve baseline for integration is to combine all the data representations one after another, as shown in Fig. 1(a). However, it does not consider interactions between different data sources. To address this, [18] built a hierarchical attention mechanism that unified text message, metadata and network representation for user geolocation prediction, as shown in Fig. 1(b). However, the attention mechanism may cause a certain loss of information when features derived from different data sources are combined via the summation operation. In addition, attention mechanism ignores the correlations between different data sources.



(a) Concatenation (b) Attention

Fig. 1. Previous baseline models

To better integrate different data sources, we take the advantage of bi-GRU architecture, which simultaneously captures the complementarity and diversity of each data source from bi-directions with the update gates and the reset gates. Different from the attention mechanism which integrates diverse inputs with a summation operation, we concatenate all the hidden states as the hybrid features, in order to retain the diversities of different data sources. We evaluate our model on a real-world dataset and experiment results show that our model outperforms the aforementioned methods. Besides, we also address the problem of scarcity in friendship network. Practically, we incorporate celebrity ties into the social network to enrich the information of user network representations.

The rest of the paper is organized as follows. In Sect. 2, we briefly review on the related work of user profiling. Then, we deliver a detailed description of our model in Sect. 3. Section 4 presents experimental results along with our analysis. Finally, we make a conclusion in Sect. 5.

2 Related Work

Most previous work in user profiling heavily relied on hand-crafted syntactic and lexical features extracted from the user-generated messages. [12] analyzed bloggers' writing styles and high frequency words at different genders and ages. [14] extracted stylistic features and lexical features from users' blogs, using SVM and logistic regression model to predict users' ages. Recently, deep learning methods have been applied in user profiling tasks and shown their effectiveness against traditional approaches. [19] devised a joint learning model with Long Short-Term Memory model to distinguish users' genders, ages, and professions.

Besides, network analysis targeting at node interactions in a connected network becomes a hot field over these years [20–22]. In network analysis methods, user profiling is treated as a node classification problem. [23] incorporated text features into network representation by matrix factorization. [4] devised a framework that preserves user profiles in the social network embeddings via a nonlinear mapping.

In social media, users are free to generate various types of data. It is found that the combination of two or more types of data is distinctly better than merely a single type in prediction tasks of user profiles. [24] established multi-scale Convolutional Neural Networks with an attention mechanism on user-generated contents, combined with user network representations. [25] unified users' tweets and other metadata (location, time zone) to predict user geolocation with a stacking approach. [18] developed a complex neural network model that joins text messages, user metadata and network representations for geolocation prediction.

In this paper, we unify user text messages, user metadata, followees' information and network representations. Different from previous methods, our model seamlessly incorporates different data sources by taking advantage of both their complementarity and diversity.

3 Model

In this section, we introduce our model, a fusion framework which joints four different types of data sources. Shown in Fig. 2, the model takes user text messages, metadata, followees' information and network representations as inputs. Then, four components are treated as a sequence while a bi-GRU layer is employed to learn their interdependency. Subsequently, all hidden units are concatenated as a new vector representation to retain their differentia, and then they are fed into the final fully connected layer. Details of the sub-models will be discussed in Sects. 3.1, 3.2, 3.3 and 3.4.

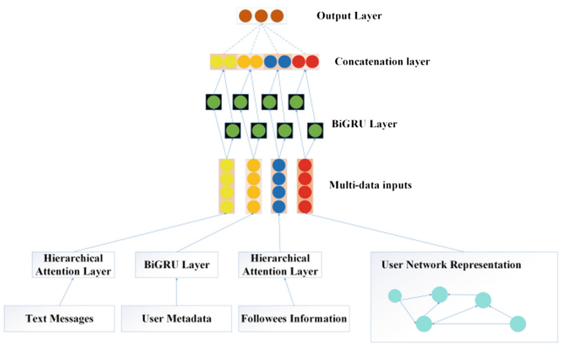


Fig. 2. Illustration of the fusion model. Hierarchical attention layer denotes hierarchical attention network. BiRNN denotes bi-recurrent neural network. Concatenation layer indicates concatenation of all hidden units learned from multi-data inputs.

3.1 Text Messages Representation

Text messages are the most important information for user profiling in social media. We take each message as one sentence formed by a sequence of words and aggregate all messages to be a document. To get the document presentation, We adopt the hierarchical attention network [26], in which there are two hierarchical layers. Figure 3 shows one typical hierarchical layer in the hierarchical attention network.

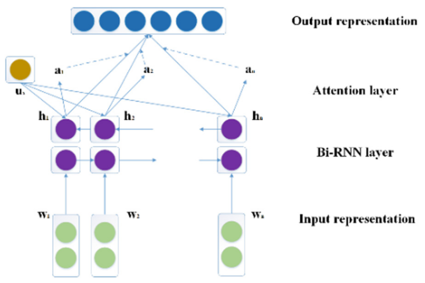


Fig. 3. The architecture of one hierarchical layer in the hierarchical attention network

The input representations can be the word embeddings, while the output comes to be a sentence embedding. Similarly, sentence embeddings can be combined into a document. To implement RNN, we use Gated Recurrent Unit (GRU) [27]. Formally, the formulations of GRU are as follows:

$$\mathrm{z_t} = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{1}$$

$$\mathrm{r_t} = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2}$$

$$\tilde{h}_{\mathrm{t}} = tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h) \tag{3}$$

$$h_{\mathrm{t}} = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_{\mathrm{t}} \tag{4}$$

where $\mathrm{z_t}$ denotes an update gate, $\mathrm{r_t}$ a reset gate, $\tilde{h}_{\mathrm{t}}$ a candidate state, h_{t} a hidden state, and x_t an input state, W_z, W_r, W_h, , U_z, U_r, U_h, b_z, b_r, b_h are model parameters, $\odot$ denotes the element-wise multiplication operator.

We get the hidden presentation h_{it} by concatenating forward hidden state $\vec{h}_{it}$ and backward hidden state $\overleftarrow{h}_{it}$. Then, a self-attention mechanism is introduced, which automatically assigns weights to different inputs. The formulation of the self-attention mechanism is defined as follows:

$$e_{\mathrm{i}} = tanh(W_w h_i + b_w) \tag{5}$$

$$\alpha_i = \frac{\exp\left(u_w^T \mathrm{e_i}\right)}{\sum_i \exp\left(u_w^T \mathrm{e_i}\right)} \tag{6}$$

$$\mathrm{s} = \sum\nolimits_i \alpha_i h_i \tag{7}$$

where α_i is the weight of i-th of the hidden unit h_i, and u_w is the context vector, W_w and b_w are model parameters, s is the output vector.

3.2 Metadata Representation

Apart from text messages, user metadata information is also useful for inferring user attributes. In this paper, we regard *user nickname*, *self-introduction*, *education information*, *work information* and *individualized labels* as user metadata. We represent the metadata by concatenating all the elements, feeding them into a BiRNN layer and an Attention layer to the metadata representation.

3.3 Network Representation

For solving the sparsity of user friendships, most of previous works construct a 2-degree friends network. However, constructing such a network is very labor-intensive and time-consuming and most users are not well-connected. Therefore, an effective measure is to intensify the relationships among users. According to our observation, celebrity ties can be an alternative way to boost the connectivity between different users. As shown in Fig. 4, although user *B* and C do not have an explicit friendship, they have an indirect relationship as they both follow celebrity D and have interactions in blogs. Specifically, we define a social network as G = (V, E), where V represents the vertices including ordinary users as well as celebrities, and E indicates the relationships between vertices. There are three types of social relationship in our network: friendships, follower-followee relationships and blog-interacted relationships (*@mention*, *repost* and *vote*). We employ LINE [22] which involves the first-order and second-order proximities between nodes to obtain users' vector representations in the social network, where we set all the weights of edges set to be 0 and 1.

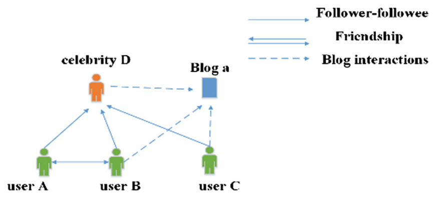


Fig. 4. An example of social network relationships

3.4 Followees' Information Representation

In addition to the construction of social network, we notice that the information of followee, especially celebrities, has strong relations with the user's traits. For example, if a user follows a certificated company that sells cosmetics, we suppose the user may be a female. In the followee list, followee usually put an explicit description of their nickname and self-introduction. Herein, we join the nickname and description to form a sentence representing one followee, and adopt another hierarchical attention network, shown as Fig. 5, to get the whole representation of followees' information.

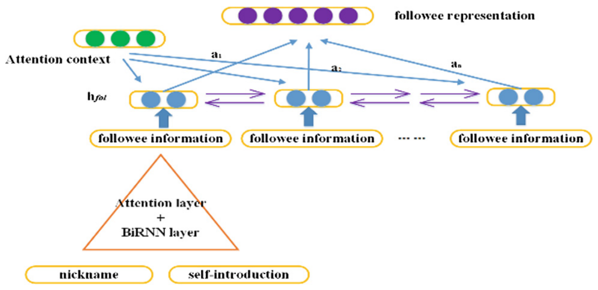


Fig. 5. Model architecture for getting followee information representation

3.5 Fusion Framework of Multi-data Sources

Given a series of data vector representations $\{s_1, s_2, s_3, s_4\}$, we adopt a Bi-GRU layer to learn their interdependencies adaptively on both forward and backward directions, resulting in corresponding hidden vectors $\{h_1, h_2, h_3, h_4\}$. Then, we concatenate all the hidden units, sending them through a fully-connected layer. In practice, the lengths of

$\{s_1, s_2, s_3, s_4\}$ may not be the same, so we adjust the hidden size of sub-models to ensure the same lengths of different inputs. Formally, a user vector representation v_u is computed by:

$$v_u = W_c[h_1 \oplus h_2 \oplus h_3 \oplus h_4] + b_c \tag{8}$$

$$h_i = f_{\mathrm{BiGRU}}(s_i) \tag{9}$$

where W_c, b_c are the model parameters.

We adopt cross-entropy error of the predicted and true distributions as the loss function for optimization. The loss function is defined as follows:

$$L = -\sum_{i=1}^{T} \sum_{j=1}^{C} y_i^j \log\left(\hat{y}_i^j\right) \tag{10}$$

where T denotes the number of training sets, C denotes the number of classes in user profiling tasks, and $y_i^j, \hat{y}_i^j$ are the true labels and prediction probabilities respectively.

4 Experiments

Since there is few public benchmark data set on user profiling yet, we collect a real-word date set and evaluate our models in three user profiling tasks: gender, age and location prediction. Users for evaluation are from Sina Micro-blog[1], one of the most popular social media websites in China. We gathered user accounts from the comment lists of Micro-blog hot-searched event from March to April, 2018. Finally, we get 31,852 users with 21,884 females and 9,968 males, as females are more likely to share their comments than males. For the fair comparison, we adopt down sampling strategy that cuts off the sample number of female to 10,942. Figure 6 shows the distribution of user in age. We split ages into four intervals: [<19, 19–23, 24–27, >27], corresponding to different educational ages. User locations are in accordance with seven regions of China. Table 1 summarizes the statistics of our data set, where *NE* denotes the Northeast of China, *N* the North of China, *C* the Center of China, *E* the East of China, *NW* the Northwest of China, *SW* the Southwest of China, and *S* the South of China. We randomly select 70% as the training set, 10% the validation set and 20% the test set.

4.1 Experiment Setting

We use the pre-trained word2vec [28] vectors with a dimension of 200, and employ an open source Chinese Segmentation tool Jieba[2] to process the Chinese text. Words not included in the word2vec vectors are endowed with a uniform distribution between [−0.25, 0.25]. We adopt Adam algorithm [29] for optimization, and mini-batch size is fixed to 32. We train LINE [22] to get network representations of first-order

[1] https://weibo.com/.

[2] https://github.com/fxsjy/jieba.

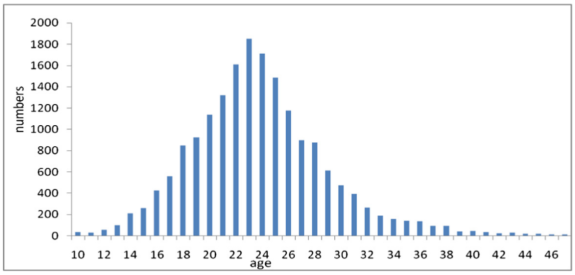


Fig. 6. Distributions of user in age

Table 1. Statistics of datasets for user profiling evaluation

Gender	Male			Female			
	9968			10942			
Age	<19		19–23	24–27		>27	
	3407		5914	5273		3716	
Location	NE	N	C	E	NW	SW	S
	2364	4359	3058	8144	1445	2951	3827

embeddings and second-order embeddings with 150 dimensions respectively. The GRU dimensions of words and sentences are set to be 50 and 150 respectively in hierarchical attention network. Since user metadata only comprises one bi-RNN layer, we set the GRU dimensions to be 150. The size of all attention units is set to 300, the dropout rate is 0.5. For evaluation, we use the metrics of F1 measures.

4.2 Baseline Methods

We compare our model with several baseline methods for user profiling tasks:

Text Feature + SVM: In traditional methods, bag of words (BOW) is regularly used for extracting text feature. In our experiments, different words in BOW model are weighted by TF-IDF, and then we adopt Singular Vector Decomposition (SVD) to perform dimensionality reduction, feeding them into an SVM classifier.

Text Feature + HAN: HAN represents the hierarchical attention network. Here, we use HAN only for extracting text messages features as the baseline method.

Multi-CNN + Network Embedding (MCNE) [24]**:** Each sentence is learnt by a convolutional neural network and an attention mechanism is used to assign weights to different sentences. Both the messages embedding and the network embedding are concatenated as the user embedding.

Multi-Data + Concatenation (MDC): Four types of data inputs are simply concatenated for the fully-connected layer, with details shown in Fig. 1(a).

Multi-Data + Attention (MDA) [18]**:** Three types of text information are aggregated by an attention mechanism layer, and the output is summed up with the network vector by another attention mechanism layer, with details shown in Fig. 1(b).

4.3 Experimental Results and Analysis

We report results of our proposed model on user profiling tasks along with baseline methods in Table 2.

Table 2. Results of user profiling tasks on Sina micro-blogs data set

	Gender	Age	Location
SVM	76.5	43.8	48.7
HAN	81.8	48.7	61.7
MCNE	84.2	46.8	65.6
MDC	85.5	51.5	68.1
MDA	85.5	52.7	67.5
Our model	**86.6**	**56.3**	**70.2**

We can see that deep learning methods outperform remarkably better than BOW. MCNE is superior to HAN in gender and location prediction significantly, but slightly inferior in age prediction. The possible reason is that user gender and location can be inferred by local salient features, like "homeboy", "Beijing", while age requires more information of different aspects and their relations. In this respect, RNN performs better than CNN. In general, methods that unifies multi-data sources boost the performance compared to merely text-messages based methods (SVM, HAN) and the MCNE method. However, MDA does not make a significant improvement compared to the baseline MDC. The reason may be that aggregating different inputs by an attention mechanism may loss the peculiarity of each data source and information is partly lost when features are shortened to a quarter of the original inputs by the summation operation. Among all the methods, our model consistently and significantly performs the best in all user profiling tasks, especially the age identification task with 4.8% improvements of performance compared with MDC. This indicates our model effectively takes advantage of the relativity and diversity of different data sources.

Visualization: To further illustrate our proposed model, we use t-SNE [30] to make a 2-dimensional visualization of user embedding vectors in the test dataset. As shown in Fig. 7, we can observe a clear division of node distributions in gender, location tasks with 2 and 7 distinct regions respectively, where users with the same attributes are clustered tightly. Although age prediction has the lowest accuracy, we still observe two separated parts, in which the left denotes users who are above 24 years old and the right under 24. Besides, we can see a gradual change of colors from right to left (dark blue, blue, green and orange), which suggests the transition of different age groups (<19, 19–23, 24–27, 27 correspondingly). Thus, results of visualization give a strong evidence of the good performance of our model.

Sequential Order Analysis: To provide more insight into our proposed fusion model, we investigate the influence of sequence order before the BiRNN-layer. Specifically, let t denote the text messages representation, i denote the user metadata representation, ne denote the network representation, and f denote the followee information. Table 3 reports results on four types of different combinations. It reveals

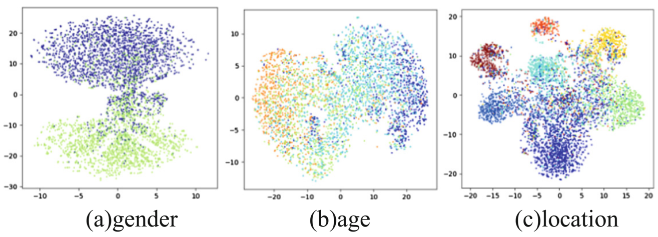


Fig. 7. 2-dimensional visualization of user embeddings in user profiling tasks

the significant differences among different orders in the sequence. The reason is mainly because of the effect of the update state and the reset state in GRU, which will reduce the information from the previous hidden state if it has little relevance to the current state. Herein, we observe that the order *ne-u-t-f* obtains the best performance, partly because the middle position of text messages can effectively capture both the information from user metadata and followee information from bi-directions since they all are the text representations and show a strong relevance intuitively. Besides, text messages preserve the most valuable information for inferring user attributes. A practical issue is how to let our model find the sequence order automatically, which we leave for our future work.

Table 3. Results of fusion framework with different orders of sequential combination.

Order	Gender	Age	Location
t_f_u_ne	85.68	55.03	68.64
t_u_f_ne	86.29	54.89	65.91
f_ne_t_u	86.24	53.46	68.59
ne_u_t_f	**86.69**	**56.38**	**70.24**

Network Analysis: To verify the effect of celebrity ties, we construct five different social networks by adding *friend nodes*, *celebrity nodes from followee lists*, *celebrity nodes from @mention behaviors*, *celebrity nodes from repost behaviors* and *celebrity nodes from vote behaviors* incrementally. We obtain the network embeddings by using LINE and feed them into the MLP classifier. Figure 8 shows the results on different network scales, from which we are apparently aware of the remarkable promotion by adding *celebrity nodes from followee lists*, with a drastic increase of 10% in F1-measure in gender and age prediction. The possible reason is that celebrity nodes boost the connectivity of different users who are not friends, and thus enrich the network structure. In addition, we also observe a significant effect of *repost* and *vote* behaviors on location predictions, probably due to the fact that users usually repost and vote the micro-blogs concerning local reports.

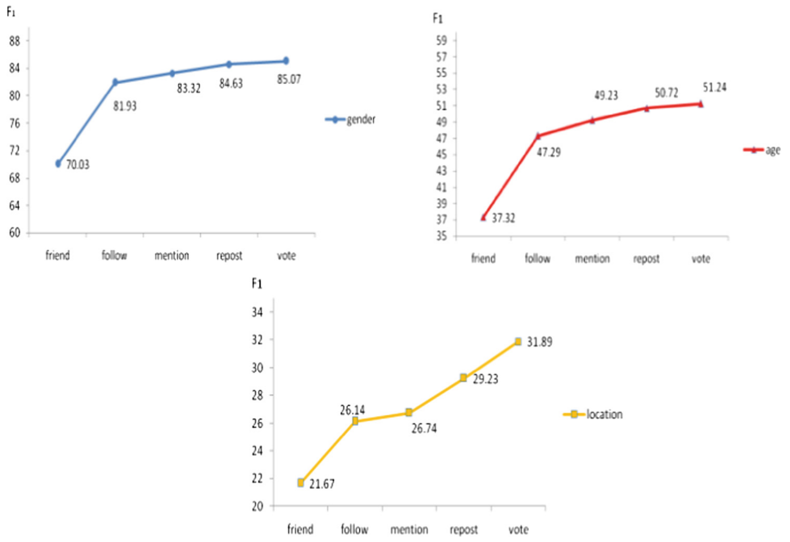


Fig. 8. Comparison on the performances of user profiling with different network scales

5 Conclusion

In this paper, we present a novel fusion model of multi-data sources for user profiling, which seamlessly integrates them by taking advantage of their relativity and diversity. Concretely, we carefully devise four different types of data sources: text messages, user metadata, followee information and network presentations, feeding them into different sub-models and integrating them via a hybrid bi-GRU framework. To alleviate the problem of weak connectivity in user-friendship based network, we innovatively incorporate celebrity nodes, noticing the indirect interactions between users via celebrity nodes. Experimental results show that our proposed model performs effectively on user profiling tasks. In the future, we will do further research on our model to implement an automatic mechanism for finding the best sequence combination of multi-data sources. In addition, we also plan to incorporate other different data sources, like images and videos that appear in user micro-blogs.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (No. 61572145) and the Major Projects of Guangdong Education Department for Foundation Research and Applied Research (No. 2017KZDXM031). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Lu, Z., Pan, S.J., Li, Y., Jiang, J., Yang, Q.: Collaborative evolution for user profiling in recommender systems. In: IJCAI International Joint Conference on Artificial Intelligence, pp. 3804–3810 (2016)
2. Zhou, M.: Gender difference in web search perceptions and behavior: does it vary by task performance? Comput. Educ. **78**(259), 174–184 (2014)
3. Preoţiuc-Pietro, D., Liu, Y., Hopkins, D., Ungar, L.: Beyond binary labels: political ideology prediction of twitter users. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 729–740 (2017)
4. Zhang, D., Yin, J., Zhu, X., Zhang, C.: User profile preserving social network embedding. In: Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 3378–3384 (2017)
5. Burger, J.D., Henderson, J., Kim, G., Zarrella, G.: Discriminating gender on Twitter. In: Conference on Empirical Methods in Natural Language Processing, pp. 1301–1309 (2011)
6. Chen, J., Li, S., Dai, B., Zhou, G.: Active learning for age regression in social media. In: China National Conference on Chinese Computational Linguistics, pp. 351–362 (2016)
7. Roller, S., Speriosu, M., Rallapalli, S., Wing, B., Baldridge, J.: Supervised text-based geolocation using language models on an adaptive grid. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1500–1510 (2012)
8. Preoţiuc-Pietro, D., Lampos, V., Aletras, N.: An analysis of the user occupational class through Twitter content. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 1754–1764 (2015)
9. Kim, H.R., Chan, P.K.: Learning implicit user interest hierarchy for context in personalization. Appl. Intell. **28**(2), 153–166 (2008)
10. Majumder, N., Poria, S., Gelbukh, A., Cambria, E.: Deep learning-based document modeling for personality detection from text. IEEE Intell. Syst. **32**(2), 74–79 (2017)
11. Lampos, V., Aletras, N.: Predicting and characterising user impact on Twitter. In: Conference of the European Chapter of the Association for Computational Linguistics, pp. 405–413 (2014)
12. Schler, J., Koppel, M., Argamon, S., Pennebaker, J.: Effects of age and gender on Blogging. In: Proceedings of AAAI Symposium on Computational Approaches for Analyzing Weblogs, pp. 199–205 (2006)
13. Ciot, M., Sonderegger, M., Ruths, D.: Gender inference of Twitter users in non-english contexts. In: Conference on Empirical Methods in Natural Language Processing, pp. 1136–1145 (2013)
14. Mukherjee, A., Liu, B.: Improving gender classification of blog authors. In: Conference on Empirical Methods in Natural Language Processing, pp. 158–166 (2010)
15. Marquardt, J., et al.: Age and gender identification in social media. In: Proceedings of CLEF 2014 Evaluation Labs, pp. 1129–1136 (2014)
16. Mislove, A., Viswanath, B., Gummadi, K., Druschel, P.: You are who you know: inferring user profiles in online social networks. In: Third ACM International Conference on Web Search and Data Mining, pp. 251–260 (2010)
17. Han, X., Wang, L., Crespi, N., Park, S., Cuevas, Á.: Alike people, alike interests? inferring interest similarity in online social networks. Decision Support Systems **69(C)**, 92–106 (2015)

18. Miura, Y., Taniguchi, M., Taniguchi, T., Ohkuma, T.: Unifying text, metadata, and user network representations with a neural network for geolocation prediction. In: Meeting of the Association for Computational Linguistics, pp. 1260–1272 (2017)
19. Wang, J., Li, S., Zhou, G.: Joint learning on relevant user attributes in micro-blog. In: IJCAI International Joint Conference on Artificial Intelligence, pp. 4130–4136 (2017)
20. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 855–864 (2016)
21. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations bryan. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
22. Tang, J., Qu, M.: LINE: large-scale information network embedding categories and subject descriptors. In: International World Wide Web Conferences Steering Committee, pp. 1067–1077 (2015)
23. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI International Joint Conference on Artificial Intelligence, pp. 2111–2117 (2015)
24. Zhao, Z., Du, J., Gao, Q., Gui, L., Xu, R.: Inferring user profile using microblog content and friendship network. In: Communications in Computer and Information Science, pp. 29–39 (2017)
25. Han, B., Cook, P., Baldwin, T.: A stacking-based approach to twitter user geolocation prediction. In: Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 7–12 (2013)
26. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489 (2016)
27. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. Computer Science (2014)
28. Mikolov, T., Corrado, G., Chen, K., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of the International Conference on Learning Representations, pp. 1–12 (2013)
29. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. Computer Science (2014)
30. Van Der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9** (2605), 2579–2605 (2008)

First Place Solution for NLPCC 2018 Shared Task User Profiling and Recommendation

Qiaojing Xie, Yuqian Wang, Zhenjing Xu, Kaidong Yu, Chen Wei(✉), and ZhiChen Yu

Turing Robot, Beijing, China
{xieqiaojing,wangyuqian,weichen}@uzoo.cn

Abstract. Social networking sites have been growing at an unprecedented rate in recent years. User profiling and personalized recommendation plays an important role in social networking, such as targeting advertisement and personalized news feed. For NLPCC Task 8, there are two subtasks. Subtask one is User Tags Prediction (UTP), which is to predict tags related to a user. We consider UTP as a Multi Label Classification (MLC) problem and proposed a CNN-RNN framework to explicitly exploit the label dependencies. The proposed framework employs CNN to get the user profile representation and the RNN module captures the dependencies among labels. Subtask two, User Following Recommendation (UFR), is to recommend friends to the users. There are mainly two approaches: Collaborative Filtering (CF) and Most Popular Friends (MPF), and we adopted a combination of both. Our experiments show that both of our methods yield clear improvements in F1@K compared to other algorithms and achieved first place in both subtasks.

Keywords: User profiling · User tags prediction
Multi label classification · Friend recommendation

1 Introduction

Nowadays, UTP and UFR have attracted great attention due to the popularity of social networks. For example, on social networks where people share information and connect to each other, users present themselves with demographical information as well as tags indicating their specialities or interests. These tags form an important part of user profiling for personalized user/item recommendation. In the absence of tags, user generated data or other information could be used to predict tags for users. Meanwhile, current social relations of users can also be used to recommend new users they would like to follow. Since user behavioral data is heterogeneous, it is still challenging to effectively leverage the heterogeneous information for user profiling and recommendation.

This shared task includes two subtasks. Subtask one is UTP which can be considered as a MLC problem. Given users' other information except tags, we need to predict related tags to the users. Subtask two is UFR, where we predict the users a user would like to follow in the future given users' following relationship and other provided information. We will introduce the two subtasks respectively in the following sections.

1.1 UTP

User Profiling can be defined as user information tagging, which is to assign user tags based on the users' past interests/behavior. Wu et al. [3] conducted unsupervised keywords extraction from Twitter messages to tag Twitter users' interests and concerns, and evaluated the performance by human annotators. Lai et al. [2] created a news recommender system to predict users' interests by analyzing their reading behaviors. Some works consider user profile inference as a supervised classification task. Li et al. [1] extracted user information from social media websites like Twitter, Google Plus or Facebook and make predictions for user attributes based on their tweets. Yin et al. [4] proposed a probabilistic model for personalized tag prediction, which integrates three factors: an egocentric effect, environmental effects and web page content.

In this subtask, we cast UTP as a MLC problem. For MLC problem, Binary Relevance (BR) [7] is a classical method. However it lacks of sufficient ability to discover dependencies among labels. To address this issue, various methods have been proposed. Classifier chains (CC) [8] extends BR by taking label dependencies into account. CC links binary classifiers as a chain and feeds the predictions of the earlier classifiers as features to the latter classifiers. Label Powerset (LP) combines multiple tags as new tags for classification. Condensed Filter Tree (CFT) [5] tries to find the best label sequences to make the best prediction. For multi label classification of images, CNN-RNN [6] extends CC by utilizing RNN to model label dependencies.

1.2 UFR

Friend recommendation is either based on topological structures of a social network, or derived from profile information of users.

Traditional methods use Friend-of-Friend (FOF) to obtain candidate friends set. This set of friends can be then sorted by several criteria including the popularity of friends and the number of shared friends between the target user and the candidate friends. This approach requires the existence of second degree linkage of users in data.

Semantic based methods recommend friends that are similar to the target user. As for similarity measurement, heterogeneous information could be used. Chin et al. [10] recommended friends to the target user using proximity and homophily. Xiao et al. [11] analysed the personality of the users by mining their tweets content and recommended friends with similar personalities. Wang et al. [12] proposed Friendbook system which collects user-centric knowledge from sensors on the smartphone and modeled life styles of users in order to suggest friends who share similar life styles with the target user. Gou et al.

[13] designed SFViz system that helps people seek friends with similar interests. Feng et al. [14] linearly combined multiple similarity measurements to find friends similar to the target user. This approach can provide with precise personalized recommendation based on carefully selected features. However a great effort is needed to collect appropriate information from users as well as their friends.

Classic recommendation approaches can also be applied to friend recommendation. In this case, a person may have the roles of user and friend at the same time. Collaborative filtering is amongst the most popular recommendation algorithms [15,16]. Memory-based collaborative filtering contains user-based and item-based algorithms, which make recommendations according to the similarity of user-item behaviors. Another category of collaborative filtering algorithm is matrix factorization, which learns a dense distributed representation for each user and item. Different variations of matrix factorization have been explored, such as SVD, LDA and ALS. The two previous algorithms are usually applied to explicit data such as movie ratings, and ALS could be more adapted to implicit datasets [17].

Recent works apply deep neural networks to friend recommendation. Liu et al. [18] combined deep learning techniques and collaborative information to explore the user representations latent behind the topology and content. Ding et al. [19] extracted deep features of users using CNN and performed recommendation using Bayesian Personalized Ranking.

2 Proposed Methods

2.1 UTP

The task of UTP is to predict tags which are related to a user. We cast UTP as a MLC problem. Compared to the multi-class classification, MLC is different: multi-class refers to classifying instances into one or more classes which does not take the label dependencies into account, while MLC explicitly models the dependencies among labels. Motivated by CNN-RNN [6], we utilize CNN to capture rich representations of users. We employ RNN to model dependencies among labels. Unlike CNN-RNN [6], which only considers the previous prediction of the maximum probability, we think and prove that the previous prediction of multiple labels will have an effect on later predictions. The illustration of the CNN-RNN framework is shown in Fig. 1.

CNN Module. We employ CNN to get the representation of each user profile. We use $c_{i,j}$ to represent the feature map element of i-th row and j-th column:

$$c_{i,j} = f(\sum_{d=0}^{D-1}\sum_{m=0}^{F-1}\sum_{n=0}^{F-1} w_{d,m,n}x_{d,i+m,j+n} + w_b) \tag{1}$$

where $D, F, w_{d,m,n}, x_{d,i+m,j+n}$ indicate the depth, filter size, the filter and pixel in the image. d, m, n represent d-th layer, m-th row and n-th column.

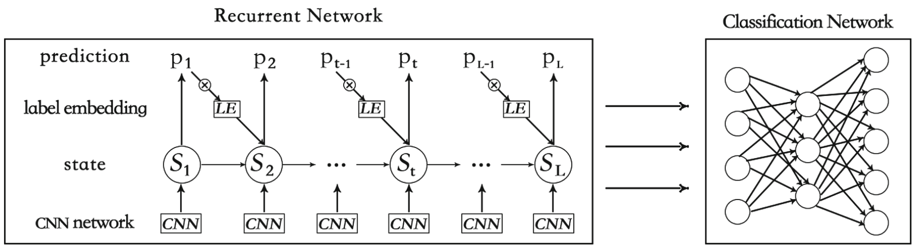


Fig. 1. The architecture of the proposed CNN-RNN model. Our proposed framework consists of three key modules: the output of CNN is employed as the user profile representation; the RNN captures the dependencies among labels, and the classification network combines the CNN output and the output of the recurrent layer as features to compute label probability.

Suppose n filters are used and the n resulting feature maps are $C^{(1)}, ..., C^{(n)}$. Then a pooling operation Pool$(\cdot)$ is applied to each of these n feature maps to produce n p-dimensional vectors Pool$(c^{(1)}), \cdots,$ Pool$(c^{(n)})$. The output of CNN is denoted by o_{CNN}:

$$o_{CNN} = \text{Flatten}(\text{Concat}(\text{Pool}(C^{(1)}), \cdots, \text{Pool}(C^{(n)}))) \tag{2}$$

RNN Module. The label prediction of time step t is represented as a vector p_t, the prediction label embedding e_t can be obtained by multiplying the predicted vector with a label embedding matrix U,

$$e_t = Up_t \tag{3}$$

We use the output of CNN and the previous label prediction information as combined features to feed to the recurrent layer:

$$x_t = \text{Concat}(o_{CNN}, e_{t-1}) \tag{4}$$
$$h_t = f(W_x x_{t-1} + W_h h_{t-1} + W_c c_t) \tag{5}$$
$$p_t = W_p h_t + b_p \tag{6}$$

where h_t, x_t, c_t indicate the hidden state of RNN, the RNN input and the cell state at time step t respectively. W_x, W_h, W_c, W_p, b_p are the parameters to be learned during the training process. f is an activation function.

Classification Network. We combine the output of CNN and the previous label prediction embedding as features to feed the classification network and calculate the predicted distribution over labels o_{out}:

$$o_{out} = W_o \text{Concat}(o_{RNN}, o_{CNN}) + b_o \tag{7}$$

where o_{RNN}, o_{CNN} indicate the last hidden state of RNN and the CNN output. W_o, b_o are parameters to be learned during training.

2.2 UFR

We assume that on social platform, users may follow friends because they know their friends in the real world, or they are interested in the tweets posted by their friends, or because the friends are celebrities or the friend accounts are popular.

Based on these hypothesis, the social and other types of information about all users and friends could be useful for recommending friends to users. However, neither sufficient tweets nor tags for friends are available in this task, we can not rely much on the features of friends. And thus, we decided to fully leverage social relations and user features.

Firstly we tried ALS, an efficient matrix factorization algorithm, which decomposes the sparse user-item matrix into low-dimensional user factors p and item factors q:

$$r'_{u,i} = p_u^T q_i \tag{8}$$

where $r'_{u,i}$ is 1 if user u follows item i, otherwise is 0.

Different from SVD which is optimized using Stochastic Gradient Descent (SGD), ALS optimizes the two types of factors alternatively by fixing one type while optimizing the other. Thus, ALS can be easily parallelized, and for implicit datasets where the user-item matrix is less sparse than explicit datasets, ALS is usually more efficient.

Apart from social following information, we also tried to incorporate other types of user information in the Collaborative Filtering algorithm, therefore we decided to implement user-based CF. For simplicity of implementation, given M users and N friends, we formulate user-based CF as following:

$$P = S \times C \tag{9}$$

where $C \in \mathbb{R}^{M \times N}$ is the user-friend matrix and each element $c_{i,j}$ is 1 if the user i follows friend j, otherwise is 0; $S \in \mathbb{R}^{M \times M}$ and each element $s_{u,v}$ represents the similarity between user u and user v; and $P \in \mathbb{R}^{M \times N}$ is the prediction result matrix and each element $p_{i,j}$ represents the score for user i and friend j. Using P, we filter out friend ids that are already friends of each user, and then sort the rest of the friends by prediction scores.

For this task, we calculated different types of similarity values between users, and the similarity matrix S is formulated as a weighted sum of different similarity matrices:

$$S = \sum \alpha_k S_k \tag{10}$$

$$S_k = \text{Norm}(G_k) \times \text{Norm}(G_k)^T \tag{11}$$

where α_k is a scalar weight, and $G_k \in \mathbb{R}^{M \times F}$ is the user-feature matrix with each element $g_{i,j}$ equals 1 if user i has feature j, otherwise equals 0. As for features, we used users' social relations, tags, check-in categories, tweets and profile information which includes gender, province and city. For social relation similarity, features are the set of friends. For tags, check-in and profile similarities, features are the total set of values of the corresponding information. Using tweets information, we compared two types of similarities: one uses the 10,000 most frequent words tokenized with LTP [21]; the other uses the 10,000 most frequent topics including hashtags surrounded by "#", user accounts initialized with "@", and emoji tags surrounded by "[" and "]". Our motivation behind topic extraction is that we assume that these topics represent users' emotions, their preferences on hot topics and popular accounts. We compared several combinations of these similarities and the results will be discussed in the next section.

3 Experiments

3.1 Data Analysis

In this task, there are several data sets given, including users social information, users tags, users check-in and tweets, as well as users profile information including their gender, city and province.

In users' social information, there are 56,217 users and 2,242,334 friends in total. 20% of the users (11,602) have only 1 friend and 82% of friends (1,860,094) are followed by only 1 user. The intersection of user and friend set is 12,674. All these statistics show that we have a quite sparse social network in question. As for user information, although all the users have profile and check-in information, only 11,714 users have tags, and only 9,343 users have tweets. Friends have even less related information. Lacking of useful user and friend information makes it hard to leverage similarities between users and their friends. Detailed statistics for users and different types of information are described in Table 1.

Table 1. Analysis of all the provided information (Coverage of users/friends is the number of users/friends having the corresponding types of information)

Information type	Number of values	Coverage of users	Coverage of friends	Total ids
Tweet	-	9,343	4,284	9,743
Tag	22,211	11,714	4,061	11,995
Check-in	260	56,217	12,936	60,000
Gender	2	56,217	12,923	59,320
Province	37	56,217	12,923	59,320
City	55	56,217	12,923	59,320

For submission of subtask one, we need to predict tags for 2,776 user ids. 2,720 of these ids have social relations, but only 769 of them have tweets. None of the ids has tag information in the provided data.

For subtask two, 33,857 user ids need to be recommended of potential friends based on their existent social relations. Among them, 6,091 user ids have no more than 5 friends in all the provided data, which would probably be involved in cold start problems.

3.2 Evaluation Metrics

The evaluation of both subtasks will use F1@K as defined below:

$$P_i@K = \frac{|H_i|}{K} \tag{12}$$

$$R_i@K = \frac{|H_i|}{|V_i|} \tag{13}$$

$$F1_i@K = \frac{P_i@K * R_i@K}{P_i@K + R_i@K} \tag{14}$$

$$F1@K = \frac{1}{N}\sum_{i=1}^{N} F1_i@K \tag{15}$$

where $|H_i|$ is the correctly predicted item set (item refers to tag in UTP and friend in UFR) for user i's top prediction, $|V_i|$ is the ground truth item set for user i. $P_i@K, R_i@K$ and $F1_i@K$ are the precision,recall and F1 for user i. In UTP, we set $K = 3$. In UFR, we set $K = 10$.

3.3 Experimental Results and Analysis

UTP. As aforementioned, for UTP submission task, only 769 users have tweets. We combined users' tweets, social links, check-ins and profile information as features for training and prediction. In this task, we chose as baselines four advanced approaches solving MLC problems: BR [7], CC [8], LP and Adaptation Algorithm (AA). The first three approaches use Decision Tree (DT), Naive Bayes (NB) and Random Forest (RF) methods, while AA is implemented with MLKNN. In addition, in order to verify the importance of label dependencies, we employ CNN [9] as a strong baseline. For tweets information, Deep learning(DL) approaches use 300-dimensional word2vec[1] vectors for training, while others use bag-of-words features.

We compared with 11 methods on the given dataset, and the results for $P@K$, $R@K$, $F1@K(K{=}3)$ are shown in Table 2. From Table 2, we can see that our proposed CNN-RNN model achieved the best performance on all metrics and our model outperformed the second best method by 39.59% in $P@K$.

It is obvious that DL approaches outperformed other traditional methods. We notice that the BOW model is not enough to represent the user profile since the BOW model does not consider the spatial correlation among features, while CNN uses convolution and pooling operation to capture richer information from different regions of the feature maps.

[1] https://code.google.com/p/word2vec.

Table 2. Results in $P@K$,$R@K$,$F1@K$, bold numbers indicate the best results each line.

Approaches	Methods	Metrics		
		P@K	R@K	F1@K
BR	DT	5.68%	3.31%	2.09%
	NB	8.07%	5.56%	3.29%
	RF	5.23%	3.29%	2.02%
CC	DT	5.41%	3.19	2.00%
	NB	9.13%	6.40%	3.76%
	RF	5.50%	3.41%	2.10%
LP	DT	5.79%	3.74%	2.27%
	NB	6.13%	4.31%	2.53%
	RF	5.43%	3.54%	2.14%
AA	MLKNN	5.39%	3.24%	2.02%
DL	CNN	19.06%	10.91%	6.94%
	CNN-RNN	**58.65%**	**31.97%**	**20.69%**

Regarding label dependencies, our model and LP outperformed BR. Compared with CNN, our method takes label dependencies into account and outperformed CNN by 13.75% in $F1@K$. We assume that users' tags in social networks are usually relevant. Users who are interested in a certain area are usually interested in related fields.

UFR. For our off-line experiments, we split social data to form off-line training and test set. For each user, we randomly choose 80% of his friends as training set and the rest as test set. In this way we assure that all the test users appear in training set. We then filter out friends set in the test set to make sure they appear in users or friends set of the training data. Since we already have the set of user ids to predict for the shared task, we only keep those user ids in our test data. It provides with the information of 56,217 users and 1,861,408 friends in training set, and 28,129 users and 167,679 friends in test set. In order to efficiently do the computation and ensure cover as much user ids in test set as possible, we filter out friends set and only keep friends with 2 or more followers. As a result of this filtering process, we have 47,957 users and 293,254 friends in training set and 24,688 users in test set. We can observe that nearly 20% of the user ids to be predicted have no more than 5 friends in all the provided data. And thus we split out our test set into two partitions: users with more than 10 friends in training data, and users with 10 or less than 10 friends. In total, the first test partition contains 10,438 users, and the second contains 14,250 users. At submission stage, we took all the provided data as training set to conduct predictions.

Firstly, we compare user-based methods (UB) using different information sources as similarity features. From Table 3, it is clear that using social relations as features is the best, and adding any other type of information will decrease the performance. The reason is possibly that it is straightforward to use users' past social behavior as features rather than other irrelevant ones when to predict users' future social relations.

Table 3. Results of user-based methods using different features on the two test set partitions. ("Coverage of users" means the number of users having recommendation results using each method; "$F1@K$ cov" refers to the F1 score within users covered by the corresponding method; "Added to UB-social" refers to the results of the user-based method on all the users, combining social and the corresponding information with equal weights as similarity features using formula(8))

methods	Test partition 1			Test partition 2		
	Coverage of users	$F1@K$ cov ($F1@K$ total)	Added to UB-social (F1@K)	Coverage of users	$F1@K$ cov ($F1@K$ total)	Added to UB-social (F1@K)
UB-social	10438	**2.59% (2.59%)**	-	14212	**1.35% (1.34%)**	-
UB-tag	1816	1.43% (0.25%)	−0.05%	3874	0.14% (0.03%)	−0.26%
UB-vocab	1573	1.41% (0.21%)	−0.07%	4048	0.09% (0.02%)	−0.34%
UB-topic	1572	1.71% (0.25%)	−0.04%	4039	0.14% (0.04%)	−0.30%
UB-gender	10438	2.19% (2.19%)	−0.40%	14250	0.48% (0.48%)	−0.86%
UB-province	10438	2.09% (2.09%)	−0.49%	14250	0.46% (0.46%)	−0.87%
UB-city	10436	2.00% (2.00%)	−0.58%	14249	0.45% (0.45%)	-0.89%

We also tried ALS and Most Popular Friends(MPF). MPF is an intuitive method simply recommending K most popular friends to each user after removing those who are already friends of the user. We set 20 dimensions for user and friend factors, and conduct 50 iterations while training ALS. Table 4 shows the performance comparisons of the three methods.

It can be seen from Table 4 that ALS achieves the best performance on the test partition 1, while UB-social performs the best on test partition 2.

4 Conclusion

In this paper, we compared our algorithms with other mainstream advanced methods in user profiling and recommendation, and our proposed methods

Table 4. Results using different methods on the two test set partitions

Methods	Test partition 1			Test partition 2		
	$P@K$	$R@K$	$F1@K$	$P@K$	$R@K$	$F1@K$
MPF	7.34%	3.01%	2.00%	0.75%	1.18%	0.44%
UB-social	9.39%	3.93%	2.59%	**1.98%**	**6.37%**	**1.34%**
ALS	**12.48%**	**5.13%**	**3.40%**	1.24%	1.99%	0.72%

achieved the best performance in both subtasks. For UTP subtask, we combined tweets, socials, check-ins and profiles as feature for training and prediction through data analysis. We proposed deep learning framework CNN-RNN, which employ CNN to get the user profile representation and model the dependencies among labels by RNN. Experiment results show that utilizing RNN mechanism to model label dependencies is effective for this MLC problem.

For subtask two, we combine collaborative filtering methods with MPF to conduct friend recommendations. Our submission results are based on ALS for most of the users, for the rest of the users who do not have ALS recommendation results due to lack of social relations, we simply propose K most popular friends after removing their existent friends. However we found that for users with no more than 10 friends in the training data, user-based CF is much more efficient than MPF and ALS. In the future, the ensemble of different recommendation methods is worth investigating deeply. Ranking methods like wide & deep [20] could also be tried to re-rank friend candidates set generated by collaborative filtering methods. Another possible way of studying this subtask is to consider friend recommendation as a MLC problem and leverage heterogeneous information in the process.

References

1. Li, J., Ritter, A., Hovy, E.: Weakly supervised user profile extraction from twitter. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, (Volume 1: Long Papers), vol. 1, pp. 165–174 (2014)
2. Lai, Y., Xu, X., Yang, Z., et al.: User interest prediction based on behaviors analysis. Int. J. Digit. Content Technol. Appl, 6(13) (2012)
3. Wu, W., Zhang, B., Ostendorf, M.: Automatic generation of personalized annotation tags for twitter users. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics 2010, pp. 689–692 (2010)
4. Yin, D., Xue, Z., Hong, L., et al.: A probabilistic model for personalized tag prediction. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 959–968 (2010)
5. Li, C.L., Lin, H.T.: Condensed filter tree for cost-sensitive multi-label classification. In: International Conference on Machine Learning, pp. 423–431 (2014)
6. Wang, J., Yang, Y., Mao, J., et al.: Cnn-rnn: a unified framework for multi-label image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2285–2294. IEEE (2016)

7. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. Data mining and knowledge discovery handbook, pp. 667–685. Springer, Boston (2009)
8. Read, J., Pfahringer, B., Holmes, G.: Classifier chains for multi-label classification. Mach. Learn. **85**(3), 333 (2011)
9. Kim, Y.: Convolutional neural networks for sentence classification (2014). arXiv preprint arXiv:1408.5882
10. Chin, A., Xu, B., Wang, H.: Who should I add as a "friend"?: a study of friend recommendations using proximity and homophily. In: International Workshop on Modeling Social Media, pp. 1–7 (2013)
11. Xiao, P., Fan, Y.Q., Du, Y.J.: A personality-aware followee recommendation model based on text semantics and sentiment analysis. In: Huang, X., Jiang, J., Zhao, D., Feng, Y., Hong, Y. (eds.) NLPCC 2017. LNCS (LNAI), vol. 10619, pp. 503–514. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73618-1_42
12. Wang, Z., Liao, J., Cao, Q.: Friendbook: a semantic-based friend recommendation system for social networks. IEEE Trans. Mob. Comput. **14**(3), 538–551 (2016)
13. Gou, L., You, F., Guo, J., et al.: SFViz:interest-based friends exploration and recommendation in social networks. In: International Symposium on Visual Information Communication, pp. 1–10. ACM (2011)
14. Feng, S., Zhang, L., Wang, D.: A Unified Microblog User Similarity Model for Online Friend Recommendation. Commun. Comput. Inf. Sci. **496**, 286–298 (2014)
15. Hannon, J., Bennett, M., Smyth, B.: Recommending Twitter users to follow using content and collaborative filtering approaches. ACM Conference on Recommender Systems, pp. 199–206. ACM (2010)
16. Chen, T., Tang, L., Liu, Q., et al.: Combining factorization model and additive forest for collaborative followee recommendation. In: KDD-Cup Workshop (2012)
17. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Eighth IEEE International Conference on Data Mining, pp. 263–272. IEEE (2009)
18. Liu, Y., Chen, X., Li, S., Wang, L.: A user adaptive model for followee recommendation on Twitter. In: Lin, C.-Y., Xue, N., Zhao, D., Huang, X., Feng, Y. (eds.) ICCPOL/NLPCC -2016. LNCS (LNAI), vol. 10102, pp. 425–436. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50496-4_35
19. Ding, D., Zhang, M., Li, S.Y., et al.: BayDNN: Friend Recommendation with Bayesian Personalized Ranking Deep Neural Network. In; Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1479–1488. ACM (2017)
20. Cheng, H.T., Koc, L., Harmsen, J., et al.: Wide & deep learning for recommender systems. In: The Workshop on Deep Learning for Recommender Systems. ACM, pp. 7–10 (2016)
21. Che, W., Li, Z., Liu, T.: LTP: A Chinese language technology platform. In: Proceedings of the Coling 2010, Demonstrations, Beijing, China, pp. 13–16, August 2010

Summary++: Summarizing Chinese News Articles with Attention

Juan Zhao[1], Tong Lee Chung[2], Bin Xu[2](✉), and Minghu Jiang[1]

[1] Lab of Computational Linguistics, School of Humanities, Tsinghua University, Beijing, China
solomagix@gmail.com, jiang.mh@tsinghua.edu.cn

[2] Computer Science Department, Tsinghua University, Beijing, China
tongleechung86@gmail.com, xubin@tsinghua.edu.cn

Abstract. We present Summary++, the model that competed in NLPCC2018's Summary task. In this paper, we describe in detail of the task, our model, the results and other aspects during our experiments. The task is News article summarization in Chinese, where one sentence is generated per article. We use a neural encoder decoder attention model with pointer generator network, and modify it to focus on words attented to rather than words predicted. Our model archive second place in the task with a score of 0.285. The highlights of our model is that it run at character level, no extra features (e.g. part of speech, dependency structure) were used and very little preprocessing were done.

Keywords: Text summarization · Sequence-to-sequence · Pointer Coverage

1 Introduction

Text summarization is the task of producing a short piece of text from a long one while preserving main information [1]. Summarization is one of the eight in NLPCC2018's evaluation task. As one of the more traditional task in NLP, it is still attracting a lot of attention and with advancements in *deep learning techniques*, there are more opportunities for improvement. As information grows rapidly, the time needed for a person to consume all these data is insufficient, summarization is helpful by providing a short text helping reader decide if they want to read the whole article. NLPCC's summarization task focuses on Chinese News articles, where research and support still lacks. The data is provided by Toutiao.com which consist of News articles. Train and evaluation datasets are provided to train the model and a test set is used to compare different models.

We participate in the task with our model Summary++, which is based of the Pointer Generator Network [2] with modifications to obtain our results. In the paper will focus on pre-processing of data, the model + modification and hyperparameter tuning. We do not use any feature extraction tools and human engineered features during pre-processing. We base our solution on character level so that we do not need to perform word segmentation. Our model is end-to-end to keep reduce human involvement during training and testing. Experiment results and other details will be followed in the paper, including improvement at each stage and training time. Models are compared with Character-based ROUGE-F metric [3].

The main contribution of this paper can be summarized as follows:

1. We present our Summary++ which obtained the second highest score of 0.285 in NLPCC2018's summary task.
2. We present a end-to-end solution for Chinese News article summarization.
3. Our model does not require word segmentation or any feature engineering.

1.1 The Summarization Task for Chinese News Article

The task description and data, provided by Toutiao.com, is a single sentence summarization task for News articles. The summary is usually a general description removing all details and comes in one sentence. Figure 1 shows one training point which contains one News article and its corresponding summary. The task can be regarded as an sequence to sequence problem with the article as input and summary as output. The standard model in deep learning for such task is encoder decoder model. Challenges when using such model include accuracy of word generated, out-of-vocabulary problem and repetition. Pointer generator network [4], which we base our solution on, is a outstanding model for such problems which incorporates both copying and generation mechanisms.

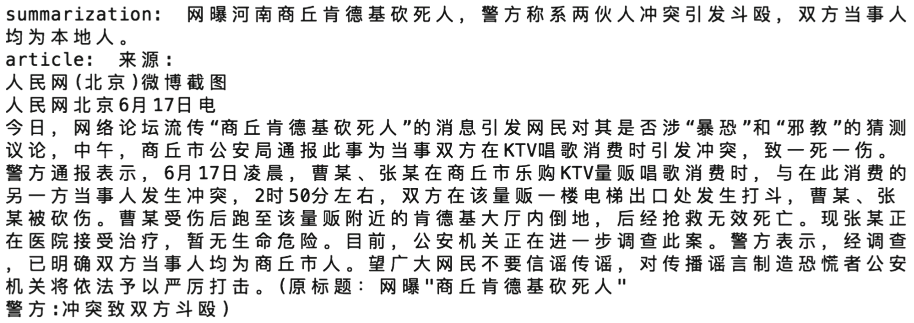

```
summarization: 网曝河南商丘肯德基砍死人，警方称系两伙人冲突引发斗殴，双方当事人
均为本地人。
article: 来源：
人民网(北京)微博截图
人民网北京6月17日电
今日，网络论坛流传“商丘肯德基砍死人”的消息引发网民对其是否涉“暴恐”和“邪教”的猜测
议论，中午，商丘市公安局通报此事为当事双方在KTV唱歌消费时引发冲突，致一死一伤。
警方通报表示，6月17日凌晨，曹某、张某在商丘市乐购KTV量贩唱歌消费时，与在此消费的
另一方当事人发生冲突，2时50分左右，双方在该量贩一楼电梯出口处发生打斗，曹某、张
某被砍伤。曹某受伤后跑至该量贩附近的肯德基大厅内倒地，后经抢救无效死亡。现张某正
在医院接受治疗，暂无生命危险。目前，公安机关正在进一步调查此案。警方表示，经调查
，已明确双方当事人均为商丘市人。望广大网民不要信谣传谣，对传播谣言制造恐慌者公安
机关将依法予以严厉打击。(原标题: 网曝"商丘肯德基砍死人"
警方:冲突致双方斗殴)
```

Fig. 1. Example of training data point. A article and its summary is provided. In the figure, the News article is about a murder incident that involves one man hacking another man to death at KFC. The article goes into deep detail of the incident, including time, place and aftermath. The summary however is very general and goes as follows, "Internet reports Shangqiu Henan hacking incident at KFC, police says two mans conflict escalated into fight, two man are locals."

When revising the data, we find some unusual datapoints, where the content consist of only html tags or punctuation. We use regular expression to remove tags and continuous punctuations for both training and testing set. Another dataset without summary was also provided but was discarded because such data was not needed in our model.

The total number of datapoints is 50000 for training, 2000 for evaluation and 2000 for testing. A only Chinese vocabulary list was provided with 2987 characters but we generated our own which includes digits, punctuation and English words. The total number of unique tokens was 213789 and ones with top 10000 frequency were used.

2 Summary++

In this section, we present our model in detail. We briefly describe the model we use and focus on the modifications we did to improve performance.

2.1 Our Model

Our model is based of Pointer Generator Summarization Network, which has a network that tell the decoder to use a generated word or a word that was attended to. In our solution, we try our model the same way, but when we perform summarization on the test set, we remove the generation process, because we find that the pointer network easily overfit the data and provided false facts in the test results. Figure 2 shows decoding process during training.

The model is encoder decoder model with attention mechanism, pointer network and coverage mechanism [5]. The encoder is a Bi-directional LSTM (BiLSTM) which takes in a sequence of characters and outputs two final states and one hidden states for each input character. BiLSTM is used in many state-of-the-art models and shows promising potentials.

During decoding, A LSTM cell takes in the previous hidden states and context vector as input, and outputs a current hidden state (also known as the decoder state). Attention is a mechanism which shows the decoder which part of the sequence to focus on while decoding. We can think of it as giving a weight to each encoder hidden states. The attention weight is calculated as follows: 1. for each token in the sequence, concatenate its corresponding encoder hidden state with decoder hidden state to form a vector; 2. run the vector through a linear transformation followed by a *tanh*; 3. run the output from the previous step through another linear transformation followed by a $softmax$. We are able to get a probability distribution for each word after the three steps.

The context vector is a state after considering all encoder hidden states and attention. We can see it as a vector that is used to predict the next word. It is calculated by multiplying the encoder hidden state of each token with it attention weight, then adding together all these weighted state. The context vector is project through a linear transformation to the size of our vocabulary.

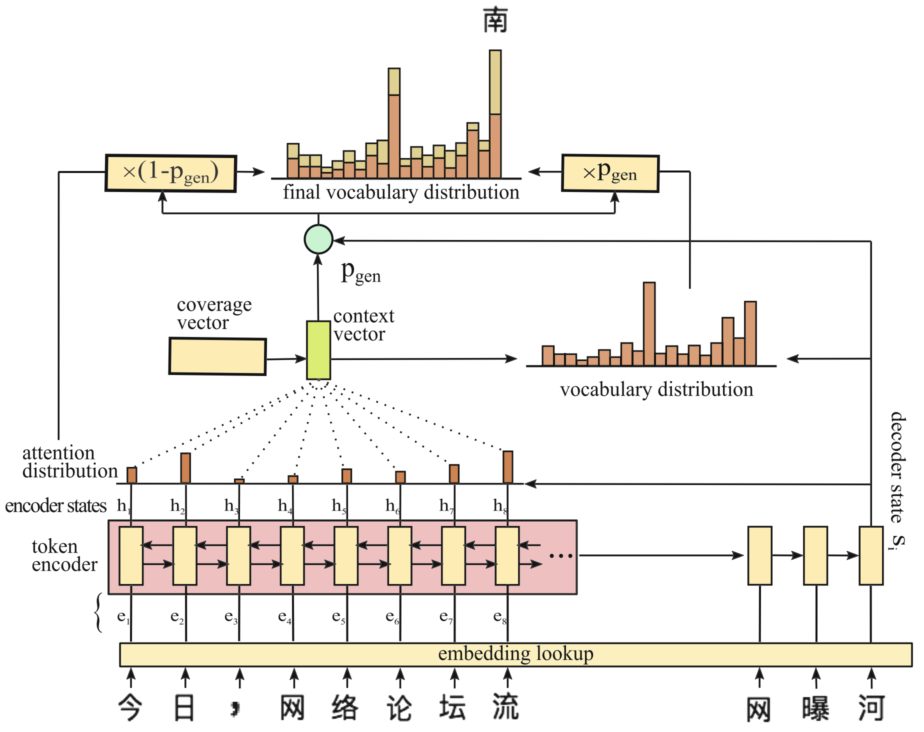


Fig. 2. Pointer Generator Network Model + Coverage that is used in our Summary++ system. During each decoding step, attention, coverage vector and generation probability is calculated to predict the next word.

Finally, a *softmax* is used to obtain the distribution of the next token. The following equations show calculation of attention and context vector:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{attn}) \tag{1}$$

$$a^t = \text{softmax}(e^t) \tag{2}$$

$$h_t^* = \sum_i a_i^t h_i \tag{3}$$

where h_i is the hidden encoder state and s_t is the hidden decoder state. v^T, W_h, W_s and b_{attn} are all trainable parameters. h_t^* is the context vector.

The coverage vector is a summation of attention during the decoding process. For each token in the input sequence, its initial coverage weight is 0, that means the token is not attended to. As the output sequence get decoded, coverage increases for each token as it gets assigned attention weight. As a token get more attention, it get more difficult to have higher attention scores. Such mechanism is used for preventing repetition in the output sequence. The coverage vector is included in the calculation of attention. Coverage is the new attention is calculated as following:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{4}$$

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + W_c c_i^t + b_{attn}) \tag{5}$$

c^t is initialized to 0. The attention and context calculation is the same as Eqs. 2 and 3.

The pointer network outputs a scalar the tells the model to look at the predicted token to look at a token that was attened to most. It can be seen as a soft switch between two choices. The network takes in the context vector and the decoder states and outputs a scalar for each token in the vocabulary. The final vocabulary distribution is the weighted sum of predicted distribution and attention of the input token. The switching scalar also known as generation probability is calculated as:

$$p_{gen} = \sigma(w_{h^*} h^* + w_s s_t + w_x x_t + w_d d + b_{ptr}) \tag{6}$$

where w_{h^*}, w_s, w_x, and w_d are trainable parameters. p_{gen} is our generation probability. Predicted distribution and final distribution is calculated as:

$$P_{vocab} = \text{softmax}_{vocab}(V'(V[s_t, h_t^*] + b) + b') \tag{7}$$

$$P(w) = p_{gen} P_{vocab}(w) + (1 - p_{gen}) \sum_{i:w_i = w} a_i^t \tag{8}$$

V, V' and b' are trainable parameters.

The loss we use is negative log likelihood of each token in the output sequence. We hope that by training, the probability of a suitable next token will have the highest. Coverage does not converge unless a coverage loss is included. This is the minimum between attention and coverage value, which slows down the gradient descent process. Loss is show as below:

$$loss_t = -\log P(w_t^*) \tag{9}$$

$$loss = \frac{1}{T} \sum_{t=0}^{T} loss_t \tag{10}$$

and loss with coverage:

$$loss_t = -\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t) \tag{11}$$

2.2 Character Embedding

For Chinese text we choose to operate at character level, thus leaving out the need for segmentation. Each character is represented by a vector which is fed into the model. We find that segmentation introduces a large number of words (token), such token is difficult to train due to the fact that data is not sufficient. A lot of words appear on a few times even though they are everyday words. Correlation of words become too sparse. Another reason is that we hope that our model is a end to end model and not a pipeline type of model. Error in pipeline can propagate along and a better solution would be minimize all errors within one model. We also do not introduce any human engineered feature of any kind (POS, DEP, etc). We believe that such feature are not a hundred percent correct, thus causing the error to be enlarged in the model.

In the dataset, we do find some pieces of data that requires pre-processing. Due to the fact that our model takes in the first 400 characters as input for an article, some articles contains only tags and punctuation in the beginning, thus causing the model to incorrectly identify crucial tokens in the model. We use a regular expression to remove some text that does not look like news content.

2.3 Attention only Generation

When evaluating our model, we find that some summarizes contain some incorrect information. For example, we find that given a article about CBA players as

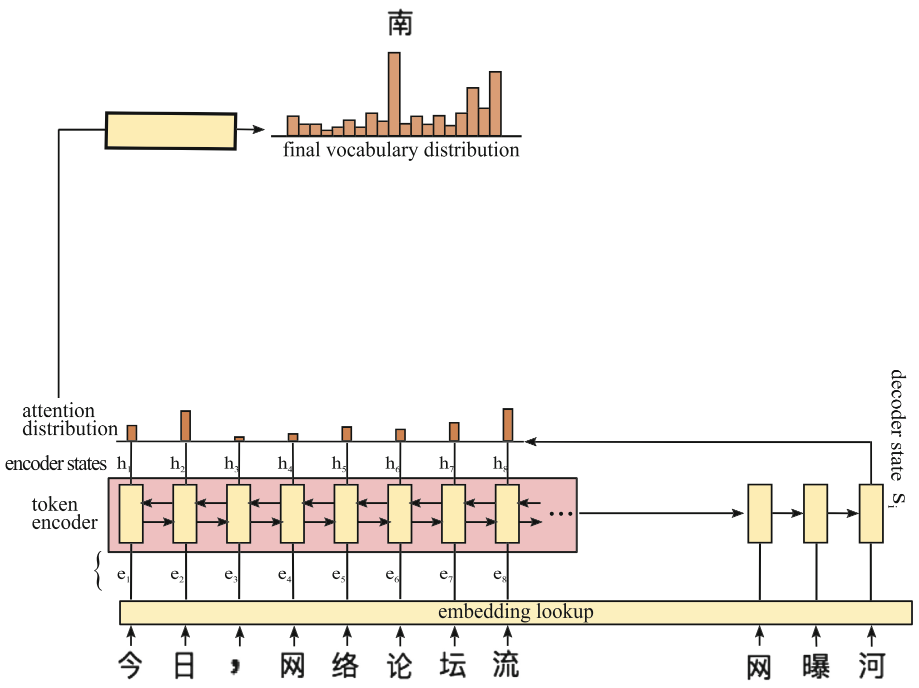

南
final vocabulary distribution
attention distribution
encoder states
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
token encoder
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8
decoder state S_i
embedding lookup
今 日 ， 网 络 论 坛 流
网 曝 河

Fig. 3. Modified model, the whole vocabulary prediction part has been removed, the whole generation process relies only on attention weights. The parameters are trained using the full model but only these parts are used for summary generation.

input, the output summary begins with *NBA*. After carefully analyzing the matter, we find that the problem is that predicted value is over confident. We find that by removing the predicted vocabulary distribution, we can achieve optimal results. Figure 3 is the model when in testing phrase. We directly use the word that has the highest attention weight. This is similar to using neural network as a copying mechanism, at each time step, we copy one word from the original article. Different from directly copying, attention distribution was trained on the full model hoping that correct words can be predicted.

3 Experiments

In this section, we describe in detail our experiment. We first discuss hyperparameter settings and results.

3.1 Hyperparameters and Training

Setting hyperparameters is one of the bigger challenges in our experiment. We report ROUGE score on the evaluation set to compare different settings and choose accordingly. We also compare the difference we using the original model and our simplified model.

On Training Steps. We find that the model easily overfits, which means the more training steps the more likely the model will overfit. We report F score on ROUGE-1, ROUGE-2 and ROUGE-L after some number of training to find the optimal step size. Table 2 shows the relation between performance and training step size (Table 1).

Table 1. ROUGE result on training step, we find that after about 100,000 training step, performance seems to stagnate.

num training steps	ROUGE-1	ROUGE-2	ROUGE-L
11353	0.43	0.27	0.37
25791	0.44	0.28	0.38
109937	0.46	0.29	0.40
233769	0.46	0.30	0.40
392774	0.46	0.30	0.40
570743	0.46	0.29	0.40

ROUGE result on evaluation set does not tell us everything about the model. High training step trigger repetition even when coverage is added. In order to have a high ROUGE score and more readable summary, we choose the training step to be around 110,000.

3.2 On Encoder Decoder Max Size Min Size

We find another important factor in competing was setting encoder input size. Due to hard-ware limitations, we were only able to increase encoder input size to 400, that is only the first 400 tokens were taken. We find that smaller size decreases the performance of the model. While sizes too large result in out-of-memory error.

We also set the minimum decoder size to 15. After some observation, we estimate the minimum length of summary is about 15 characters. We also try other sizes such as 20 and 35, we find that performance is slightly worse than 15, so we choose 15 as minimum decoder size. We set the maximum decoder size to 35. Increasing this size does not have any improvement as well.

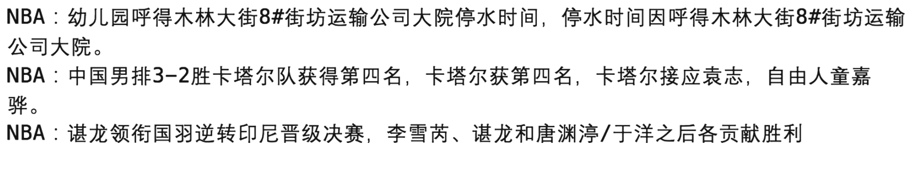
3.3 On Attention only Summary Generation

We finally test the results of using pointer generator probability to switch between generating the next word and using the attended character. We test the vanilla approach and also the two alternate, where one uses only the attended character and the other uses only the generated word. We operate our experiment on the final test data and report the official ROUGE score

Table 2. ROUGE score when using different generation methods, Vanilla uses both generated character and attention with a soft switch. Attention and Generation uses either only the attended sum or predicted vocabulary.

num training steps	ROUGE score
Vanilla	0.243
Generation only	0.270
Attention only	0.285

We also evaluate some example and find that even with the switch method, we find some obvious mistake in the summary with generated summaries. Figure 4 shows some examples when using a switch between generation and attending. When using attention guided summary, such problem do not appear.


NBA：幼儿园呼得木林大街8#街坊运输公司大院停水时间，停水时间因呼得木林大街8#街坊运输公司大院。

NBA：中国男排3-2胜卡塔尔队获得第四名，卡塔尔获第四名，卡塔尔接应袁志，自由人童嘉骅。

NBA：谌龙领衔国羽逆转印尼晋级决赛，李雪芮、谌龙和唐渊渟/于洋之后各贡献胜利

Fig. 4. Examples of incorrect summary, the model is over confident starting with "NBA" is a correct way to begin a summary.

3.4 Final Results Compared in the Evaluation Task

We finally compare results with other team in the evaluation task. Our team made second place on the score board. All methods are not disclose at the time of writing, so this is reference to where our model stand in the task. Table 3 shows the top five on the leader board. Difference is rather small.

Table 3. Evaluation result on the final score board. Our mode Summary++ made second place.

num training steps	ROUGE score
WILWAL	0.2938
Summary++	0.285
CCNU_NLP	0.282
freefolk	0.281
kakami	0.278

4 Related Works

Summarization has been around for many decades [1,6], this paper focuses on multi-sentence summarization where the task is to generate summaries with multiple sentences. The task has attracted attention when a large multi-sentence summary corpus was introduced [7].

Neural Extractive Summarization. The extractive approach is based on a hypothesis that the main idea of a document can be summarized in a few phrases or words in the document. Then the task of the summarization turns to find the most important words in the document. The neural extractive approaches [8–10] are mainly based on the CNN model and some of its deformations. The greatest problem of this kind of approach is that the generated summarization may be incoherent and inconsistent.

Neural Abstractive Summarization. The abstractive approach needs to understand the meaning of the document, and then briefly summarize it by a highly readable human language. For this target, the RNN/LSTM models and some of their deformations are adopted to complete the neural abstractive task [11,12]. Recently, some researchers have used the latest neural networks model to summarize, such as the sequence-to-sequence model and attention model.

Sequence-to-Sequence Models. Most of current state-of-the-art models are based of sequence-to-sequence models which have gained many successes in machine translation [13,14]. Attention [15], pointer network [4], coverage [5] and controllable summarization [16,17] are some techniques adapted to the task of summarization.

5 Conclusion

This paper present our model in the NLPCC2018 summarization task. We modify the pointer generator network model to achieve SOTA results on the test set. We show that the model can sometimes be over confident with its prediction and we simplify the model to only using attended tokens. We also show that character level summary in Chinese language is not only possible but also practical. Our model requires very little pre-processing and no human-engineered feature. We believe that there is more potentials in the model.

Acknowledgement. This work was supported by the National Key Research and Development Program of China 2017YFB1401903, National Social Science Major Fund (14ZDB154; 15ZDB017) of China, and the National Natural Science Key Foundation of China (61433015).

References

1. Nenkova, A., McKeown, K.: Automatic summarization. Found. Trends Inf. Retr. **5**(2–3), 103–233 (2011)
2. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (2017)
3. Lin, C.-Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop
4. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28, pp. 2692–2700. Curran Associates Inc. (2015)
5. Mi, H., Sankaran, B., Wang, Z., Ittycheriah, A.: Coverage embedding models for neural machine translation. In: EMNLP (2016)
6. Saggion, H., Poibeau, T.: Automatic text summarization: past, present and future. In: Poibeau, T., Saggion, H., Piskorski, J., Yangarber, R. (eds.) Multi-source. Theory and Applications of Natural Language Processing, pp. 3–21. Multilingual Information Extraction and Summarization. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-28569-1_1
7. Nallapati, R., Zhou, B., dos Santos, C.N., Gülçehre, Ç., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, 11–12 August 2016 (2016)
8. Cheng, J., Lapata, M.: Neural summarization by extracting sentences and words. In: Meeting of the Association for Computational Linguistics, pp. 484–494 (2016)
9. Nallapati, R., Zhou, B., Ma, M.: Classify or select: Neural architectures for extractive document summarization. CoRR, abs/1611.04244 (2016)
10. Cao, Z., Li, W., Li, S., Wei, F., Li, Y.: Attsum: joint learning of focusing and summarization with neural attention. In: International Conference on Computational Linguistics, pp. 547–556 (2016)
11. Lopyrev, K.: Generating news headlines with recurrent neural networks. arXiv:1512.01712 Computation and Language (2015)
12. Ayana, S.S., Zhao, Y., Liu, Z., Sun, M.: Neural headline generation with sentence-wise optimization. arXiv:1604.01904v2 Computation and Language (2016)

13. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations (ICLR) (2015)
14. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, vol. 2, NIPS 2014 (2014)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, Q.V., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 6000–6010. Curran Associates Inc. (2017)
16. Fan, A., Grangier, D., Auli, M.: Controllable abstractive summarization. CoRR, abs/1711.05217 (2017)
17. Kikuchi, Y., Neubig, G., Sasano, R., Takamura, Y., Okumura, M.: Controlling output length in neural encoder-decoders. CoRR, abs/1609.09552 (2016)

NLP Fundamentals

Paraphrase Identification Based on Weighted URAE, Unit Similarity and Context Correlation Feature

Jie Zhou[1], Gongshen Liu[1](✉), and Huanrong Sun[2]

[1] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
{sanny02, lgshen}@sjtu.edu.cn

[2] SJTU-Shanghai Songheng Information Content Analysis Joint Lab., Shanghai, China
sunhuanrong@021.com

Abstract. A deep learning model adaptive to both sentence-level and article-level paraphrase identification is proposed in this paper. It consists of pairwise unit similarity feature and semantic context correlation feature. In this model, sentences are represented by word and phrase embedding while articles are represented by sentence embedding. Those phrase and sentence embedding are learned from parse trees through Weighted Unfolding Recursive Autoencoders (WURAE), an unsupervised learning algorithm. Then, unit similarity matrix is calculated by matching the pairwise lists of embedding. It is used to extract the pairwise unit similarity feature through CNN and k-max pooling layers. In addition, semantic context correlation feature is taken into account, which is captured by the combination of CNN and LSTM. CNN layers learn collocation information between adjacent units while LSTM extracts the long-term dependency feature of the text based on the output of CNN. This model is experimented on a famous English sentence paraphrase corpus, MSRPC, and a Chinese article paraphrase corpus. The results show that the deep semantic feature of text could be extracted based on WURAE, unit similarity and context correlation feature. We release our code of WURAE, deep learning model for paraphrase identification and pre-trained phrase end sentence embedding data for use by the community.

Keywords: Paraphrase identification · Recursive Autoencoders
Phrase embedding · Sentence embedding · Deep learning · Semantic feature

1 Introduction

In general, paraphrase means expressing the same meaning in different words. With the development of NLP and paraphrase generation, there is a phenomenon that AI machine writers paraphrase similar news or stories on different websites and social medias. Paraphrase identification is useful in news event detection and first story detection. It is also helpful to other NLP applications, including question answering, information retrieval, plagiarism detection, machine translation evaluation and so on.

Paraphrase identification is a subtask in natural language processing (NLP), which aims at recognizing if the given pair of text convey same meaning. That pair of text might have different length and be expressed in different way. If a pairwise text have equivalent semantic, it would be labelled as paraphrase. In another way, a pairwise text is non-paraphrase if they have different meaning.

In this paper, a deep learning model is proposed for paraphrase identification, based on pairwise unit similarity feature and semantic context correlation feature. The pairwise unit similarity feature is extracted from given pairs of text through a convolutional neural model. Moreover, the work of [1, 9, 10] are extended to get the semantic context correlation feature based on CNN and LSTM. Also, for the purpose of learning phrase and sentence embedding, the work of [18] is extended to Weighted Unfolding Recursive Autoencoders (WURAE).

The model is adaptive to both sentence-level and article-level paraphrase identification (PI) task. The sentence-level PI task is experimented in an English sentence corpus, Microsoft Research Paraphrase Corpus (MSRPC), and compared with the state-of-art models. In our work, an extension to existing problem is made by introducing article-level paraphrase detection, detecting whether the given pair of articles talk about the same matter. The article-level PI task is experimented in a Chinese article paraphrase dataset, which is generated from sports and entertainment news.

In the rest of paper, we first review related works in Sect. 2. In Sect. 3, our methodology is introduced in detail. Experimental setup and results are discussed in Sect. 4. Finally, conclusion and future work plans are exposed in Sect. 5.

2 Related Work

The coverage of existing literature is about paraphrase identification (PI) and sentence embedding. The part of PI is divided into lexical similarity, semantic feature, syntactic feature and traditional features. The issue of sentence embedding is mainly about unsupervised learning method and certain-task-supervised learning method.

2.1 Paraphrase Identification (PI)

To compare the meaning of given pairwise text, a traditional method is based on their lexical similarity. The basic method includes Longest Common Subsequence (LCS) [2], similarity of name entity, calculating the cosine distance of word embedding, obtaining statistics feature by Vector Space Model (VSM), n-gram overlap and so on. [16] used corpus-based and knowledge-based measures of similarity with WordNet. A set of words in different order may differ in meaning. Thus, meaning of phrases should be taken into account. Considering continuous and discontinuous linguistic phrases, [8] extended TF-IDF by discriminative weights of words and phrases. With the development of neural networks and word embedding, deep learning algorithm is widely used in NLP. [6] proposed a pairwise semantic and lexical similarity measurement based on CNN. [9] figured out a method of using wide one-dimension convolution to get n-gram feature, which [1] have used in paraphrase detection. [1] also combined it with LSTM to get semantic representation of sentences. [10] used multiple filter widths, getting various n-gram feature maps, in sentence classification task.

Syntactic feature is helpful for deep semantic comprehension. Structured alignment in syntactic feature based on dependency trees is explained in [15]. [18] used dynamic pooling layer to construct a fixed-sized similarity matrix from phrase embedding.

Some other features can be added to improve the accuracy of identification. Number feature was applied in [18]. The use of machine translation (MT) evaluation in paraphrase identification was explained by [14] which made use of 8 different MT metrics.

2.2 Sentence Embedding

The distributed representation of nature language makes the computer process natural language more convenient. Recently, many studies have proposed various methods for distributed representation of phrase, sentence or even paragraph. [9, 10] explained a way of modelling a sentence by CNN while they are both concerned on one certain topic, training sentence embedding with the labelled data. [13] advised a self-attention mechanism and a special regularization term. [12] proposed Paragraph Vector, an unsupervised learning algorithm, which learns fixed-length representation from variable-length pieces of sentences, paragraph or documents. [18] proposed Unfolding Recursive Autoencoders, an unsupervised learning method to calculate phrase or sentence embedding based on parse tree. [11] used continuity of text from books, training an encoder-decoder model that tries to reconstruct surrounding sentences of an encoded passage.

3 Methodology

The inputs of our deep learning models are the distributed representation of words. Then, the phrases and sentences embedding are learned from WURAE, an unsupervised learning algorithm trained by a large scale of both English and Chinese sentence corpus. In the sentence-level PI task, word and phrase embedding are regarded as the units of sentence like the nodes in parse tree. By analogy, sentence embedding is considered as the units of article in the article-level PI task. With the distributed representation of text, pairwise unit similarity feature is extracted from the unit similarity matrix through CNN and k-max pooling layers. In addition, semantic context correlation feature is learned from the combination of CNN and LSTM. Some other features are also added to the model. The probability of being paraphrased is predicted by the combination of features. The overall architecture of sentence-level paraphrase identification would be described in Sect. 3.5, including lexical, syntactic and semantic feature. And the entire architecture of article-level one would be explained in Sect. 3.6.

3.1 Distributed Representation of Words

Distributed representation of data is a must for applying deep learning method into NLP. Word embedding can convert one word in natural language into a node of vector space, which helps computer process NLP tasks more convenient. With the implementation of word embedding, a sentence could be represented with a list of fixed-dimensional vectors. If a sentence is composed of n words and the dimension of word embedding is m, the sentence could be expressed as $(w_1, w_2, \cdots, w_i, \cdots, w_n)$ where w_i equals $(x_1, x_2, \cdots, x_i, \cdots, x_m)$. In the work of learning phrase or sentence embedding, word embedding is the data of every leaf node in parse tree. A pre-trained word embedding with the dimension of 300, Google News vectors[1], is used in the experiment of sentence-level PI task. Since the article-level paraphrase dataset is constructed from Chinese sports and entertainment news, we trained 300-dimensional vectors through Word2Vec algorithm based on the corpus of Chinese Wiki data[2] and Sogou News data[3].

3.2 Distributed Representation of Phrase and Sentence

Owing to the diversity and complexity of natural language, although word embedding could represent sentences as lists of vectors, it is still difficult to get the accurate semantic feature. The phrases composed of ordered words are more important than separate words while understanding meaning of sentences. For the purpose of extracting deep semantic feature, there is a need to train on phrase or sentence embedding, capturing syntactic and semantic feature besides lexical one. In this research, the work of [18] is extended to Weighted Recursive Autoencoders (WURAE). Here we will introduce their previous work briefly and then propose our improvement on it.

Unfolding Recursive Autoencoders (URAE). Based on parse tree of sentence, we can obtain a binary tree structure representing the sentence. The leaf nodes of the tree are word embedding of words in the sentence. The internal nodes representing phrases and root node representing sentence are computed from their children, which is called as encoding part. The child node could be a leaf node or an internal node. For the given n-length sentence S, represent it with a list of m-dimensional vectors as $S = (w_1, w_2, \cdots, w_i, \cdots, w_n)$ where $w_i = (x_1, x_2, \cdots, x_i, \cdots, x_m)$. In the encoding part, the parent node p is calculated from its children (c_1, c_2) by a standard neural network layer:

$$p = f(W_e[c_1; c_2] + b_e) \tag{1}$$

where $[c_1; c_2]$ means the concatenation of its children, f is an element-wise activation function such as *tanh*, $b_e \in R^m$ is the encoding bias vector and $W_e \in R^{m \times 2m}$ is the encoding matrix to learn.

[1] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit?usp=sharing.

[2] https://dumps.wikimedia.org/zhwiki/latest/zhwiki-latest-pages-articles.xml.bz2.

[3] http://www.sogou.com/labs/resource.

To optimize the training and improve the representation of phrase or sentence, the reconstruction is calculated during the decoding part. The decoding calculation of one parent node p reconstructs its children as $\left(c_1^{'}, c_2^{'}\right)$:

$$\left[c_1^{'}; c_2^{'}\right] = f(W_d p + b_d) \tag{2}$$

where f is an element-wise activation function, $W_d \in R^{2m \times m}$ is the decoding matrix and $b_d \in R^{2m}$ is the decoding bias vector. In the URAE, decoding part of node p_i reconstructs the entire subtree underneath p_i. With all the reconstructed leaf nodes underneath p_i, we could get the reconstruction error by computing Euclidean distance between the concatenation of original inputs and its reconstructions:

$$E_{rec}\left(y_{(i,j)}\right) = \left\| \left[w_i; \cdots; w_j\right] - \left[w_i^{'}; \cdots; w_j^{'}\right] \right\|^2 \tag{3}$$

where node $y_{(i,j)}$ is encoded from leaf nodes $\left(w_i, \cdots, w_j\right)$.

Weighted Unfolding Recursive Autoencoders (WURAE). As is mentioned above, URAE could calculate the distributed representation of phrases and sentence. Its method of reconstruction error ensures the increased importance of the child which has larger subtree. However, the method also causes that the more a word occurs in the corpus, the more times it would be reconstructed, the more contributions it would make to the reconstruction error. Because URAE optimizes weights by minimizing the reconstruction error, the more time a word occurs, the more effect it would have on the model weights, and it would be reconstructed better. It would happen that stopwords like ‘the’, ‘a’ and etc. have the same effect or even more effect than the others while representing phrases. But different words affect semantic meaning of phrases in different degrees. So, we propose that reconstruction error of every leaf nodes should be weighted by the reciprocal of its frequency:

$$E_{rec}\left(y_{(i,j)}\right) = \sum\nolimits_{k=i}^{j} \frac{1}{count(w_k)} \cdot \left\|w_k - w_k^{'}\right\|^2 \tag{4}$$

where $count(w_k)$ means the count of the word in the corpus.

WURAE Training. A large set of sentences is used to train this unsupervised learning algorithm. The model minimizes the sum of all node’s reconstruction errors in a mini-batch. It uses backward propagation through structure [5] to compute the gradient and optimizes with L-BFGS in the mini-batch training.

After learning phrase and sentence embedding, we can get the sentence represented as $(w_1, w_2, \cdots, w_n, w_{n+1}, \cdots, w_{2n-1})$, where $(w_1, \cdots, w_n)$ are word vectors, $(w_{n+1}, \cdots, w_{2n-2})$ are phrase vectors and w_{2n-1} is the sentence vector.

3.3 Pairwise Unit Similarity from CNN

In order to find out whether the given pair of text convey same meaning, we take the similarity of basic units into account. Words and phrases are regarded as the units of sentence. By analogy of sentence and article, sentences are considered as the basic units of article. For the given pair of l_1-length text T_1 and l_2-length text T_2, T_1 and T_2 are represented by the unit embedding lists, like S mentioned above. To extract basic unit similarity feature, we firstly compute similarity matrix via the unit embedding lists of the pairwise text. The similar or same pair of units, matched from T_1 and T_2, might appear in different positions of the two lists. Thus, we need to compare every unit vector in one text with all the unit vectors in another one. A similarity matrix with the size of $l_1 \times l_2$ is constructed by calculating cosine distance between the matched-pairs of units.

Convolution neural network is used to learn the patterns of pairwise semantic resemblance. The architecture of pairwise unit similarity measurement is as described in Fig. 1. The model consists of 3 convolution layers and the former two convolution layers are both followed by a max-pooling layer. The output of the third convolution layer is fed into a k-max-pooling layer, which extracts the top k most important features and gets the result of a flattened feature *F_unit_similarity*. Due to the dissymmetry of two text, the pairwise unit resemblance is calculated through two directions of the input similarity matrix. Then the pairwise unit similarity feature could be obtained by concatenating the two features, $F\_unit\_similarity_1$ and $F\_unit\_similarity_2$.

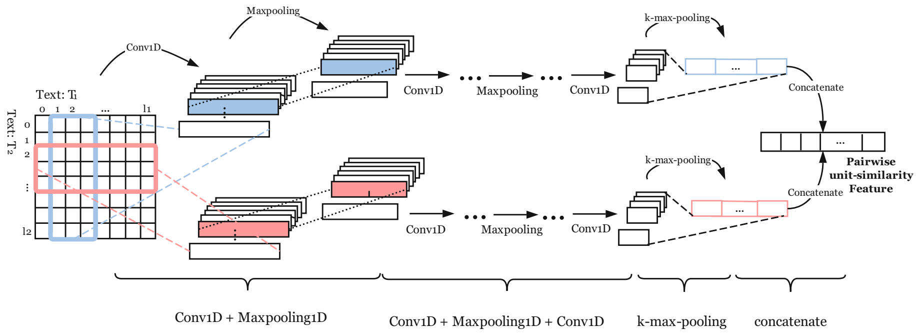


Fig. 1. The architecture of pairwise unit similarity measurement

3.4 Semantic Context Correlation from CNN and LSTM

Different arrangements of words and phrases express various semantics in the sentences. Also, various sequences of sentences make the meaning of articles different. So, besides pairwise unit similarity feature, we also regard semantic correlation among the units as an important feature. In this part, a combination of CNN and LSTM is used to get semantic context correlation feature as depicted in the Fig. 2. The input of this model is a list of basic unit embedding which represents the text.

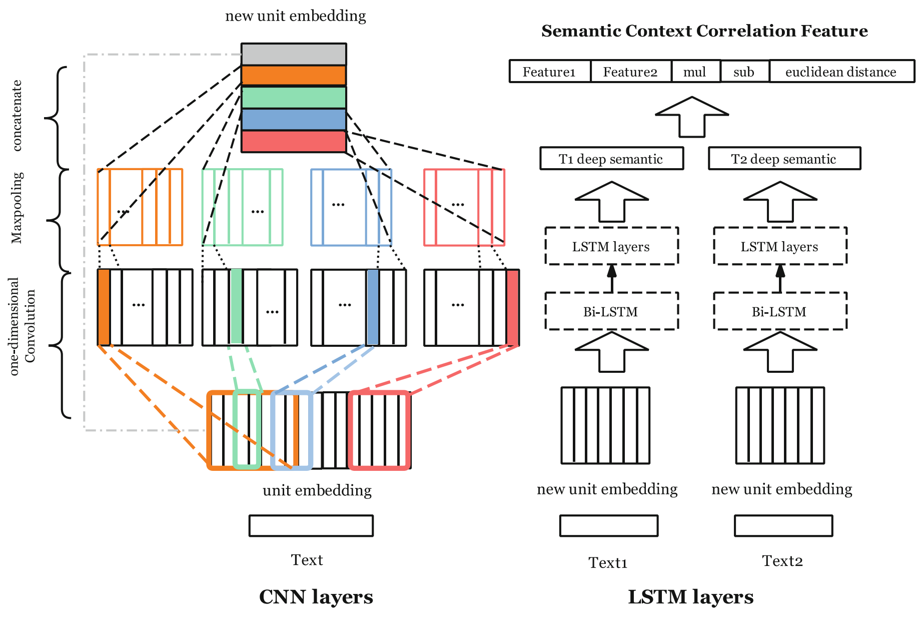


Fig. 2. The architecture of semantic context correlation similarity measurement

Firstly, CNN is utilized to get the collocation information between adjacent units in context, like n-gram feature in sentences. The model uses 4 one-dimensional convolutional layers with window sizes of 2, 3, 5 and 7 to get features from embedding list, resembling 2-gram, 3-gram, 5-gram and 7-gram feature. The input of embedding list is fed into the four different convolutional layers respectively. New unit embedding is constructed by concatenating these 4 results and the original unit embedding list. For the given input of m-dimensional embedding, the new unit embedding would have the dimension of 5 × m.

The new unit embedding is fed into LSTM so as to learn long-term dependencies from sequential units of text. Here a bidirectional LSTM performs both forward pass and backward pass on the new unit embedding matrix. Then, other two LSTM layers learn more from its output. The last hidden state is taken as deep semantic feature of the input text.

For the given pair of texts, each one is fed into the model separately to get its own deep semantic feature $F\_semantic$. And then the pairwise deep semantic features $(F\_semantic_1, F\_semantic_2)$ generate the semantic context correlation feature:

$$F_{sub\_sem} = F\_semantic_1 - F\_semantic_2 \tag{5}$$

$$F_{mul\_sem} = F\_semantic_1 \cdot *F\_semantic_2 \tag{6}$$

$$F_{euclidean\_sem} = F_{sub\_sem} \cdot *F_{sub\_sem} \tag{7}$$

where semantic context correlation feature equals to the concatenate of those features, $[F\_semantic_1; F\_semantic_2; F_{mul\_sem}; F_{sub\_sem}; F_{euclidean\_sem}]$.

3.5 Paraphrase Identification on Pairwise Sentence

MSRPC, an English paraphrase sentences corpus, is used in the sentence-level PI task. The entire architecture is shown in Fig. 3. Firstly, WURAE is trained in a large scale of English news sentences and then it calculates the phrase embedding of sentence. The sentence is represented by pre-trained word embedding and phrase embedding. For the purpose of classification, we extract its pairwise word & phrase similarity feature and semantic context correlation feature from models proposed in Sects. 3.3 and 3.4. Moreover, other features are added to the model, including number feature, BLEU score, ratio of Longest Common Subsequence (LCS), ratio of edit distance and similarity based on TF-IDF.

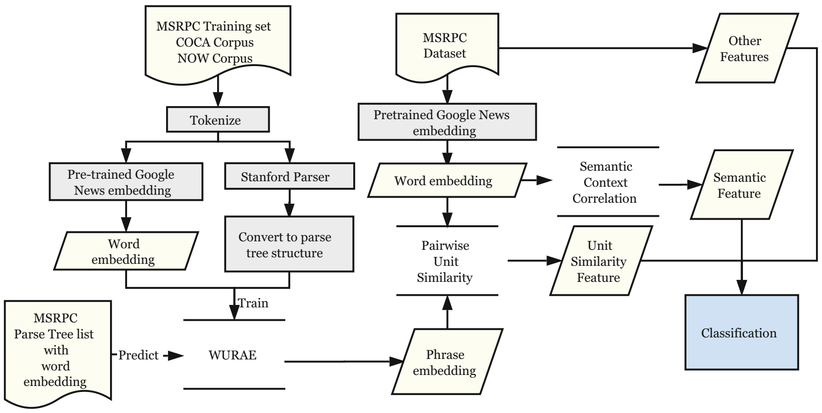


Fig. 3. The overall architecture of sentence-level paraphrase identification

3.6 Paraphrase Identification on Pairwise Article

The overall methodology of article-level PI task is depicted in Fig. 4. A dataset of Chinese news paraphrase article (CNPA) is used in this task. Chinese word embedding is trained by Word2Vec. Then, WURAE is trained in a large scale of Chinese sports & entertainment news sentences. Sentence embedding, calculated by WURAE, represents the articles. For classification, pairwise sentence similarity feature and semantic context correlation feature are extracted from the given pair of articles through the models mentioned above. To get better performance, number feature is also added to this method. The probability of being paraphrased is predicted by the combination of features.

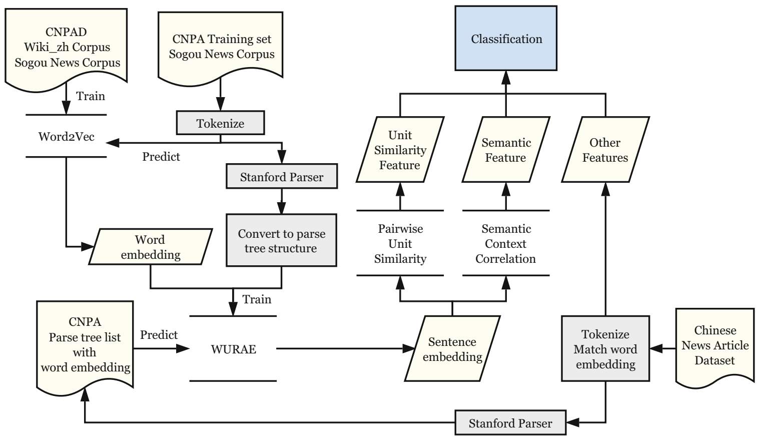


Fig. 4. The overall architecture of article-level paraphrase identification

4 Experiments

4.1 Datasets and Settings

MSRPC. In our sentence-level PI task, we use the benchmark Microsoft Research Paraphrase Corpus. The length of sentences in this corpus ranges from 7 to 35 and 67% of the pairs are paraphrased. The origin train set has 4,076 pairs and we split it into train set and validation set with the ratio of 9 to 1. And the origin test set has 1,725 pairs of sentences. Owing to the asymmetry of two sentences, we expand the dataset by exchanging position of two sentences in one pair. As is mentioned above, a 300-dimensional English word embedding of Google News vectors is applied in this task.

Chinese News Paraphrase Article Dataset (CNPA). An article paraphrase corpus of Chinese sports & entertainment news is used in our article-level PI task. Non-paraphrase pairs are constructed in this corpus by randomly matching articles from different paraphrase pairs. We further introduce comparison on length and TF-IDF to prevent negative pairs from differing too much. The length of articles, which means the number of its sentences, varies from 10 and 55. We split the dataset into train set, validation set and test set, as shown in Table 1. The Chinese word embedding is trained from Word2Vec with Chinese Wiki data and Sogou News data in the dimension of 300.

Table 1. Statistics of Chinese news paraphrase article dataset

Set	Article pairs	Paraphrase	Non-paraphrase
Train	10191	5721	4470
Val	2909	1633	1276
Test	1455	817	638

Settings. The hyperparameters are tuned on the validation set of MSRPC. The settings of English sentence-level PI experiment are chosen as Adadelta optimizer, learning rate of 0.175, dropout rate of 0.1 and mini-batch size of 50. We adjusted the mini-batch size of Chinese article-level PI experiment to 64. The size of k-max pooling in pairwise unit similarity measurement is separately 15 for word-level or sentence-level unit and 17 for phrase-level unit.

4.2 Distributed Representation of Phrase and Sentence

WURAE is trained in the mini-batch of the sentences from a large scale of English and Chinese corpus. The English corpus is constructed by COCA (Corpus of Contemporary American English), NOW (News on the Web)[4] and MSRPC train set, which have 80,697 sentences. The Chinese corpus is composed of Sogo News data and sentences in train set, which has 421,293 sentences. To get the parse tree, we preprocessed the corpus by Stanford Parser. Based on WURAE, the phrase and sentence embedding is learned from the parse tree with the initial word embedding.

Table 2. Performance of different features

Model	Accuracy	F1-score	Model	Accuracy	F1-score
WE + Pairwise Similarity	71.94%	79.13%	With Other Features	75.01%	82.23%
WE + Semantic Context Correlation	71.42%	80.73%		73.80%	81.58%
WE + PE + Pairwise Similarity	72.70%	81.35%		75.48%	82.38%
WE + PE + Pairwise Sim + Context Corr	73.91%	81.99%		76.70%	83.44%

4.3 Results

MSRPC. Firstly, we test performance of separate and combined features, shown in Table 2. We can find that phrase embedding improve the performance of pairwise similarity by 0.76% on accuracy and 2.22% on F1-score. A performance of 71.42% is obtained from semantic context correlation. Our entire sentence-level paraphrase identification model gained the accuracy of 76.70% and F1-score of 83.44%.

We also compare our methodology with lots of state-of-art methods. The comparison is shown in Table 3. Our method achieves a competitive result compared with the

[4] https://corpus.byu.edu/.

Table 3. Experimental results of english sentence-level paraphrase detection

Method		Open resources	Acc	F1-score
All paraphrase (Baseline)			66.5%	79.9%
Hu et al. [7]	Convolutional Matching Model	Project homepage	69.9%	80.91%
Socher et al. [18]	URAE with Dynamic Pooling	Pre-trained phrase vector data, PI code	76.8%	83.6%
Madnani et al. [14]	8 Machine Translation Metrics	Error analysis data	77.4%	84.1%
Pang et al. [17]	Text Matching via CNN		75.94%	83.01%
El-Sayed et al. [3]	Similarity & Abductive Network		73.91%	81.25%
Eyecioglu et al. [4]	Character-Based Features		74.2%	82.7%
Our Work	WURAE with K-Max, CNN and LSTM	WURAE, PI code, pre-trained data[5]	76.70%	83.44%

existing methods. It shows that deep semantic features of sentences could be extracted by the combination of WURAE, pairwise similarity and context correlation method.

CNPAD. This dataset is experimented on both separate and combined features, shown in Table 4, where SE means sentence embedding. The pairwise sentence similarity gets the accuracy of 96.15% and semantic context correlation gets the accuracy of 96.91%. Through combination of those two methods, we could get an improvement of 0.55% on accuracy. And the overall architecture obtains the accuracy of 99.31% and F1-score of 99.39%. We can see that the combination of sentence embedding, pairwise similarity and semantic context correlation do capture the deep semantic feature of articles.

Table 4. Experimental result of Chinese article-level paraphrase detection

Method	Accuracy	F1-score
All paraphrase (Baseline)	56.15%	71.92%
SE + Pairwise Sentence Similarity	96.15%	96.57%
SE + Semantic Context Correlation	96.91%	97.23%
SE + Pairwise Similarity + Context Correlation	97.46%	97.72%
SE + Pairwise Similarity + Context Correlation + Number feature	99.31%	99.39%

[5] https://github.com/SannyZhou/WURAE_Paraphrase_Identification_CNN_LSTM.

5 Conclusion

In this paper, we proposed a method of sentence-level paraphrase identification and introduced an article-level paraphrase identification method by analogy. Also, Weighted Unfolding RAE, an unsupervised learning algorithm, is proposed for learning phrase and sentence embedding. In the sentence-level PI task, words and phrases embedding represents the sentences while the articles are represented by sentence embedding in the article-level PI task. Pairwise unit similarity feature is captured from unit similarity matrix through CNN and k-max pooling layers. After getting region information from sequences by multiple CNN layers with different window sizes, the model implements LSTM to learn the long-term dependency of text. The experimental results prove that our methodology could capture deep semantic feature and perform well in paraphrase identification. It also shows that we can get better semantic feature with the distributed representation of phrases and sentences based on WURAE. In the future, we could build an open domain Chinese paraphrase corpus. Also, we would adjust our paraphrase identification method and our algorithm of phrase & sentence embedding in different NLP applications, such as question answering, information retrieval, text classification, etc.

Acknowledgements. This research work has been funded by the National Natural Science Foundation of China (Grant No. 61772337, U1736207 and 61472248), the SJTU-Shanghai Songheng Content Analysis Joint Lab, and program of Shanghai Technology Research Leader (Grant No. 16XD1424400).

References

1. Agarwal, B., Ramampiaro, H., Langseth, H., Ruocco, M.: A deep network model for paraphrase detection in short text messages. arXiv preprint arXiv:1712.02820 (2017)
2. Chitra, A., Kumar, S.: Paraphrase identification using machine learning techniques. In: Proceedings of the 12th International Conference on Networking, VLSI and Signal Processing, pp. 245–249 (2010)
3. El-Alfy, E.S.M., Abdel-Aal, R.E., Al-Khatib, W.G., Alvi, F.: Boosting paraphrase detection through textual similarity metrics with abductive networks. Appl. Soft Comput. **26**, 444–453 (2015)
4. Eyecioglu, A., Keller, B.: Knowledge-lean paraphrase identification using character-based features. In: Filchenkov, A., Pivovarova, L., Žižka, J. (eds.) AINL 2017. CCIS, vol. 789, pp. 257–276. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-71746-3_21
5. Goller, C., Kuchler, A.: Learning task-dependent distributed representations by backpropagation through structure. In: IEEE International Conference on Neural Networks, vol. 1, pp. 347–352. IEEE (1996)
6. He, H., Lin, J.: Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 937–948 (2016)

7. Hu, B., Lu, Z., Li, H., Chen, Q.: Convolutional neural network architectures for matching natural language sentences. In: Advances in Neural Information Processing Systems, pp. 2042–2050 (2014)
8. Ji, Y., Eisenstein, J.: Discriminative improvements to distributional sentence similarity. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 891–896 (2013)
9. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
10. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
11. Kiros, R., Zhu, Y., Salakhutdinov, R.R., Zemel, R., Urtasun, R., Torralba, A., Fidler, S.: Skip-thought vectors. In: Advances in Neural Information Processing Systems, pp. 3294–3302 (2015)
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
13. Lin, Z., et al.: A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)
14. Madnani, N., Tetreault, J., Chodorow, M.: Re-examining machine translation metrics for paraphrase identification. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 182–190. Association for Computational Linguistics (2012)
15. Mahajan, R.S., Zaveri, M.A.: Machine learning based paraphrase identification system using lexical syntactic features. In: 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp. 1–5. IEEE (2016)
16. Mihalcea, R., Corley, C., Strapparava, C., et al.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI, vol. 6, pp. 775–780 (2006)
17. Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., Cheng, X.: Text matching as image recognition. In: AAAI, pp. 2793–2799 (2016)
18. Socher, R., Huang, E.H., Pennin, J., Manning, C.D., Ng, A.Y.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Advances in Neural Information Processing Systems, pp. 801–809 (2011)

Which Embedding Level is Better for Semantic Representation? An Empirical Research on Chinese Phrases

Kunyuan Pang, Jintao Tang(✉), and Ting Wang

College of Computer, National University of Defense Technology Changsha, Hunan 410073, People's Republic of China
{pangkunyuan,tangjintao,tingwang}@nudt.edu.cn

Abstract. Word embeddings have been used as popular features in various Natural Language Processing(NLP) tasks. To overcome the coverage problem of statistics, compositional model is proposed, which embeds basic units of a language, and compose structures of higher hierarchy, like idiom, phrase, and named entity. In that case, selecting the right level of basic-unit embedding to represent semantics of higher hierarchy unit is crucial. This paper investigates this problem by Chinese phrase representation task, in which language characters and words are viewed as basic units. We define a phrase representation evaluation tasks by utilizing Wikipedia. We propose four intuitionistic composing methods from basic embedding to higher level representation, and investigate the performance of the two basic units. Empirical results show that with all composing methods, word embedding out performs character embedding on both tasks, which indicates that word level is more suitable for composing semantic representation.

Keywords: Word embedding · Phrase representation
Composing model

1 Introduction

Word embeddings have been working as popular features in nearly every NLP task like named entity recognition [15], similarity measurement [8,10], machine translation [3,14], etc. Popular embeddings methods such as skip-gram, CBOW [8], and Glove [11] adhere to the *Distributional Hypothesis* [5]. They first generate a token list from a specific vocabulary of a language, and then calculate embeddings for each token with token cooccurrence information.

However, limited by the vocabulary and resource, embeddings do not cover all language phenomenon, like idioms, named entities and phrases.

The research is supported by the National Key Research and Development Program of China (2018YFB1004502) and the National Natural Science Foundation of China (61472436, 61532001).

Table 1. Semantics of a Chinese phrase 大学城北站

Segmentation	Semantics Candidate	Evaluation
大学城北站	Higher Education Mega Center North Station	precise, but hard to include in vocabulary
大学城 | 北 || 站	advanced education district| north part/north of|| station	high word semantic selection, but less generative / potential segmentation error
大学 || 城 || 北站	university|| city|| north (key) station	segmentation error
大 | 学 | 城 | 北 | 站	big/advanced| mimic/knowledge/school| city/town/center| north part/north of/defeat| station/stand/stay	flexible in generation, but complex combinations to choose from

[8] generate embeddings for phrases in a statistical way. Frequent bigrams in corpus are viewed as *idiomatic phrases*. These frequent bigrams are recorded as new tokens and participate in embeddings with other words. This method partially breaks from vocabulary restrictions, but is still restricted to the corpus. Named entities, for example that come into existence after the corpus are not embedded.

Hierarchical structure of language makes it possible to form composed entities and phrases with basic units. Based on this idea, compositional models [18] embed basic units and compose into higher hierarchy structures. This overcomes the coverage problem in both vocabulary and corpus.

For languages like Chinese, token definition is also a problem in embeddings, in which vocabulary can be built on characters or words. Embeddings of characters and words are semantically different, and this difference affects the semantic representing ability of compositional model. Generally, a character embedding is an average of more senses while word embeddings are more specific. Whether with characters or with words is an important question in composing models. As is shown in Table 1, a location entity is composed with different units. The quality of compositional semantic representation is largely decided by composing unit selection. Consequently, which embedding level is better for semantic representing becomes an important research question.

To be specific, We evaluate the quality of semantic representation with (1) measuring the distance between composed embedding and trained embedding for Wikipedia titles, and (2) comparing semantic similarity of phrase embeddings against Wikipedia redirection. Under this evalutaion measure, we use three intuitionistic composing methods, component average(CA), neighbor average(NA) and neighbor cluster average(NCA) and investigate performance of these models on word embeddings and character embeddings.

Results of experiments on this task show that with each composing model, word embeddings outperform character embeddings, which suggests in semantic analyzation tasks, word embeddings might be more suitable.

This paper is organized as follows. Section 2 summarizes related works, especially on how to generate word embeddings, how to use them and how to ensemble component embedding into greater parts embedding. Section 3 illustrates our methods, from embedding short words and characters to calculating long words and phrases. Section 4 shows experiments on embedding error and on Wikipedia redirection prediction tasks. Section 5 summarizes our model and discusses remaining research points.

2 Related Works

Word embedding can be categorized into 3 classes, which are language model based, task based, and direct generation.

Both language model based and direct generation models follow *Distributional Hypothesis* [5], which states that *words with same contexts tend to have similar meanings.* In different models, this hypothesis is implemented with different optimization objectives. In NNLM [1], a 3-layer neural network is used to estimate $P(w_i|w_{i-(n-1)}, ..., w_{i-1})$, where embeddings of $(w_{i-(n-1)}, ..., w_i)$ serve and get trained as the first layer.

CBOW and skip-gram [8] try to generate word embedding with a simple task. The task is to predict the word with context words or reversely. CBOW predicts by dot production current word and the average of embeddings in context window. Skip-gram is similar. GloVe [11] records co-occurrence of words and suppose words co-occurrence frequency ratio as similarity ratio. Ideally GloVe and skip-gram converge to the same embedding if an optimum embedding exists.

Task based embedding solve supervised tasks with neural network. The embedding layer can be generalized to other tasks and serve as word embedding. Fasttext [6], for example, use a shallow neural network for text classification. The first layer is taken as word embedding.

All word embeddings mentioned above claim to represent syntactic and semantic embedding. These trained embeddings are released for use in other tasks.

Models are put forward to improve word embedding or solve problems in application with hierarchical structure of language, especially in dealing with OOV(out of vocabulary) words. [12] model words with a convolutional network over its characters. Character patterns in English is believed to be strong in syntactic and the combination of characters can make up any word. [?] used Byte-Pair Encoding(BPE) to control vocabulary size in machine translation. Words are first encoded with character pairs before fed into neural machine translator. [17] built a recursive neural network on its syntactic tree to encode a sentence. When encountering an OOV word, the recursive network is formed in a primitive left-to-right way on its sub-word parts. These sub-word parts are found by BPE and serve as leaf embedding nodes in the syntactic tree.

In Chinese, most researchers use word embedding as is parallel in English. Still, character embedding is also used, especially in word segmentation [20] and text classification [19]. Characters can also enrich word embeddings or even be divided into sub-character parts. CWE [2] modifies CBOW, Skip-gram and GloVe by adding character embedding average into context vector, and improve the quality of target word embedding. [13] breaks Chinese characters into radicals. They use CBOW for radical embedding, and feed radical embedding to different neural networks for short text classification, Chinese word segmentation and web search ranking.

[9] work on composing phrase with its components on early vector based models of word meaning. They designed additive model and multiplication model. Both models are enlightening for composing on word embedding.

3 Methodology

This section illustrates the framework of word embedding generation and phrase embedding composing.

3.1 Problem Definition

With a given segmented corpus $\mathbf{D}$ and an embedding method, sets of chars $\mathbf{S_C}$, words $\mathbf{S_W}$ and phrases $\mathbf{S_P}$ are defined. To avoid repetitive description, we denote these language units **tokens**, $\mathbf{S_T} = S_C \cup S_W \cup S_P$. Embedding methods give every token t a vector representation $\boldsymbol{e}(t)$. We research on long tokens $t = (c_1 c_2 ... c_n)$ where $c_1, c_2, ..., c_n \in S_T$. These substrings $c_1, ..., c_n$ are called **components**. The aim of this research is to compose estimation of $\boldsymbol{e}(t)$ with $\boldsymbol{e}(c_1), ..., \boldsymbol{e}(c_n)$, and make the composed estimation $\boldsymbol{e}_{\mathbf{comp}}(\mathbf{t})$ as close to trained $\boldsymbol{e}(t)$ as possible if $t \in S_T$.

3.2 Modified CBOW

CBOW is used to generate word embedding as a basis of phrase embedding composing. In order to capture composability from Chinese characters to words, a modification is made.

The original version of CBOW works on Chinese words as follows. First, sentences in the corpus are segmented into words with segmentation algorithm. Secondly, words with frequency above the threshold are selected as tokens and form token list. Finally CBOW iterates on the corpus several times, on each word in token list with objective described in Eq. 1. $\boldsymbol{e}(word)$ and $\boldsymbol{e}'(word)$ are two sets of embeddings, and $\boldsymbol{e}_{context}$ is the average vecter of context words.

$$Cost = \sum_{(word,context) \in D} \frac{\exp(\boldsymbol{e}'(word)\boldsymbol{e}_{context})}{\sum_{word' \in V} \exp(\boldsymbol{e}'(word')\boldsymbol{e}_{context})} \quad (1)$$

In order to compare estimating precision of different composition methods, embedding of characters, words and phrases are needed. CBOW can produce phrase embedding by including them in user segmentation dictionary and segmenting by large-grain. Thus, those phrases in the corpus are included in the token list and get an embedding.

A large majority of characters are also included in the token list, but they are not character embeddings. Single characters appear in word-level token list for two reasons. It is a single-character word, or it is left because of segmentation error. A Chinese character has a lot more senses than words. Senses of a character as an independent word are usually different from those composing other words. Embedding of single-character words in CBOW is thus infeasible in character to word/phrase composition.

An option is to train CBOW on character level separately and generate character embedding with composing senses. However, the alignment between character embedding space and word embedding space takes extra effort, and existing research of alignment is not satisfying in Chinese character to word alignment task. CWE produce character embedding and word embedding at the same time, too, but that character embedding is an additive part in context, which is not in word embedding space either.

We modify CBOW by randomly replacing a word as a character composing it in each iteration. With enough iteration, the character embedding is a combination of single-character word sense and word composing sense. Experiments show that this modification dose not harm CBOW in its ability to embed semantic information of words. Character sense is improved since when looking for similar words of a character, more words that it composes are recalled.

3.3 Trivia Combination Model

In the discussion to follow, we discuss methods to calculate representation of tokens via components of lower level linguistic units.

CA (Component Average) is a trivia model of estimation is component average. Let T be the component sepuence of a token t.

$$\boldsymbol{e}_{comp}(t) = \frac{1}{|T|}\sum_{c\in T}\boldsymbol{e}(c) \tag{2}$$

NA (Neighbor Average) explores more information of the components by finding m most similar words in the embedding space and then calculates average of these neighbors. Let N be the set of neighbours, i.e. $N = \{q|rank(q, c_i) < m, c_i \in T\}$

$$\boldsymbol{e}_{comp}(t) = \frac{1}{|N|}\sum_{n\in N}\boldsymbol{e}(n) \tag{3}$$

3.4 Neighbor Cluster Average

NCA is based on the following hypothesis in composing tokens from components: **Sense Activation Hypothesis**: A component as words or characters has several senses. When composing high level structures, i.e. tokens or sentences, one sense of the component is activated. Activated senses of components compose semantic meaning of tokens and sentences.

In the embedding point of view, this hypothesis means that component embedding is an average of its sense embeddings. This is in accordance with *Distributional Hypothesis* and the process of CBOW. Each sense of a component can be represented by a distribution of context words. Training on the corpus by CBOW is training the embedding of a component by a superposition of its sense context distributions and results in an average of senses.

It is observed from word similarity task that similar words, or neighbors of a word requires interpretation from different senses of a word, concluding that more information of different senses can be recovered from word neighors.

NCA model aims to discover combination of senses by clustering. Neighbors of each component are retrieved with a large window, ensuring that as many senses represented by neighbors are included as possible. Since the embedding model views cooccurance as similarity. If two senses of two components are likely to be acitvated in the same token, their corresponding neighbour clusters should be close in the embedding space and forms one cluster when clusting all neighbors of all components. Selecting the largest cluster is thus selecting the most likely average of combined senses. Let C_{max} be the largest cluster over all $n \in N$ as defined in 3.3

$$\boldsymbol{e}_{comp}(t) = \frac{1}{|C_{max}|} \sum_{n \in C_{max}} \boldsymbol{e}(n) \tag{4}$$

We use k-means cluster algorithm. k is set to size of components. We take the centroid of the largest cluster as the representation of the token.

3.5 Self Attention Model

Self attention model(denoted as **ATTN**) follows attention machenism preopsed in [16].

$$\boldsymbol{e}_{comp}(t) = \frac{1}{|T|} \sum_{c_i \in T} \alpha(c_i) \boldsymbol{e}(c_i) \tag{5}$$

$$\alpha(c_i) = tanh(w * cos(\boldsymbol{e}_{context}, \boldsymbol{e}(c_i)) + b_p) \tag{6}$$

$$\boldsymbol{e}_{context} = \frac{1}{|T| - 1} \sum_{c_j \in T, j \neq i} \boldsymbol{e}(c_j) \tag{7}$$

The importance or weight of a component c_i is estimated as coherence with other components in the token. w is used for controlling weight ratio of high relavance to low relavance. $b_p \in \{begin, middle, end\}$ is the position bias of weight encoding positional information of a component. The cost to train w and b_p is: every combosed estimation embedding is close to its trained embedding.

$$Cost = -\sum_{t \in S_T} cos(\boldsymbol{e}_{comp}(t), \boldsymbol{e}(t)) \tag{8}$$

4 Experiments

To find the right level of representing phrase semantics, we compare segmenting phrase into words and characters. We also experiment on composing words with characters to show composability of characters. Absolute Embedding Error compares the precision of composing compared to standard embedding trained by CBOW.

4.1 Experiment Settings

Corpus and Dictionary. We use modified CBOW on cleaned Chinese Wikipedia corpus. We extracted 1GB pure text from dumped wiki-pages. Jieba[1] is used to segment the pure text. Wikipedia title list is added as user dictionary to ensure that we retrieve enough phrases and train their embedding for comparison.

Embedding Algorithm and Parameters. We run our modified CBOW on the text. Replace ratio is 0.1, and iteration is set to 20 times, larger than usual to ensure replacement balance. The embedding dimension is 60 and minimum occurrence of a token is set to 3.

Character, Word and Phrase Selection. The identification of characters, words and phrases is by length. We take into consideration only tokens purely consist of Chinese characters.

We select tokens with a length of 1 as characters, 2–3 as words and longer than 5 as phrases. This selection is based on reasons that follows. First of all, it is hard to separate characters and single-character words. Thanks to our modification over CBOW, embedding of tokens with the surface form of single characters always contain semantic information as characters.

According to [7], Chinese linguists listed the most frequently used words. Among 56008 of them, only 162 are of 5 characters and above and most of them have an inner structure of shorter words. We are confident that these long tokens are phrases.

2–3 characters long tokens are words without doubt. Words with 4 characters are a mixture with independent words, short phrases, and a lot of Chinese traditional idioms of weak composability. As a result, we end up with 10386 characters, 118348 words and 49878 phrases for composing experiments.

Compose Levels. Table 2 shows all 4 composing levels that we test on. Composing from characters to words is also included, in case the number of components affects composing quality.

Though our separation and composing method is abrupt and simple. It separates composing from words and composing from characters well. Any other

[1] http://www.oss.io/p/fxsjy/jieba.

Table 2. Illustration of different composing levels

Level	Component Selection	example(广州市大学城北站)
p_w	non-overlapping words found in the phrase with forward maximum matching	广州市 | 大学城北
p_c	all characters in the phrase	广 | 州 | 市 | 大 | 学 |······
p_l	words in p_w and characters **not** in p_w	广州市 | 大学城北 | **站**
w_c	all characters in a **word**	广 | 州 | 市

given composing divides into word-to-phrase and character-to-phrase composing patterns.

4.2 Relative Composing Precision

Relative Composing Precision experiment compares the composed phrase embedding with trained phrase embedding. Formally, this precision is defined as Eq. 9. $\boldsymbol{e}(n)$ is the embedding of the most similar token in S_T.

$$RCP = cos(\boldsymbol{e}_{comp}(t), \boldsymbol{e}(t)) - cos(\boldsymbol{e}(n), \boldsymbol{e}(t)) \tag{9}$$

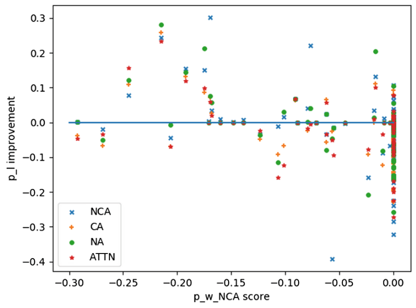
Note that $\boldsymbol{e}_{comp}(t)$ is the only variable for a given sample token. The reason not using bit-wise L2-loss is, in CBOW similiarity is valued by cosine. The norm of token vecters is not 1. $|\boldsymbol{e}_{comp}(t) - \boldsymbol{e}(t)|^2$ can still be large even if we get the exact meaning. It is acceptable that our composed embedding is synonym of the original phrase. The reason for adding reference score $cos(\boldsymbol{e}(n), \boldsymbol{e}(t))$ is to align samples at different composing difficulties. For tokens that lies in dense parts of embedding space, the error is penalised by likely higher reference score.

Table 3. Relative composing precision on different levels and methods

	CA			ATTN			NA			NCA		
	p_w	p_c	p_l	p_w	p_c	p_l	p_w	p_c	p_l	p_w	p_c	p_l
Mean RCP	−0.28	−0.56	−0.30	−0.28	−0.56	−0.29	−0.21	−0.46	−0.22	**−0.20**	−0.43	−0.21
Best sample	0.01	−0.13	0.01	0.02	−0.12	0.02	**0.06**	−0.08	**0.06**	**0.06**	−0.05	0.04
RCP @75%	−0.14	−0.47	−0.17	−0.13	−0.45	−0.15	**−0.08**	−0.35	−0.09	−0.09	−0.31	−0.10
RCP @50%	−0.26	−0.56	−0.28	−0.24	−0.55	−0.26	−0.17	−0.44	−0.20	**−0.16**	−0.40	−0.19
RCP @25%	−0.36	−0.65	−0.40	−0.39	−0.67	−0.42	−0.30	−0.58	−0.30	**−0.28**	−0.57	−0.29
Worst sample	−0.75	0.93	−0.80	−0.75	−1.00	−0.80	−0.68	−1.00	−0.71	−0.70	−0.95	**−0.64**

We compare composing methods in Table 3. On each composing level, NCA always achieves the best performance. On p_w, NCA advantage over NA exists in the low-quality cases. Attention model improves at most 0.05 points over CA model. The improvement of introducing neighbor for more information is significant, as on each level, NA and NCA are a lot better than CA.

Comparing between p_w level and p_c level. Even the worst model for p_w is better than p_c. This comparison shows the importance of level selection. Pure character level is not suitable for semantic composing task.



	CA	NA	NCA	ATTN
average affect	-0.02	-0.01	-0.02	-0.02
p_w leads by	**0.03**	**0.03**	**0.04**	**0.03**
p_l leads by	0.01	0.01	0.02	0.01

Fig. 1. Comparison between p_w and p_l

Improvements of information from left characters is not significant. Table 3 shows the result of introducing leftover characters for more information(level p_l). We also scatter sample points in Fig. 1 to show p_l improvments over p_w results when the original p_w scores differently. Overall, p_l result is better when p_w score is small, but becomes noise when p_w socre is large.

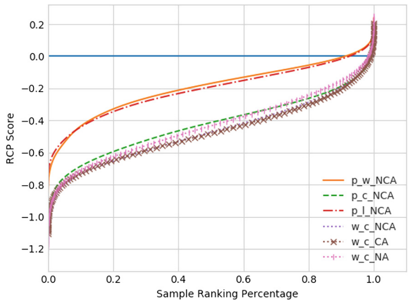


Fig. 2. w_c Results

Figure 2 shows the performance of composing words with characters. A potential reason why p_c performs badly is segments the phrase into too many components for the model to process. w_c has 2–3 components and is similar to p_w segmentation. If character embedding were also a good level of composing, w_c should achieve similar performance as p_w. However, as is shown in the figure, w_c performs similar to p_c level. The reason lies in characters or character embedding, but not in component number.

These experiments show that word alone is the only level that compose the embedding of phrase with low difference. Character level is not only unsuitable itself, but also bring noise when integrating with words.

4.3 Wikipedia Redirections Prediction

The most direct semantic task is word similarity test like word-sim 393 in English and wordsim245 and wordsim297 in Chinese. However, wordsim245 and wordsim297 contains very few phrases and we have to compose our own semantic similarity task.

We compose phrase-word semantic similarity task by utilizing Wikipedia redirections. Redirections in Wikipedia are paraphrases of the same thing or closely related things noted by Wikipedia editors. A pair of redirections are thus semantically identical or very close.

We construct positive set by finding all redirections with at least one embedded phrase. Negative set is constructed by sampling a pair of words and phrases from positive set, making sure that the pair is not in positive set. The size of positive and negative set each is 426.

We use word similarity directly for this classification task and adopt AUC [4] to examine precision of similarity without setting threshold manually.

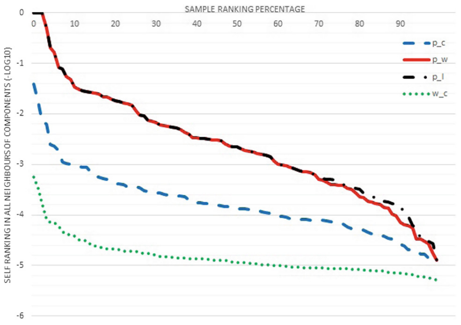
Table 4. AUC of Wikipedia redirection prediction

	Composing level	
	p_w	p_c
CA	0.9485	0.7845
NA	0.9336	0.6841
NCA	0.9295	0.7037

Trained value of words and phrases are used as standard reference value. Composed embedding of phrases are used for each test case and The AUC values are shown in Table 4. High AUC of standard reference show that our embedding and cosine similarity is a good feature for the task. With each method, p_w is a lot better than p_c. This again proves word is the only right level to compose semantic embedding of phrases.

4.4 Case Study: Difference in Component Quality

Opposite from common sense that if we understand all characters in a word well, we will understand the word. Composing words and phrases from characters is impossible. We explain this phenomenon with descriptive experiments.



	nearest neighbor		
	p_w	p_c	p_l
mean RCP	-0.0364	-0.0894	-0.0312
best sample	0.0000	-0.0000	0.0000
RCP @75%	0.0000	-0.0163	0.0000
RCP @50%	0.0000	-0.0824	0.0000
RCP @25%	-0.0468	-0.1304	-0.0284
worst sample	-0.2924	0.3693	-0.2924

Fig. 3. Difference in component quality

Figure 3 left illustrates the most ‘precise’ neighbor we retrieved in NA and NCA at different levels.[2] This precise neighbor is useless in models because we need the standard answer to identify its precision, and the composing model is mostly sorting this neighbor by all information. Still, it helps to illustrate quality of our components. It is shown that half p_w components include synonym of the target token in neighbor set, while character levels finds only close words.

We also try to retrieve the target phrase as most similar word of its components. As Fig. 3 shows, to achieve a 50% recall rate, Character need to expand similarity words window to 4,000 tokens. In contrast, 1,000 tokens window retrieves 63.8% phrases with word level components.

We conclude that the failure with character level composing lies in the character embeddings being too far away from words and phrases that it forms.

5 Conclusion and Future Work

We investigate the token definition problem of embedding for semantic representation by phrase composition task in Chinese. Evaluated on different composing methods, composing precision and Wikipedia redirection prediction both show that each method with word embedding outperforms the same method with character embedding. This indicates word embedding might be better in semantic representation then character embedding.

Our future work includes 2 directions. (1) We plan to conduct more experiments on semantic analyze tasks and evaluate on semantic representativeness of word embedding and character embedding from more perspectives. (2) We plan to create more complex and precise phrase semantic composing models and try to compose phrase, entities and out of vocabulary tokens better.

[2] We have excluded token themselfs in neighbors in all composing experiments to avoid bias.

References

1. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
2. Chen, X., Xu, L., Liu, Z., Sun, M., Luan, H.: Joint learning of character and word embeddings. In: International Conference on Artificial Intelligence, pp. 1236–1242 (2015)
3. Cho, K., Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: EMNLP, pp. 16–37 (2014)
4. Fawcett, T.: An introduction to ROC analysis. Pattern Recognit. Lett. **27**(8), 861–874 (2006)
5. Harris, Z.S.: Distributional Structure. Springer, Netherlands (1981). https://doi.org/10.1007/978-94-009-8467-7_1
6. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. In: Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 427–431 (2017)
7. Li, X., Wang, T.: Lexicon of Common Words in Contemporary Chinese. The Commercial Press, Beijing (2008)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
9. Mitchell, J., Lapata, M.: Vector-based models of semantic composition. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, pp. 236–244 (2008)
10. Parikh, A., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 2249–2255 (2016)
11. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)
12. Shen, Y., He, X., Gao, J., Deng, L.: Learning semantic representations using convolutional neural networks for web search. In: International Conference on World Wide Web, pp. 373–374 (2014)
13. Shi, X., Zhai, J., Yang, X., Xie, Z., Liu, C.: Radical embedding: delving deeper to chinese radicals. In: 2010 European Signal Processing Conference, pp. 572–575 (2015)
14. Sundermeyer, M., Alkhouli, T., Wuebker, J., Ney, H.: Translation modeling with bidirectional recurrent neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 14–25 (2014)
15. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 384–394 (2010)
16. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 6000–6010 (2017)
17. Yang, B., Wong, D.F., Xiao, T., Chao, L.S., Zhu, J.: Towards bidirectional hierarchical representations for attention-based neural machine translation. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 1432–1441 (2017)

18. Yu, M., Dredze, M.: Learning composition models for phrase embeddings. Trans. Assoc. Comput. Linguist. **3**, 227–242 (2015)
19. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems, pp. 649–657 (2015)
20. Zheng, X., Chen, H., Xu, T.: Deep learning for chinese word segmentation and pos tagging. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 647–657 (2013)

Improving Word Embeddings for Antonym Detection Using Thesauri and SentiWordNet

Zehao Dou(✉), Wei Wei, and Xiaojun Wan

Peking University, Beijing, China
{zehaodou,weiwei,wanxiaojun}@pku.edu.cn

Abstract. Word embedding is a distributed representation of words in a vector space. It involves a mathematical embedding from a space with one dimension per word to a continuous vector space with much lower dimension. It performs well on tasks including synonym and hyponym detection by grouping similar words. However, most existing word embeddings are insensitive to antonyms, since they are trained based on word distributions in a large amount of text data, where antonyms usually have similar contexts. To generate word embeddings that are capable of detecting antonyms, we firstly modify the objective function of Skip-Gram model, and then utilize the supervised synonym and antonym information in thesauri as well as the sentiment information of each word in SentiWordNet. We conduct evaluations on three relevant tasks, namely GRE antonym detection, word similarity, and semantic textual similarity. The experiment results show that our antonym-sensitive embedding outperforms common word embeddings in these tasks, demonstrating the efficacy of our methods.

Keywords: Antonym detection · Word embedding · Thesauri SentiWordNet

1 Introduction

Word embedding plays an important role in word representation since it effectively captures semantic information of words. A good embedding provides vector representations of words such that the relationship between two vectors mirrors the linguistic relationship between the two words. Such distributed representations of words in a vector space contribute to achieving better performance in many natural language processing tasks such as text classification [7], abstraction generation, and entity recognition.

By grouping similar words, the existing word embeddings perform well on synonyms, hyponyms, and analogies detection. However, most of them are insensitive to antonyms, since they are trained based on the distributional hypothesis [4] and word distributions in a large amount of text data, where antonyms usually have similar contexts. To be specific, a pair of antonyms, for example, "long"

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 67–79, 2018.
https://doi.org/10.1007/978-3-319-99501-4_6

and "short", tend to appear in similar context environment. This leads to the serious problem that it is extremely difficult to discriminate antonyms from synonyms. It is important to solve this problem and obtain antonym-sensitive word embedding, since such embedding has the potential to make contribution in some certain tasks such as semantic textual similarity.

A key characteristic of current word embeddings attracts our attention: the vectors of related words are supposed to have close directions while unrelated words correspond to vectors in opposite directions. Therefore, the cosine distance between completely unrelated words is close to -1, while the cosine distance between a pair of related words, including both synonyms and antonyms, is close to 1. In other words, instead of evaluating sematic similarity, these embeddings evaluate the relatedness between words, treating synonyms and antonyms equally without discrimination.

To obtain antonym-sensitive word embedding, we start from modifying the above characteristic. In our model, the cosine distance between related words is supposed to be either close to 1 or close to -1, while the cosine distance between unrelated words is close to 0. Then in the vector space, the related words of a particular word distribute in two areas, either around the position of the word or around the head of the reversed vector. In this situation, we can design a more reasonable model to detect antonyms. Given that both antonyms and synonyms are highly-related word pairs, our goal is to make the cosine distance of synonyms close to 1 and the cosine distance of antonyms close to -1. Then a pair of words with a negative cosine distance close to -1 are likely to be antonyms.

With this idea, we propose a novel approach to train our antonym-sensitive embedding, named Word Embedding Using Thesauri and SentiWordNet with Distributional Corpus-based Information (WE-TSD). Firstly, we modify the objective function of Skip-Gram model in order to achieve the goal stated in the previous paragraph. Secondly, our model uses thesauri information to get supervised synonym and antonym word pairs, and we also utilize SentiWordNet provided by [1] to make sentimental analysis of every word in vocabulary. SentiWordNet is a dataset which labels each English word with a corresponding 3-dimensional vector. The three components of the vector respectively represent the positive emotion rate, negative emotion rate and objective rate of the word with their sum 1.

To demonstrate the efficacy and task adaptability of our antonym-sensitive embedding, we conduct evaluations on three relevant tasks, namely GRE antonym detection, word similarity, and semantic textual similarity. The experiment results show that our antonym-sensitive embedding outperforms common word embeddings in all these tasks, convincingly demonstrating the effectiveness of our methods.

2 Related Works

In the past decades, research on natural language processing has made great success, where a good word representation plays an crucial role. At first, one-hot word vector has been widely used in the bag-of-words (BOW) text model. The success of text categorization [5] with BOW popularized this model. However, one-hot vector has a serious weak point: with each word represented as a completely independent entity, this word representation hardly provides any notion of similarity.

To reduce the size of vector space and encode the relationship between words, Rumelhart et al. [17] described a new learning procedure to learn representations. With the word embedding model, many natural language processing tasks, such as entity recognition and dependency parsing, gained second life. In this word embedding model, semantic relationship between words are well encoded and can be easily detected.

However, the huge time cost and computational complexity became an obstacle, leading researchers to find a more efficient and less complex model. In 2013, two highly efficient models were proposed by Mikolov et al. [9], namely CBOW (continuous bag-of-words) and Skip-Gram. CBOW aims to predict a center word from the surrounding context in terms of word vectors. Skip-gram does the opposite, and predicts the distribution of context words from a center word. With these two models, we are able to learn word embeddings with large corpus.

On antonym detection tasks, Polarity Inducting Latent semantic Analysis proposed by Yih et al. abstracts polarity information from thesauri and they use context vectors to cover those words out of thesauri vocabulary. Ono et al. [14] proposes a word embedding-based antonym detection using thesauri and distributional information, which combined the traditional Skip Gram objective function [8] and the polarity thesauri information. This model contributes to finding a balance between the Skip Gram model and WE-T (Word Embedding with Thesauri Information) model. Nguyen et al. [13] proposes a novel model which integrates distributional lexical contrast into word embedding. However, the results of these previous work are far from perfection, motivating us to conduct the research in this paper and make some improvement.

3 Our Approach to Obtain Antonym-Sensitive Word Embeddings

3.1 Skip-Gram Model and Our Modification

In this section, we firstly introduce the original Skip-Gram Model and negative sampling proposed by Mikolov et al. [8], which is one of the most popular methods to train word embeddings. Next, to adapt for our antonym-sensitive embedding, we make some modification on the objective function in the Skip-Gram Model.

Skip-Gram Model with Negative Sampling. A word embedding is a mapping $V \rightarrow R^D$ that maps a word to its corresponding D-dimensional word vector. This is called a D-dimensional word embedding. The most widely used approaches of training word embeddings are Skip-Gram and CBOW [8]. In this paper, we mainly focus on the former one.

In the standard Skip-Gram Model, we aim to obtain a word embedding that can predict the context words around a given target center word effectively. To be specific, our goal is to maximize the following objective function,

$$\frac{1}{T}\sum_{t=1}^{T}(\sum_{t-c\leq i\leq t+c, i\neq t} \log p(w_i|w_t)), \tag{1}$$

where $w_1, w_2, \cdots w_T$ are the words in the whole training corpus, and c is a hyperparameter corresponding to window size, determining the number of context words induced by the target center word w_t. The most essential part of this model is the conditional probability $p(w_i|w_t)$. Following Mikolov et al.'s idea [9], the conditional probability is expressed as follows,

$$p(w_i|w_t) = \frac{e^{u'^T_{w_i} u_{w_t}}}{\sum_{j=1}^{|W|} e^{u'^T_{w_j} u_{w_t}}}, \tag{2}$$

where u'_w, u_w denote the representations of word w respectively as context and target word. W is the vocabulary set extracted from the training corpus and $|W|$ denotes the vocabulary size. Further, Eq. (2) can also be written as

$$p(w_i|w_t) = softmax\left(u'^T_{w_i} u_{w_t}\right). \tag{3}$$

Although the basic model described above seems reasonable, its performance is actually not satisfying enough. Due to the normalization term, the time complexity of the above conditional probability equation is $O(|W|)$, which is unacceptable especially when W is large. Therefore, two modifications have been proposed. Firstly, Mikolov et al. [8] present hierarchical softmax as a much more efficient alternative to the normal softmax. With hierarchical softmax, the time complexity can be reduced to $O(\log(|W|))$.

Another idea to improve the efficacy of word embeddings and reduce the training cost is negative sampling, which is also provided by Mikolov et al. [8]. For every training step, instead of looping over the entire vocabulary, we just sample several negative examples. Although negative sampling is based on the Skip-Gram model, it is in fact optimizing a different objective. The new objective function tries to maximize the probability of a word and context being in the corpus data if it indeed is, and maximize the probability of a word and context no being in the corpus data if it indeed is not, which is shown as follows,

$$L = \sum_{w\in W}(\sum_{t-c\leq i\leq t+c, i\neq t} \log(\sigma(v'^T_w v_{w_i})) + \sum_{u\in NEG(w)} \log(\sigma(-v'^T_w v_u))), \tag{4}$$

where σ denotes the sigmoid function, and $NEG(w)$ denotes a small subset of all the negative examples of target word w sampled from a modified unigram distribution [8]. The number of components in the negative sampling subset is called "negative size". Besides, we employ the subsampling [8] which discards the words according to the following probability:

$$P(w) = 1 - \sqrt{\frac{t}{p(w)}}, \tag{5}$$

where t is a threshold to control the discard action and $p(w)$ is the occurrence probability of w in the training corpus. Subsampling is very useful and it aims to make less frequent words be sampled more often.

Modified Skip-Gram Model. However, the widely used Skip-Gram Model described above can not fulfill our goal. In the result embeddings, the cosine distance between unrelated words is close to -1, while the cosine distance between a pair of related words, including both synonyms and antonyms, is close to 1. In other words, instead of evaluating sematic similarity, these embeddings evaluate the relatedness between words, treating synonyms and antonyms equally without discrimination.

To obtain antonym-sensitive word embedding, we want the cosine distance between related words to be either close to 1 or close to -1, while the cosine distance between unrelated words to be close to 0. Then in the vector space, the related words of a particular word distribute in two areas, either around the position of the word or around the head of the reversed vector. Next, given that both antonyms and synonyms are highly-related word pairs, our goal is to make the cosine distance of synonyms close to 1 and the cosine distance of antonyms close to -1.

Therefore, in our model, the objective function with distributional information of unsupervised training corpus should be:

$$Func1 = \sum_{w \in C} (\sum_{t-c \le i \le t+c, i \ne t} \log(\sigma((v'^T_w v_{w_i})^2)) + \sum_{u \in NEG(w)} \log(\sigma(-(v'^T_w v_u)^2))). \tag{6}$$

Our modification is the square functions inside the sigmoid function σ. With the square functions, the absolute value of the dot product of positive samples and target word will get closer to 1 during training, which means their cosine distance will be either close to 1 or close to -1. This result fulfills our expectation.

The modified Skip-Gram Model with the new objective function (6) plays an important role in our final model. However, both antonym and synonym are considered "related" and we still can not discriminate antonyms from synonyms based on their cosine distance. Therefore, we need to utilize supervised dataset including Thesauri Dataset and SentiWordNet, in order to make the cosine distance between synonyms close to 1 while the cosine distance between antonyms close to -1. As a result, the lower their cosine distance is, the more likely to be antonyms they are.

3.2 Word Embedding Injecting Thesauri Dataset Information with Max Margin Framework (WE-TM)

In this section, we introduce a sub-model using thesauri dataset information. Following Ono et al.'s work [14], we are going to embed all the words in thesauri into vectors.

According to our target, we need to increase the dot product between synonyms and decrease the dot product between antonyms. Therefore, we set up an objective function as shown below,

$$Func2 = -\sum_{w \in V} max(0, \gamma - \frac{1}{|S(w)|} \sum_{s \in SYN(w)} sim(w, s) + \frac{1}{|A(w)|} \sum_{a \in AYN(w)} sim(w, a)), \tag{7}$$

where V denotes the vocabulary in thesauri; $SYN(w)$ denotes all the synonyms of word w and $ANT(w)$ denotes all the antonyms of w. $|S(w)|$ and $|A(w)|$ denote the sizes of $SYN(w)$ and $ANT(w)$. The hyper-parameter γ will be set later. $sim(w, s)$ denotes the similarity of the two words in our model, which can be mathematically expressed as:

$$sim(w, s) = v_w'^T v_s. \tag{8}$$

The maximization of $Func2$ makes the similarity score between synonyms very high and that between antonyms very low. Besides, for some indirect antonyms, for example, "beautiful" and "bitter", although they are not antonyms according to thesauri information, their similarity score will also be relatively low because "beautiful" is the synonym of "nice" and "bitter" is the antonym of "nice". The directions of word vectors of "beautiful" and "nice" are almost the same while the directions of word vectors of "nice" and "bitter" are almost opposite. This sub-model is reasonable and effective, and we name this model WE-TM (Word Embedding using Thesauri Dataset Information with Max Margin Framework). Comparing with the WE-TD model proposed by [14], our model introduces the Max Margin Framework which can significantly decrease the risk of overfitting.

3.3 Word Embeddings Based on SentiWordNet (WE-S)

Besides thesauri, the other supervised knowledge base we are going to use is SentiWordNet [1], which is explicitly devised for supporting sentiment classification and opinion mining applications. It is a lexical resource in which each WORDNET synset w is associated to three numerical scores $[Pos(w), Neg(w), Obj(w)]$, describing how objective, positive, and negative the terms contained in the synset are. The triple has the following property:

$$Pos(w) + Neg(w) + Obj(w) = 1. \tag{9}$$

From this dataset, we need to abstract some senti-synonym word pairs and some senti-antonym word pairs according to their corresponding sentiment triple.

We hope the senti-synonym word pairs have high word similarity scores and senti-antonym word pairs have low word similarity scores, just like those in thesauri information. Firstly, we drop all the synsets with the sentiment triple $[0, 0, 1]$ because words in these categories are completely objective and we can hardly conduct any sentimental analysis on them. Then we define a concept named senti-similarity, which evaluates the degree one word is similar to another in respect of sentimental inclination. Mathematically, it is expressed as follows,

$$SentiSim(w_1, w_2) = \frac{Pos(w_1)Pos(w_2) + Neg(w_1)Neg(w_2)}{\sqrt{((Pos(w_1)^2 + Neg(w_1)^2)(Pos(w_2)^2 + Neg(w_2)^2)}}. \tag{10}$$

In this expression, we ignore the objective judgement ratio $Obj(w)$ of these words and calculate the normalized dot product of $[Pos(w_1), Neg(w_1)]$ and $[Pos(w_2), Neg(w_2)]$. The higher the senti-similarity is, the more similarly these two words express in sentimental inclination. In our model, we think that those senti-synonyms should have not only high senti-similarity score but also high word similarity score and vice versa. So our objective function is:

$$Func3 = \sum_{w_1, w_2 \in SWN} (SentiSim(w_1, w_2) - \overline{SentiSim}) \cdot \log \sigma(sim(w_1, w_2)), \tag{11}$$

where SWN denotes the vocabulary of SentiWordNet, and $\overline{SentiSim}$ denotes the average senti-similarity value of all word pairs in SWN; σ is the sigmoid function. Through the maximization of $Func3$, word pairs with higher senti-similarity score will also have higher word similarity.

3.4 Our Approach(WE-TSD)

Our final model is the integration of the three sub-models above. Our objective function is:

$$Func = Func1 + c_1 Func2 + c_2 Func3, \tag{12}$$

and our goal is to maximize the function. All the three sub-models are necessary. The first model Modified Skip-Gram makes use of distributional corpus information and describes the relatedness of words based on unsupervised corpus. The WE-T model controls the word similarity between synonyms and antonyms, and the WE-S model controls the similarity between senti-antonyms and senti-synonyms. The vocabulary size of WE-S is a lot larger than WE-T and it almost covers the whole vocabulary. Each of the three models has its own effect and none of them can be discarded.

In our objective function, c_1 and c_2 are two coefficients used to balance the importance of the three sub-models. While conducting experiments, we need to tune the coefficients and parameters to achieve better performance. We call this novel model WE-TSD which means Word Embeddings Based on Thesauri, SentiWordNet and Distributional information.

4 Experiments

4.1 Evaluation Settings

In this section, we introduce three relevant tasks which we utilize to evaluate our methods, namely GRE antonym detection task, word similarity task and textual similarity task. Both the dataset and the evaluation metrics are described in detail below. In our experiments, we not only compare our model with several baselines but also with some advanced embeddings including WE-T (Word Embedding using Thesauri only), WE-TD (Word Embedding using Thesauri and discributional condition [14]) and WE-S (Word Embedding using SentiWordNet only).

GRE Antonym Detection Task. GRE antonym questions dataset is widely used in antonym detection tasks which is originally provided by [11]. It is a set of questionnaires with several hundreds of single-choice questions. Each question has a target word and five candidates. We need to choose the only antonym of the target word in the five candidates. Moreover, this dataset is divided into two parts, development set which contains 162 questions and test set which contains 950 questions. Since there are 160 questions appear in both of the two sets, we will report results on both the whole test set and the remaining test set with 790 (= 950 −160) questions just like [14] do in their evaluation experiment of WE-TD model.

When we evaluate our model on these single-choice questions, we calculate the similarity of the target word and the five candidates one by one, and then the candidate who has the lowest similarity with the target word is declared the winner. After finishing all the questions, we evaluate our embedding by F-score following Zhang et al. (2014). If our model doesn't contain the target word or the five candidates, the question will be left unanswered. Unanswered questions are regarded the same as wrong-answered.

Word and Semantic Text Similarity Task. In word similarity experiments, one of the most widely used dataset is WordSim353 provided by [3]. In this experiment, we use dataset WordSim353. Since this dataset consists of two parts, the relatedness part and the similarity part, we conduct our experiment on the two parts respectively. Obtaining three results on WordSim353 (Rel), WordSim353 (Sim) and WordSim353 (Combined), we can show the effectiveness of our model comprehensively. In WordSim353, there are 353 word pairs and a human labeled word similarity score for each word pair.

We compute the similarity of these word pairs according to the absolute value of the cosine distance of their corresponding word vectors. The higher the calculated value is, the more related or similar the word pair is supposed to be. Then, we calculate the Spearman correlation between the word similarity scores from our model and the human labeled scores for comparison.

Semantic Textual Similarity (STS) is the task of determining the degree of semantic similarity between two sentences. STS task is an important foundation of many natural language processing applications, but the performance still remains to be improved. One of the difficulties is that common systems are insensitive to antonyms. Two sentences with high overlap but also a pair of antonyms usually indicate opposite meanings, but common systems tend to generate a high similarity score. Therefore, our antonym-sensitive embedding has a great potential to improve the results by avoiding such errors.

We conduct experiments on STS Benchmark [2], which comprises a selection of the English datasets used in the STS tasks organized in the context of SemEval between 2012 and 2017. The performance is measured by the Pearson correlation of machine scores with human judgments. We build a convolutional neural network, which achieves best results on this dataset, as described in [18], and use different word embeddings as the input of the model.

4.2 Training Resource and Parameter Settings of Our Model

Our supervised datasets used to train our WE-TSD model include two parts. The first one is antonym and synonym pairs in two thesauri, WordNet provided by [10] and Roget provided by [6], the other is SentiWordNet provided by [1]. The unsupervised training corpus comes from Wikipedia. We lowercase all the words and drop all the stopping words and punctuations. The size of raw text is over 10GB and the huge size of unsupervised dataset helps us to train the word embedding more accurately.

When training our WE-TSD model, the dimension of embeddings is set to 300, the negative size is 10, window size is 5, and the threshold for subsampling was 10^{-8}. We utilize Adam as the optimizer. During training, we use learning rate decay to fit the training corpus, and the number of iteration epochs is set to 50. In Func2, $\gamma = 0.6$. The parameter c_1 is 100, c_2 was 2.5. While determining these two hyper-parameters, we take the proportion of the size of Wiki Corpus, Thesauri and SentiWordNet vocabulary into account.

4.3 Results of Experiments

GRE Antonym Detection Task. In experiments, we compare our model with baselines including Encarta lookup from [?], S2Net from [?], WordNet & Roget BFTP from [12], WE-T and WE-TD from [14] and so on. Since we evaluate on the same data as [14], we simply report the evaluation results reported by them.

In the GRE Antonym Detection Task, two models obviously surpass the others, namely **WE-TD** proposed by [14] and our **WE-TSD**. Both these two models make use of supervised information about synonyms and antonyms, which may play an important role in this task. The major difference between **WE-TD** and our **WE-TSD** is our utilization of SentiWordNet. On the complete test set **TestSet(950)**, we get an F-score of 92% and outperform the **WE-TD** model which achieves an F-score of 89%, demonstrating the contribution of the semantic information in SentiWordNet.

More detailed results are listed in Table 1. Obtaining the state-of-the-art results, we can state that our antonym-sensitive embedding is capable of detecting antonyms more effectively than other existing embeddings.

Word and Text Similarity Task. As a tool of antonym detection, it is worth celebrating that our WE-TSD method performs well on the GRE antonym detection task. However, it is far from enough as a general word embedding. In order to demonstrate the efficacy and task adaptability of our methods, we then conduct experiments on a basic task, word similarity evaluation.

Table 1. Results on the GRE antonym detection task. The best values are marked in bold font.

Model	DevSet			TestSet(950)			TestSet(790)		
	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
Encatra lookup	0.65	0.61	0.63	0.61	0.56	0.59	-	-	-
WordNet and Roget lookup	1.00	0.49	0.66	0.98	0.45	0.62	0.98	0.45	0.61
WE-T Model	0.92	0.71	0.80	0.90	0.72	0.80	0.90	0.72	0.80
WE-D Model	0.09	0.08	0.09	0.08	0.07	0.07	0.07	0.07	0.07
EnCarta PILSA	0.88	0.87	0.87	0.81	0.80	0.81	-	-	-
WordNet & Roget BPTF	0.88	0.88	0.88	0.82	0.82	0.82	-	-	-
WE-TD Model	0.92	0.91	0.91	0.90	0.88	0.89	0.89	0.87	0.88
WE-TM Model	0.92	0.91	0.91	0.91	0.89	0.90	0.89	0.87	0.88
WE-S Model	0.88	0.87	0.87	0.83	0.81	0.82	0.81	0.79	0.80
WE-TSD Model	0.95	0.92	**0.935**	0.93	0.91	**0.92**	0.92	0.90	**0.91**

In word similarity experiment, we compare our model with many baselines such as path similarity based on hypernym-hyponym structure in WordNet by [16], mutual information based on WordNet by [15], Word2Vec and LDA similarity proposed by [9] based on English Wikipedia data, WebJaccard algorithm based on Google Search webpages, WE-TD by [14] and so on. Our experimental results are listed in Table 2. The results of our model **WE-TSD** surpass both the **Gensim Word2Vec** and **WE-TD** model on WS353(Sim), WS353(Combined) and MEN-TR-3k dataset. Since our embedding aims to reflect the semantic similarity rather than the relatedness between words, **WE-TSD** performs not that well on WS353(REL). On WS353(REL) dataset, our model is slightly inferior to the **Gensim Word2Vec** model, but the results are still acceptable and outperform the other methods, which shows the effectiveness and versatility of our model.

In text similarity experiment, we compare our word embeddings with WE-TD Model, Gensim Word2Vec Model and GLOVE Word Representation Model to

Table 2. Spearman's rank correlation coefficients in different models. The best values are marked in bold font.

Model	WS353(Sim)	WS353(Rel)	WS353(Com)	MEN-TR-3k
Path Similarity on WordNet	0.347	0.262	0.315	0.298
Mutual Information on WordNet	0.388	0.257	0.349	0.325
Gensim Word2Vec on Wiki	0.651	**0.648**	0.650	0.611
LDA Method on Wiki	0.660	0.487	0.575	0.614
WebJaccard on Google Search	0.277	0.050	0.157	0.105
WE-TD Model on Wiki	0.621	0.617	0.620	0.717
WE-TM Model on Wiki	0.625	0.615	0.621	0.720
WE-S Model on Wiki	0.669	0.635	0.650	0.727
WE-TSD Model on Wiki	**0.675**	0.635	**0.656**	**0.732**

Table 3. The calculation of text similarity using CNN. The best values are marked in bold font.

Model	Test set	Validation set
GLOVE Word Representation	0.790	0.832
Gensim Word2Vec	0.794	0.831
WE-TD Model	0.715	0.733
WE-TM Model	0.725	0.746
WE-S Model	0.784	0.831
WE-TSD Model	**0.808**	**0.859**

demonstrate the applicability and efficacy of our antonym-sensitive embeddings on this task. The experimental results are listed in Table 3. Our WE-TSD model outperforms the other embeddings on both test set and validation set.

As is shown above, our model consists of three important parts, namely Modified Skip-Gram, Thesauri based and SentiWordNet based model. In fact, all of them are necessary. The first part is the most basic one and it assures that all the existing word in Wiki Corpus are taken into account in our model. The second part shows its strength in antonym detection task, and the third part plays a vital role in word and task similarity tasks. Our model performs well on all of the three tasks, demonstrating its effectiveness and task adaptability.

5 Conclusions

In this paper, we propose a novel word embedding model to get better performance on discriminating antonyms from synonyms and our model achieves an F-score of 92% on GRE antonym detection task, outperforming the current state-of-the-art. Also this model has an satisfying performance on both word and textual similarity tasks, demonstrating its effectiveness and task adaptability. In future work, we plan to extend our ideas to train word embeddings which are capable of capturing other semantic relations, such as hyponyms and hypernyms.

References

1. Baccianella, S., Esuli, A., Sebastiani, F.: Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In: LREC, vol. 10, pp. 2200–2204 (2010)
2. Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., Specia, L.: Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055 (2017)
3. Finkelstein, L., et al.: Placing search in context: the concept revisited. In: Proceedings of the 10th International Conference on World Wide Web, pp. 406–414. ACM (2001)
4. Harris, Z.S.: Distributional structure. Word **10**(2–3), 146–162 (1954)
5. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 137–142. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0026683
6. Kipfer, B.A.: Roget's 21st century thesaurus in dictionary form: the essential reference for home, school, or office. Laurel (1993)
7. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning, pp. 1188–1196 (2014)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
9. Mikolov, T., Yih, W.T., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 746–751 (2013)
10. Miller, G.A.: Wordnet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
11. Mohammad, S., Dorr, B., Hirst, G.: Computing word-pair antonymy. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 982–991. Association for Computational Linguistics (2008)
12. Mohammad, S.M., Dorr, B.J., Hirst, G., Turney, P.D.: Computing lexical contrast. Comput. Linguist. **39**(3), 555–590 (2013)

13. Nguyen, K.A., Walde, S.S.I., Vu, N.T.: Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. arXiv preprint arXiv:1605.07766 (2016)
14. Ono, M., Miwa, M., Sasaki, Y.: Word embedding-based antonym detection using thesauri and distributional information. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 984–989 (2015)
15. Pantel, P., Lin, D.: Discovering word senses from text. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 613–619. ACM (2002)
16. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet: similarity: measuring the relatedness of concepts. In: Demonstration Papers at HLT-NAACL 2004, pp. 38–41. Association for Computational Linguistics (2004)
17. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533 (1986)
18. Shao, Y.: HCTI at semeval-2017 task 1: use convolutional neural network to evaluate semantic textual similarity. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp. 130–133 (2017)

Neural Chinese Word Segmentation with Dictionary Knowledge

Junxin Liu[1], Fangzhao Wu[2], Chuhan Wu[1], Yongfeng Huang[1](✉), and Xing Xie[2]

[1] Department of Electronic Engineering, Tsinghua University, Beijing, China
{ljx16,wuch15}@mails.tsinghua.edu.cn, yfhuang@mail.tsinghua.edu.cn
[2] Microsoft Research Asia, Beijing, China
{fangzwu,Xing.Xie}@microsoft.com

Abstract. Chinese word segmentation (CWS) is an important task for Chinese NLP. Recently, many neural network based methods have been proposed for CWS. However, these methods require a large number of labeled sentences for model training, and usually cannot utilize the useful information in Chinese dictionary. In this paper, we propose two methods to exploit the dictionary information for CWS. The first one is based on pseudo labeled data generation, and the second one is based on multi-task learning. The experimental results on two benchmark datasets validate that our approach can effectively improve the performance of Chinese word segmentation, especially when training data is insufficient.

Keywords: Chinese word segmentation · Dictionary · Neural network

1 Introduction

Different from English texts, in Chinese texts there is no explicit delimiters such as whitespace to separate words. Thus, Chinese word segmentation (CWS) is an important task for Chinese natural language processing [3,17], and an essential step for many downstream tasks such as POS tagging [20], named entity recognition [9], dependency parsing [2,15] and so on.

Since a Chinese sentence is usually a sequence of Chinese characters, Chinese word segmentation is usually modeled as a sequence labeling problem [13,17]. Many sequence modeling methods such as hidden Markov model (HMM) [5] and conditional random field (CRF) [6] have been applied to the CWS task. A core problem in these sequence modeling based CWS methods is building the feature vector for each character in sentences. In traditional CWS methods these character features are constructed via manual feature engineering [10,19]. These handcrafted features need a large amount of domain knowledge to design, and the size of these features is usually very large [3].

In recent years, many neural network based methods have been proposed for CWS [3,16,17,20]. For example, Peng et al. [11] proposed to use Long Short-Term Memory Neural Network (LSTM) to learn the character representations for CWS and use CRF to jointly decode the labels. However, these neural network

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 80–91, 2018.
https://doi.org/10.1007/978-3-319-99501-4_7

based methods usually rely on a large number of labeled sentences. For words which are scarce or absent in training data, these methods are very difficult to correctly segment the sentences that contain these words [17]. Since these words are in large quantity, it is very expensive and even unpractical to improve the coverage of these words via annotating more sentences. Luckily, many of these words are well defined in existing Chinese dictionaries. Thus, Chinese dictionaries have the potential to improve the performance of neural network based CWS methods and reduce the dependence on labeled data [17].

In this paper we propose to incorporate the dictionary information into neural network based CWS approach in an end-to-end manner without any feature engineering. More specifically, we propose two methods to incorporate the dictionary information for CWS. The first one is based on pseudo labeled data generation, where we build pseudo labeled sentences by randomly sampling words from Chinese dictionaries. The second one is based on multi-task learning. In this method we introduce another task named Chinese word classification (i.e., classifying a sequence of Chinese characters based on whether they can form a Chinese word), and jointly train this task with CWS by sharing the parameters of neural networks. We conducted extensive experiments on two benchmark datasets. The experimental results validate that our methods can effectively improve the performance of CWS, especially when training data is insufficient.

2 Related Work

In recent years, many neural network based methods have been proposed for Chinese word segmentation [3,16,17,20]. Most of these methods model CWS as a sequence labeling task [3,17]. The core difference between these methods mainly lies in how they learn the contextual feature representation for each character in sentence. For example, Zheng et al. [20] proposed to use multilayer perceptrons to learn feature representations of characters from a fixed window. Chen et al. [3] used LSTM to capture global contextual information. They also explicitly captured the local context by combining the embedding of current character with the embeddings of neighbouring characters as the input of LSTM. In [11], LSTM is used to learn character representations and CRF is used to jointly decode the labels. These methods rely on a large number of labeled sentences to train CWS models and cannot exploit the useful information in Chinese dictionaries [17]. Since there are massive Chinese words which are scarce or absent in the labeled sentences, these neural CWS methods usually have difficulty in correctly segmenting sentences containing these words [17].

Recently, incorporating the dictionary information into neural Chinese word segmentation has attracted increasing attentions [14,17]. For example, Yang et al. [14] proposed to incorporate external information such as punctuation, automatic segmentation and POS data into neural CWS via pretraining. However, the useful information in Chinese dictionaries is not considered in their method. Zhang et al. [17] proposed to incorporate the dictionary information into an LSTM based neural CWS method via feature engineering. They used several handcrafted templates to build an additional feature vector for each character using the dictionary and the neighbouring characters. These additional feature vectors are fed to another LSTM network to learn additional character representations. However, designing these handcrafted feature templates needs a lot of domain knowledge. In addition, more model parameters are introduced in their method, making it more difficult to train neural CWS model especially when training data is insufficient. Different from [17], our method to incorporate dictionary information into neural CWS can be trained in an end-to-end manner and does not need manual feature engineering. Experimental results show that our approach can achieve better performance than the method in [17].

3 Our Approach

In this section we first present the basic neural architecture for Chinese word segmentation used in our approach. Then, we introduce our methods of incorporating dictionary information for neural CWS.

3.1 Basic Neural Architecture

Following many previous works [3,17], in this paper we model Chinese word segmentation as a character-level sequence labeling problem. For each character in a sentence, our model will assign one of the tags in a predefined tag set to it, indicating its position in a word. We use the *BMES* tagging scheme, where B, M and E mean the beginning, middle and end position in the word, and S represents single character word.

The basic neural architecture for CWS used in our approach is CNN-CRF. This neural architecture contains three main layers. The first layer is the character embedding layer. In this layer, the input sentence is converted to a sequence of vectors. Denote the input sentence as $\mathbf{x} = [c_1, c_2, ..., c_M]$, where M is the sentence length and c_i is the i-th character in this sentence. After the embedding layer, the input sentence will become $\mathbf{x} = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_M]$, where $\mathbf{c}_i \in \mathcal{R}^D$ is the embedding of character c_i and D is the embedding dimension.

The second layer is the CNN layer. Previous studies show that local context information is important for Chinese word segmentation [1,14]. In addition, many researchers have shown that CNN is effective in capturing local context information [7,12,18]. Motivated by these observations, we use CNN to learn the contextual representations of characters for CWS. Denote $\mathbf{w} \in \mathcal{R}^{KD}$ as the parameter of a filter with kernel size K, then the hidden representation of the i-th character generated by this filter is formulated as follows:

$$h_i = f(\mathbf{w}^T \times \mathbf{c}_{i-\lceil\frac{k-1}{2}\rceil:i+\lfloor\frac{k-1}{2}\rfloor} + b), \tag{1}$$

where $\mathbf{c}_{i-\lceil\frac{k-1}{2}\rceil:i+\lfloor\frac{k-1}{2}\rfloor}$ is the concatenation of the embeddings of neighbouring characters, f is the ReLU function, and $\mathbf{w}$ and b are the parameters of the filter.

Multiple filters with different kernel sizes are used. The final hidden representation of the i-th character is the concatenation of the output of all filters at this position, which is denoted as $\mathbf{h}_i \in \mathcal{R}^F$ (F is the number of filters).

The third layer is the CRF layer. In Chinese word segmentation there are usually strong dependencies among neighbouring tags [3]. For example, the tag M cannot follow tag S or E. Following many previous works on CWS [11,17], we use CRF to capture the dependencies among neighbouring tags. Denote the input sentence as $\mathbf{x} = [c_1, c_2, ..., c_M]$, and the predicted tag sequence as $\mathbf{y} = [y_1, y_2, ..., y_M]$, then the score of this prediction is formulated as:

$$g(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{M} (S_{i,y_i} + A_{y_{i-1},y_i}), \tag{2}$$

where S_{i,y_i} is the score of assigning tag y_i to the i-th character, and A_{y_{i-1},y_i} is the score of jumping from tag y_{i-1} to tag y_i. In our approach, S_i is defined as:

$$S_i = \mathbf{W}^T \mathbf{h}_i + \mathbf{b}, \tag{3}$$

where $\mathbf{h}_i$ is the hidden representation of the i-th character learned by the CNN layer, and $\mathbf{W} \in \mathcal{R}^{F \times T}$ and $\mathbf{b} \in \mathcal{R}^T$ (T is the size of the tag set) are the parameters for character score prediction. In CRF, the probability of sentence $\mathbf{x}$ having tag sequence $\mathbf{y}$ is defined as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{\exp(g(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathcal{Y}(\mathbf{x})} \exp(g(\mathbf{x}, \mathbf{y}'))}, \tag{4}$$

where $\mathcal{Y}(\mathbf{x})$ is the set of all possible tag sequences of sentence $\mathbf{x}$.

Then the loss function can be formulated as:

$$\mathcal{L} = -\sum_{i=1}^{N} \log(p(\mathbf{y}_i|\mathbf{x}_i)), \tag{5}$$

where N is the number of labeled sentences for training, and $\mathbf{y}_i$ is the ground-truth tag sequence of the i-th sentence.

For prediction, given a sentence $\mathbf{x}$ to be segmented, the predicted tag sequence $\mathbf{y}^\star$ is the one with the highest likelihood:

$$\mathbf{y}^\star = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} p(\mathbf{y}|\mathbf{x}). \tag{6}$$

We use Viterbi algorithm to solve the decoding problem in Eq. (6).

3.2 Incorporating Dictionary Information for Neural CWS

Existing neural CWS methods usually rely on a large number of labeled sentences for model training. Researchers have found that the neural models trained on labeled sentences usually have difficulties in segmenting sentences which contain OOV or rarely appearing words [17]. For example, a Chinese sentence is

"人工智能最近很火" (Recently AI is hot). Its ground-truth segmentation is "人工智能/最近/很火". However, if "人工智能" (AI) does not appear in the labeled data or only appears for a few times, then there is a large probability that this sentence will be segmented into "人工/智能/最近/很火", since "人工" and "智能" are both popular words which may frequently appear in the labeled data. Luckily, many of these rare words are included in Chinese dictionary. If the neural model is aware of that "人工智能" is a Chinese word, then it can better segment the aforementioned sentence. Thus, dictionary information has the potential to improve the performance of neural CWS methods.

In this paper we propose two methods for incorporating dictionary information into training neural CWS models. Next we will introduce them in detail.

Pseudo Labeled Data Generation. Our first method for incorporating dictionary information into neural CWS model training is based on pseudo labeled data generation. More specifically, given a Chinese dictionary which contains a list of Chinese words, we randomly sample U words and use them to form a pseudo sentence. For example, assuming that three words "很火", "最近" and "人工智能" are sampled, then a pseudo sentence "很火最近人工智能" can be built. Since the boundaries of these words are already known, the tag sequence of the generated pseudo sentence can be automatically inferred. For instance, the tag sequence of aforementioned pseudo sentence is "B/E/B/E/B/M/M/E" under the *BMES* tagging scheme. Then we repeat this process until N_p pseudo labeled sentences are generated. These pseudo labeled sentences are added to labeled data set to enhance the training of neural CWS model.

Since the pseudo labeled sentences may have different informativeness from the manually labeled sentences, we assign different weights to the loss on these two kinds of training data, and the final loss function is formulated as:

$$\mathcal{L} = -\sum_{i=1}^{N} \log(p(\mathbf{y}_i|\mathbf{x}_i)) - \lambda_1 \sum_{i=1}^{N_p} \log(p(\mathbf{y}_i^s|\mathbf{x}_i^s)), \tag{7}$$

where $\mathbf{x}_i^s$ and $\mathbf{y}_i^s$ represent the i-th pseudo labeled sentence and its tag sequence, and λ_1 is a non-negative coefficient.

Multi-task Learning. Our second method for incorporating dictionary information into neural CWS model training is based on multi-task learning. In this method, we design an additional task, i.e., word classification, which means classifying a sequence of Chinese characters based on whether it can be a Chinese word. For example, the character sequence "人工智能" will be classified to be true, while the character sequence "人重智新" will be classified to be false. The positive samples are obtained from a Chinese dictionary. The negative samples are obtained via randomly sampling a word from the dictionary, and then each character in this word will be randomly replaced by a random selected character with a probability p. This step is repeated multiple times until a predefined

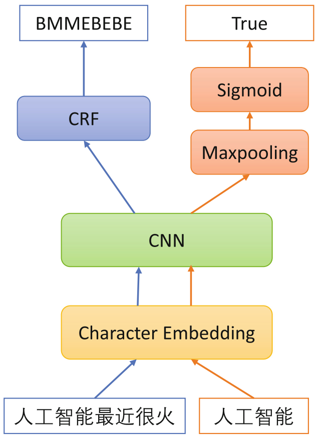


Fig. 1. Our proposed framework for jointly training CWS and word classification models. The left part is for CWS and the right part is for word classification.

number of negative samples are obtained. We use a neural method for the word classification task, whose architecture is similar with the CNN-CRF architecture for CWS, except that the CRF layer is replaced by a max-pooling layer and a sigmoid layer for binary classification. The loss function of the word classification task is formulated as:

$$\mathcal{L} = \sum_{i=1}^{N_w} \log(1 + e^{-y_i s_i}), \tag{8}$$

where N_w is the number of training samples for word classification, s_i is the predicted score of the i-th sample, and y_i is the word classification label which can be 1 or -1 (1 represents true and -1 represents false).

Motivated by multi-task learning, we propose a unified framework to jointly train the Chinese word segmentation model and the word classification model, which is illustrated in Fig. 1. In our framework, the CWS model and the word classification model share the same embedding layer and CNN layer. In this way, these two layers can better capture the word information in Chinese dictionary via jointly training with the word classification task, and the performance of CWS can be improved. In model training we assign different weights to the loss of these two tasks, and the final loss function is:

$$\mathcal{L} = -(1 - \lambda_2) \sum_{i=1}^{N} \log(p(\mathbf{y}_i|\mathbf{x}_i)) + \lambda_2 \sum_{i=1}^{N_w} \log(1 + e^{-y_i s_i}), \tag{9}$$

where λ_2 is a coefficient ranging from 0 to 1.

4 Experiment

4.1 Dataset

In our experiments we used two benchmark datasets released by the third international Chinese language processing bakeoff[1] [8]. The detailed statistics of these two datasets are summarized in Table 1. We used the last 10% data of the training set as development set.

Table 1. The statistics of datasets.

Dataset		#Sentence	#Word	#Character	OOV Rate
MSRA	Train	46.3K	1.27M	2.17M	-
	Test	4.4K	0.10M	0.17M	3.4%
UPUC	Train	18.8K	0.51M	0.83M	-
	Test	5.1K	0.15M	0.26M	8.8%

4.2 Experimental Settings

The character embeddings used in our experiments were pretrained on the Sogou news corpus[2] using the word2vec[3] tool. The dimension of character embedding is 200. We used 400 filters in the CNN layer and the kernel sizes of these filters range from 2 to 5. Rmsprop [4] was used as the algorithm for neural model training. The learning rate was set to 0.001 and the batch size was 64. Dropout was applied to the embedding layer and the CNN layer. The dropout rate was set to 0.3. We use early stopping strategy. When the loss on the development set doesn't reduce after 3 consecutive epochs, the training is stopped. We repeated each experiment for 5 times and reported the average results.

4.3 Performance Evaluation

In this section we compare our approach with several baseline methods. These baseline methods include: (1) Chen et al. [3], a LSTM based CWS method which also considers local contexts; (2) LSTM-CRF, a popular neural CWS method based on the LSTM-CRF architecture [11,17]; (3) CNN-CRF, a neural CWS method based on the CNN-CRF architecture, which is the basic model for our approach; (4) Zhang et al. [17], a neural CWS method which can incorporate dictionary information via feature templates. In order to evaluate the performance of different methods under different amounts of labeled data, we randomly sampled different ratios of labeled data for training. The experimental results are summarized in Tables 2 and 3. According to Tables 2 and 3, we have two observations.

[1] http://sighan.cs.uchicago.edu/bakeoff2006/download.html.
[2] http://www.sogou.com/labs/resource/ca.php.
[3] https://code.google.com/archive/p/word2vec/.

Table 2. The performance of different methods on the *MSRA* dataset. *P*, *R* and *F* represent precision, recall and Fscore respectively. *Ours_Pseudo* represents our approach based on pseudo labeled data generation, and *Ours_Multi* represents our approach based on multi-task learning.

	1%			10%			100%		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Chen et al. [3]	75.50	75.80	75.64	87.71	86.22	86.96	94.24	93.35	93.80
LSTM-CRF	75.88	74.86	75.36	85.52	84.81	85.16	94.26	93.29	93.78
CNN-CRF	75.59	74.43	75.00	89.72	89.14	89.43	95.03	94.53	94.78
Zhang et al. [17]	75.75	75.95	75.85	89.52	89.01	89.27	95.71	95.41	95.56
Ours_Pseudo	80.58	77.97	79.25	90.49	89.59	90.04	95.36	94.71	95.03
Ours_Multi	78.47	77.31	77.88	89.91	89.27	89.59	95.10	94.50	94.80

Table 3. The performance of different methods on the *UPUC* dataset.

	5%			25%			100%		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Chen et al. [3]	82.31	82.60	82.44	88.00	89.90	88.94	90.79	92.92	91.84
LSTM-CRF	81.08	80.88	80.98	86.76	88.40	87.57	91.39	92.58	91.98
CNN-CRF	82.44	84.50	83.46	89.95	91.57	90.75	92.22	93.84	93.02
Zhang et al. [17]	83.38	84.98	84.17	89.93	91.41	90.66	92.60	93.89	93.24
Ours_Pseudo	87.37	86.56	86.97	90.97	92.04	91.50	92.77	94.09	93.43
Ours_Multi	84.59	86.22	85.40	90.43	91.68	91.05	92.35	93.93	93.13

First, both of our approaches perform better than various neural CWS methods which do not consider dictionary information, and the performance advantage becomes larger when training data is insufficient. This result validates that by incorporating the dictionary information our approaches can effectively improve the performance of neural CWS. This is because there are many words which do not appear or rarely appear in the training data, and the neural CWS models which are trained purely on labeled data usually have difficulty in segmenting sentences containing these words. Many of these words are usually included in Chinese dictionaries, and exploiting the useful information in dictionaries can help the neural CWS model better recognize these words.

Second, although the method proposed in [17] can also incorporate the dictionary information for CWS, our approaches usually can outperform it, especially when training data is insufficient. This result shows that our approaches are more appropriate for incorporating dictionary information for CWS than the method proposed in [17]. This is maybe because in [17] the feature templates for incorporating dictionary information are manually designed, which may not be optimal.

In addition, in [17] an additional LSTM network is used to learn character representations from these dictionary based features. Thus, more model parameters are incorporated, making it more difficult to train the CWS model especially when training data is insufficient. Our approaches do not rely on feature engineering and the additional model parameters introduced in our approaches are limited. Thus, our approach can achieve better performance than [17].

4.4 Influence of Dictionary

In this section we conducted several experiments to explore the influence of the type and the size of Chinese dictionary on the performance of our approach.

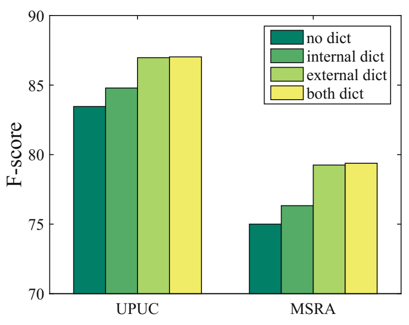


Fig. 2. The influence of dictionary type.

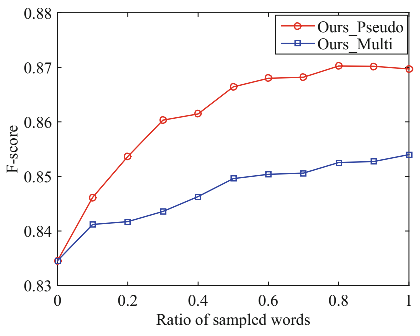


Fig. 3. The influence of dictionary size.

First, we explore the influence of dictionary type. In previous section, the Chinese dictionary used in our approach is the Sogou Chinese Dictionary, which can be regarded as an external dictionary. We also built an internal dictionary using the words appearing in the training data. The results of our approach without any dictionary, with only internal dictionary, with only external dictionary, and with both dictionaries are summarized in Fig. 2. We randomly sampled 5% training data of *UPUC* dataset and 1% training data of *MSRA* dataset for model training.

According to Fig. 2, with external dictionary our approach can improve the performance of CWS. In addition, our approach can also improve the performance with only internal dictionary. This result is promising, since the internal dictionary is built on the words appearing in training data and no external resource is involved. In addition, incorporating both internal and external dictionaries can further improve the performance of our approach, which indicates that these two dictionaries contain complementary information.

Next, we explore the influence of the dictionary size on the performance of our approach. We randomly sampled different numbers of words from the Sogou dictionary, and the experimental results on *UPUC* dataset are summarized in Fig. 3.

From Fig. 3, with the size of dictionary grows the performance improves. This result is intuitive since when a dictionary contains more words it can have a better coverage of the Chinese words, and our approach can benefit from this by incorporating the useful information in these words into training neural CWS model.

4.5 The Influence of Parameters

There are two most important parameters in our approaches. The first one is λ_1, which controls the relative importance of pseudo labeled samples. The second one is λ_2, which controls the relative importance of word segmentation task. The influence of these parameters on the performance of our approaches is illustrated in Figs. 4 and 5.

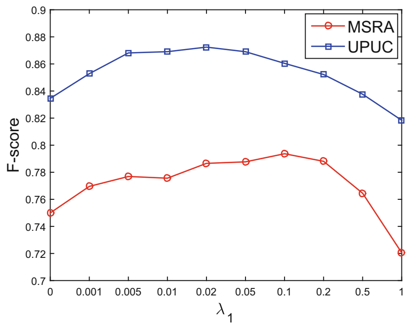


Fig. 4. The influence of λ_1.

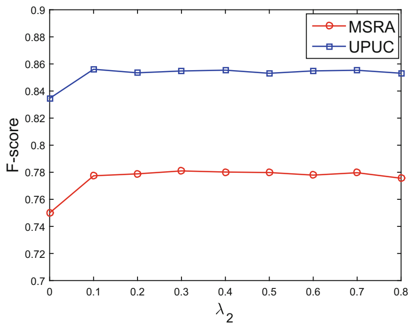


Fig. 5. The influence of λ_2.

From Figs. 4 and 5 we can see that when λ_1 and λ_2 are too small, the performance of our approach is not optimal, and improves as λ_1 and λ_2 increase. This is because when these parameters are too small, the useful information in the dictionary is not fully exploited. However, when λ_1 and λ_2 become too large, the performance of our approach decreases. This is because in these cases the pseudo labeled samples and the word classification task are over-emphasized. Accordingly, the manually labeled samples and the CWS task are not fully respected. Thus, a moderate value is most appropriate for λ_1 and λ_2.

4.6 Case Study

In this section we conducted several case studies to explore why our approach can improve the performance of Chinese word segmentation via incorporating the dictionary information. Several segmentation results of our approach without dictionary (i.e., the CNN-CRF method), with internal dictionary and with external dictionary are shown in Table 4. For illustration purpose, we only show the results of our approach based on pseudo labeled data generation.

Table 4. Several Chinese word segmentation examples.

	Example 1	Example 2
Original	5 名男子和被害人有恩怨	警方一口气带回了 5 0 多人
CNN-CRF	5 /名/男子/和/被/害/人/有/恩怨	警方/一/口/气/带回/了/ 5 0 多/人
+Internal dictionary	5 /名/男子/和/被/害/人/有/恩怨	警方/一口气/带回/了/ 5 0 多/人
+External dictionary	5 /名/男子/和/被害人/有/恩怨	警方/一口气/带回/了/ 5 0 多/人

According to Table 4, after incorporating the dictionary information, our approach can correctly segment many sentences where the basic CNN-CRF method has difficulties. For instance, in the first example, the true segmentation of "被害人" is "被害人". However, CNN-CRF incorrectly segments it into "被/害/人", because "被害人" is an OOV word which does not appear in training data. Our approach with external dictionary can correctly segment this sentence because the word "被害人" is in the external dictionary and our approach can fully exploit this useful information. In the second example, CNN-CRF incorrectly segments "一口气" into "一/口/气", because "一口气" is an rare word which only appears 2 times in the training data which is difficult for neural CWS model to segment it. Since this word is in both internal and external dictionaries, our approach with either dictionary can correctly segment this sentence. Thus, these results clearly show that incorporating dictionary information into training neural CWS methods is beneficial.

5 Conclusion

In this paper we present two approaches for incorporating the dictionary information into neural Chinese word segmentation. The first one is based on pseudo labeled data generation, where pseudo labeled sentences are generated by combining words randomly sampled from dictionary. The second one is based on multi-task learning, where we design a word classification task and using the dictionary to build labeled samples. We jointly train the Chinese word segmentation and the word classification task via sharing the same network parameters. Experimental results on two benchmark datasets show that our approach can effectively improve the performance of Chinese word segmentation, especially when training data is insufficient.

References

1. Cai, D., Zhao, H., Zhang, Z., Xin, Y., Wu, Y., Huang, F.: Fast and accurate neural word segmentation for Chinese. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, vol. 2, pp. 608–615 (2017)
2. Chen, W., Zhang, Y., Zhang, M.: Feature embedding for dependency parsing. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 816–826 (2014)

3. Chen, X., Qiu, X., Zhu, C., Liu, P., Huang, X.: Long short-term memory neural networks for Chinese word segmentation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1197–1206 (2015)
4. Dauphin, Y., de Vries, H., Bengio, Y.: Equilibrated adaptive learning rates for non-convex optimization. In: Advances in Neural Information Processing Systems, pp. 1504–1512 (2015)
5. Eddy, S.R.: Hidden markov models. Curr. Opin. Struct. Biol. **6**(3), 361–365 (1996)
6. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: probabilistic models for segmenting and labeling sequence data (2001)
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
8. Levow, G.A.: The third international Chinese language processing bakeoff: word segmentation and named entity recognition. In: Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pp. 108–117 (2006)
9. Luo, W., Yang, F.: An empirical study of automatic Chinese word segmentation for spoken language understanding and named entity recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 238–248 (2016)
10. Peng, F., Feng, F., McCallum, A.: Chinese segmentation and new word detection using conditional random fields. In: Proceedings of the 20th International Conference on Computational Linguistics, p. 562. Association for Computational Linguistics (2004)
11. Peng, N., Dredze, M.: Multi-task domain adaptation for sequence tagging. In: Proceedings of the 2nd Workshop on Representation Learning for NLP, pp. 91–100 (2017)
12. dos Santos, C., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pp. 69–78 (2014)
13. Xue, N.: Chinese word segmentation as character tagging. Int. J. Comput. Linguisti. Chin. Lang. Process. **8**(1), 29–48 (2003). Special Issue on Word Formation and Chinese Language Processing
14. Yang, J., Zhang, Y., Dong, F.: Neural word segmentation with rich pretraining. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, pp. 839–849 (2017)
15. Zhang, M., Zhang, Y., Che, W., Liu, T.: Chinese parsing exploiting characters. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics. Long Papers, vol. 1, pp. 125–134 (2013)
16. Zhang, M., Zhang, Y., Fu, G.: Transition-based neural word segmentation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 421–431 (2016)
17. Zhang, Q., Liu, X., Fu, J.: Neural networks incorporating dictionaries for Chinese word segmentation (2018)
18. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Advances in Neural Information Processing Systems, pp. 649–657 (2015)
19. Zhao, H., Huang, C.N., Li, M., Lu, B.L.: Effective tag set selection in Chinese word segmentation via conditional random field modeling. In: Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation, pp. 87–94 (2006)
20. Zheng, X., Chen, H., Xu, T.: Deep learning for Chinese word segmentation and pos tagging. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 647–657 (2013)

Recognizing Macro Chinese Discourse Structure on Label Degeneracy Combination Model

Feng Jiang, Peifeng Li(✉), Xiaomin Chu, Qiaoming Zhu, and Guodong Zhou

School of Computer Science and Technology, Soochow University, Suzhou, China
{fjiang, xmchu}@stu.suda.edu.cn, {pfli, qmzhu, gdzhou}@suda.edu.cn

Abstract. Discourse structure analysis is an important task in Natural Language Processing (NLP) and it is helpful to many NLP tasks, such as automatic summarization and information extraction. However, there are only a few researches on Chinese macro discourse structure analysis due to the lack of annotated corpora. In this paper, combining structure recognition with nuclearity recognition, we propose a Label Degeneracy Combination Model (LD-CM) to find the solution of structure recognition in the solution space of nuclearity recognition. Experimental results on the Macro Chinese Discourse TreeBank (MCDTB) show that our model improves the accuracy by 1.21%, compared with the baseline system.

Keywords: Label degeneracy · Combination model
Macro discourse structure

1 Introduction

Nowadays, the focus of most previous works on Natural Language Processing (NLP) has shifted from word to larger semantic levels, such as sentence and event. This trend makes discourse structure analysis more important because it is the foundation of many discourse-based NLP tasks. Discourse refers to a series of clauses, sentences or paragraphs as a whole [1]. It includes not only the text sequence but also the structural and logical relationships among them. Commonly, discourse structure analysis is divided into micro and macro structure analysis. The former studies the intra- or inter-sentence relationship, while the latter studies the discourse relationships among sentence groups, paragraphs and chapters [2], which pays attention to understanding the full text from higher-level semantics.

Macro discourse analysis uses paragraphs as elementary discourse units, and constructs a discourse structure tree between the paragraphs. It is a challenging task due to there is no connective between macro discourse units, and the length of the discourse unit is longer. In general, macro discourse analysis includes three tasks: structure recognition, nuclearity recognition, and relationship recognition. Structure recognition

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 92–104, 2018.
https://doi.org/10.1007/978-3-319-99501-4_8

is to identify whether there is a relationship between adjacent discourse units. Nuclearity recognition is to identify which is more important among discourse units. Moreover, relation recognition is to identify the logical relationship between discourse units. These three tasks constitute the analysis of macro discourse structure, and ultimately build a discourse structure tree of an article.

Table 1 is a sample text including a title and five paragraphs. Figure 1 is its macro discourse structure tree, in which the leaf nodes are paragraphs, and the parent nodes, the larger discourse units, connect the adjacent child nodes. A directed edge connected parent and child nodes, with the arrow's edge pointing to the important child node and the non-arrowed edge pointing to the secondary child node. In Fig. 1, P_2, P_3 and P_4 form a *Joint* relation, and they are equally important; P_1 and $DU_{2\text{-}4}$ form a *Commentary*

Table 1. Contents of chtb_0156.

Asia's Largest Fluorine-Free Coolant Manufacturing Factory Entered Production In Tianjin (亚洲最大无氟制冷剂生产厂在天津投产)
(P_1) Asia's largest fluorine-free coolant manufacturing base - GreenKel coolant (China) limited company completed … in Tianjin's development district. (亚洲最大的无氟制冷剂生产基地——格林柯尔制冷剂（中国）有限公司日前在天津开发区建成投产。)
(P_2) With headquarters established in Canada, GreenKel is a … company manufacturing and selling fluorine-free coolant. (总部设在加拿大的格林柯尔集团是生产和销售无氟新型制冷剂的大型跨国集团公司。) GreenKel coolant (China) limited company is … company, China's Tianjin … office and … Company, ltd. with a total investment value of … dollars, among which the foreign investment value is … dollars. (格林柯尔制冷剂（中国）有限公司是格林柯尔集团北美公司与中国天津开发区总公司和中国南方证券有限公司合建的合资企业，总投资额五千万美元，其中外方投资额四千一百八十五万美元。)
(P_3) The company occupies … meters, and introduced … advanced production line and testing equipment, to produce … fluorine-free coolants. (公司占地七万平方米，引进全套欧美先进的生产线和检测设备，生产格林柯尔无氟新型系列制冷剂。)
(P_4) The currently completed first phase project annual production volume that … , with annual production values reaching … dollars , is the fluorine-free coolant production base that … the most advanced equipment and technology in Asia .(目前竣工的一期工程年产量可达一万吨，全部出口外销，年产值达二亿美元，是目前亚洲生产规模最大，设备和技术最先进的无氟制冷生产基地。)
(P_5) The completion and entering into production of this production base, signifies that … the world's leading ranks. (该生产基地的建成投产，标志着中国无氟制冷生产跨进世界领先行列。)

relation, in which P_1 is more important; $DU_{1\text{-}4}$ and P_5 form an *Evaluation* relation, in which $DU_{1\text{-}4}$ is more important. A good article always has a good discourse structure tree. If we can construct a discourse structure tree, it will play an active role in downstream tasks such as automatic summarization and information extraction. In the task of automatic summarization, after constructing a macro structure tree, we can follow the arrow from the top down to the leaf node to get a more natural summary. For example, according to the Fig. 1, chtb_0156's abstract in Table 1 is the topic sentence of P_1.

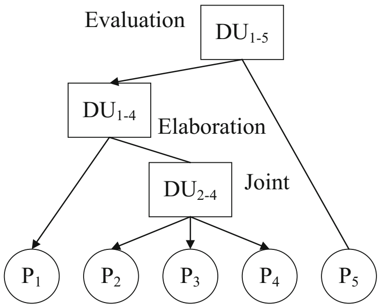


Fig. 1. The macro structure tree of chtb_0156.

Discourse structure recognition is the first step and the most important step in the tasks of discourse structure analysis. In this paper, we first use the features of structure information and semantic macro-information to form a combination model from different views. On this basis, it uses the label degeneracy to find out the solution of structure recognition in the solution space of the nuclearity recognition to obtain more detailed feature expression. Finally, we use the minimum probability post-editing to ensure that the model can automatically build a complete macro structure tree. The experimental results on MCDTB, a RST style macro Chinese discourse TreeBank, justify the effectiveness of our model.

2 Related Work

Discourse structure analysis is divided into micro and macro structure analysis. In micro discourse analysis, Hernault et al. [3] was the first to implement a complete discourse analyzer in English, and used SVM with rich textual features (including structure features, syntactic features and dominant sets) in structure recognition. Recent studies followed this research line and focused on effective features. Joty et al. [4] and Feng et al. [5] used sequence labeling instead of classification. The former used Dynamic Conditional Random Field (DCRF) model combining structure recognition with relationship recognition, and the latter used Conditional Random Field (CRF) model with post-editing avoiding meaningless sequence labeling. Li et al. [6] and Li et al. [7] used the recursive neural network and attention-based hierarchical neural network with the distribution representation of text for recognizing structure, respectively. Besides, Feng et al. [9] and Wang et al. [10] introduced N-gram and the tree features to recognize structure on the previous work [3, 8]. Due to the lack of unified theories and corpus, there are few researches on Chinese discourse analysis. Lv et al. [11] using Chinese FrameNet (CFN) and Li [1] using Connective-driven Dependency Tree (CDT) with maximum entropy model attempted to recognize the discourse structure on their self-built corpus, respectively.

There is less research at the macro level due to lack of corpus. In English, Sporleder et al. [12] used a maximum entropy model to recognize the macro discourse structure of RST-DT after correction and clipping. In Chinese, Jiang et al. [13] used the maximum entropy model to identify the macro discourse nuclearity with annotated discourse structures in MCDTB, the only one Chinese corpus labeled on the macro level.

3 Label Degeneracy Combination Model

Label Degeneracy Combination Model (LD-CM) is shown in Fig. 2. We first use structural information to train the Structure Model, and use semantic and macro information to train the Semantic and Macro Information Model to recognize the macro discourse nuclearity. Then, we use these models to form a combination model. The combination model can learn different features from different views, and decode the prediction results in a unified way to compensate for the bias caused by a single model, thereby improving system performance.

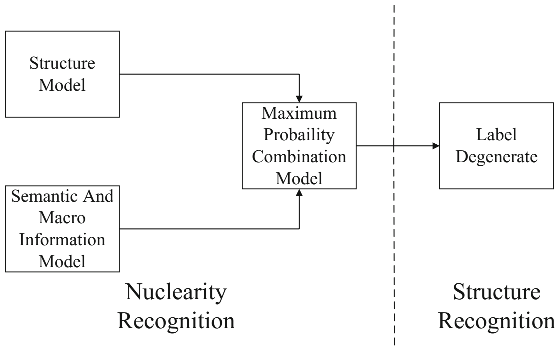


Fig. 2. The label degeneracy combination model.

Finally, we use the label degeneracy to map the nuclearity results to reveal the macro-structure. In previous work [1, 5], structure recognition and nuclearity recognition were mainly cascading tasks. That is to say, structure recognition is binary classification (whether the adjacent discourse units can be merged) and nuclearity is three-classification (*Nucleus Ahead, Nucleus Behind* and *Multi-Nucleus*) on the merging part that is obtained from structure recognition. We improve the nuclearity recognition and regard the *No-Relationship* as the fourth class. In this way, there is a mapping relationship between structure recognition and nuclearity recognition: the adjacent discourse units should be merged (*Nucleus Ahead, Nucleus Behind* and *Multi-Nucleus*) or not (*No-Relationship*). Therefore, it can make the model capture more detailed feature expression from three classes of nuclearity to be distinguished from *No-Relationship*, which helps to recognize structure.

3.1 Sequence Labeling on Chinese Discourse Structure Recognition

In Chinese discourse structure recognition, Li et al. [1] and Lv et al. [11] both regarded it as a classification problem, that is, to determine whether two given discourse units should be merged. However, their methods have certain disadvantages, such as the inability to consider contextual information and to recognize multi-relationship structure.

Therefore, we regard this problem as sequence labeling following Joty et al. [4] and Feng et al. [5], which has achieved good performance in English discourse structure recognition. For the task of predicting macro discourse structure in Chinese, the sequence labeling model has the following advantages: (1) it can consider the context information; (2) it can keep the original discourse structure; (3) it can achieve a balance between greedy algorithm and global optimization.

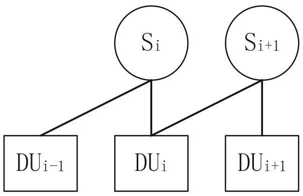


Fig. 3. Macro-structure sequence labeling model.

As shown in Fig. 3, DU_i is the ith discourse unit and S_i is the structure label whether the ith discourse unit is merged with the previous DU_{i-1}. On the one hand, when annotating S_i, we can take into account the information of DU_{i-1} and DU_{i+1}, so as to increase the accuracy of discourse structure recognition. On the other hand, there is no need to convert multi-relationship structure. For example, when DU_{i-1}, DU_i and DU_{i+1} form a multi-relationship structure, we only need to consider whether DU_i and DU_{i-1} should be merged and whether DU_{i+1} and DU_i should be merged. When both of them labeled as merged, three successive discourse units form a multi-relationship structure.

3.2 Feature Combination

Table 2 shows the features used in the previous studies on RST-DT, an English discourse corpus, where Sporleder et al. [12] focused on macro structure analysis and the rest focused on micro level. Table 2 illustrates that syntax information and dominant set are very useful in microstructure analysis. However, we cannot introduce them to macro structure analysis because the elementary of macro discourse unit is paragraph

Table 2. Statistics of the features used in recent studies.

Features	Sporleder [12]	Hernault [3]	Feng [9]	Joty [4]	Feng [5]	Wang [10]
Location information and distance information	√	√		√	√	√
Structure information						√
Syntax information, dominance set		√	√	√	√	√
N-gram			√	√	√	√
Number of paragraphs and sentences	√	√		√	√	√
Tree features				√	√	√
Status information						√
Word co-occurrence, semantic similarity	√		√			
Punctuation, tense	√					
Cue words, lexical chains.	√			√	√	
Contextual features				√	√	
Entity transfer					√	

that does not have syntax information and dominant set. Therefore, we select the other common features used in the previous studies as the structural features as follows:

- The position of the beginning and the end of a discourse unit;
- The number of sentences and the number of paragraphs contained in a discourse unit;
- The comparison of the number of sentences in a discourse unit to its previous unit;
- The comparison of the number of paragraphs in a discourse unit to its previous unit.

Inspired by the distributed representation of text [14], we improved the “word co-occurrence” feature adopted by Sporleder et al. [12] and used semantic similarity to measure the semantic connection between two discourse units. We used Word2Vec model to train word vectors on CTB8.0, and used the method proposed by Xu [15] to calculate semantic similarity.

In macro structure, it is not possible to display the connectives explicitly, but there may be a connective in the first sentence of a discourse unit. Therefore, we regard the first connective and its part of speech in the discourse unit as one of the features. In addition, because macro-structure analysis focuses more on macroscopic understanding, we add some macro information as features, such as whether a discourse unit is a leaf node and whether a discourse unit was merged at the previous round. Due to differences in language characteristics and discourse unit granularity, we do not use tense and N-gram features. Finally, we select the semantic and macro information features as follows:

- The semantic similarity between a discourse unit and its previous unit;
- The connective of the first sentence and it’s part of speech in a discourse unit;

- Whether a discourse unit is a leaf node;
- Whether a discourse unit was merged in the previous round.

We first use the structural features to train Structure Model and use semantic and macro information features to train Semantic and Macro Information Model on the training set, respectively. Then we use the above two models to predict nuclearity of two given discourse units in the test set, respectively. For each nuclearity label in the prediction sequence, the one with the highest probability from the above two models is selected as the final result.

3.3 Label Degeneracy

Previous studies have considered nuclearity recognition as part of relationship recognition following the structure recognition. Different from them, we use label degeneracy to recognize the structure after nuclearity recognition. In nuclearity recognition, we added *No-Relationship* as the fourth class to form a model for recognizing structure and nuclearity simultaneously. In this way, there is a mapping relationship between structure recognition and nuclearity recognition: the adjacent discourse units should be merged *(Nucleus Ahead*, *Nucleus Behind* and *Multi-Nucleus*) or not (*No-Relationship*).

Our label degeneracy approach is as follows: when DU_i is merged with the previous discourse unit DU_{i-1}, the label S_i is 1, otherwise 0. In nuclearity recognition, when DU_i isn't merged with DU_{i-1}, the label N_i is 0. When DU_i is less important than DU_{i-1} (*Nucleus Ahead*), N_i is 1; when DU_i is more important than DU_{i-1} (*Nucleus Behind*), N_i is 2; when DU_i and DU_{i-1} are equally important (*Multi-Nucleus*), N_i is 3. The degenerate mapping relationship between the labels of the discourse structure recognition and the nuclearity recognition is shown in Eq. (1).

$$S_i = LD(N_i) = \begin{cases} 0, & N_i = 0 \\ 1, & N_i = 1, 2, 3 \end{cases} \tag{1}$$

Therefore, it can make the model capture more detailed feature expression from three classes of nuclearity to be distinguished from *No-Relationship*, which helps to recognize structure. Jiang's experiment [13] shows that the main mistakes of nuclearity recognition is that it always recognizes *Nuclearity Behind* as other two types by mistake because this class is scarce. In contrast, the other two relations *Nuclearity Ahead* and *Multi-Nucleus* can be well recognized. This result means that their models can capture differences between different nuclearity classes, making the characteristics of the merged part more detailed (not only labeled 1 in structure recognition, but also labeled 1, 2 and 3 in nuclearity recognition), thereby enhancing distinction with *No-Relationship* (labeled 0 in structure and nuclearity).

For example, if one relationship is labeled as 3 (*Multi-Nucleus*), usually it is a *Coordination* relation which may have more than two children. In this way, the label 3 (*Multi-Nucleus*) can strengthen the feature expression of this discourse structure (the adjacent discourse units should be merged). So that it is not only distinguished from *No-relationship* (label 0), but also distinguishable from the *Nuclearity Ahead* (label 1) and *Nuclearity Behind* (label 2), thus improving the accuracy of the structure recognition. In addition, even if the error is confused with three nuclearity classes (*Nuclearity Ahead, Nuclearity Behind* and *Multi-Nucleus*), label degeneracy can also make it not reduce the accuracy of the result.

4 Experiment

In this section, we first introduce the experimental setting, and then report the experimental results and analysis.

4.1 Experimental Setup

There are few genuine macro discourse corpora, especially in Chinese language. The Macro Chinese Discourse TreeBank (MCDTB) [13] is the only available corpus in Chinese. Different from RST-DT, MCDTB adopts the three categories of 15 classes of discourse relationships formed by the improved CDTB [1] in macro-level discourse relationship annotation. In addition, MCDTB also includes macro discourse information, such as paragraph topic sentences, summaries, abstracts and pragmatic functions. MCDTB annotated 720 news reports (0001-0325, 0400-0454, 0500-0554, 0600-0885 and 0900-0931) from CTB 8.0. As shown in Table 3, the corpus has 3,981 paragraphs, and the average number of paragraphs is 5.53 per document. Besides, a document contains at most 22 paragraphs and at least 2 paragraphs. The corpus contains 8,319 sentences, 398,829 words, with an average of 553.93 words per document.

Table 3. The statistical of MCDTB.

Items	Number
#documents	720
#paragraphs	3,981
#paragraphs of the longest document	22
#paragraphs of the shortest document	2
#sentences	8,319
Average length (paragraphs/document)	5.53
Average length (sentences/paragraph)	2.09

We use Conditional Random Field (CRF) to train the Structure Model and the Semantic and Macro Information Model, with the parameter *C* of 4, the feature window of 3 and other parameters are default. There are 8,863 samples in total, including 3,261 positive samples and 5,602 negative samples. We use five-fold cross validation to ensure the objectivity of the experiments. In particular, according to the article lengths, we divided the articles of different lengths into five sets, so that the size of each set is almost the same.

Feng et al. [5] pointed out that there were two constraints used in the serialization labeling method of discourse structure recognition: discourse units cannot be consecutively merged and there is no merger at all in the sequence result. For the above two constraints, we take the measures as following:

First, we maintain the multi-relationship structure. In RST-DT, the multi-relationship structure is relatively small. According to statistics, 95% of annotated relations are binary. In MCDTB, the multi-relationship structure accounts for 8.6%, reaching 246 and the multi-relationship has 16 child nodes at most. Using the right-branching tree to replace the multi-relationship tree, the number of newly generated relations will account for 19.66%. At the same time, it also increases the distortion of the structure tree. We use the original structure to ensure the objectivity of recognition.

Second, we use the minimum probability post-editing to ensure that the second case does not occur. When there is no merger at all in the sequence result, we traverse the probability values of each label in prediction, and replace the label 0, that has minimum probability, with label 1 to ensure that the prediction meets the constraint condition.

4.2 Experimental Results and Analysis

We use the features of Sporleder [12] that can be migrated to Chinese (distance information, location information, number of sentences and paragraphs, sentences and paragraphs' number comparison, semantic similarity and connectives) to form Sporleder Liked Model as a benchmark system. As shown in Table 4, the performance of the Structure Model (M1) and the Semantic and Macro Information Model (M2) in discourse structure recognition is slightly worse than the baseline. However, composed of M1 and M2, the Combination Model (M1 + M2) obtained 77.56% accuracy, which is 0.5% higher than Sporleder Liked Model. Our model LD-CM outperforms all other six models in accuracy and this justifies the effectiveness of the combination model and label degeneracy.

Table 4. The comparison of each model's accuracy. Significant differences between SLM and LD-CM* (with $p < 0.01$).

#	Model	Accuracy
M1	Structure Model (structure recognition)	76.09%
M2	Semantic and Macro Information Model (structure recognition)	77.01%
M3	Sporleder Liked Model (SLM)	77.06%
M4	Combine Model (M1 + M2)	77.56%
M5	Label Degeneracy Model (based on M2)	77.78%
M6	Label Degeneracy Combination Model (LD-CM)	**78.27%***
M7	LD-CM with Post-Editing (LD-CMWP)	78.24%

As shown in Table 5, there is 14.11% difference prediction between the Structure Model (M1) and the Semantic and Macro Information Model (M2). This proves that different features can learn different views, thereby enhancing the ability of discourse structure recognition.

Table 5. The difference between the predictions of M1 and M2.

Prediction	Wrong (M2)	Correct (M2)
Wrong (M1)	16.39%	**7.51%**
Correct (M1)	**6.60%**	69.49%

Label Degeneracy Model (M5) based on M2 achieved a correct rate of 77.78%, which was 0.72% higher than that of Sporleder Liked Model and was about the same as Combine Model (M4). Therefore, we also statistics the differences between the predictions of M4 and M5 as shown in Table 6. There is 11.94% difference prediction between these two models. This proves that different methods have improvement in the different direction, so it is effective to combine them into a Label Degeneracy Combination Model (LD-CM).

Table 6. The difference between the predictions of M4 and M5.

Prediction	Wrong (M5)	Correct (M5)
Wrong (M4)	16.36%	**6.08%**
Correct (M4)	**5.86%**	71.70%

Label Degeneracy Combination Model (LD-CM) achieves an optimal value of 78.27%, 1.21% higher than Sporleder Liked Model. We use LD-CM with Post-Editing (LD-CMWP) to ensure that it can build a complete tree automatically with only reduce the accuracy slightly. To find out how LD-CM can improve the performance of macro-structure recognition, we analyzed the experimental results and found two phenomena:

1. LD-CM can overcome overfitting caused by a large amount of samples from short articles. LD-CM is calmer than other models for whether the third and fourth discourse unit should be merged. In those short articles, the third and fourth discourse unit (usually paragraphs) are often merged. However, in those long articles, they are acted as higher-level discourse unit in high probability.
2. LD-CM is better at recognizing complex higher-level macro structure. Figure 4 shows the prediction results of each model for the high-level macro-structure of chtb_0112. LD-CM predicts the structure correctly (a). Structures (b), (c), and (d) are the results of Sporleder Liked Model, Combination Model and Label Degeneracy Model, respectively. In complex high-level macro structure, more attention should be paid to the information in the front part of the article, due to the fact that if the article is longer, its topic is more dispersed, and its important part is more likely to gather in the front part. Sporleder Liked Model did not show learning this. Combination Model shows such a predictive behavior: if a discourse unit is not merged with other discourse unit before, it is more likely to be merged with the next discourse unit. However, it does not take into account that the important part in the front of the article. Label Degeneracy Model also captures discourse units, which are more likely to be merged in the front part of the article, but does not consider the fact that the behind part has been merged.

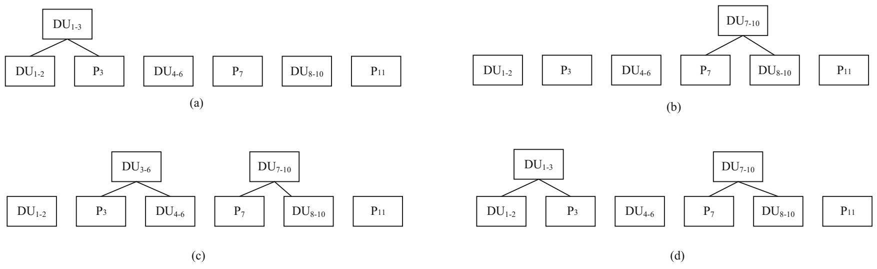


Fig. 4. High-level macro discourse structure of chtb_0112.

In addition, to study the performance of each model for articles on different lengths, we divide the corpus into two parts: the short articles whose paragraph lengths are less than or equal to 6 and the long articles whose paragraph lengths are more than 6. We model the corpora of 535 short articles and 185 long articles respectively and analyzed the results. Figure 5 shows the performance on different paragraph lengths. In those short articles, compared with the benchmark system M3, Combination Model (M4) and Label Degeneracy Model (M5) have improved 0.8% and 0.5% in accuracy, reaching 81.63% and 81.33%, respectively. The reason that M4 is better than M5 is that the structural information and semantic and macro information are equally important in the short article. It can capture different features from different aspects, to compensate for the bias caused by the single model.

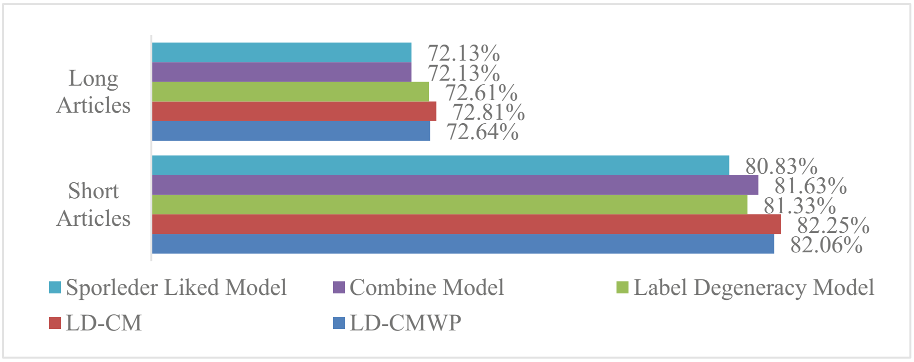


Fig. 5. The performance of models in different length articles.

In those long articles, M5 performed better than M4, reaching 72.61% accuracy. With the increasing of the article length, the structural information is more difficult to express for the structural characteristics of the discourse, and sometimes even excessively learning the low-level structure will cause the prediction error of the high-level structure. In those long articles, multi-relationships (usually *Joint* relation) appear more frequently, and this structure usually is labeled as 3 (*Multi-Nucleus*) in nuclearity, so it is easier to capture the special structure by the label degeneracy.

It is worth noting that LD-CM has achieved the best performance in both long and short article, reaching 72.81% and 82.25% respectively. Even LD-CMWP with slightly lower performance is better than other models, reaching 72.64% and 82.06% accuracy in long and short article respectively. This proves the effectiveness of our model.

5 Conclusions

In this paper, we propose a Label Degeneracy Combination Model (LD-CM) in recognizing macro discourse structure. We combine structural features with the semantic and macro-information features to form a combination model, and use the label degeneracy to find the solution of structure recognition in the solution space of nuclearity recognition. In this way, the model can capture a more detailed feature expression. The experimental results on MCDTB show that our LD-CM improves the accuracy by 1.21%, compared with the benchmark system. In particular, we use the post-editing to ensure generating a complete discourse structure tree automatically. In the future work, we will focus on recognizing discourse nuclearity and relationship, and eventually form an end-to-end macro discourse analyzer.

Acknowledgments. The authors would like to thank three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant Nos. 61772354, 61773276 and 61472265, and was also supported by the Strategic Pioneer Research Projects of Defense Science and Technology under Grant No. 17-ZLXDXX-02-06-02-04.

References

1. Li, Y.: Research on the structure of Chinese text structure and the construction of resources. Soochow University (2015)
2. Van Dijk, T.A.: Narrative macro-structures. PTL J. Descr. Poet. Theory Lit. **1**, 547–568 (1976)
3. Hernault, H., Prendinger, H., duVerle, D.A., Ishizuka, M.: HILDA: a discourse parser using support vector machine classification. Dialogue Discourse **1**(3), 1–33 (2010)
4. Joty, S., Carenini, G., Ng, R., Mehdad, Y.: Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In: ACL 2013, vol. 1, pp. 486–496. ACL, Sofia (2013)
5. Feng, V.W., Hirst, G.: A linear-time bottom-up discourse parser with constraints and post-editing. In: ACL 2014, pp. 511–521. ACL, Baltimore (2014)
6. Li, J., Li, R., Hovy, E.: Recursive deep models for discourse parsing. In: EMNLP 2014, pp. 2061–2069. ACL, Doha (2014)
7. Li, Q., Li, T., Chang, B.: Discourse parsing with attention-based hierarchical neural networks. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 362–371. ACL, Austin (2016)
8. Sagae, K.: Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In: International Workshop on Parsing Technologies, pp. 81–84. ACL, Paris (2009)

9. Feng, V.W., Hirst, G.: Text-level discourse parsing with rich linguistic features. In: ACL 2012, vol. 1, pp. 60–68. ACL, Jeju (2012)
10. Wang, Y., Li, S., Wang, H.: A two-stage parsing method for text-level discourse analysis. In: ACL 2017, vol. 2, pp. 184–188. ACL, Vancouver (2017)
11. Lv, G., Na, S., Li, R., Wang, Z., Chai, Q.: Frame-based discourse structure modeling and relation recognition for Chinese sentence. J. Chin. Inf. Process. **29**(6), 98–109 (2015)
12. Sporleder, C., Lascarides, A.: Combining hierarchical clustering and machine learning to predict high-level discourse structure. In: Coling 2004, pp. 43–49. ACM, Geneva (2004)
13. Jiang, F., Chu, X., Sheng, X., Li, P., Zhu, Q.: A macro discourse primary and secondary relation recognition method. J. Chin. Inf. Process. **32**(1), 43–50 (2018)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Comput. Sci. (2013)
15. Xu, S.: Research and Implementation of Paraphrase Recognition for Question Answering. Harbin Institute of Technology (2009)

LM Enhanced BiRNN-CRF for Joint Chinese Word Segmentation and POS Tagging

Jianhu Zhang[1], Gongshen Liu[1](✉), Jie Zhou[1], Cheng Zhou[2], and Huanrong Sun[2]

[1] School of Electric Information and Electronic Engineering, Shanghai Jiaotong University, Shanghai, China
{zhangjianhu3290,lgshen,sanny02}@sjtu.edu.cn
[2] SJTU-Shanghai Songheng Content Analysis Joint Lab, Shanghai, China
{zhoucheng,sunhuanrong}@021.com

Abstract. Word segmentation and part-of-speech tagging are two preliminary but fundamental components of Chinese natural language processing. With the upsurge of deep learning, end-to-end models are built without handcrafted features. In this work, we model Chinese word segmentation and part-of-speech tagging jointly on the basis of state-of-the-art BiRNN-CRF architecture. LSTM is adopted as the basic recurrent unit. Apart from utilizing pre-trained character embeddings and trigram features, we incorporate neural language model and conduct multi-task training. Highway layers are applied to tackle the discordance issue of the naive co-training. Experimental results on CTB5, CTB7, and PPD datasets show the effectiveness of the proposed method.

Keywords: Chinese word segmentation · POS tagging · LSTM
Language model

1 Introduction

Word segmentation and part-of-speech(POS) tagging are two preliminary but fundamental components of Chinese natural language processing(NLP). Ng and Low [18] demonstrate that word segmentation and POS tagging could be modeled as a sequence labeling problem, in which target labels are the combinations of segmentation boundaries and POS tags, namely the joint S&T. Based on traditional handcrafted features, researchers have done a lot of remarkable work [8,10,30,33].

With the rapid development of the artificial neural network and deep learning, many neural models are applied to the joint S&T, reducing the efforts of feature engineering and boosting the performance. Currently, character-based BiRNN-CRF model achieves state-of-the-art performance on the Chinese joint S&T [2,32], in which bidirectional recurrent neural network(BiRNN) is the main backbone for sequence tagging, and conditional random fields(CRF) [11] on top of BiRNN is used to gain the optimal tag sequence over the entire sentence. Moreover, LSTM [7] and GRU [3] are often applied to capture long-term relationship. Unlike previous pipeline methods, in which segmentation is put at the very first stage and then followed by POS tagging, the joint S&T model does not suffer from error propagation problem and word segmentation could make the most of information extracted from POS tagging to alleviate the ambiguity problem. Therefore, the joint S&T model outperforms the pipeline model significantly [2,24,33].

Recently, most researchers have focused on enriching the features of character embeddings [2,14,24,27,31]. However, given the complexity of neural network(NN) models and limited resources of the labeled corpus, it could be insufficient to train complicated models with annotations alone. Actually, there is a lot of semantic and syntactic knowledge that could be extracted from raw texts. Consequently, some semi-supervised and multi-task methods are proposed to improve sequence labeling performance [15,22,23].

In this paper, we extend the spirit of semi-supervising to the Chinese joint S&T and go a few steps further. Basic BiRNN-CRF framework for sequence labeling is adopted. We pre-train the Chinese character embeddings on large raw texts with GloVe [21]. In addition, we utilize n-gram embeddings to enrich the character features. More importantly, we propose to argument the current Chinese joint S&T architecture with a neural language model. The intuition is that it may be difficult for RNN to learn the proper representation as the hidden state considering the large number of parameters. Thus, it could be beneficial to add an extra but direct objective for the learning. The neural language model is simple and requires no additional data or annotation. It could be used as a training method and brings no extra computing cost during decoding.

To sum up, our contributions in this work are as follows:

1. Apply BiRNN-CRF model to the Chinese joint S&T and achieve state-of-the-art performance.
2. Propose to argument the current Chinese joint S&T architecture with a neural language model, which helps RNN to learn the proper hidden state but requires no additional data or annotation.
3. Conduct extensive experiments on three different datasets. Experimental results show that our best model outperforms previous state-of-the-art models.

2 LM Enhanced BiRNN-CRF Model

2.1 Overview of the Proposed Model

As visualized in Fig. 1, the proposed model is an adaptation of BiRNN-CRF enhanced by a neural language model. For a Chinese sentence $X = (c_1, \ldots, c_n)$, where c_i is the i-th character, all characters are represented as vectors and

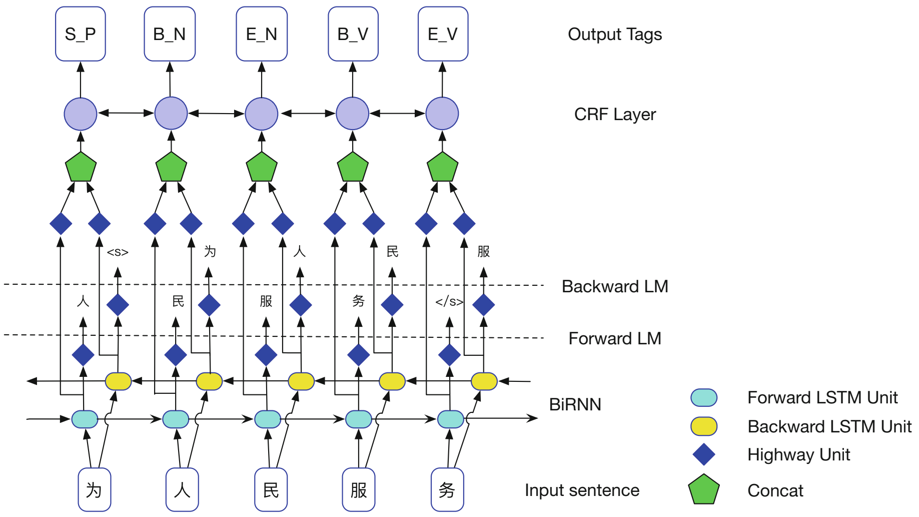


Fig. 1. LM Enhanced BiRNN-CRF Model Architecture

fed into BiRNN. LSTM is adopted as the basic recurrent unit. We employ the dropout [25] strategy to the output of BiRNN and then feed it into the first-order CRF layer. The sentence-level optimal tag sequence is predicted at the end. As for the tagging scheme, following the work of Kruengkrai et al. [10], we employ `B`, `I`, `E`, `S` as the word boundary tags, which represent a character at the beginning, inside, end of a word or a single character word respectively. In addition, in order to make the most of raw texts, output of BiRNN is also used as the input of a neural language model, which predicts the previous or next character. Highway layers are placed between output of BiRNN and target tasks to map the hidden state to different semantic space.

2.2 Character Representations

Character representations is of wide interest in Chinese NLP. Sun et al. [27] propose to enhance Chinese character embeddings with radical information, Li et al. [13] develop two component-enhanced Chinese character embedding models and their bigram extensions. Shao et al. [24] propose three different approaches to effectively represent Chinese characters, namely the concatenated n-gram, radicals and orthographical features as well as the pre-trained character embeddings. In order to keep the model as simple as possible, only pre-trained character embeddings and n-gram embeddings are employed in this paper.

n-gram Embeddings. Although RNN is good at extracting contextual features from context-free character representations, many researchers have shown that traditional n-gram embeddings is beneficial for many NLP tasks [5,12,29].

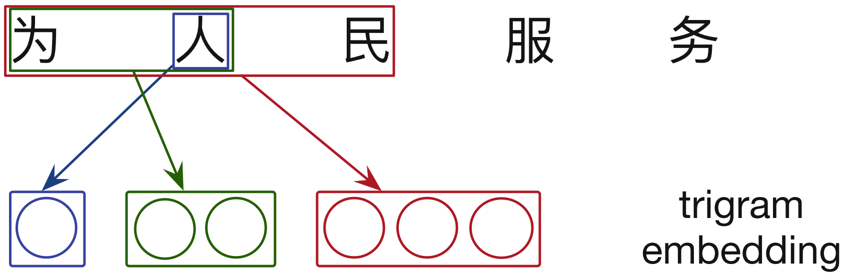


Fig. 2. Trigram embedding of a Chinese character in a given context

n-gram embeddings employed in this paper is demonstrated in Fig. 2. In this example, trigram embeddings of the pivot character 人 in the given context is the concatenation of the context-free vector representation of 人 itself along with the bigram 为人 as well as the trigram 为人民 .

Pre-trained Character Embeddings. The context-free vector representations of single characters used above could be replaced by pre-trained character embeddings trained from raw texts. In this paper, we pre-train character embeddings on Wikipedia[1] using GloVe [21]. This kind of pre-trained character embeddings is used to initialize the very bottom input of our neural networks. For those characters that are not in the embedding vocabulary, we randomly initialize them.

2.3 Neural Language Model

Although the character embeddings described above could carry a lot of general language knowledge, it is not task-specific, thus may contain a considerable irrelevant portion and maybe not optimal for the Chinese joint S&T. Moreover, due to the large number of parameters, it may be difficult for BiRNN to learn the proper hidden state. In order to address these problems, we propose to incorporate a language model with the joint S&T model and conduct multi-task learning.

As shown in Fig. 1, the language model and the S&T model share the same BiRNN, which fits the setting of multi-task learning and transfer learning. However, these two tasks are apparently not strongly related, which means directly using the output from the recurrent neural network layer could hurt the performance of the joint S&T model. Thus, highway layer [26] is applied to further transform the hidden state of RNN into different semantic space for different objectives. The highway layer can be illustrated as follows:

$$H(\mathbf{X}) = g(W_H\mathbf{X} + b_H) \odot T(\mathbf{X}) + \mathbf{X} \odot C(\mathbf{X}), \qquad (1)$$

[1] https://dumps.wikimedia.org/.

where operator $\odot$ indicates element-wise product, $g(\cdot)$ is a certain type of non-linear transformation. $T(\cdot)$ represents the transform gate and $C(\cdot)$ is the carry gate. In our experiments, $C(\cdot) = 1 - T(\cdot)$ is adopted. Transform gate $T(\cdot)$ could be formalized as:

$$T(\mathbf{X}) = \sigma(W_T \times \mathbf{X} + b_T) \tag{2}$$

where W_T and b_T are both trainable parameters.

In order to extract language knowledge from both directions, we adopt two language model, namely the forward language model(i.e., from left to right) and the backward language model(i.e., from right to left), which makes predictions for the next and previous character respectively. The forward language model is defined as

$$P_f(c_1, \ldots, c_n) = \prod_{i=1}^{N} P_f(c_i | c_1, \ldots, c_{i-1}) \tag{3}$$

where $P_f(c_i | c_1, \ldots, c_{i-1})$ is computed by the neural network with the following formula

$$P_f(c_i | c_0, \ldots, c_{i-1}) = \frac{\exp(w_{c_i}^T f_{i-1})}{\sum_{\hat{c}_j} \exp(w_{\hat{c}_j}^T f_{i-1})} \tag{4}$$

where w_{c_i} is the weight vector for predicting character c_i, f_{i-1} is output of corresponding highway unit. Consequently, the average negative log probability of the target words is applied as the object function of the forward language model:

$$\mathcal{J}_{F-LM} = -\frac{1}{N} \sum_i \log P_f(c_i) \tag{5}$$

Accordingly, the backward language model is defined as

$$P_b(c_1, \ldots, c_n) = \prod_{i=1}^{N} P_b(c_i | c_{i+1}, \ldots, c_N) \tag{6}$$

where $P_b(c_i | c_{i+1}, \ldots, c_N) = \frac{\exp(w_{c_i}^T b_i)}{\sum_{\hat{c}_j} \exp(w_{\hat{c}_j}^T b_i)}$. And the loss function is calculated as

$$\mathcal{J}_{B-LM} = -\frac{1}{N} \sum_i \log P_b(c_i) \tag{7}$$

With Eqs. 5 and 7, the overall objective function of our language model could be written as

$$\mathcal{J}_{LM} = \mathcal{J}_{F-LM} + \mathcal{J}_{B-LM} \tag{8}$$

2.4 CRF

Since we model the joint S&T as a sequence labeling task, it is beneficial to consider the correlations between labels in the neighborhood and jointly decode the optimal label sequence for a given input sentence. Therefore, we build a

CRF layer upon the BiRNN to decode each label jointly instead of independently. Formally, we use $z = (z_1, \ldots, z_n)$ to represent the BiRNN output(transformed by highway layer and concatenated with both directions) for an input sentence $x = (c_1, \ldots, c_n)$. $y(z) = (y_1, \ldots, y_n)$ is the corresponding label sequence for z. The probabilistic model of CRF describes the conditional probability of generating the whole label sequence y given z. Similar to Ma and Hovy [16], we define this probability with the following form:

$$p(y|z; W, b) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, z)}{\sum_{y' \in Y(z)} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, z)} \tag{9}$$

where $\psi_i(y', y, z) = \exp(W^T_{y',y} z_i + b_{y',y})$ are potential functions, and $W^T_{y',y}$ and $b_{y',y}$ are the weight vector and bias corresponding to label pair (y', y), respectively.

In the training phase, the maximum conditional likelihood is applied, i.e., minimize the negative log-likelihood as follows:

$$\mathcal{J}_{CRF} = -\sum_i \log p(y_i|z_i) \tag{10}$$

In the testing or decoding phase, our target is to search for the optimal label sequence y^* with the maximum conditional probability:

$$y^* = \underset{y \in Y(z)}{\operatorname{argmax}}\, p(y|z; W, b) \tag{11}$$

In this paper, we employ first-order CRF layer; the Viterbi algorithm can solve the training and decoding in an efficient way.

2.5 Multi-task Learning

With Eqs. 8 and 10, our multi-task loss function could be written as

$$\mathcal{J} = \mathcal{J}_{CRF} + \lambda \mathcal{J}_{LM} \tag{12}$$

where λ is a weight parameter to make a balance between the language model and the joint S&T model.

3 Experiments

We conduct our experiments on three different datasets: Chinese Treebank 5.1(CTB5), Chinese Treebank 7.0(CTB7), as well as PKU's People's Daily (PPD). Table 1 summarizes the brief statistics of these corpora in terms of sentence number and word number. CTB5 is split according to [8], CTB7 is split according to [30], and PPD datasets are from the Fourth International Chinese Language Processing Bakeoff [9]. We apply the standard F1-score to evaluate both the word segmentation and the joint S&T performance.

Table 1. Datasets' statistics

Datasets	Splits	Num of sentences	Num of words
CTB5	Train	18,086	494 k
	Dev	350	6.8 k
	Test	348	8.0 k
CTB7	Train	31,088	718 k
	Dev	10,036	237 k
	Test	10,291	245 k
PPD	Train	66,691	1.1 M
	Test	6,424	160 K

Table 2. Hyper-parameters.

Char. embedding size	64
LSTM state size	200
LSTM depth	1
Highway depth	1
Optimiser	Adagrad
Initial learning rate	0.1
Decay rate	0.05
Gradient clipping	5.0
Dropout rate	0.5
Batch size	20

3.1 Implementation

We implement our model based on the TensorFlow library [1]. Bucket strategy is adopted in our word, namely grouping the sentences of similar length into the same bucket and pad them to the equivalent length accordingly. Each bucket has their own computational graph.

The hyper-parameters adopted in this paper is shown in Table 2. We use the same hyper-parameters for all the experiments on different datasets without further fine-tuning. The weights of the neural networks are initialized as Glorot and Bengio [6]. We use the error back propagation algorithm to train the network. Mini-batch stochastic gradient descent with momentum is employed for optimization [4]. We apply dropout in our model, and the dropout rate is fixed at 0.5. Gradient clipping [19] of 5.0 is used for model stability.

3.2 Component Effects

In order to find out the effects of each component of our neural network architecture, we run some ablation experiments, and the results are shown in Table 3.

Table 3. Effects of each component on test sets of CTB5, CTB7, PDD datasets.

Models	CTB5			CTB7			PPD		
	P	R	F	P	R	F	P	R	F
BiLSTM-CRF	91.23	91.08	91.15	89.44	88.46	88.95	89.12	89.21	89.16
+ pre-trained embeddings	92.43	92.03	92.23	89.74	89.16	89.45	89.52	89.74	89.63
+ trigram embeddings	93.16	92.88	93.02	90.32	90.14	90.23	90.11	90.03	90.07
+ language model	93.04	92.92	92.98	90.82	90.04	90.43	89.93	89.89	89.91
+ highway layer	94.83	94.25	94.54	91.65	90.38	91.01	91.76	90.39	91.07

Table 4. Comparisons with previous models on test sets in word segmentation and joint S&T F1-scores.

Models	CTB5		CTB7		PPD	
	Seg	Seg&Tag	Seg	Seg&Tag	Seg	Seg&Tag
ZPar	97.69	93.79	94.59	89.71	94.62	89.94
Shap et al. [24]	97.98	94.06	95.37	90.54	94.95	90.76
Ours	98.72	94.88	95.74	91.24	95.45	91.32

The base model is BiRNN-CRF, in which character embeddings are randomly initialized. Pre-trained embeddings, n-gram embeddings, neural language model, and highway layer are incrementally incorporated into the base model.

Pre-trained Embeddings and Trigram Embeddings. From Table 3, we could learn that employing the pre-trained embeddings as initial vector representations for characters achieves improvements on all three datasets. Intuitively, the pre-trained character embeddings give a more reasonable initialization for NN. Further improvement is obtained by adding trigram embeddings, which reveals rich local information could be encoded in n-gram with a statistical co-occurrence way.

Language Model and Highway Layer. The third and fourth row of Table 3 elucidate the effects of the neural language model and the highway layer. As the third row shown, incorporating language model but without highway layer yields a little or no performance improvement. We conjecture the reason for this is that the joint S&T model and the language model are not strongly related so that they could not get benefits easily from each other. However, by employing highway layer, we get notable improvements. From these results, we could see that the language model is able to capture task-specific language knowledge, and the highway layer plays an important role in mediating the joint S&T model and the language model. In general, we incorporate all the essential components: pre-trained embeddings, trigram embeddings, neural language model, and highway layer into the basic BiRNN-CRF as our final model.

Table 5. Comparisons with [2] on test sets in joint S&T Precision, Recall and F1-scores.

Models	CTB5			CTB7			PPD		
	P	R	F	P	R	F	P	R	F
Chen et al. [2]	92.88	93.49	93.19	84.40	86.25	85.31	90.27	90.05	90.16
Ours	94.28	95.48	94.88	85.51	89.04	87.24	92.03	90.62	91.32

3.3 Performance Comparison

In this section, we focus on comparisons between the proposed model and previous state-of-the-art models. Ensemble decoding of three models trained independently is employed to get the best performance. Experimental results are shown in Table 4.

We first compare our model with ZPar [34], which is one of the most prevalent joint tagger using structured perceptron and beam decoding. We retrain a ZPar model and reproduce the performance reported in the original paper on CTB5. Then we train and evaluate ZPar on CTB7 and PPD respectively. As shown in Table 4, our model outperforms ZPar on all the three datasets. Theoretically, ZPar utilizes discrete local information at both character and word levels and employs structured perceptron for global optimization [32], while we employ BiRNN to model complex features and capture long-term dependencies. The CRF layer is applied on top of BiRNN for sentence level optimization. Moreover, we propose to incorporate a neural language model to conduct co-training to extract task-specific language knowledge from the raw text. We further employ highway layer to unify the joint S&T model and the language model. In contrast to the traditional methods, our model need no feature engineering or data preprocessing and benefits from large raw texts.

Recently, Shao et al. [24] propose several vector representations of Chinese characters to improve the joint S&T performance. We retrain their model on CTB5 using the best setting reported in their paper and reproduce their evaluation scores. We also retrain and evaluate their model on PDD and CTB7 for comparison. As shown in Table 4, besides without any feature engineering our model still outperforms theirs.

In Table 5, we compare our model with Chen et al. [2] in terms of the joint S&T F1 score. Chen et al. [2] introduce the convolution layer into the Chinese joint S&T and use different filters to model the complex compositional features of Chinese characters. We take the performance scores from their paper directly. Notably, they split the CTB7 dataset in a different way to estimate the model robustness. Following their setting, we also test our model on web blogs and train it on the rest of dataset. As shown in Table 5, our model outperforms theirs by a relatively large margin.

4 Related Work

As two of the most fundamental tasks in Chinese NLP, word segmentation and POS tagging have been studied for decades. Traditional methods like CRFs, HMMs, and maximum entropy classifiers [17,20,28] focused on feature engineering to create powerful handcrafted features for each specific task, which means that it is difficult to apply them to new tasks or domains. Recently, with the upsurge of deep learning, we have gotten rid of feature engineering but build end-to-end models. Meanwhile, CRF layer has also been demonstrated to be effective in capturing the dependency among labels and is widely used as the tag inference layer for sequence labeling tasks. Our model is built upon the success of BiRNN-CRF model and is further extended to better capture the character information for the Chinese S&T with a co-training neural language model.

Recently, Shao et al. [24] propose a character-based joint S&T model for Chinese using BiRNN-CRF. They mainly focus on enrich the character embeddings such as extracting radicals and orthographical features with convolutional neural networks(CNN), whereas we propose to enhance the current BiRNN-CRF model with a neural language model in a multi-task learning way. Peters et al. [22] propose to enrich word embeddings with pre-trained language model and obtain notable performance improvement. But this method requires expensive resources and time. Liu et al. [15] and Rei [23] both propose to empower sequence labeling with semi-supervised language model. We extend this spirit and modified the model architecture to make it applicable for the Chinese S&T.

5 Conclusion

In this paper, we model the Chinese S&T jointly as a fully character-based sequence labeling task. BiRNN-CRF is adopted and LSTM is used as the basic recurrent unit. In order to effectively extract language knowledge from the unstructured corpus, in addition to utilizing pre-trained character embeddings and trigram embeddings, we propose to incorporate neural language model and conduct multi-task training to help RNN learn the proper hidden state. Highway layers are applied to overcome the discordance issue of the naive co-training. Our model is extensively evaluated on CTB5, CTB7, and PPD datasets. The experimental results on the test sets show that the proposed model outperforms ZPar and other state-of-the-art models.

Acknowledgement. This research work has been funded by the National Natural Science Foundation of China (Grant No.61772337, U1736207 and 61472248), the SJTU-Shanghai Songheng Content Analysis Joint Lab, and program of Shanghai Technology Research Leader (Grant No.16XD1424400).

References

1. Abadi, M. et al.: Tensorflow: a system for large-scale machine learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI 2016, pp. 265–283. USENIX Association, Berkeley (2016)
2. Chen, X., Qiu, X., Huang, X.: A long dependency aware deep architecture for joint Chinese word segmentation and POS tagging. CoRR abs/1611.05384 (2016)
3. Chung, J., Gülçehre, Ç., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3555 (2014)
4. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**, 2121–2159 (2011)
5. Durme, B.V., Rastogi, P., Poliak, A., Martin, M.P.: Efficient, compositional, order-sensitive n-gram embeddings. In: EACL (2017)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y.W., Titterington, M. (eds.) Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, PMLR, Chia Laguna Resort, Sardinia, Italy, vol. 9, pp. 249–256, 13–15 May 2010
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
8. Jiang, W., Huang, L., Liu, Q., Lü, Y.: A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (2008)
9. Jin, G., Chen, X.: The fourth international Chinese language processing bakeoff: Chinese word segmentation, named entity recognition and Chinese POS tagging. In: IJCNLP (2008)
10. Kruengkrai, C., Uchimoto, K., Kazama, J., Wang, Y., Torisawa, K., Isahara, H.: Joint Chinese word segmentation and POS tagging using an error-driven word-character hybrid model. IEICE Trans. Inf. Syst. **E92**(D12), 2298–2305 (2009). https://doi.org/10.1587/transinf.E92.D.2298
11. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML 2001, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco (2001)
12. Li, B., Liu, T., Zhao, Z., Wang, P., Du, X.: Neural bag-of-ngrams. In: AAAI, pp. 3067–3074 (2017)
13. Li, Y., Li, W., Sun, F., Li, S.: Component-enhanced Chinese character embeddings. CoRR abs/1508.06669 (2015)
14. Ling, W., Luís, T., Marujo, L., Astudillo, R.F., Amir, S., Dyer, C., Black, A.W., Trancoso, I.: Finding function in form: Compositional character models for open vocabulary word representation. CoRR abs/1508.02096 (2015)
15. Liu, L., Shang, J., Xu, F.F., Ren, X., Gui, H., Peng, J., Han, J.: Empower sequence labeling with task-aware neural language model. CoRR abs/1709.04109 (2017)
16. Ma, X., Hovy, E.H.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. CoRR abs/1603.01354 (2016)
17. McCallum, A., Freitag, D., Pereira, F.C.N.: Maximum entropy markov models for information extraction and segmentation. In: Proceedings of the Seventeenth International Conference on Machine Learning, ICML 2000, pp. 591–598. Morgan Kaufmann Publishers Inc., San Francisco (2000)

18. Ng, H.T., Low, J.K.: Chinese part-of-speech tagging: one-at-a-time or all-at-once? word-based or character-based? In: Lin, D., Wu, D. (eds.) Proceedings of EMNLP 2004, pp. 277–284. Association for Computational Linguistics, Barcelona, July 2004
19. Pascanu, R., Mikolov, T., Bengio, Y.: Understanding the exploding gradient problem. CoRR abs/1211.5063 (2012)
20. Peng, F., Feng, F., McCallum, A.: Chinese segmentation and new word detection using conditional random fields. In: Proceedings of the 20th International Conference on Computational Linguistics, COLING 2004. Association for Computational Linguistics, Stroudsburg (2004). https://doi.org/10.3115/1220355.1220436
21. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
22. Peters, M.E., Ammar, W., Bhagavatula, C., Power, R.: Semi-supervised sequence tagging with bidirectional language models. CoRR abs/1705.00108 (2017)
23. Rei, M.: Semi-supervised multitask learning for sequence labeling. CoRR abs/1704.07156 (2017)
24. Shao, Y., Hardmeier, C., Tiedemann, J., Nivre, J.: Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF. CoRR abs/1704.01314 (2017)
25. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
26. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. CoRR abs/1505.00387 (2015)
27. Sun, Y., Lin, L., Tang, D., Yang, N., Ji, Z., Wang, X.: Radical-enhanced Chinese character embedding. CoRR abs/1404.4714 (2014)
28. Uchiumi, K., Tsukahara, H., Mochihashi, D.: Inducing word and part-of-speech with Pitman-Yor hidden semi-Markov models. In: ACL (2015)
29. Vajjala, S., Banerjee, S.: A study of n-gram and embedding representations for native language identification. In: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, pp. 240–248 (2017)
30. Wang, Y., Kazama, J., Tsuruoka, Y., Chen, W., Zhang, Y., Torisawa, K.: Improving Chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In: IJCNLP (2011)
31. Wieting, J., Bansal, M., Gimpel, K., Livescu, K.: Charagram: Embedding words and sentences via character n-grams. CoRR abs/1607.02789 (2016)
32. Yang, Z., Salakhutdinov, R., Cohen, W.W.: Transfer learning for sequence tagging with hierarchical recurrent networks. CoRR abs/1703.06345 (2017)
33. Zhang, Y., Clark, S.: Joint word segmentation and POS tagging using a single perceptron. In: Proceedings of ACL-08: HLT, pp. 888–896. Association for Computational Linguistics, Columbus, June 2008
34. Zhang, Y., Clark, S.: A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, pp. 843–852. Association for Computational Linguistics, Stroudsburg (2010)

Chinese Grammatical Error Correction Using Statistical and Neural Models

Junpei Zhou[1,2], Chen Li[1(✉)], Hengyou Liu[1], Zuyi Bao[1], Guangwei Xu[1], and Linlin Li[1]

[1] Alibaba Group, 969 West Wenyi Road, Hangzhou, China
{puji.lc,hengyou.lhy,zuyi.bzy,linyan.lll}@alibaba-inc.com, kunka.xgw@taobao.com

[2] Zhejiang University, 38 Zheda Road, Hangzhou, China
zhoujunpei@zju.edu.cn

Abstract. This paper introduces the Alibaba NLP team's system for NLPCC 2018 shared task of Chinese Grammatical Error Correction (GEC). Chinese as a Second Language (CSL) learners can use this system to correct grammatical errors in texts they wrote. We proposed a method to combine statistical and neural models for the GEC task. This method consists of two modules: the correction module and the combination module. In the correction module, two statistical models and one neural model generate correction candidates for each input sentence. Those two statistical models are a rule-based model and a statistical machine translation (SMT)-based model. The neural model is a neural machine translation (NMT)-based model. In the combination module, we implemented it in a hierarchical manner. We first combined models at a lower level, which means we trained several models with different configurations and combined them. Then we combined those two statistical models and a neural model at the higher level. Our system reached the second place on the leaderboard released by the official.

Keywords: Grammatical Error Correction · Combination
Statistical machine translation · Neural machine translation

1 Introduction

With the economy booming, China becomes more and more attractive to foreign businesses, students, and travelers, and learning Chinese is becoming more and more popular. The number of CSL learners grows up rapidly, but learning Chinese would not be easy for them, because Chinese is quite different from other languages, especially from English. For example, in Chinese, questions are conveyed by intonation and the subject and verb are not inverted as in English. Nouns cannot be post-modified as in English, and adverbials usually precede

J. Zhou—Work done during an internship at Alibaba Group

J. Zhou and C. Li—Equal Contribution.

verbs, unlike in English where complex rules govern the position of such sentence elements. It has quite flexible expressions and loose structural grammar. These traits bring a lot of trouble to CSL learners, leading to the rapid growth of the demands for Chinese GEC.

GEC for English has been studied for many years, with many shared tasks such as CoNLL-2013 [24] and CoNLL-2014 [23], while those kinds of studies on Chinese are less yet. This NLPCC shared task gives researchers an opportunity to build systems and exchange opinions, which can promote progress in this field. Another important contribution of this shared task is that it released a huge dataset for Chinese GEC. The details of this dataset will be described in Sect. 3. This shared task could make the community more flourish which benefits all CSL learners.

This paper is organized as follows: Sect. 2 describes some related works in English as well as Chinese GEC task. Dataset will be described in Sect. 3. Section 4 illustrates our system and explains two modules of it, including three models. The evaluation and discussion of the combination of statistical and neural models are shown in Sect. 5. Section 6 concludes the paper and discusses the future work.

2 Related Work

2.1 English GEC

Earlier methods for English GEC mainly use rule-based approaches [4,13] and classifier-based models [11,25,30], which can correct limited and specific type of errors. To address more complex errors, Machine Translation (MT) models are proposed and developed by many researchers. Statistical Machine Translation (SMT) has been dominant for a long time. In the work of Brockett et al. [2], they propose an SMT GEC model.

Since 2013, the GEC shared tasks in CoNLL2013 [24] and CoNLL2014 [23] boost this field, with a great many approaches developed. A POS-factored SMT system is proposed [34] to correct five types of errors in the text. In the work of Felice et al. [8], they propose a pipeline of the rule-based system and a phrase-based SMT system augmented by a web-based language model. The word-level Levenshtein distance between source and target is used as a translation model feature [15] to enhance the model.

Nevertheless, Neural Machine Translation (NMT) systems have achieved substantial improvements in this field [1,29]. Inspired by this phenomenon, Sun et al. [27] utilize the Convolutional Neural Network (CNN) for the article error correction. The Recurrent Neural Network (RNN) is also used [33] to map the sentence from learner space to expert space.

2.2 Chinese GEC

A great number of resources including annotated corpus are available in English GEC. However, the resource for Chinese is much less, and previous works related to

Chinese GEC is relatively scarce. The NLPTEA CGED shared task [9,16,17,32] boosts the Chinese Grammatical Error Diagnosis (GED) field greatly, and most works in Chinese GEC focus on the detection of errors instead of correction.

A probabilistic first-order inductive learning algorithm [5] outperforms many basic classifiers for error classification. In 2014, Lee et al. propose a judgment system at sentence level [18] combining N-gram statistical features and predefined rules. Several methods including CRF and SVM, together with frequency learning from a large N-gram corpus are used to detect and correct word ordering errors [7]. The work of Chang et al. [6] utilizes rules manually constructed as well as automatically generated. In the work of NTOU [19] they propose a traditional supervised model, which extracts word N-grams and POS N-grams as features. Rule-based methods and n-gram statistical methods are combined [31] to get a hybrid system for the CGED shared task.

3 Dataset Description

The dataset is provided by the NLPCC 2018 GEC shared task. The training data is collected from Lang-8 and each input sentence may have zero to k different corrections. The test data is texts written by foreign students and carefully corrected by professors. Both the training data and the test data are collated into the same form.

Each instance in the training data is in the form of $[s_o,\ k,\ \mathbf{C}]$, where s_o is the original sentence written by CSL learners, k denotes the number of correction candidates written by native speakers for s_o, and $\mathbf{C}$ is the set which contains k correction candidates as $\{c_1, c_2, ..., c_k\}$. After thresholding invalid lines whose k is 0, and filtering 216 lines whose $k > 0$ but $\mathbf{C}$ is empty, we got 593,524 valid lines. For each line of the data, we had two options of generating training instances. The first choice is to only use the candidate which has the minimal edit distance from the original sentence. In this method, for an original sentence s_o, we form a training instance (s_o, c_i) where $1 < i < k$ and $c_i = \arg\min_{c_i} \mathit{EditDistance}(c_i, s_o)$. We denote the training set generated by this method as 'NLPCC MinEd', which contains 593,524 data pairs. Another choice is to use all the candidates in $\mathbf{C}$. For an original sentence s_o, we make pairs of the original sentence and each of the candidates to form a training set as $\{(s_o, c_1), (s_o, c_2), ..., (s_o, c_k)\}$. We denote the training set generated by this method as 'NLPCC Expand', which contains 1,097,189 data pairs. This dataset is much larger than previous datasets released for the Chinese GEC field.

4 System Description

Our system combined statistical models as well as neural models, including the correction module and the hierarchical combination module. The pipeline is shown in Fig. 1.

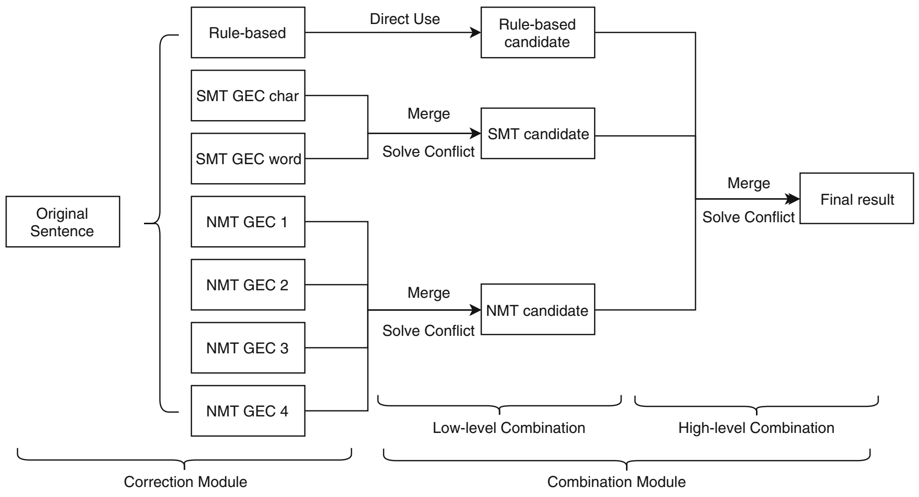


Fig. 1. Pipeline of our system with two modules

4.1 Correction Module

In the correction module, we used both statistical models and neural models with different configurations for the GEC task. The statistical models include the rule-based GEC model and SMT-based GEC models with different configurations. The neural models consist of several NMT-based GEC models with different structures.

Rule-Based GEC Model. The rule-based model starts by segmenting Chinese characters into chunks, which incorporates useful prior grammatical information to identify possible out-of-vocabulary errors. The segments are looked up in the dictionary built by Gigawords [10], and if a segment is out of vocabulary, it will go through the following steps:

1. If the segment consists of two or more characters, and turns out to be in the dictionary by permuting the characters, it will be added to the candidate list.
2. If the concatenation with a previous or next segment is in the dictionary, it will be added to the candidate list.
3. All possible keys in the dictionary with the same or similar Pinyin (the Romanization system for Standard Chinese) or similar strokes to the segment are generated. The generated keys for the segment itself, concatenated with those of previous or next segments, will be added to the candidate list of possible corrections.

After the steps, a candidate list of all possible corrections will be processed to identify whether there might be out-of-vocabulary error and its probability using a language model. The negative log likelihood of a size-5 sliding window suggests whether the top-scored candidate should be a correction of the original segment.

SMT-based GEC Model. The SMT GEC model consists of two components. One is a language model, which assigns a probability $p(e)$ for any target sentence e, and the other one is a translation model, which assigns a conditional probability $p(f|e)$. The language model is learned from a monolingual corpus of the target language, while the parameters of the translation model are calculated from the parallel corpus. We used the noisy channel model [3] to combine the language model and the translation model, and incorporated beam search to decode the result.

To explore the ability of SMT GEC models with different configurations, we trained two SMT GEC models with different data granularity as described in Sect. 5.1, including a char-level model S_{char} and a word-level model S_{word}. The correction result of sentence s_i generated by S_m was denoted as C_{iS_m} where $m \in \{char, word\}$.

NMT-based GEC Model. We used the encoder-decoder structure [1] with the general attention mechanism [20]. The NMT GEC model can capture complex relationships between the original sentence and the corrected sentence in GEC. We used a two-layer LSTM model for both encoder and decoder. To enhance the ability of NMT GEC models, we trained four NMT GEC models with different data pairs and configurations as described in Sect. 5.1. Those four NMT models were denoted as N_j, where $j \in \{1, 2, 3, 4\}$ was the model index. The correction result of sentence s_i generated by N_j was denoted as C_{iN_j}.

We used the character-based NMT because most characters in Chinese have their own meanings, which is quite different from English characters, and the Chinese word's meaning often depends on the meaning of its characters. On the other hand, the errors in original sentences can make the word-based tokenization worse, which will introduce larger and lower quality vocabulary list.

4.2 Combination Module

We performed a hierarchical combination of the correction candidates generated by models in the correction module. The hierarchical combination was composed of a low-level combination and a high-level combination. The low-level combination was used within each category of models, such as combining two SMT GEC models, combining four NMT GEC models, and so on. The high-level combination aimed to combine the candidates generated by the low-level combination, which means that it merged statistical and neural models.

One of the most significant problems in combination is to solve conflicts. The conflict means that when we want to merge several models, a sentence has different candidates from two or more models. For the conflict between two models, we designed five methods to solve the conflict, denoted as M_t where $t \in \{0, 1, 2, 3, 4\}$. M_0 is the simplest method to solve conflicts, which picked one side as the prior side, and then always chose the candidate in the prior side if conflicts occur. M_1 took the union operation on editing sets of two models if conflict occurred. Here the editing set is generated by the difference between the original sentence and the corresponding candidate sentence. Because the editing set is not unique between two sentences, we chose to use the editing set which could minimize the editing distance between two sentences. M_2 took the intersection operation on editing sets of two models if the conflict occurred. M_3 and M_4 both used the language model to assess the quality of candidate sentences. The language model was implemented by *KenLM* [12] to score each of the candidates and picked up the one with the higher score. The only difference is that M_4 used the length of the sentence to normalize the score, which means we divided the score by the length of the candidate sentence.

Low-Level Combination. For the two SMT GEC models S_{char} and S_{word}, we used M_3 to solve the conflict. For the four NMT GEC models, we first picked up two models, because incorporating too many models would confuse the model and provide many wrong candidates. Then we used those M_t where $t \in \{1, 2, 3, 4\}$ methods to solve conflicts between two models. We ranked those four models by the score on the development dataset split from the training dataset. Then we explored the combination of those four models with method M_0, and the order of combination is decided by the ranking order. For example, if model N_3 ranked prior to N_1, it will be used as higher priority during combination, which means that if a sentence s_i has different candidates in C_{iN_3} and C_{iN_1}, we picked up C_{iN_3} as the final result. Following this rule, we found that N_3 combined with N_4 with M_0 performed best, so we used this combination as the backbone, and tested M_t where $t \in \{1, 2, 3, 4\}$ on this combination. The detailed experimental results of the combination can be found in Sect. 5.2.

High-Level Combination. After the low-level combination, for each original sentence s_i, we had three candidates $\{C_{iR}, C_{iS}, C_{iN}\}$ generated by rule-based model, SMT GEC model, and NMT GEC model separately. We performed high-level combination on these candidates. If there were only two candidates which conflicted with each other, we could still use the method described as M_t where $t \in \{0, 1, 2, 3, 4\}$, but when all three candidates conflicted at the same time, we expanded the method M_t to fit three candidates, Those operations in M_t such as union and intersection could be easily expanded. We also designed a protection mechanism for the high-level combination according to the degree of agreement of three GEC models. If those three candidates of GEC models conflicted a lot, we assumed none of them is right and protected the sentence to keep it untouched. According to the sensitivity of the trigger of the protection mechanism, we designed two degrees of protection denoted as P_1 and P_2. The detailed experimental results of the combination can be found in Sect. 5.2.

5 Evaluation and Discussion

5.1 Experimental Settings

We randomly picked 10% of the training dataset as the development dataset, on which we tested our models and chose the combination method according to the scores. Because the official scorer was not released during the contest, we used the script written by ourselves to score the result on the development dataset. Firstly we converted the candidate to the editing set in the form of 'm2', which contains the editing steps between the original sentence and the candidate. Then we also converted the ground truth correction sentence to the 'm2' form. For all the valid editing sets we chose the one which minimized the editing distance.

For the SMT GEC model, we used different data granularities. S_{char} was trained on char-level data and S_{word} was trained on word-level data. Because most characters in Chinese have their own meanings, so it is reasonable to train a char-level SMT GEC model. We simply split every char by space to get the char-level data, and we used the Jieba [28] segmentation tool to split the word in a sentence by space to produce the word-level data.

For the NMT GEC model, we used the pre-trained embedding in different parts of the model. The first choice was to use it for the whole model, which forced the model to learn a proper embedding by itself. Considering the dataset is not large enough for the model to learn the embedding from scratch, we also tested the pre-trained embedding used for both encoder and decoder parts. But the embedding was trained on the Gigaword [10], which was quite different from the sentences written by CFL learners, so we also used the pre-trained embedding only in the decoder part. The configurations of our four different NMT GEC models N_j, $j \in \{1, 2, 3, 4\}$ are shown in Table 1. For the 'Network' column, the 'BiLSTM' means bi-directional LSTM [26].

5.2 Experimental Results

As described in Sect. 4.2, we used our own scorer during the contest, and used the scorer tool released by NLPCC to assess the performance again after the contest. It is worth to mention that the official scorer was released after the contest, so we chose the model combination based on the unofficial scorer written by ourselves. Because the official document released before contest used the F_1 score as the evaluation example, we calculated the F_1 score in our unofficial scorer instead of $F_{0.5}$ score. According to the evaluation of the single model performance of four NMT GEC models by our unofficial scorer, we ranked those models as $N_3 > N_4 > N_1 > N_2$, which determined the order of combination of NMT GEC models in Table 2. In Tables 2 and 3, the 'Precision', 'Recall', and '$F_{0.5}$ (official)' columns are calculated by the official scorer, and the 'F_1 (unofficial)' column is generated by our own scorer.

Table 1. Configurations of four NMT models

Model	Network	Embed	Dataset
N_1	LSTM	No Pre-trained Embedding	NLPCC MinEd
N_2	BiLSTM	Pre-trained Embedding for Encoder and Decoder	NLPCC MinEd
N_3	BiLSTM	Pre-trained Embedding for Encoder and Decoder	NLPCC Expand
N_4	BiLSTM	Pre-trained Embedding Only for Decoder	NLPCC Expand

According to the results shown in Table 2, we chose $S_{char} + S_{word}$ for the SMT GEC model, and $N_3 + N_4$ with M_3 for the NMT GEC model. For a specific sentence s_i, with the candidate C_{iR} generated by the rule-based model, the combination candidates C_{iS} and C_{iN} generated by the low-level combination of SMT and NMT GEC models separately, we used the high-level combination to generate the final result. According to Table 3, we first explored the influence of different M_i on combining three candidates, and chose the best one to add protection mechanism on it. Because we used the unofficial scorer during the contest, we chose M_2 as the conflict solving method and add P_2 protection as our final submission.

Table 2. Low-level Combination

Model	Solve conflict	Precision	Recall	$F_{0.5}$ (official)	F_1 (unofficial)
S_{char}	None	0.2096	0.0758	0.1549	0.1366
S_{word}	None	0.2107	0.0597	0.1399	0.1090
$S_{char} + S_{word}$	M_3	0.2376	0.0928	**0.1811**	**0.1462**
N_3	M_0	0.362	0.0996	0.2371	0.1166
$N_3 + N_4$	M_0	0.3453	0.1196	**0.2507**	**0.1260**
$N_3 + N_4 + N_1 + N_2$	M_0	0.3187	0.1292	0.2464	0.1152
$N_3 + N_4$	M_1	0.3363	0.1283	0.2540	0.1266
$N_3 + N_4$	M_2	0.3433	0.1130	0.2439	0.1259
$N_3 + N_4$	M_3	0.3485	0.1241	0.2559	**0.1318**
$N_3 + N_4$	M_4	0.3493	0.1238	**0.2561**	0.1304

Table 3. High-level combination

Model	Solve conflict	Precision	Recall	$F_{0.5}$ (official)	F_1 (unofficial)
$R + S + N$	M_0	0.3321	0.1714	0.2797	0.2731
$R + S + N$	M_1	0.3145	0.1969	0.2809	0.2342
$R + S + N$	M_2	0.3397	0.1664	0.2811	**0.2786**
$R + S + N$	M_3	0.3382	0.1782	**0.2867**	0.2370
$R + S + N$	M_4	0.336	0.1781	0.2854	0.2573
$R + S + N$	$M_2 + P1$	0.3528	0.1622	0.2856	0.2853
$R + S + N$	$M_2 + P2$	0.4100	0.1375	**0.2936**	**0.3371**

5.3 Case Analysis

We picked up some cases from the test dataset to illustrate the strengths and weaknesses of models in different categories.

As shown in Table 4, different models focus on different types of errors, so it is necessary to combine candidates generated by different models. The rule-based model is good at correcting errors which share similar intonation or font with the original character, such as 工司 to 公司 (similar intonation), 持别 to 特别 (similar font) and so on. The rule-based model can also solve a more complicated situation defined as character-order problem in a word, for example, correct 富 to 富裕 (change the order of characters). The SMT GEC model can also cope with some error-writing characters if they appear frequently in the training corpus. However, the most useful field of this model is to deal with errors which need adding or deleting a character, such as 都一起 to 一起 and so on. The NMT GEC model is good at correcting some complex errors, which need change the whole word with several characters, or reorder and add characters at the same time. For example, it can correct 太污染 to 污染太严重, and 对我看来 to 对我来, and so on, which to some extent 'understand' the collocation of characters.

As shown in Table 5, there still exists many limitations for those GEC models. The rule-based model would accidentally hurt some correct words if the statistical threshold is not set properly, such as 痴想 to 吃香. Although the SMT GEC

Table 4. The cases which can be corrected by our GEC models

Model	Original Sentence	Correction Candidate Generated
Rule	找工作的人比**工**司更多。	找工作的人比**公**司更多。
Rule	现在人们会认为中国，**持**别是北京	现在人们会认为中国，**特**别是北京
Rule	大家希望**裕富**的生活。	大家希望**富裕**的生活。
SMT	官吏腐败，尤其几个贪官污吏老是**枪**夺人民的财产。	官吏腐败，尤其**是**几个贪官污吏老是**枪**夺人民的财产。
SMT	**大使管**所查到的都可以上网。	**大使馆**所查到的都可以上网。
SMT	我和家人**都一起**在关岛过了很幸福的时间。	我和家人**一起**在关岛过了很幸福的时间。
NMT	对我**看来**，曼古空气污染的问题与日俱增。	对我**来说**，曼古空气污染的问题与日俱增。
NMT	北京的空气**太污染**了。	北京的空气**污染太严重**了。
NMT	大学是让学生们能**作为**一名好成人，好的社会的一部分。	大学是让学生们能**成为**一名好成人，好的社会的一部分。
NMT	结果全世界的环境问题**越严重**。	结果全世界的环境问题**越来越严重**。

Table 5. The cases which cannot be corrected by our GEC models

Model	Original Sentence	Correction Candidate Generated
Rule	因此，以自国文化的价值标准来批评别的文化是不合理的，而且应当抛弃的一种**痴想**。	因此，以自国文化的价值标准来批评别的文化是不合理的，而且应当抛弃的一种**吃香**。
SMT	学习是以得到智慧，不是**以**就业。	学习是以得到**的**智慧，不是就业。
NMT	大学的功能不是这样的。**就是学的**。	大学的功能不是这样的。

model can deal with errors about adding or deleting a character, it sometimes would add or delete wrong characters, such as add 的 and delete 以. The NMT GEC model sometimes would direct throw away a part of the sentence if it is too difficult to correct, as the part 就是学的 in the table.

6 Conclusion and Future Work

In this paper, we proposed a system for the GEC task, which combined statistical and neural models. This method consisted of two modules: the correction module and the combination module. In the correction module, two statistical models, including a rule-based model and an SMT GEC model, and an NMT GEC model generated correction candidates for each input sentence. In the combination module, we implemented it in a hierarchical manner. In the low-level combination, we combined models with different configurations within the same category. Then, in the high-level combination, we combined candidates of two statistical models and the neural model generated in the low-level combination. Our system reached the second place on the leaderboard released by the official.

In the future, we will further explore the strengths as well as limitations of three GEC models and combination methods in our system. We will focus on improving the 'Recall' metric of our system.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Brockett, C., Dolan, W.B., Gamon, M.: Correcting ESL errors using phrasal SMT techniques. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pp. 249–256. Association for Computational Linguistics (2006)
3. Brown, P.F., Pietra, V.J.D., Pietra, S.A.D., Mercer, R.L.: The mathematics of statistical machine translation: parameter estimation. Comput. Linguist. **19**(2), 263–311 (1993)
4. Bustamante, F.R., León, F.S.: GramCheck: a grammar and style checker. In: Proceedings of the 16th Conference on Computational Linguistics-Volume 1, pp. 175–181. Association for Computational Linguistics (1996)
5. Chang, R.Y., Wu, C.H., Prasetyo, P.K.: Error diagnosis of Chinese sentences using inductive learning algorithm and decomposition-based testing mechanism. ACM Trans. Asian Lang. Inf. Process. (TALIP) **11**(1), 3 (2012)
6. Chang, T.H., Sung, Y.T., Hong, J.F., Chang, J.I.: KNGED: a tool for grammatical error diagnosis of Chinese sentences. In: 22nd International Conference on Computers in Education, ICCE 2014. Asia-Pacific Society for Computers in Education (2014)
7. Cheng, S.M., Yu, C.H., Chen, H.H.: Chinese word ordering errors detection and correction for non-native Chinese language learners. In: Proceedings of COLING 2014, The 25th International Conference on Computational Linguistics: Technical Papers, pp. 279–289 (2014)

8. Felice, M., Yuan, Z., Andersen, Ø.E., Yannakoudakis, H., Kochmar, E.: Grammatical error correction using hybrid systems and type filtering. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pp. 15–24 (2014)
9. Gaoqi, R., Zhang, B., Endong, X., Lee, L.H.: IJCNLP-2017 task 1: Chinese grammatical error diagnosis. In: Proceedings of the IJCNLP 2017, Shared Tasks, pp. 1–8 (2017)
10. Gra, D., Chen, K.: Chinese gigaword. LDC Catalog No.: LDC2003T09, ISBN 1, 58563-58230 (2005)
11. Han, N.R., Chodorow, M., Leacock, C.: Detecting errors in English article usage with a maximum entropy classifier trained on a large, diverse corpus. In: LREC (2004)
12. Heafield, K.: KenLM: faster and smaller language model queries. In: Proceedings of the Sixth Workshop on Statistical Machine Translation, pp. 187–197. Association for Computational Linguistics (2011)
13. Heidorn, G.E., Jensen, K., Miller, L.A., Byrd, R.J., Chodorow, M.S.: The EPISTLE text-critiquing system. IBM Syst. J. **21**(3), 305–326 (1982)
14. Ji, J., Wang, Q., Toutanova, K., Gong, Y., Truong, S., Gao, J.: A nested attention neural hybrid model for grammatical error correction. arXiv preprint arXiv:1707.02026 (2017)
15. Junczys-Dowmunt, M., Grundkiewicz, R.: The AMU system in the CoNLL-2014 shared task: grammatical error correction by data-intensive and feature-rich statistical machine translation. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pp. 25–33 (2014)
16. Lee, L.H., Gaoqi, R., Yu, L.C., Endong, X., Zhang, B., Chang, L.P.: Overview of NLP-TEA 2016 shared task for Chinese grammatical error diagnosis. In: Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016), pp. 40–48 (2016)
17. Lee, L.H., Yu, L.C., Chang, L.: Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis, 07 March 2015
18. Lee, L.H., Yu, L.C., Lee, K.C., Tseng, Y.H., Chang, L.P., Chen, H.H.: A sentence judgment system for grammatical error detection. In: Proceedings of COLING 2014, The 25th International Conference on Computational Linguistics: System Demonstrations, pp. 67–70 (2014)
19. Lin, C.J., Chan, S.H.: Description of NTOU Chinese grammar checker in CFL 2014. In: Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2014), Nara, Japan, pp. 75–78 (2014)
20. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
21. Madnani, N., Tetreault, J., Chodorow, M.: Exploring grammatical error correction with not-so-crummy machine translation. In: Proceedings of the Seventh Workshop on Building Educational Applications using NLP, pp. 44–53. Association for Computational Linguistics (2012)
22. Napoles, C., Callison-Burch, C.: Systematically adapting machine translation for grammatical error correction. In: Proceedings of the 12th Workshop on Innovative use of NLP for Building Educational Applications, pp. 345–356 (2017)
23. Ng, H.T., Wu, S.M., Briscoe, T., Hadiwinoto, C., Susanto, R.H., Bryant, C.: The CoNLL-2014 shared task on grammatical error correction. In: Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pp. 1–14 (2014)

24. Ng, H.T., Wu, S.M., Wu, Y., Hadiwinoto, C., Tetreault, J.: The CoNLL-2013 shared task on grammatical error correction (2013)
25. Rozovskaya, A., Roth, D.: Algorithm selection and model adaptation for ESL correction tasks. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pp. 924–933. Association for Computational Linguistics (2011)
26. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
27. Sun, C., Jin, X., Lin, L., Zhao, Y., Wang, X.: Convolutional neural networks for correcting English article errors. In: Li, J., Ji, H., Zhao, D., Feng, Y. (eds.) NLPCC 2015. LNCS (LNAI), vol. 9362, pp. 102–110. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25207-0_9
28. Sun, J.: 'jieba' Chinese word segmentation tool (2012)
29. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
30. Tetreault, J.R., Chodorow, M.: The ups and downs of preposition error detection in ESL writing. In: Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1, pp. 865–872. Association for Computational Linguistics (2008)
31. Wu, X., Huang, P., Wang, J., Guo, Q., Xu, Y., Chen, C.: Chinese grammatical error diagnosis system based on hybrid model. In: Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications, pp. 117–125 (2015)
32. Yu, L.C., Lee, L.H., Chang, L.P.: Overview of grammatical error diagnosis for learning Chinese as a foreign language. In: Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications, pp. 42–47 (2014)
33. Yuan, Z., Briscoe, T.: Grammatical error correction using neural machine translation. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 380–386 (2016)
34. Yuan, Z., Felice, M.: Constrained grammatical error correction using statistical machine translation. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, pp. 52–61 (2013)
35. Zampieri, M., Tan, L.: Grammatical error detection with limited training data: the case of Chinese. In: Proceedings of ICCE (2014)

Text Mining

Multi-turn Inference Matching Network for Natural Language Inference

Chunhua Liu[1], Shan Jiang[1], Hainan Yu[1], and Dong Yu[1,2](✉)

[1] Beijing Language and Culture University, Beijing, China
chunhualiu596@gmail.com, jiangshan727@gmail.com, ericeryu@gmail.com, yudong_blcu@126.com

[2] Beijing Advanced Innovation for Language Resources of BLCU, Beijing, China

Abstract. Natural Language Inference (NLI) is a fundamental and challenging task in Natural Language Processing (NLP). Most existing methods only apply one-pass inference process on a mixed matching feature, which is a concatenation of different matching features between a premise and a hypothesis. In this paper, we propose a new model called Multi-turn Inference Matching Network (MIMN) to perform multi-turn inference on different matching features. In each turn, the model focuses on one particular matching feature instead of the mixed matching feature. To enhance the interaction between different matching features, a memory component is employed to store the history inference information. The inference of each turn is performed on the current matching feature and the memory. We conduct experiments on three different NLI datasets. The experimental results show that our model outperforms or achieves the state-of-the-art performance on all the three datasets.

Keywords: Natural Language Inference · Multi-turn inference
Memory mechanism

1 Introduction

Natural Language Inference (NLI) is a crucial subtopic in Natural Language Processing (NLP). Most studies treat NLI as a classification problem, aiming at recognizing the relation types of hypothesis-premise sentence pairs, usually including "Entailment", "Contradiction" and "Neutral".

NLI is also called Recognizing Textual Entailment (RTE) [7] in earlier works and a lot of statistical-based [9] and rule-based approaches [19] are proposed to solve the problem. In 2015, Bowman released the SNLI corpus [3] that provides more than 570 K hypothesis-premise sentence pairs. The large-scale data of SNLI allows a Neural Network (NN) based model to perform on the NLI. Since then, a variety of NN based models have been proposed, most of which can be divided into two kinds of frameworks. The first one is based on "Siamense" network [3,22]. It first applies either Recurrent Neural Network (RNN) or Convolutional Neural Networks (CNN) to generates sentence representations on both premise and hypothesis, and then concatenate them for the final classification. The second one is called "matching-aggregation" network [33,36]. It matches two sentences at word level, and then aggregates the matching results to generate a fixed vector for prediction. Matching is implemented by several functions based on element-wise operations [25,34]. Studies on SNLI show that the second one performs better.

Though the second framework has made considerable success on the NLI task, there are still some limitations. First, the inference on the mixed matching feature only adopts one-pass process, which means some detailed information would not be retrieved once missing. While the multi-turn inference can overcome this deficiency and make better use of these matching features. Second, the mixed matching feature only concatenates different matching features as the input for aggregation. It lacks interaction among various matching features. Furthermore, it treats all the matching features equally and cannot assign different importance to different matching features.

In this paper, we propose the MIMN model to tackle these limitations. Our model uses the matching features described in [5,33]. However, we do not simply concatenate the features but introduce a multi-turn inference mechanism to infer different matching features with a memory component iteratively. The merits of MIMN are as follows:

- MIMN first matches two sentences from various perspectives to generate different matching features and then aggregates these matching features by multi-turn inference mechanism. During the multi-turn inference, each turn focuses on one particular matching feature, which helps the model extract the matching information adequately.
- MIMN establishes the contact between the current and previous matching features through memory component. The memory component store the inference message of the previous turn. In this way, the inference information flows.

We conduct experiments on three NLI datasets: SNLI [3], SCITAIL [12] and MPE [14]. On the SNLI dataset, our single model achieves 88.3% in accuracy and our ensemble model achieves 89.3% in terms of accuracy, which are both comparable with the state-of-the-art results. Furthermore, our MIMN model outperforms all previous works on both SCITAIL and MPE dataset. Especially, the model gains substantial (8.9%) improvement on MPE dataset which contains multiple premises. This result shows our model is expert in aggregating the information of multiple premises.

2 Related Work

Early work on the NLI task mainly uses conventional statistical methods on small-scale datasets [7,20]. Recently, the neural models on NLI are based on large-scale datasets and can be categorized into two central frameworks: (i) Siamense-based framework which focuses on building sentence embeddings separately and integrates the two sentence representations to make the final prediction [4,17,22–24,29,32]; (ii) "matching-aggregation" framework which uses various matching methods to get the interactive space of two input sentences and then aggregates the matching results to dig for deep information [8,10,15,16,21,25,27,28,32,36,38].

Our model is directly motivated by the approaches proposed by [5,34]. [34] introduces the "matching-aggregation" framework to compare representations between words and then aggregate their matching results for final decision.

[5] enhances the comparing approaches by adding element-wise subtraction and element-wise multiplication, which further improve the performance on SNLI. The previous work shows that matching layer is an essential component of this framework and different matching methods can affect the final classification result.

Various attention-based memory neural networks [37] have been explored to solve the NLI problem [6,15,23]. [15] presents a model of deep fusion LSTMs (DF-LSTMs) (Long Short-Term Memory) which utilizes a strong interaction between text pairs in a recursive matching memory. [6] uses a memory network to extend the LSTM architecture. [23] employs a variable sized memory model to enrich the LSTM-based input encoding information. However, all the above models are not specially designed for NLI and they all focus on input sentence encoding.

Inspired by the previous work, we propose the MIMN model. We iteratively update memory by feeding in different sequence matching features. We are the first to apply memory mechanism to matching component for the NLI task. Our experiment results on several datasets show that our MIMN model is significantly better than the previous models.

3 Model

In this section, we describe our MIMN model, which consists of the following five major components: encoding layer, attention layer, matching layer, multi-turn inference layer and output layer. Figure 1 shows the architecture of our MIMN model.

We represent each example of the NLI task as a triple (p, q, y), where $p = [p_1, p_2, \cdots, p_{l_p}]$ is a given premise, $q = [q_1, q_2, \cdots, q_{l_q}]$ is a given hypothesis, p_i and $q_j \in \mathbb{R}^r$ are word embeddings of r-dimension. The true label $y \in \mathcal{Y}$ indicates the logical relationship between the premise p and the hypothesis q, where $\mathcal{Y} = \{neutral, entailment, contradiction\}$. Our model aims to compute the conditional probability $Pr(y|p, q)$ and predict the label for examples in testing data set by $y^* = argmax_{y \in \mathcal{Y}} Pr(y|p, q)$.

3.1 Encoding Layer

In this paper, we utilize a bidirectional LSTM (BiLSTM) [11] as our encoder to transform the word embeddings of premise and hypothesis to context vectors.

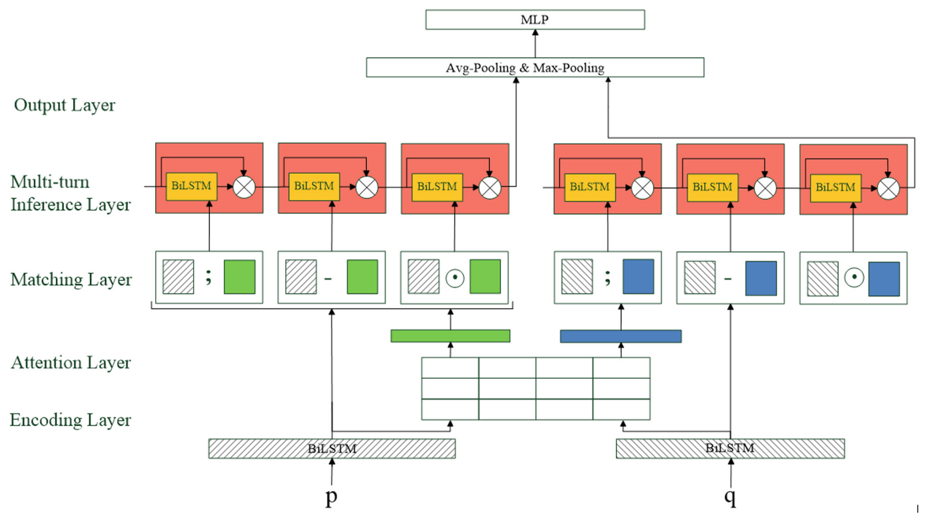


Fig. 1. Architecture of MIMN Model. The matching layer outputs a matching sequence by matching the context vectors with the aligned vectors (green and blue) based on three matching functions. The multi-turn inference layer generates inference vectors by aggregating the matching sequence over multi-turns. (Color figure online)

The premise and the hypothesis share the same weights of BiLSTM.

$$\bar{p}_i = \text{BiLSTM}_{enc}(p, i), \qquad i \in [1, 2, \cdots, l_p] \tag{1}$$

$$\bar{q}_j = \text{BiLSTM}_{enc}(q, j), \qquad j \in [1, 2, \cdots, l_q] \tag{2}$$

where the context vectors $\bar{p}_i$ and $\bar{q}_j$ are the concatenation of the forward and backward hidden outputs of BiLSTM respectively. The outputs of the encoding layer are the context vectors $p \in \mathbb{R}^{l_p \times 2d}$ and $q \in \mathbb{R}^{l_q \times 2d}$, where d is the number of hidden units of BiLSTM_{enc}.

3.2 Attention Layer

On the NLI task, the relevant contexts between the premise and the hypothesis are important clues for final classification. The relevant contexts can be acquired by a soft-attention mechanism [2,18], which has been applied to a bunch of tasks successfully. The alignments between a premise and a hypothesis are based on a score matrix. There are three most commonly used methods to compute the score matrix: linear combination, bilinear combination, and dot product. For simplicity, we choose dot product in the following computation [25]. First, each element in the score matrix is computed based on the context vectors of $\bar{p}_i$ and $\bar{q}_j$ as follows:

$$e_{ij} = \bar{p}_i^T \bar{q}_j, \tag{3}$$

where $\bar{p}_i$ and $\bar{q}_j$ are computed in Eqs. (1) and (2), and e_{ij} is a scalar which indicates how $\bar{p}_i$ is related to $\bar{q}_j$.

Then, we compute the alignment vectors for each word in the premise and the hypothesis as follows:

$$\tilde{p}_i = \sum_{j=1}^{l_q} \frac{exp(e_{ij})}{\sum_{t=1}^{l_q} exp(e_{it})} \bar{q}_j, \qquad \tilde{q}_j = \sum_{i=1}^{l_p} \frac{exp(e_{ij})}{\sum_{t=1}^{l_p} exp(e_{tj})} \bar{p}_i, \tag{4}$$

where $\tilde{p}_i \in \mathbb{R}^{2d}$ is the weighted summaries of the hypothesis in terms of each word in the premise. The same operation is applied to $\tilde{q}_j \in \mathbb{R}^{2d}$. The outputs of this layer are $\tilde{p}_i \in \mathbb{R}^{l_p \times 2d}$ and $\tilde{q}_j \in \mathbb{R}^{l_q \times 2d}$. For the context vectors $\bar{p}$, the relevant contexts in the hypothesis $\bar{q}$ are represented in $\tilde{p}$. The same is applied to $\bar{q}$ and $\tilde{q}$.

3.3 Matching Layer

The goal of the matching layer is to match the context vectors $\bar{p}$ and $\bar{q}$ with the corresponding aligned vectors $\tilde{p}$ and $\tilde{q}$ from multi-perspective to generate a matching sequence.

In this layer, we match each context vector p_i against each aligned vector $\tilde{p}_i$ to capture richer semantic information. We design three effective matching functions: f^c, f^s and f^m to match two vectors [5,31,33]. Each matching function takes the context vector $\bar{p}_i$ ($\bar{q}_j$) and the aligned vector $\tilde{p}_i$ ($\tilde{q}_j$) as inputs, then matches the inputs by an feed-forward network based on a particular matching operation and finally outputs a matching vector. The formulas of the three matching functions f^c, f^s and f^m are described in formulas (5)–(7). To avoid repetition, we will only describe the application of these functions to $\bar{p}$ and $\tilde{p}$. The readers can infer these equations for $\bar{q}$ and $\tilde{q}$.

$$u^c_{p,i} = f^c(\bar{p}_i, \tilde{p}_i) = \text{ReLU}(W^c([\bar{p}_i \,;\, \tilde{p}_i]) + b^c), \tag{5}$$

$$u^s_{p,i} = f^s(\bar{p}_i, \tilde{p}_i) = \text{ReLU}(W^s(\bar{p}_i - \tilde{p}_i]) + b^s), \tag{6}$$

$$u^m_{p,i} = f^m(\bar{p}_i, \tilde{p}_i) = \text{ReLU}(W^m(\bar{p}_i \odot \tilde{p}_i]) + b^m), \tag{7}$$

where ; , $-$, and $\odot$ represent concatenation, subtraction, and multiplication respectively, $W^c \in \mathbb{R}^{4d \times d}$, $W^s \in \mathbb{R}^{2d \times d}$ and $W^m \in \mathbb{R}^{2d \times d}$ are weight parameters to be learned, and $b^c, b^s, b^m \in \mathbb{R}^d$ are bias parameters to be learned. The outputs of each matching function are $u^c_{p,i}, u^s_{p,i}, u^m_{p,i} \in \mathbb{R}^d$, which represent the matching result from three perspectives respectively. After matching the context vectors $\bar{p}$ and the aligned vectors $\tilde{p}$ by f^c, f^s and f^m, we can get three matching features $u^c_p = \{u^c_{p,i}\}_1^{l_p}$, $u^s_p = \{u^s_{p,i}\}_1^{l_p}$ and $u^m_p = \{u^m_{p,i}\}_1^{l_p}$.

The u^c_p can be considered as a joint-feature of combing the context vectors $\bar{p}$ with aligned vectors $\tilde{p}$, which preserves all the information. And the u^s_p can be seen as a diff-feature of the $\bar{p}$ and $\tilde{p}$, which preserves the different parts and removes the similar parts. And the u^m_p can be regarded as a sim-feature of p and $\bar{p}$, which emphasizes on the similar parts and neglects the different parts between

$\bar{p}$ and $\tilde{p}$. Each feature helps us focus on particular parts between the context vectors and the aligned vectors. These matching features are vector representations with low dimension, but containing high-order semantic information. To make further use of these matching features, we collect them to generate a matching sequence u_p.

$$u_p = [u_p^1, u_p^2, u_p^3] = [u_p^c, u_p^s, u_p^m], \tag{8}$$

where $u_p^1, u_p^2, u_p^3 \in \mathbb{R}^{l_p \times d}$.

The output of this layer is the matching sequence u_p, which stores three kinds of matching features. The order of the matching features in u_p is inspired by the attention trajectory of human beings making inference on premise and hypothesis. We process the matching sequence in turn in the multi-turn inference layer. Intuitively, given a premise and a hypothesis, we will first read the original sentences to find the relevant information. Next, it's natural for us to combine all the parts of the original information and the relevant information. Then we move the attention to the different parts. Finally, we pay attention to the similar parts.

3.4 Multi-turn Inference Layer

In this layer, we aim to acquire inference outputs by aggregating the information in the matching sequence by multi-turn inference mechanism. We regard the inference on the matching sequence as the multi-turn interaction among various matching features. In each turn, we process one matching feature instead of all the matching features [5,8]. To enhance the information interaction between matching features, a memory component is employed to store the inference information of the previous turns. Then, the inference of each turn is based on the current matching feature and the memory. Here, we utilize another BiLSTM for the inference.

$$c_{p,i}^k = \text{BiLSTM}_{inf}(W_{inf}[u_{p,i}^k; m_{p,i}^{(k-1)}]), \tag{9}$$

where $c_{p,i}^k \in \mathbb{R}^{2d}$ is an inference vector in the current turn, $k = [1, 2, 3]$ is the index current turn, $i = [1, 2, 3, \cdots, l_p]$, $m_{p,i}^{(k-1)} \in \mathbb{R}^{2d}$ is a memory vector stores the historical inference information, and $W_{inf} \in \mathbb{R}^{3d \times d}$ is used for dimension reduction.

Then we update the memory by combining the current inference vector $c_{p,i}^k$ with the memory vector of last turn $m_p^{(k-1)i}$. An update gate is used to control the ratio of current information and history information adaptively [35]. The initial values of all the memory vectors are all zeros.

$$\begin{aligned} m_{p,i}^k &= g \odot c_{p,i}^k + (1 - g) \odot m_{p,i}^{(k-1)}, \\ g &= \sigma(W_g[c_{p,i}^k; m_{p,i}^{(k-1)}] + b_g), \end{aligned} \tag{10}$$

where $W_g \in \mathbb{R}^{4d \times 2d}$ and $b_g \in \mathbb{R}^{2d}$ are parameters to be learned, and σ is a sigmoid function to compress the ratio between 0–1. Finally, we use the latest memory matrix $\{m_p^{3i}\}_1^{l_p}$ as the inference output of premise m_p^{inf}. Then we calculate m_q^{inf} in a similar way. The final outputs of this layer are m_p^{inf} and m_q^{inf}.

3.5 Output Layer

The final relationship judgment depends on the sentence embeddings of premise and hypothesis. We convert m_p^{inf} and m_q^{inf} to sentence embeddings of premise and hypothesis by max pooling and average pooling. Next, we concatenate the two sentence embeddings to a fixed-length output vector. Then we feed the output vector to a multilayer perceptron (MLP) classifier that includes a hidden layer with *tanh* activation and a softmax layer to get the final prediction. The model is trained end-to-end. We employ multi-class cross-entropy as the cost function when training the model.

4 Experiment

4.1 Data

To verify the effectiveness of our model, we conduct experiments on three NLI datasets. The basic information about the three datasets is shown in Table 1.

The large SNLI [3] corpus is served as a major benchmark for the NLI task. The MPE corpus [14] is a newly released textual entailment dataset. Each pair in MPE consists of four premises, one hypothesis, and one label, which is different from the standard NLI datasets. Entailment relationship holds if the hypothesis comes from the same image as the four premises. The SCITAIL [12] is a dataset about science question answering. The premises are created from relevant web sentences, while hypotheses are created from science questions and the corresponding answer candidates.

Table 1. Basic information about the three NLI datasets. Sentence pairs is the total examples of each dataset. N, E, and C indicate Neutral, Entailment, and Contradiction, respectively.

Dataset	Sentence pairs	Train	Valid	Test	Labels
SNLI	570k	549,367	9,842	9,824	N, E, C
MPE	10k	8,000	1,000	1,000	N, E, C
SCITAIL	24k	23,596	1,304	2,126	N, E

4.2 Models for Comparison

We compare our model with "matching-aggregation" related and attention-based memory related models. In addition, to verify the effectiveness of these major components in our model, we design the following model variations for comparison:

- **ESIM:** We choose the ESIM model as our baseline. It mixes all the matching feature together in the matching layer and then infers the matching result in a single-turn with a BiLSTM.
- **600D MIMN:** This is our main model described in Sect. 3.
- **600D MIMN-memory:** This model removes the memory component. The motivation of this experiment is to verify whether the multiple turns inference can acquire more sufficient information than one-pass inference. In this model, we process one matching feature in one iteration. The three matching features are encoded by BiLSTM$_{inf}$ in multi-turns iteratively without previous memory information. The output of each iteration is concatenated to be the final output of the multi-turn inference layer:

$$c^k_{p,i} = \text{BiLSTM}_{inf}(W_{inf}[u^k_{p,i}]), \tag{11}$$

$$m^{inf}_p = [\{c^1_{p,i}\}^{l_p}_1; \{c^2_{p,i}\}^{l_p}_1; \{c^3_{p,i}\}^{l_p}_1]. \tag{12}$$

- **600D MIMN-gate+ReLU:** This model replaces the update gate in the memory component with a ReLU layer. The motivation of this model is to verify the effectiveness of update gate for combining current inference result and previous memory. Then the Eq. (10) is changed into Eq. (13).

$$m^k_{p,i} = \text{ReLU}(W_m[c^k_{p,i}; m^{(k-1)}_{p,i}]). \tag{13}$$

4.3 Experimental Settings

We implement our model with Tensorflow [1]. We initialize the word embeddings by the pre-trained embeddings of 300D GloVe 840B vectors [26]. The word embeddings of the out-of-vocabulary words are randomly initialized. The hidden units of BiLSTM$_{enc}$ and BiLSTM$_{inf}$ are 300 dimensions. All weights are constrained by L2 regularization with the weight decay coefficient of 0.0003. We also apply dropout [30] to all the layers with a dropout rate of 0.2. Batch size is set to 32. The model is optimized with Adam [13] with an initial learning rate of 0.0005, the first momentum of 0.9 and the second of 0.999. The word embeddings are fixed during all the training time. We use early-stopping (patience = 10) based on the validation set accuracy. We use three turns on all the datasets. The evaluation metric is the classification accuracy. To help duplicate our results, we will release our source code at https://github.com/blcunlp/RTE/tree/master/MIMN.

4.4 Experiments on SNLI

Experimental results of the current state-of-the-art models and three variants of our model are listed in Table 2. The first group of models (1)–(3) are the attention-based memory models on the NLI task. [15] uses external memory

Table 2. Performance on SNLI

Model (memory related)	Parameters	Train(% acc)	Test(% acc)
(1) 100D DF-LSTM [15]	320k	85.2	84.6
(2) 300D MMA-NSE with attention [23]	3.2m	86.9	85.4
(3) 450D LSTMN with deep attention fusion [6]	3.4m	88.5	86.3
Model (bidirectional inter-attention)	Parameters	Train (% acc)	Test (% acc)
(4) 200D decomposable attention [25]	380k	89.5	86.3
(5) "compare-aggregate" [33]	–	89.4	86.8
(6) BiMPM [36]	1.6m	90.9	87.5
(7) 600D ESIM [5]	4.3M	92.6	88.0
(8) 300D CAFE [32]	4.7m	89.8	**88.5**
(9) 450D DR-BiLSTM [8]	7.5m	94.1	**88.5**
(10) BiMPM (Ensemble) [36]	6.4m	93.2	88.8
(11) 450D DR-BiLSTM (Ensemble) [8]	45m	94.8	89.3
(12) 300D CAFE (Ensemble) [32]	17.5m	92.5	89.3
Human Performance (Estimated)	–	–	87.7
Model (this paper)	Parameters	Train (%acc)	Test (%acc)
(13) 600D MIMN	5.3m	92.2	**88.3**
(14) 600D MIMN-memory	5.8m	87.5	87.5
(15) 600D MIMN-gate+ReLU	5.3m	90.7	88.2
(16) 600D MIMN (**Ensemble**)	–	92.5	**89.3**

to increase the capacity of LSTMs. [23] utilizes an encoding memory matrix to maintain the input information. [6] extends the LSTM architecture with a memory network to enhance the interaction between the current input and all previous inputs.

The next group of models (4)–(12) belong to the "matching-aggregation" framework with bidirectional inter-attention. Decomposable attention [25] first applies the "matching-aggregation" on SNLI dataset explicitly. [33] enriches the framework with several comparison functions. BiMPM [36] employs a multi-perspective matching function to match the two sentences. ESIM [5] further sublimates the framework by enhancing the matching tuples with element-wise subtraction and element-wise multiplication. ESIM achieves 88.0% in accuracy on the SNLI test set, which exceeds the human performance (87.7%) for the first time. [9,32] both further improve the performance by taking the ESIM model as a baseline model. The studies related to "matching-aggregation" but without bidirectional interaction are not listed [27,34].

Motivated by the attention-based memory models and the bidirectional inter-attention models, we propose the MIMN model. The last group of models (13)–(16) are models described in this paper. Our single MIMN model obtains an accuracy of 88.3% on SNLI test set, which is comparable with the current state-of-the-art single models. The single MIMN model improves 0.3% on the test set compared with ESIM, which shows that multi-turn inference based on the matching features and memory achieves better performance. From model (14), we also observe that memory is generally beneficial, and the accuracy drops 0.8% when the memory is removed. This finding proves that the interaction between matching features is significantly important for the final classification. To explore the way of updating memory, we replace the update gate in MIMN with a ReLU layer to update the memory, which drops 0.1%.

To further improve the performance, an ensemble model MIMN is built for comparison. We design the ensemble model by simply averaging the probability distributions [36] of four MIMN models. Each of the models has the same architecture but initialized by different seeds. Our ensemble model achieves the state-of-the-art performance by obtains an accuracy of 89.3% on SNLI test set.

4.5 Experiments on MPE

The MPE dataset is a brand-new dataset for NLI with four premises, one hypothesis, and one label. In order to maintain the same data format as other textual entailment datasets (one premise, one hypothesis, and one label), we concatenate the four premises as one premise.

Table 3 shows the results of our models along with the published models on this dataset. LSTM is a conditional LSTM model used in [27]. WbW-Attention aligns each word in the hypothesis with the premise. The state-of-the-art model on MPE dataset is SE model proposed by [14], which makes four independent predictions for each sentence pairs, and the final prediction is the summation of four predictions. Compared with SE, our MIMN model obtains a dramatic improvement (9.7%) on MPE dataset by achieving 66.0% in accuracy.

To compare with the bidirectional inter-attention model, we re-implement the ESIM, which obtains 59.0% in accuracy. We observe that MIMN-memory model achieves 61.6% in accuracy. This finding implies that inferring the matching features by multi-turns works better than single turn. Compared with the ESIM, our MIMN model increases 7.0% in accuracy. We further find that the performance of MIMN achieves 77.9% and 73.1% in accuracy of entailment and contradiction respectively, outperforming all previous models. From the accuracy distributions on N, E, and C in Table 3, we can see that the MIMN model is good at dealing with entailment and contradiction while achieves only average performance on neural.

Consequently, the experiment results show that our MIMN model achieves a new state-of-the-art performance on MPE test set. All of our models perform well on the entailment label, which reveals that our models can aggregate information from multiple sentences for entailment judgment.

4.6 Experiments on SCITAIL

In this section, we study the effectiveness of our model on the SCITAIL dataset. Table 4 presents the results of our models and the previous models on this dataset. Apart from the results reported in the original paper [12]: Majority

Table 3. Performance on MPE. Models with # are reported from [14].

Models	Test (%acc)	N	E	C
LSTM#	53.5	**39.2**	63.1	53.5
WbW-Attention#	53.9	30.2	61.3	66.5
SE#	**56.3**	30.6	48.3	71.2
ESIM (our_imp)	59.0	34.1	68.3	65.1
MIMN	**66.0**	35.3	**77.9**	**73.1**
MIMN-memory	61.6	28.4	72.7	70.8
MIMN-gate+ReLU	64.8	37.5	77.9	69.1

Table 4. Performance on SCITAIL. Models with * are reported from [12].

Models	Valid (%acc)	Test (%acc)
Majority class*	63.3	60.3
decomposable attention*	75.4	72.3
ESIM*	70.5	70.6
Ngram*	65.0	70.6
DGEM*	79.6	77.3
CAFE [32]	–	83.3
MIMN	84.7	**84.0**
MIMN-memory	81.3	82.2
MIMN-gate+ReLU	83.4	83.5

class, ngram, decomposable attention, ESIM and DGEM, we compare further with the current state-of-the-art model CAFE [32].

We can see that the MIMN model achieves 84.0% in accuracy on SCITAIL test set, which outperforms the CAFE by a margin of 0.5%. Moreover, the MIMN-gate+ReLU model exceeds the CAFE slightly. The MIMN model increases 13.3% in test accuracy compared with the ESIM, which again proves that multi-turn inference is better than one-pass inference.

5 Conclusion

In this paper, we propose the MIMN model for NLI task. Our model introduces a multi-turns inference mechanism to process multi-perspective matching features. Furthermore, the model employs the memory mechanism to carry proceeding inference information. In each turn, the inference is based on the current matching feature and previous memory. Experimental results on SNLI dataset show that the MIMN model is on par with the state-of-the-art models. Moreover, our model achieves new state-of-the-art results on the MPE and the SCITAL datasets. Experimental results prove that the MIMN model can extract important information from multiple premises for the final judgment. And the model is good at handling the relationships of entailment and contradiction.

Acknowledgements. This work is funded by Beijing Advanced Innovation for Language Resources of BLCU, the Fundamental Research Funds for the Central Universities in BLCU (No. 17PT05) and Graduate Innovation Fund of BLCU (No. 18YCX010).

References

1. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR,abs/1409.0473, September 2014

3. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: EMNLP (2015)
4. Chen, Q., Zhu, X.D., Ling, Z.H., Wei, S., Jiang, H., Inkpen, D.: Recurrent neural network-based sentence encoder with gated attention for natural language inference. In: EMNLP (2017)
5. Chen, Q., Zhu, X., Ling, Z.H., Wei, S., Jiang, H., Inkpen, D.: Enhanced LSTM for natural language inference. In: ACL, Vancouver, pp. 1657–1668, July 2017
6. Cheng, J., Dong, L., Lapata, M.: Long short-term memory-networks for machine reading. In: EMNLP, Austin, pp. 551–561, November 2016
7. Dagan, I., Glickman, O., Magnini, B.: The PASCAL recognising textual entailment challenge. In: Quiñonero-Candela, J., Dagan, I., Magnini, B., d'Alché-Buc, F. (eds.) MLCW 2005. LNCS (LNAI), vol. 3944, pp. 177–190. Springer, Heidelberg (2006). https://doi.org/10.1007/11736790_9
8. Ghaeini, R., et al.: DR-BiLSTM: dependent reading bidirectional LSTM for natural language inference. arXiv preprint arXiv:1802.05577 (2018)
9. Glickman, O., Dagan, I.: A probabilistic setting and lexical cooccurrence model for textual entailment. In: Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, EMSEE 2005 (2005)
10. Gong, Y., Luo, H., Zhang, J.: Natural language inference over interaction space. In: ICLR (2018)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
12. Khot, T., Sabharwal, A., Clark, P.: SciTail: a textual entailment dataset from science question answering. In: AAAI (2018)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR,abs/1412.6980 (2014)
14. Lai, A., Bisk, Y., Hockenmaier, J.: Natural language inference from multiple premises. In: IJCNLP, pp. 100–109 (2017)
15. Liu, P., Qiu, X., Chen, J., Huang, X.: Deep fusion LSTMs for text semantic matching. In: ACL, pp. 1034–1043 (2016)
16. Liu, P., Qiu, X., Zhou, Y., Chen, J., Huang, X.: Modelling interaction of sentence pair with coupled-LSTMs. In: EMNLP (2016)
17. Liu, Y., Sun, C., Lin, L., Wang, X.: Learning natural language inference using bidirectional LSTM model and inner-attention. CoRR,abs/1605.09090 (2016)
18. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: EMNLP, pp. 1412–1421 (2015)
19. MacCartney, B., Galley, M., Manning, C.D.: A phrase-based alignment model for natural language inference. In: EMNLP 2008 (2008)
20. Marelli, M., Menini, S., Baroni, M., Bentivogli, L., Bernardi, R., Zamparelli, R.: A sick cure for the evaluation of compositional distributional semantic models. In: LREC (2014)
21. McCann, B., Bradbury, J., Xiong, C., Socher, R.: Learned in translation: contextualized word vectors. arXiv preprint arXiv:1708.00107 (2017)
22. Mou, L., et al.: Natural language inference by tree-based convolution and heuristic matching. In: ACL (2016)
23. Munkhdalai, T., Yu, H.: Neural semantic encoders. In: EACL, pp. 397–407 (2016)
24. Nie, Y., Bansal, M.: Shortcut-stacked sentence encoders for multi-domain inference. In: RepEval@EMNLP (2017)
25. Parikh, A., Täckström, O., Das, D., Uszkoreit, J.: A decomposable attention model for natural language inference. In: EMNLP, Austin, pp. 2249–2255, November 2016

26. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: EMNLP (2014)
27. Rocktäschel, T., Grefenstette, E., Hermann, K.M., Kocisky, T., Blunsom, P.: Reasoning about entailment with neural attention. In: ICLR (2016)
28. Sha, L., Chang, B., Sui, Z., Li, S.: Reading and thinking: re-read LSTM unit for textual entailment recognition. In: COLING, pp. 2870–2879, December 2016
29. Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., Zhang, C.: DiSAN: directional self-attention network for RNN/CNN-free language understanding. CoRR,abs/1709.04696 (2017)
30. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
31. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: ACL, Beijing, pp. 1556–1566, July 2015
32. Tay, Y., Tuan, L.A., Hui, S.C.: A compare-propagate architecture with alignment factorization for natural language inference. arXiv preprint arXiv:1801.00102 (2017)
33. Wang, S., Jiang, J.: A compare-aggregate model for matching text sequences. CoRR,abs/1611.01747 (2016)
34. Wang, S., Jiang, J.: Learning natural language inference with LSTM. In: NAACL, San Diego, pp. 1442–1451, June 2016
35. Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M.: Gated self-matching networks for reading comprehension and question answering. In: ACL, pp. 189–198 (2017)
36. Wang, Z., Wael, H., Radu, F.: Bilateral multi-perspective matching for natural language sentences. In: IJCAI, pp. 4144–4150 (2017)
37. Weston, J., Chopra, S., Bordes, A.: Memory networks. In: ICLR (2015)
38. Yu, H., Munkhdalai, T.: Neural tree indexers for text understanding. In: ACL, vol. 1, pp. 11–21 (2017)

From Humour to Hatred: A Computational Analysis of Off-Colour Humour

Vikram Ahuja(✉), Radhika Mamidi, and Navjyoti Singh

International Institute of Information Technology, Hyderabad, India
vikram.ahuja@research.iiit.ac.in, radhika.mamidi@iiit.ac.in

Abstract. Off-colour humour is a category of humour which is considered by many to be in poor taste or overly vulgar. Most commonly, off-colour humour contains remarks on particular ethnic group or gender, violence, domestic abuse, acts concerned with sex, excessive swearing or profanity. Blue humour, dark humour and insult humour are types of off-colour humour. Blue and dark humour unlike insult humour are not outrightly insulting in nature but are often misclassified because of the presence of insults and harmful speech. As the primary contributions of this paper we provide an original data-set consisting of nearly 15,000 instances and a novel approach towards resolving the problem of separating dark and blue humour from offensive humour which is essential so that free-speech on the internet is not curtailed. Our experiments show that deep learning methods outperforms other n-grams based approaches like SVM's, Naive Bayes and Logistic Regression by a large margin.

Keywords: Off-colour humour · Deep learning · Insult detection

1 Introduction

In the last decade, there has been an exponential increase in the volume of social media interactions (twitter, reddit, facebook etc.). It took over three years, until the end of May 2009, to reach the billionth tweet [1]. Today, it takes less than two days for one billion tweets to be sent. Social media has increasingly become the staple medium of communication via the internet. However, due to the non-personal nature of online communication, it presents a unique set of challenges. Social media has become a breeding ground for hate speech and insults as there is a lack of accountability that can be abused.

We need to discern between content which is an honest attempt at humour as opposed to content which is purely derogatory and insulting. Humour is an essential part of communication and allows us to convey our emotions and feelings. Humour is the tendency of particular cognitive experiences to provoke laughter and provide amusement.

Humour that is sometimes considered to be purely offensive, insulting or a form of hate speech is described as off-colour humour. Off-colour humour (also known as vulgar humour, crude humour, or shock humour) is humour that deals with topics that may be considered to be in poor taste or overly vulgar. It primarily consists of three sub-categories, dark humour, blue humour and insult humour.

Dark Humour has been more frequently discussed in literary, social research as well as psychology but not much attention has been given to it in linguistics. It is a comic style that makes light of subject matter that is generally considered taboo, particularly subjects that are normally considered serious or painful to discuss such as death as defined by wikipedia. Dark humour aims at making fun of situations usually regarded as tragic, such as death, sickness, disability, and extreme violence, or of the people involved or subject to them [4]. It is inspired by or related to these tragic events and do not in any way make fun of them [5]. In dark humour a gruesome or a tragic topic is mixed with an innocuous topic which creates shock and inappropriateness. This invoked inappropriateness or shock generally amusing to the listeners [6]. Dynel [7] in their paper show that dark humour inspired by tragic events such as a terrorist attacks just addresses topics tangential to them and do not in any way make fun of them directly. Blue humour is a style of humour that is indecent or profane and is largely about sex. It contains profanity or sexual imagery that may shock. It is also referred to as Ribaldry. Insult humour is that kind of humour which consists of offensive insults directed to a person or a group. Roasting is a form of insult comedy in which specific individual, a guest of honor, is subjected to jokes at their expense, intended to amuse the event's wider audience as defined by Wikipedia. All the three categories mentioned above seem to be interrelated to each other but have very fine differences. Dark humour is different from straightforward obscenity (blue humour) in the way that it is more subtle. Both dark and blue humour are different from insult humour in the sense that there is no intent of offending someone in the former two whereas in insult humour the main aim is to jokingly offend or insult the other person or a group of people [8].

People often get offended on such misunderstood instances of humour more than would otherwise be the case. The significance of our contribution can be fully conceived only when we realise that such occurrences can lead to gratuitous censorship and therefore curtailment of free speech. It is in this context that we are trying to formulate our problem of separating dark humour and blue humour from insult humour. In short our contributions can be summarised as follows:

- We present a dataset of nearly 15,000 jokes out of which 4,000 are of positive types.
- A novel approach towards resolving the problem of separating dark and blue humour from offensive humour.

The remainder of the paper is structured as follows. Section 2 gives a detail about related work along with it's criticisms. Section 3 presents the proposed framework. Section 4 presents the dataset used, Sect. 5 presents the experiments used. Section 6 gives the results and analysis of the experiment conducted in this study and Sect. 7 concludes the paper (Table 1).

Table 1. Few examples of jokes used in our dataset

Dark joke	1. My girlfriend is a porn star. She will kill me if she finds out 2. When someone says, "Rape jokes are not funny," I don't care. It's not like I asked for their consent anyway
Blue joke	1. Sex is not the answer. Sex is the question. "Yes" is the answer 2. How can you tell if a man is sexually excited? He's breathing
Insult joke	1. Your mama so fat when she stepped on the weighing scale it said: "I need your weight, not your phone number." 2. You were beautiful in my dreams, but a fucking nightmare in reality
Normal/safe joke	1. What's red and bad for your teeth? A brick 2. What did the German air force eat for breakfast during WW2? Luftwaffle

2 Related Work

Humour has always been an important topic for researchers. There has been a lot of study in humour in the field of linguistics, literature, neuroscience, psychology and sociology. Research in humour has revealed many different theories of humour and many different kinds of humour including their functions and effects personally, in relationships, and in society. For the scope of this paper we are restricting ourselves to off-colour humour that has been explained in the above sections.

There has been some studies on offensive humour which is usually used a form of resistance in tragic situations. Weaver [10] in their paper talks how racism could be undermined by using racial stereotypes by blacks and minority ethnic comedians. Lockyer [11] in their study analyse how disabled comedians have also ridiculed stereotypes of the disabled by reversing the offensive comments of the non-disabled.

The study by Billig [12] examines the relationship between humour and hatred, which it claims is the topic that is often ignored by researchers of prejudice. It analyses websites that present racist humour and display sympathies with the Ku Klux Klan. The analysis emphasizes the importance of examining the *metadiscourse*, which presents and justifies the humour and also suggests that the extreme language of racist hatred is indicated to be a matter for enjoyment. In the book "Jokes and their Relation to the Unconscious", Freud refers to off-colour humour as the "economy of pity" and claims that it is "one of the most frequent sources of humourous pleasure" and these jokes (off-colour) provides a socially accepted means of breaking taboos, particularly in relation to sex and aggression. In [7] the authors analyse the importance of off colour humour (dark humour) in post terrorist attack discourse. The paper claims that dark humour is a coping mechanism under oppressive regimes and in crisis situations. Davies [14] in his book argues that those who engage in racist and sexist jokes do not necessarily believe the stereotypes that the jokes express. Maxwell [15] in his paper brings forth the importance of dark humour as a cognitive and/or behavioral coping strategy which is considered to be a reaction to a traumatic event and proposes a model including progressive steps of humour, ranging from respectful to sarcastic.

Similarly there has been a lot of studies in the field of insult detection. Mahmud et al. [16] in their paper create a set of rules to extract the semantic information of a given sentence from the general semantic structure of that sentence to separate information from abusive language but is very limited in the sense that this system can annotate and distinguish any abusive or insulting sentence only bearing related words or phrases that must exist in the lexicon entry. So, it only looks at insulting words and not sentences that are used in an insulting manner.

Xiang et al. [17] in their work dealt with offensive tweets with the help of Topical Feature Discovery over a Large Scale Twitter Corpus by using Latent Dirichlet Allocation model. The work by Ravazi et al. [18] describe an automatic flame detection method which extracts features at different conceptual levels and applies multilevel classification for flame detection but is very limited due to the dataset used by them and does not consider the syntactical structure of the messages explicitly.

The above works tell us there has been quite a lot of studies in both these fields but no such computational study in the intersection of these two topics. This study is the first such attempt which tries to create a separating boundary between different types of off-colour humor and insults. We discuss our framework used to separate different types of off-colour humour in the next section.

3 Proposed Framework

The domain of jokes we are dealing with, viz. dark humour, blue humour and insult humour all are generally classified under the umbrella category of NSFW or Off-Colour Humour. It is due to their apparent similarities that on one glance they can be dismissed as being of one and the same type. As we go to finer levels of granularity it becomes evident that two separate buckets can be defined even inside off-colour humour, one pertaining to insults resulting in insult humour and the other consisting of dark humour and blue humour as shown in the Fig. 1 below. Insults being the common denominator does not mean all insults and non-jokes can be classified as insult humour, thus defining a demarkation separating the two is also required.

As mentioned above we are able to define clear boundaries between within off-colour humour between insulting and non-insulting humour. But in order to further differentiate between dark and blue humour we identify more features that can give us a clear distinction between the two. One of the primary indicators in helping us draw this line is the ability to detect and extract sexual terms leading us to blue humour and dark themes such as violence (murder, abuse, domestic violence, rape, torture, war, genocide, terrorism, corruption), discrimination (chauvinism, racism, sexism, homophobia, transphobia), disease (anxiety,

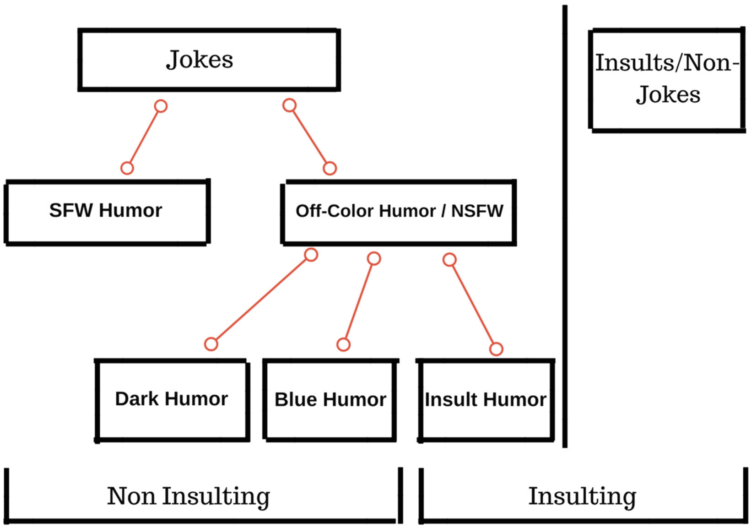
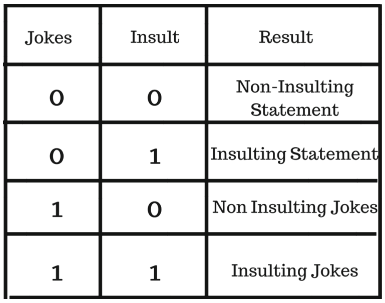


Fig. 1. Differentiating between Insulting and Non-Insulting Humor

depression, suicide, nightmares, drug abuse, mutilation, disability, terminal illness, insanity), sexuality (sodomy, homosexuality, incest, infidelity, fornication), religion and barbarism [2] leading us to dark humour. In order to define strong outlines we also ensure that even if an insulting joke or an insult contains sexual content or dark themes the primary focus of such content is on the insult part and not its sexual content or dark theme (which more than anything aides in providing a backdrop).



Jokes	Insult	Result
0	0	Non-Insulting Statement
0	1	Insulting Statement
1	0	Non Insulting Jokes
1	1	Insulting Jokes

Fig. 2. In this figure Dark and Blue Humor belongs to the category of Non-Insulting Jokes which are differentiated by dark and blue humor features mentioned in the above paragraph

The focus of this paper is to classify between these three categories of off-colour and hence, integrating the task of classification of text between humourous and non-humourous has been deemed out of scope and is limited by the creation of a dataset which consists of only the humourous content.

4 Dataset

To test our hypothesis that automatic classification models can classify between dark humour, blue humour, Insult humour and (normal humour), we needed a dataset consisting of all types of above mentioned examples. Since there is no such corpus available for the task because of limited study in this field, we collected and labeled our own data. The only data source that was available is the twitter dataset [17] which has been used to detect offensive tweets but due to the fact it was very limited in terms of themes and could not have been used for our study.

The dataset that we created consisted of multiple one-liners. We used one-liners because they are very small generally and must produce humourous effect, unlike longer jokes which usually have a relatively complex narrative structure. These characteristics make this type of humour particularly suitable for use in an automatic learning setting. The dataset is defined as follows:

– **Insult jokes:** We collected multiple one-liners jokes from the subreddits /r/insults and /r/roastme. Apart from those we also mined various jokes websites and collected jokes with tags "Insult". After removing duplicates and verifying manually we were left with nearly 4000 jokes belonging to the category of insult jokes.
– **Dark Jokes:** We collected multiple jokes from the subreddit /r/darkjokes and /r/sickipedia. These subreddits contains one-liner jokes which are highly moderated. Apart from that we mined various jokes websites and collected jokes with the tags dark, darkjokes and darkhumour. After removing the duplicates and manual verification we were left with a final dataset of approximately 3500 jokes under the category of dark jokes.
– **Blue Jokes:** Blue jokes are the types of jokes which are most famous on the internet. Since these types of jokes are mainly associated with heavy nudity, sexual content and slangs we collected one liner jokes from subreddit /r/dirtyjokes and apart from that we took jokes from various jokes websites with tags NSFW, dirty, adult and sexual. After duplicates removal and manual verification we were left with approximately 2500 jokes under the category of blue jokes.
– **Normal Jokes/Safe jokes:** We collected jokes from subreddits r/cleanjokes and /r/ oneliners. These subreddits contain clean, non offensive jokes and non disrespectful jokes. Jokes in this category does not belong to any of the above category. This types of jokes are referred to as SFW (safe for work) jokes for all future references. After collecting these jokes we searched for insult words (talk about a lexical dictionary from the paper opened in one of the tabs). After duplicates removal we were left were approximately 5000 jokes under this category.

The dataset collected is important because of the fact that it is multidimensional in the sense that it contains insulting and non insulting jokes as well jokes with taboo topics (mentioned in the above section) as well as jokes without these topics. This leaves us with a combined total of nearly 4000 jokes under the category of insult jokes, 3500 under the category of dark jokes (non-insult), 2500 jokes in the category of blue jokes (non-insult) and 5000 jokes from the category of safe jokes (non insult). Thus we have a 4000 positive examples and 11,000 negative examples. All the dataset that has been mined has been taken from websites which have strict moderation policies, thus leaving very little room for error in our dataset.

5 Experiment

Treating the problem of separating different types of jokes as a classification problem, there are wide variety of methods that can be used which can greatly affect the results. Some of the features explicitly used were:

- Dark jokes are usually limited to or their main topic is violence (murder, abuse, domestic violence, rape, torture, war, genocide, terrorism, corruption), discrimination (chauvinism, racism, sexism, homophobia, transphobia), disease (anxiety, depression, suicide, nightmares, drug abuse, mutilation, disability, terminal illness, insanity), sexuality (sodomy, homosexuality, incest, infidelity, fornication), religion and barbarism. In order to detect these relations a common sense engine called "Concept Net" [19]. It is a multilingual knowledge base, representing words and phrases that people use and the common-sense relationships between them. Concept Net was used too see use of words related to the above mentioned topics.
- Sentiment score of every joke was calculated because of the hypothesis that off-colour humour tends to have a negative sentiment compared to jokes without any vulgar or any such topics mentioned above.
- It is our hypothesis that most of the insult jokes have mainly first (self-deprecating humour) and second (directed towards someone) person feature words like "I", "you", "your". This is done in order to detect insulting jokes with contains phrases like "your mother", "your father" which are usually meant to be insults.

After preprocessing of the data we extracted n-grams from the dataset, precisely speaking unigrams, bigrams and trigrams and a feature dictionary was created using of these collected ngrams. We compare results from five different classification algorithms mentioned below with different settings along with features mentioned above.

- We used LDA, which provides a transitive relationship between words such that each document has a set of words and each word belong to multiple topics (here categories of jokes) which transitively indicates that each document is a collection of topics.

– Used n-grams trained a logistic regression and naive bayes with it.
– Along with this we used brown clustering which helped us to put similar kinds of words in same cluster and trained a SVM with it.
– We also experimented with CNN's like Kim et al. [20] work which uses CNN for sentence classification. A model as shown in [20] was created with pre-trained vectors from *word2vec*, which has been trained on google news corpus and the vectors have dimensionality of 300.

Various experiments were performed on our dataset. For evaluation metrics, the dataset was randomly divided into 90% training and 10% testing. All the experiments were performed 10 fold and the final result was then taken to be average of those results.

6 Analysis

We can see the results of our classifiers in the Table 2 below. In the case of Logistic Regression, the introduction of features suggested proved to be an important factor as it increased the accuracy by nearly 10%. LDA had a slight better result than Logistic Regression without any features but is outperformed by Logistic Regression when the features such as sentiment scores, first, second person features and dark words are added. Naive bayes outperforms both LDA and Logistic Regression when features such as sentiment scores, first and second person features and dark word are not added, but we see equal results when those are added. SVM outperforms LDA, Naive bayes and Logistic Regression by a big margin and thus proving to be a better algorithm to classify. We see that the feature set used proved to be a very valuable addition to our experiment and an increase in accuracy in every case when those features are introduced. Finally, CNN's are introduced along with word2vec outperforms every other classifier used in this study. Thus we achieve the best accuracy of 81% using CNN's with *word2vec*.

Table 2. Table showing the accuracies of various classifiers used

Results	
Features	Accuracy
Logistic Regression	59%
LR + Ngrams	62%
LR + Ngrams + features mentioned	69%
LDA	61%
Naive Bayes + Ngrams + features mentioned	69%
SVM	68%
SVM + Ngrams + features	74%
CNN + word2vec	81%

7 Future Work

Given the constraints of the scope of the paper as well as the research conducted we have not attempted to integrate the differentiate humourous and non-humourous text in our study. This could also be incorporated in the pipeline to match with other studies. Also, in this paper we have restricted sexual topics in dark humour (not to be confused with sexuality in blue humour) but in reality, there are some dark jokes which have some common features with blue jokes or talks about nudity and profanity. This can be taken up for future work and this whole system could be implemented on various social media platforms to a more holistic classification to insults in social media and practice free speech.

References

1. numbers. Twitter Official Blog, 14 March 2011
2. Black Comedy. https://en.wikipedia.org/wiki/Black_comedy
3. Ribaldry. https://en.wikipedia.org/wiki/Ribaldry
4. Bucaria, C.: Dubbing dark humour: a case study in audiovisual translation. Lodz Pap. Pragmat. **4**(2), 215–240 (2008)
5. Lewis, P.: Three Jews and a blindfold: the politics of gallows humour. In: Avner, Z., Anat, Z. (eds.) Semites and Stereotypes: Characteristics of Jewish Humour, pp. 47–58. Greenwood Press, Westport (1993)
6. Aillaud, M., Piolat, A.: Influence of gender on judgment of dark and non-dark humour. Individ. Differ. Res. **10**(4), 211–222 (2012)
7. Dynel, M., Poppi, F.I.M.: In tragoedia risus, analysis of dark humour in post-terrorist attack discourse. Discourse Commun. (2018)
8. Kuipers, G.: 'Where was King Kong when we needed him?' Public discourse, digital disaster jokes, and the functions of laughter after 9/11 (2011)
9. Gournelos, T., Greene, V., (eds.): A Decade of Dark Humour: How Comedy, Irony and Satire Shaped Post-9/11 America, pp. 20–46. University Press of Mississippi, Jackson
10. Weaver, S.: Developing a rhetorical analysis of racist humour: examining anti-black jokes on the Internet. Soc. Semiot. **20**(5), 537–555 (2010). https://doi.org/10.1080/10350330.2010.513188
11. Lockyer, S.: From comedy targets to comedy-makers: disability and comedy in live performance. Disabil. Soc. **30**(9), 1397–1412 (2015). https://doi.org/10.1080/09687599.2015.1106402
12. Billig, M.: Humour and hatred: the racist jokes of the Ku Klux Klan. Discourse Soc **12**(3), 267–289 (2001)
13. Freud, S.: Jokes and their relation to the unconscious. WW Norton & Company (1960)
14. Davies, C.: Ethnic humour around the world: a comparative analysis. Indiana University Press (1990)
15. Maxwell, W.: The use of gallows humour and dark humour during crisis situations. Int. J. Emerg. Ment. Health **5**(2), 93–98 (2003)
16. Mahmud, A., Ahmed, K.Z., Khan, M.: Detecting flames and insults in text (2008)
17. Xiang, G., Hong, J., Rose, C.P.: Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In: Proceedings of the 21st ACM Conference on Information and Knowledge Management, Sheraton, Maui Hawaii, 29 October–2 November 2012 (2012)

18. Razavi, A.H., Inkpen, D., Uritsky, S., Matwin, S.: Offensive language detection using multi-level classification. In: Farzindar, A., Kešelj, V. (eds.) AI 2010. LNCS (LNAI), vol. 6085, pp. 16–27. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13059-5_5
19. Liu, H., Singh, P.: Concept net - a practical commonsense reasoning tool-kit. BT Technol. J. **22**(4), 211–226 (2004)
20. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP, pp. 1746–1751 (2014)

Classification of the Structure of Square Hmong Characters and Analysis of Its Statistical Properties

Li-Ping Mo(✉), Kai-Qing Zhou, Liang-Bin Cao, and Wei Jiang

College of Information Science and Engineering,
Jishou University, Jishou 416000, Hunan, China
zmx89@jsu.edu.cn

Abstract. Analysis of the character structure characteristics can lay an information foundation for the intelligent processing of square Hmong characters. Combined with the analysis of character structure characteristics, this paper presents a definition of the linearization of square Hmong characters, a definition of equivalence class division of the structure of square Hmong characters, and proposes a decision algorithm of structure equivalence class. According to the above algorithm, the structure of square Hmong characters is divided into eight equivalent classes. Analysis of the statistical properties, including the cumulative probability distribution, complexity, and information entropy of square Hmong characters appearing in practical documents, shows that, first, more than 90% of square Hmong characters appearing in practical documents are composed of two components, and more than 80% of these characters possess a left-right, top-bottom, or lower-left-enclosed structure, second, the number of mean components in a square Hmong character is slightly greater than 2, third, the information entropy of the structure of Hmong characters is within the interval (1.19, 2.16). Results reveal that square Hmong characters appearing frequently in practical documents follow the principle of simple structure orientation.

Keywords: Information entropy · Probability distribution
Square Hmong character · Statistical analysis

1 Introduction

Analysis of character structure characteristics is the basis of character attributes analysis, and plays an important role in the intelligent generation and character recognition. In the field of Chinese information processing, many methods for intelligent character generation, recognition and other related processing methods based on the analysis of character structure characteristics were proposed. Instituse put for-ward a refinement method for smoothing Chinese character images using the nearest neighbor pixel correlation function to filter out noise that affects the structure characteristics of Chinese characters [1]. This method uses the nearest neighbor pixel correlation function to filter out noise that affects the structure characteristics of Chinese characters. Moreover, according to the structure characteristics of Chinese characters, Shin et al. proposed a method for generating handwritten Chinese characters on the basis of stroke correspondence [2]. Liu et al. presented a method for acquiring knowledge related to the basic elements required for intelligent generation of Chinese characters [3]. Tan et al. analyzed the structure characteristics of Chinese characters and presented a method of radical extraction using affine sparse matrix factorization for printed Chinese character recognition [4]. Dobres et al. discussed the influence of factors, such as font design difference, fine stroke difference, and color contrast polarity difference, on the scanning time of readable subtitles by analyzing the structure characteristics of Chinese characters [5].

In the past few decades, studies on the structure characteristics of Tibetan and Korean characters had also explored. Ai et al. carried out a statistical analysis of the glyph structure of Tibetan characters [6]. Cai et al. classified the structure of Tibetan characters and developed a statistical system model of glyph characters by analyzing the structure characteristics of Tibetan [7]. Kwon used the division of the Korean character structure to achieve rough classification in the process of syllable matching to preprocess alphabet division in the Korean character matching process [8]. Xu et al. utilized the rules of character structure to implement post-processing of handwritten Korean text [9]. Meanwhile, Cui et al. calculated the contribution of the information provided by letters at different positions in the spatial structure to the structure classification of Korean characters, further explored the distribution structure of Korean text information [10].

Square Hmong characters are ideograms with a fixed structure, and they are commonly used in the daily life of Hmong people in the Wuling mountain area of China. Information processing for square Hmong characters has not been extensively studied. Mo et al. conducted a number of studies on computer coding, keyboard input, glyph generation, and font creation of square Hmong characters [11–13]. However, the structure characteristics of square Hmong characters have not been investigated so far.

To address this limitation, this paper presents a classification method for square Hmong characters on the basis of structural distance. The method is implemented based on a linear representation of square Hmong characters and provides a convenient means to analyze the statistical characteristics of the structure of square Hmong characters in practical documents.

The rest of this paper is organized as follows. Section 2 introduces square Hmong characters and their linear representation. Section 3 introduces the proposed classification method and decision algorithm for the structure of square Hmong characters. Section 4 presents an analysis of the statistical characteristics of the structure of square Hmong characters in practical documents. Section 5 provides the conclusions.

2 Square Hmong Characters and Their Linear Representation

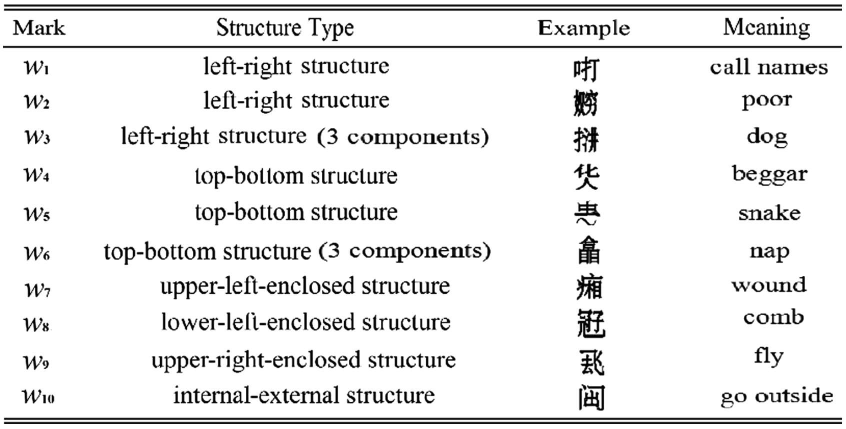
2.1 Word Information Principles of Square Hmong Characters

Square Hmong characters were created in the late Qing Dynasty for use by local Hmong people to record Hmong songs. According to [14, 15], the word information principles of square Hmong characters can be summarized from two aspects.

(1) Several simple Chinese characters, Chinese radicals, and symbols without pronunciation and meaning (such as ~ and X) are used to represent the component of phonetic-symbol, meaning-symbol, or shape-symbol.
(2) The "one word and one syllable" method is utilized to mark a morpheme or word.

The structure of square Hmong characters can be divided into four types, namely, left-right, top-bottom, part-enclosed, and internal-external structures. The part-enclosed structure can be further classified into upper-left-enclosed, lower-left-enclosed, and upper-right-enclosed types.

Several typical examples of square Hmong characters with different structures are listed in Fig. 1.



Mark	Structure Type	Example	Meaning
w_1	left-right structure	[illegible]	call names
w_2	left-right structure	[illegible]	poor
w_3	left-right structure (3 components)	[illegible]	dog
w_4	top-bottom structure	[illegible]	beggar
w_5	top-bottom structure	[illegible]	snake
w_6	top-bottom structure (3 components)	[illegible]	nap
w_7	upper-left-enclosed structure	[illegible]	wound
w_8	lower-left-enclosed structure	[illegible]	comb
w_9	upper-right-enclosed structure	[illegible]	fly
w_{10}	internal-external structure	[illegible]	go outside

Fig. 1. Examples of square Hmong characters in different structures.

According to the word information principle, if a square Hmong character consists of three or more components, two or three of these components can be combined into a simple Chinese character. This Chinese character is regarded as a component of the square Hmong character. A statistical analysis of the 1,129 square Hmong characters indicated that most Hmong characters are composed of two components, and only a few Hmong characters with left-right and top-bottom structures are composed of three components.

2.2 Linearization Representation of Square Hmong Characters

Definition 1 Linearization of square Hmong characters: The process of decomposing a square Hmong character into a uniquely identified component sequence according to its spelling order is called linearization.

To unify the description, we decompose all square Hmong characters into component sequences with a uniform length. When a component of a position in the sequence does not exist, the symbol "ε" is used to replace the missing component. According to the 16 types of components shown in Table 1, the uniform length of a component sequence can be 16 when a square Hmong character is linearized.

Table 1. Components of square Hmong characters

Number	Name	Mark
1	left component	C_{l}
2	right component	C_{r}
3	top component	C_{t}
4	bottom component	C_{b}
5	upper-left-external component	C_{olu}
6	lower-right-internal component	C_{ird}
7	lower-left-external component	C_{old}
8	upper-right-internal component	C_{iru}
9	upper-right-external component	C_{oru}
10	lower-left-internal component	C_{ild}
11	external component	C_{oa}
12	internal component	C_{ia}
13	right-left component	C_{rl}
14	right-right component	C_{rr}
15	bottom-left component	C_{bl}
16	bottom-right component	c_{br}

Considering that a two-component square Hmong character can be regarded as a three-component square Hmong character lacking a component, the right component (C_{r}) can be regarded as a right-left component (C_{rl}) when the right-right component (C_{rr}) is absent. The bottom component (C_{b}) can also be regarded as a bottom-left component (C_{bl}) when the right-bottom-right component (C_{br}) is absent. Therefore, only 14 types of components are required. That is, any square Hmong character can be decomposed into a 14-component combination sequence.

2.3 Linearization Function and Component Extraction Function

Definition 2 Linearization function of square Hmong characters: Assuming that Σ is a finite set of square Hmong characters, w is a square Hmong character ($w \in \Sigma$), and C_i (i = 1, 2, …, 14) is a finite set of components, the mapping (f) in Eq. (1) is called the linearization function of square Hmong characters.

$$f : \Sigma \rightarrow C_{\mathrm{l}} \times C_{\mathrm{rl}} \times C_{\mathrm{rr}} \times C_{\mathrm{t}} \times C_{\mathrm{bl}} \times C_{\mathrm{br}} \times C_{\mathrm{olu}} \times C_{\mathrm{ird}} \times C_{\mathrm{old}} \times C_{\mathrm{iru}} \times C_{\mathrm{oru}} \times C_{\mathrm{ild}} \times C_{\mathrm{oa}} \times C_{\mathrm{ia}} \quad (1)$$

A component combination sequence is practically a regular expression. Thus, f is a regular replacement from Σ to C_i that converts a square Hmong character into a uniquely identified component sequence. Assume that the components included in w are marked as s_{i} (i = 1, …, 14) and arranged in the order of "C_{l}-C_{rl}-C_{rr}-C_{t}-C_{bl}-C_{br}-C_{olu}-C_{ird}-C_{old}-C_{iru}-C_{oru}-C_{ild}-C_{oa}-C_{ia}." Then, $f(w)$ can be expressed by

$$f(w) = s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} \quad (2)$$

Considering that a square Hmong character contains only two or three components, valid components can only be found in two or three positions. Other locations are empty and marked as "ε". Figure 2 shows the linearization of the 10 characters in Fig. 1.

Square Hmong character	Linearization function	Components sequence													
		s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}
w_1	$f(w_1)$	布	兑	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε
w_2	$f(w_2)$	身	左	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε
w_3	$f(w_3)$	口	目	墨	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε
w_4	$f(w_4)$	ε	ε	ε	知	面	ε	ε	ε	ε	ε	ε	ε	ε	ε
w_5	$f(w_5)$	ε	ε	ε	化	人	ε	ε	ε	ε	ε	ε	ε	ε	ε
w_6	$f(w_6)$	ε	ε	ε	合	目	目	ε	ε	ε	ε	ε	ε	ε	ε
w_7	$f(w_7)$	ε	ε	ε	ε	ε	ε	厂	朗	ε	ε	ε	ε	ε	ε
w_8	$f(w_8)$	ε	ε	ε	ε	ε	ε	ε	ε	色	白	ε	ε	ε	ε
w_9	$f(w_9)$	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	飞	去	ε	ε
w_{10}	$f(w_{10})$	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	门	出

Fig. 2. Examples of linear representation of square Hmong characters.

Definition 3 Component extraction function of square Hmong characters: The function of extracting the *i*-th component from linearization result *f*(*w*) is called the extraction function of component s_i. It is denoted as $f_i(w)$ and expressed as

$$f_i(w) = s_i (i = 1, 2, \ldots, 14) \tag{3}$$

With the component extraction function, the linearization result *f*(*w*) of *w* can also be expressed by

$$f(w) = f_1(w)f_2(w)f_3(w)f_4(w)f_5(w)f_6(w)f_7(w)f_8(w)f_9(w)f_{10}(w)f_{11}(w)f_{12}(w)f_{13}(w)f_{14}(w) \tag{4}$$

3 Equivalence Class of the Structure of Square Hmong Characters and Its Judgment

3.1 Definition of Structural Distance

Structural distance is used to measure structural differences between different characters. The structural distance of square Hmong characters is defined as follows.

Definition 4 Structural distance of square Hmong characters: Given square Hmong characters w_1 and w_2, the structural distance between w_1 and w_2 is expressed as *Distance*(w_1, w_2). The calculation formula of *Distance*(w_1, w_2) is shown in Eq. (5).

$$Distance(w_1, w_2) = \sum_{i=1}^{14} f_i(w_1)\theta f_i(w_2) \tag{5}$$

Where, operation θ is defined as

$$x\,\theta\,y = \begin{cases} 1, & \text{for } \forall(x,y) \in \{(x,y)|(x = \varepsilon \text{ and } y \neq \varepsilon) \text{ or } (x \neq \varepsilon \textit{ and } y = \varepsilon)\} \\ 0, & \text{for } \forall(x,y) \in \{(x,y)|(x \neq \varepsilon \text{ and } y \neq \varepsilon) \text{ or } (x = \varepsilon \textit{ and } y = \varepsilon)\} \end{cases} \tag{6}$$

Equations (5) and (6) show that the distance between two characters with the same structure is 0, and the distance between two characters with a different structure is a positive integer. For example, $Distance(w_1, w_2) = 0$, $Distance(w_1, w_3) = 1$, $Distance(w_1, w_4) = 4$, $Distance(w_1, w_6) = 5$, $Distance(w_1, w_7) = 4$, $Distance(w_3, w_6) = 6$, $Distance(w_3, w_7) = 5$, and $Distance(w_4, w_5) = 0$.

3.2 Structure Equivalence Class of Square Hmong Characters and Its Division

Definition 5 Equivalent structure relation: Given square Hmong characters w_1 and w_2, when $Distance(w_1, w_2) = 0$, w_1 and w_2 have an equivalent structure relation.

Definition 6 Structure equivalence class of square Hmong characters: As shown in Eq. (7), a given Hmong document (D) is divided into m disjoint subsets D_i ($i = 1, 2, \ldots, m$) according to the equivalence structure relation.

$$\begin{cases} D = D_1 \cup D_2 \cup \ldots \cup D_m \\ D_i \cap D_j = \phi, \text{ for all } i \neq j \ (i,j = 1, 2, \ldots, m) \end{cases} \tag{7}$$

If it exists,

$$Distance(w_1, w_2)\left|\begin{array}{lll} = & 0, & \text{for } \forall w_1, w_2 \in D_i \\ > & 0, & \text{for } \forall w_1 \in D_i, \forall w_2 \in D_j \ (i \neq j;\ i,j = 1, 2, \ldots, m) \end{array}\right. \tag{8}$$

Then, D_i ($i = 1, 2, \ldots, m$) is the i-th structure equivalence class. Square Hmong characters belonging to the same set D_i have the same structure. Square Hmong characters belonging to different D_i have different structures.

Eight equivalence classes of the structure of square Hmong characters can be calculated by using Eq. (8). The structure description of each equivalence class is shown in Table 2.

Table 2. Equivalence classes of the structure of square Hmong characters and structure description

Class number	Structure description	Linear representation	Structure type
Class 1	C_{l}-C_{r}	s_1s_2	left-right structure with two components
Class 2	C_{l}-C_{rl}-C_{rr}	$s_1s_2s_3$	left-right structure with three components
Class 3	C_{u}-C_{d}	s_4s_5	top-bottom structure with two components
Class 4	C_{u}-C_{dl}-C_{dr}	$s_4s_5s_6$	top-bottom structure with three components
Class 5	C_{olu}-C_{ird}	s_7s_8	part-enclosed structure with two components
Class 6	C_{old}-C_{iru}	s_9s_{10}	part-enclosed structure with two components
Class 7	C_{oru}-C_{ild}	$s_{11}s_{12}$	part-enclosed structure with two components
Class 8	C_{oa}-C_{ia}	$s_{13}s_{14}$	part-enclosed structure with two components

3.3 Judgment of Structure Equivalence Class of Square Hmong Characters

The position of each component in the spatial structure of a square Hmong character can be regarded as an attribute of the character. According to the linear representation of a square Hmong character, if $A(i)$ is used to represent the i-th attribute, each character has 14 attributes $A(1)$, $A(2)$, …, $A(14)$. This attribute value only indicates whether a certain type of component exists in a character, and its range is $\{0, 1\}$. “1” means that such a component exists, and “0” means that such component does not exist.

The process of determining the structure equivalence class of a square Hmong character is divided into two stages. In the first stage, the value of each attribute $A(i)$ (i = 1, 2, …, 14) is calculated. In the second stage, the structure equivalence class of this character is determined according to the combination of attribute values. The decision process is described in Algorithm 1.

```
Algorithm 1. Structure equivalence class decision algorithm
Input: Given square Hmong character $w$
Output: Number of structure equivalence class $T_w$
Steps:
Step 1: Linearize $w$ to a uniquely identified component sequence $f(w)$ using Equation (2).
Step 2: Extract the $i$-th component $f_i(w)$ ($i$=1,2,…,14) from $f(w)$ according to Equation (3) and check whether $f_i(w)$ is "ε". If so, set $A(i)$=0; otherwise, set $A(i)$=1.
Step 3: Check the values of each attribute $A(i)$ ($i$=1,2,…,14) of $w$ and determine $T_w$ according to the combination of $A(i)$ values.
    (1) If $A(i)$=1 ($i$=1,2) and $A(j)$=0 ($j$=3,5,…,14), then $T_w$=Class 1.
    (2) If $A(i)$=1 ($i$=1,2, 3) and $A(j)$=0 ($j$=4,5,…,14), then $T_w$=Class 2.
    (3) If $A(i)$=0 ($i$=1,2, 3), $A(j)$=1($j$=4, 5), and $A(k)$=0 ($k$=6,7,…,14), then $T_w$=Class 3.
    (4) If $A(i)$=0 ($i$=1,2,3), $A(j)$=1 ($j$=4,5,6), and $A(k)$=0 ($k$=8,9,…,14), then $T_w$=Class 4.
    (5) If $A(i)$=0 ($i$=1,2,…,6), $A(j)$=1 ($j$=7,8), and $A(k)$=0 ($k$=9,…,14), then $T_w$=Class 5.
    (6) If $A(i)$=0 ($i$=1,2,…,8), $A(j)$=1 ($j$=9,10), and $A(k)$=0 ($k$=11,…,14), then $T_w$=Class 6.
    (7) If $A(i)$=0 ($i$=1,2,…,10), $A(j)$=1 ($j$=11,12), and $A(k)$=0 ($k$=13,14), then $T_w$=Class 7.
    (8) If $A(i)$=1 ($i$=1,2, …,12) and $A(j)$=1 ($j$=13,14), then $T_w$=Class 8.
    (9) In other cases, $w$ is not a valid square Hmong character.
```

4 Analysis of the Statistical Characteristics of the Structure of Square Hmong Characters

4.1 Cumulative Probability Distribution of the Structure of Square Hmong Characters

The occurrence probability of the i-th equivalence class of the structure of square Hmong characters in a given document D can be calculated by

$$P_i = \frac{|D_i|}{|D|}\ (i = 1, 2, \ldots, m) \tag{9}$$

Where, $|D|$ and $|D_i|$ indicate the number of characters in documents D and D_i, respectively, and m is the number of equivalence classes.

Document D_1 consists of all square Hmong characters that have been collected, and documents D_2, D_3, and D_4 are composed of square Hmong characters appearing in [14], [15], and [16], respectively. The probability distribution of the equivalence class of the structure of square Hmong characters in each of the four documents (D_1, D_2, D_3, and D_4) is shown in Fig. 3.

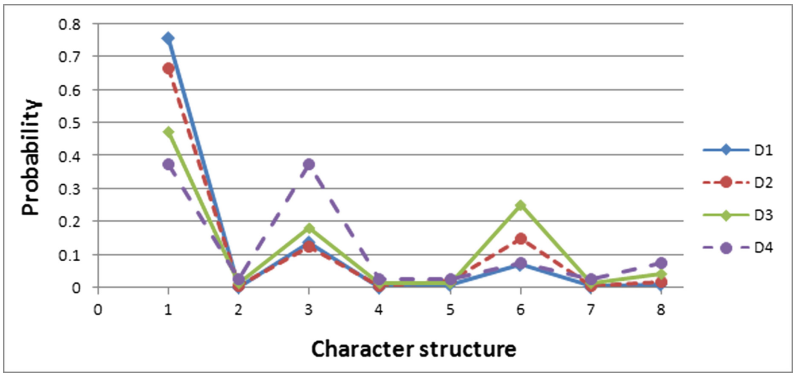


Fig. 3. Probability distribution of the equivalence class of the structure of square Hmong characters

The following points are determined from Fig. 3.

(1) Class 1 (C_l-C_r), Class 3 (C_u-C_d), and Class 6 (C_{old}-C_{iru}) have high probability characteristics. In documents D_1, D_2, D_3, and D_4, the cumulative probabilities of these three classes are as high as 0.9672, 0.9405, 0.9028, and 0.8250, respectively. The high probability of Class 1 (C_l-C_r) is particularly significant.

(2) Class 2 (C_l-C_{rl}-C_{rr}), Class 4 (C_u-C_{dl}-C_{dr}), Class 5 (C_{olu}-C_{ird}), Class 7 (C_{oru}-C_{ild}), and Class 8 (C_{oa}-C_{ia}) have a significantly low probability characteristic. In documents D_1, D_2, D_3, and D_4, the cumulative probabilities of these five classes are as low as 0.0328, 0.0595, 0.0972, and 0.1750, respectively.

(3) In documents D_1, D_2, D_3, and D_4, the cumulative probabilities of the occurrence of Class 2 (C_l-C_{rl}-C_{rr}) and Class 4 (C_u-C_{dl}-C_{dr}) are extremely low at only 0.0053, 0.0119, 0.0278, and 0.050, respectively.

Eight equivalence classes of the structure of square Hmong characters in the practical documents are not evenly distributed. More than 80% of the characters appearing in the practical documents belong to Class 1, Class 3, and Class 6, and more than 90% of them consist of two components.

4.2 Complexity of the Structure of Square Hmong Characters

The results on the cumulative probability distribution of the structure of square Hmong characters reflect the simplicity of the structure. This simplicity can be verified by calculating the complexity of the structure of square Hmong characters. Complexity can be expressed by the average number of components in each square Hmong character appearing in the practical documents. *Length* is calculated by

$$Length = \sum_{i=1}^{m} P_i l_i \tag{10}$$

Where, P_i is the probability of occurrence of various structures derived from Eq. (9) and l_i is the number of components in the i-th equivalence class.

In documents D_1, D_2, D_3, and D_4, the *Length* values calculated with Eq. (10) are 2.0053, 2.0119, 2.0278, and 2.0500, respectively. This result fully confirms the simplicity of the structure of square Hmong characters.

4.3 Information Entropy of the Structure of Square Hmong Characters

Information entropy is used to describe the distribution characteristics of square Hmong characters with different structures in the same document. A large entropy value equates to a large number of structure equivalence classes included in the document and to a uniform distribution of square Hmong characters with different structures. On the contrary, a small entropy value equates to a few structure equivalence classes included in the document and to a concentrated distribution of square Hmong characters in a few classes.

Given document D, the information entropy of the structure of square Hmong characters is calculated as

$$Entropy(D) = -\sum_{i=1}^{m} P_i log_2 P_i \tag{11}$$

Where, P_i and m are similar to those in Eq. (9).

Figure 4 presents the information entropy of the structure of square Hmong characters in documents D_1, D_2, D_3, and D_4. As shown in Fig. 4, the structure of square Hmong characters in the four documents reveals a first-order approximation of the information. The entropy is small, falling in the interval (1.19, 2.16). The results also show that the probability of occurrence of various structures appearing in the practical documents is extremely uneven.

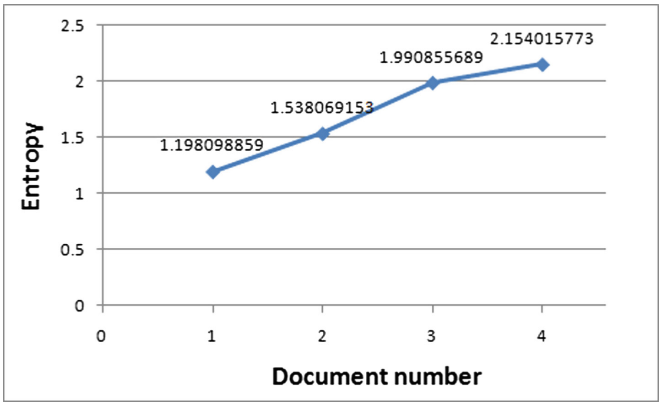


Fig. 4. Information entropy of the structure of square Hmong characters in the practical documents

5 Conclusion and Future Work

In this work, we study the linear representation of square Hmong characters and the classification method for the structure of square Hmong characters. We also analyze the statistical properties of the structure of square Hmong characters appearing in practical documents. The analysis results can be used as a basis for the design of a heuristic rule in the post-processing of an intelligent input and recognition system for square Hmong characters. The study provides a good foundation for the further investigation of the application of the structure characteristics of such characters.

Notably, this study focuses on character structure and not on specific characters. To comprehensively understand the application characteristics of square Hmong characters in different areas, a variety of practical documents with different themes, styles, and nature must be collected. An in-depth analysis of the differences in the use of specific characters should be conducted, and the effectiveness of specific characters should be evaluated.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (Nos. 61462029 and 61741205).

References

1. Instituse, R.: A fast smoothing & thinning method based on character structure. J. Chin. Inf. Process. **4**(2), 49–55 (1990)
2. Shin, J., Suzuki, K., Hasegawa, A.: Handwritten Chinese character font generation based on stroke correspondence. Int. J. Comput. Process. Orient. Lang. **18**(3), 211–226 (2005)
3. Liu, M.Y., Duan, C.S., Pi, Y.G.: Basic elements knowledge acquisition study in the Chinese character intelligent formation system. J. Softw. Eng. Appl. **2**(5), 316–322 (2009)
4. Tan, J., Xie, X.H., Zheng, W.H., et al.: Radical extraction using affine sparse matrix factorization for printed Chinese characters recognition. Int. J. Pattern Recognit. Artif Intell. **26**(3), 211–226 (2012)
5. Dobres, J., Chahine, N., Reimer, B., et al.: The effects of Chinese typeface design, stroke weight, and contrast polarity on glance based legibility. Displays **41**, 42–49 (2016)
6. Ai, J.Y., Yu, H.Z., Li, Y.H.: Statistical analysis on Tibetan shaped structure. J. Comput. Appl. **29**(7), 2029–2031 (2009)
7. Cai, Z.J., CaiRang, Z.M.: Research on the distribution of Tibetan character forms. J. Chin. Inf. Process. **30**(4), 98–105 (2016)
8. Kwon, Y.B.: Hangul tree classifier for type clustering using horizontal and vertical strokes. In: Proceedings of the 16th International Conference on Pattern Recognition, pp. 228–231. IEEE, Quebec City (2002)
9. Xu, R.J., Liu, C.P.: Grapheme segmentation and recognition in machine printed Hangul characters. J. Chin. Inf. Process. **20**(2), 66–71 (2006)
10. Cui, R.Y., Kim, S.J.: Research on information structure of Korean characters. J. Chin. Inf. Process. **25**(5), 114–119 (2011)
11. Mo, L.P., Zhou, K.Q.: Formal description of dynamic construction method for square Hmong language characters. J. Comput. Appl. **34**(3), 861–864, 868 (2014)
12. Mo, L.P., Zhou, K.Q., Jiang, X.H.: Research on square Hmong language characters fonts based on OpenType technology. J. Chin. Inf. Process. **129**(2), 150–156 (2015)

13. Mo, L.P., Zhou, K.Q.: A dynamical glyph generation method of Xiangxi Folk Hmong characters and its implementation approach. Acta Scicentiarum Naturalum Universitis Pekinesis **52**(1), 141–147 (2016)
14. Zhao, L.M., Liu, Z.Q.: Xiangxi square Hmong characters. Minor. Lang. China **12**(1), 44–49 (1990)
15. Yang, Z.B., Luo, H.Y.: On the folk coinage of characters of the Miao People in Xiangxi area. J. Jishou Univ. (Soc. Sci. Edn.) **29**(6), 130–134 (2008)
16. Long, Z.H.: Re-study of the coinage method of square characters of Miao language in the Youshui river basin of Yu, Xiang and E. J. Chongqing Educ. Coll. **25**(5), 56–59 (2012)

Stock Market Trend Prediction Using Recurrent Convolutional Neural Networks

Bo Xu, Dongyu Zhang, Shaowu Zhang, Hengchao Li, and Hongfei Lin(✉)

School of Computer Science and Technology, Dalian University of Technology, Dalian, China
hflin@dlut.edu.cn

Abstract. Short-term prediction of stock market trend has potential application for personal investment without high-frequency-trading infrastructure. Existing studies on stock market trend prediction have introduced machine learning methods with handcrafted features. However, manual labor spent on handcrafting features is expensive. To reduce manual labor, we propose a novel recurrent convolutional neural network for predicting stock market trend. Our network can automatically capture useful information from news on stock market without any handcrafted feature. In our network, we first introduce an entity embedding layer to automatically learn entity embedding using financial news. We then use a convolutional layer to extract key information affecting stock market trend, and use a long short-term memory neural network to learn context-dependent relations in financial news for stock market trend prediction. Experimental results show that our model can achieve significant improvement in terms of both overall prediction and individual stock predictions, compared with the state-of-the-art baseline methods.

Keywords: Stock market prediction · Embedding layer
Convolutional neural network · Long short-term memory

1 Introduction

Financial information on the internet has increased explosively with the rapid development of the internet. Daily financial news, as an important resource of financial information, contains a large amount of valuable information, such as the changes in senior management of listed companies and the releases of new products. The information is highly useful for investors to make crucial decisions on their personal investment. The key issue on generating a high return on the stock market lies in how well we are able to successfully predict the future movement of financial asset prices. Therefore, it is necessary to fully exploit the financial information from news for stock market trend predictions.

Existing studies have addressed stock market trend prediction using various machine learning methods, such as Support Vector Machines (SVMs) [1–4], Least Squares Support Vector Machines (LS-SVMs) [5–7] and Artificial Neural Networks (ANNs) [8–10]. Most of these studies have focused on extracting effective features for training a good prediction model [11–14]. Feature selection and feature engineering have been fully investigated for the improvement of performance. For example, Kogan et al. [11] addressed the volatility of stock returns using regression models based on financial reports to reduce the financial risk. Schumaker et al. [12] used SVM with several different textual representations, bag of words, noun phrases, and named entities for financial news article analysis. However, handcrafted features cost much manual labor and partly limit the scalability of the learned models. How to automatically generate effective features for stock market predictions remains a great challenge.

Recently, deep learning models have exhibited powerful capability in automatically generating effective features, and been successfully applied in different natural language processing tasks. Existing studies on stock trend prediction have also focused on automatically generating effective features based on deep neural network models. For example, Hsieh et al. [13] integrated bee colony algorithm into wavelet transforms and recurrent neural networks for stock prices forecasting. Ding et al. [14] proposed convolutional neural network to model the influences of events on stock price movements. However, how to accurately model the relationship between financial news and stock market movement poses a new challenge for stock market trend prediction.

In this paper, we attempt to introduce deep neural network based models for stock market trend prediction. Deep neural network models, such as convolutional neural network and long short-term memory network, have been widely used in natural language processing tasks [15–19]. We address two key issues for applying deep neural network based models in this task. One is how to generate effective entity embedding based on the contents of financial news, and the other is how to incorporate the complex mutual relationship between financial news and stock market movement into the prediction model.

In order to construct useful word embedding for financial news contents, we introduce an entity embedding method [20] to represent financial news. This method actually introduce an entity embedding layer between one-hot input layer and neural network model for automatically learning entity embedding for news contents. To predict stock market trend of the listed companies, we first extract key influential information from daily financial news. We then propose a convolutional recurrent neural network to represent the news for extracting key information. The proposed network can capture the context-dependence relations in financial news, and use their internal memory to process arbitrary sequences of inputs for better prediction. Experimental results show that the proposed model outperforms other baseline models and effectively predicts the stock market trends.

The main contribution of this paper is as follows: (1) We introduce an entity embedding layer to automatically learn distributed representation of financial news contents without any handcrafted feature. (2) We propose a recurrent convolutional neural network to extract the key information from financial news and model context-dependent relation for predicting stock market movements. (3) We conduct extensive experiments to evaluate the proposed model. Experimental results show that our model achieves significant improvement in terms of the prediction accuracy.

The rest of the paper is organized as follows: Sect. 2 introduces the overall framework and details of our model; Sect. 3 provides our experimental results and comparative analysis of the experimental results; Sect. 4 concludes the paper and introduces our future work.

2 Methodology

In this section, we introduce details about the proposed model. We first illustrate the overall framework of our model for stock market trend prediction shown in Fig. 1. The whole framework includes four modules: the financial data acquisition module, the data preprocessing module, the data labeling module and the model training module.

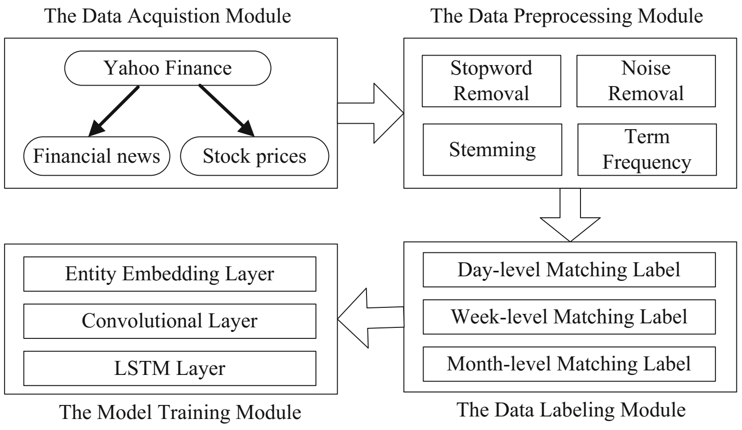


Fig. 1. The overall framework of our model

The financial data acquisition module crawls financial data from Yahoo Finance[1]. We acquire two types of data, financial news and stock prices, for model training. The financial news are used as the information source of model inputs, and the stock prices are used as the source of the ground truth labels for model targets.

The data preprocessing module transforms the webpages into texts by removing useless data, such as images and links. This module also preprocesses the stock prices data by removing stopwords, stemming the contents, and counting the term frequency in news for subsequent processing.

The data labeling module then matches the financial news with stock prices based on their timestamps, which is used to generate ground truth labels for model training at different levels, including the day-level, week-level and month-level matching labels.

The model training module is the core of our predictive model, the recurrent convolutional neural network model (RCNN). This module includes three layers, the embedding layer, the convolutional layer and the long short-term memory (LSTM)

[1] https://finance.yahoo.com/.

layer. We illustrate these layers in Fig. 2. The embedding layer learns entity embedding based on the financial news contents, the convolutional layer extracts the key local information of news, and the LSTM layer captures the relationship of dependency context for final prediction of stock market movements by a dense neural network (NN) layer. We introduce the details about each layer in the following subsections.

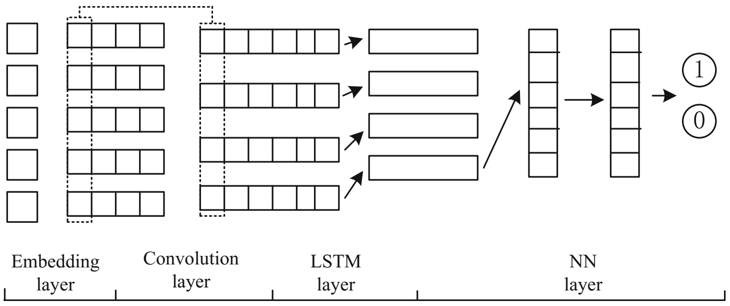


Fig. 2. Recurrent convolutional neural network

2.1 The Embedding Layer

For the embedding layer, we first count the term frequency of the crawled financial news to build a financial entity dictionary with high frequency entity terms. We then align the input sentences with diverse lengths using the financial dictionary as the inputs of the embedding layer. We adopt a state-of-the-art embedding method [20] to map the words to matrix. The used embedding method can represent key financial entities into vectors in Euclidean spaces, and map similar values close to each other in the embedding space to reveal the intrinsic properties of the categorical variables. The method can effectively represent financial entities into a vector space as the inputs of the following convolutional layer. Specifically, we first map each state of a discrete variable based on term frequency to a vector for learning vector representations of entities as follows.

$$e_i : x_i \longrightarrow \mathbf{x}_i \tag{1}$$

The mapping is equivalent to build an extra layer on top of the input one-hot representations. We encode the inputs as follows.

$$u_i : x_i \longrightarrow \delta_{x_i \alpha} \tag{2}$$

where $\delta_{x_i \alpha}$ is Kronecker delta and the range of α is the same as x_i. If m_i is the number of possible values of x_i, then $\delta_{x_i \alpha}$ becomes a vector of length m_i, where the element is non-zero when $\alpha = x_i$. Given the input x_i, the output of this fully connected layer is defined as follows.

$$\mathbf{x}_i \equiv \sum_{\alpha} w_{\alpha\beta} \delta_{x_i \alpha} = w_{x_i \beta} \tag{3}$$

where β is the index of the embedding layer and $w_{\alpha\beta}$ is the weight between the one-hot encoding layer and the embedding layer. It can be seen that the mapped vector representations of the entities are actually the weights of embedding layer. The weights are the parameters in the neural network, and can be learned during model training.

We provide a toy example of our data in Fig. 3 for better understanding the embedding layer. The example is taken from Apple news in April 26, 2016. The raw input of the example is "apple becomes the dow's worst performer". We preprocess the sentences by removing stopwords and stemming, and then we obtain the sentence "appl becom dow worst perform". Based on pre-built financial entity dictionary, we map the input sentence to the matrix using the entity embedding method, which will be taken as the inputs for the convolutional layer.

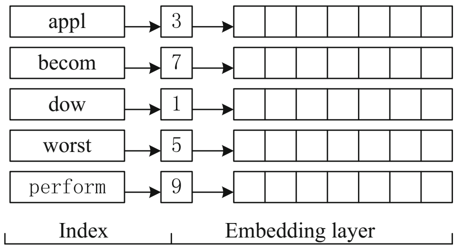


Fig. 3. A toy example for using the embedding layer to automatically learn distributed representation of financial entities.

2.2 The Convolutional Layer

Convolutional neural networks (CNN) are inspired by biological processes and are designed to use minimal amounts of preprocessing for encoding abundant semantic information in different natural language processing tasks. Convolutional neural networks, as variations of multilayer perceptrons, include three characteristics, local connectivity, parameter sharing and pooling. These characteristics make CNN an effective network model in extracting key information in texts

In our model, we use CNN as the convolutional layer, which treats the outputted matrix of the embedding layer as inputs for modeling the key information of financial news. The number of columns of the matrix is the dimensionality of the entity embedding, which is taken as the number of the feature maps of the convolutional layer. The number of rows of the matrix is taken as the number of convolutional kernels. We perform convolutional operation on the columns of the input matrix using max pooling to extract the critical information affecting stock movements. The outputs of the convolutional layer are then regarded as the inputs of the following LSTM layer.

We illustrate our convolutional layer in Fig. 4. The convolutional layer uses convolutional operation with max pooling to extract semantic and context information from financial news, and embeds the information into low dimensional representations for tracking the stock market movement.

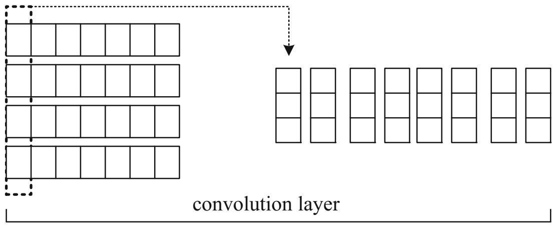


Fig. 4. Using convolution layer to extract the key information from financial news

2.3 The LSTM Layer

Recurrent neural network model (RNN) is widely used in NLP tasks, which equivalents to the multilayer feedforward neural network. Long short-term memory network (LSTM), as a variation of RNN, avoids the gradient vanish issue in RNN and uses historical information through the input, forget and output gate.

We adopt LSTM as a layer in our model. Our model takes the outputted matrix of the convolutional layer as the inputs of the LSTM layer for capturing the relationship of dependency contexts for final prediction of stock market movements. The rows of the matrix are taken as the hidden units of the LSTM layer, and the last hidden unit of the LSTM layer is then regarded as the inputs of the LSTM layer. LSTM has been proved to be effective in capturing temporal sequential information in other natural language processing tasks. Since financial data comprises abundant temporal information, we use LSTM to capture latent information in financial news, particularly to model the relationship between the stock market movement and the news. We illustrate the LSTM layer used in our model in Fig. 5.

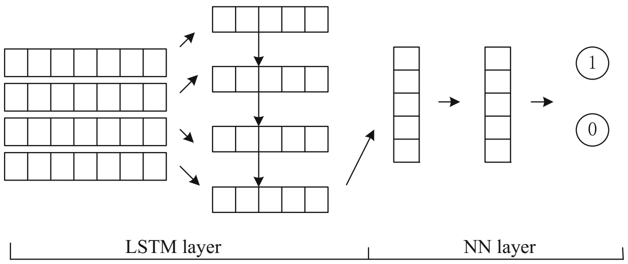


Fig. 5. Using LSTM to extract the context-dependent relation from financial news

Finally, we use a dense neural network layer to classify the financial news for predicting stock movements. We then evaluate our model using extensive experiments.

3 Experiments

3.1 Experimental Setup

In this section, we introduce the experimental setup and report the experimental results for evaluating the proposed model. We fetch the financial news using a web crawler from Yahoo Finance focusing on the shares of listed companies. The date range of the fetched news is from October, 2014 to May, 2016. The obtained data involves 447 listed companies, such as Apple Inc. (AAPL), Google Inc. (GOOG) and Microsoft Inc. (MSFT). We provide the statistics of our data in Table 1.

Table 1. Statistics of the used data

Statistics	Quantity
The number of listed companies	447
Date range of financial news	2014.10–2016.05
The number of the news	322,694

We crawl historical stock prices from Yahoo Finance website, and use the prices to generate ground truth labels of the financial news. Specifically, if the stock price moves up in the next day, we label the current day's financial news as 1, indicating it is useful. Otherwise, if stock price moves down in the next day, we labeled the current day's news as 0, indicating it is useless for stock market trend prediction.

In addition, we use the headlines of financial news as the training data in our experiments following the work by Ding et al. [14] and Tetlock et al. [21], which showed that news titles are more useful than news contents for the prediction of stock market trend. In order to detect diminishing effects of reported events on stock market volatility, we label news at day level, week level and month level, respectively. Our preliminary experimental results show that week-level and month-level labels are of little use for stock trend prediction. Therefore, we adopt the day-level labels in the following experiments, which is the same setting as other existing studies on stock movement prediction [22–24].

In our experiments, we compare our model with two state-of-the-art baseline models. One is to represent financial news using bag of words features and SVM classifier proposed by Luss et al. [23], denoted as SVM. The other adopted neural tensor network to learn distributed representations of financial events, and used convolution neural network model for predicting the stock market [14], denoted as E-CNN. We evaluate the performance of prediction in terms of two standard evaluation metrics, the accuracy (Acc) and the Matthews correlation coefficient (MCC). We conduct 5-fold cross validations to evaluate the results, and report the average results for fair comparison.

3.2 Experimental Results and Analysis

Hyper-parameter Selection

Compared to the baseline models, we introduce the embedding layer to automatically learn the entity embedding of financial news, and then used the convolutional layer and the LSTM layer to extract critical information and the context-dependent relation for final prediction. There are six hyper-parameters used in our model, including the length of the inputs, the dimensionality of the embedding layer, the length of each CNN kernel, the number of CNN kernels, the dimensionality of the LSTM layer and the number of iterations. We switch these parameters for the baseline models and our model on the development set for selecting the optimal parameters. We report the selected optimal parameter in Table 2.

Table 2. The hyper-parameters of our models

Hyper-parameters	E-CNN	EB-CNN	E-CNN-LSTM	EB-CNN-LSTM
Length of the inputs	30	30	30	30
Dim. of embedding layer	50	128	50	128
Length of CNN kernel	3	3	3	3
Number of CNN kernels	250	250	64	64
Dim. of LSTM layer	–	–	70	70
Number of iterations	30	30	30	30

Experimental Results

We report our experimental results in this section. In the experiments, we train four different neural network models to demonstrate the effectiveness of the proposed embedding layer and the LSTM layer. We introduce these models as follows and report the experimental results in Table 3.

Table 3. The results of experiments

Experiments	Accuracy	MCC
SVM [23]	58.42%	0.1425
E-CNN [14]	63.44%	0.4198
EB-CNN	64.56%	0.4265
E-CNN-LSTM	65.19%	0.4356
EB-CNN-LSTM	**66.31%**	**0.4512**

- **E-CNN:** The model proposed by Ding et al. [14], which is one of the state-of-the-art models for predicting stock market trend. The model includes an event embedding layer and one CNN layer.
- **EB-CNN:** We use the model to examine the effectiveness of the proposed embedding layer for financial entity representation. The model includes the proposed embedding layer and the CNN layer.

- **E-CNN-LSTM:** We use this model to examine the effectiveness of the LSTM layer. The model includes the event embedding, the CNN layer and the LSTM layer.
- **EB-CNN-LSTM:** This model is the proposed model, including the entity embedding layer, the CNN layer and the LSTM layer.

From the table, we observe that the EB-CNN model achieves better prediction performance than the E-CNN model, which indicates the effectiveness of entity embedding used in our model. One possible explanation for this finding is that the entity embedding layer better encodes semantic information of financial entities for word and entity representations for financial news, while the event embedding layer used in E-CNN is designed to solve the sparsity of data and used to extract the key elements of events for the representation. Therefore, we obtain better performance using the CB-CNN model.

Furthermore, we observe that the E-CNN-LSTM model outperforms the E-CNN model, which indicates the effectiveness of the LSTM layer used in our model. We believe that this is because the LSTM layer contributes to extracting the context-dependent relationship between the financial news and the stock market trends. The proposed model finally achieves the best performance among all the baseline models, which demonstrates that our model is effective in capturing the stock market movement and predicting the stock market trends. We illustrate the experimental results with the change of the number of iterations in Fig. 6. The figure clearly shows that our model outperforms other baseline models with the number of iterations changing from 0 to 30.

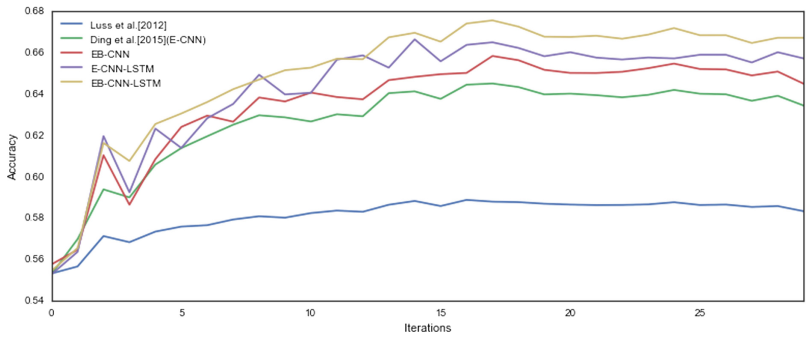


Fig. 6. Comparison of different models

Comparisons on Individual Stock Predictions

To further evaluate our model, we compare our approach with the baseline models in terms of individual stock predictions. We select nine companies as the individuals from our dataset. These companies cover high-ranking companies (GOOG, MSFT, AAPL), middle-ranking companies (AMAT, STZ, INTU), and low-ranking companies (HST, ALLE, JBHT). The ranking of companies are based on the S&P 500 from the Fortune Magazine[2]. We report the accuracy of individual stocks in Fig. 7.

[2] http://fortune.com/.

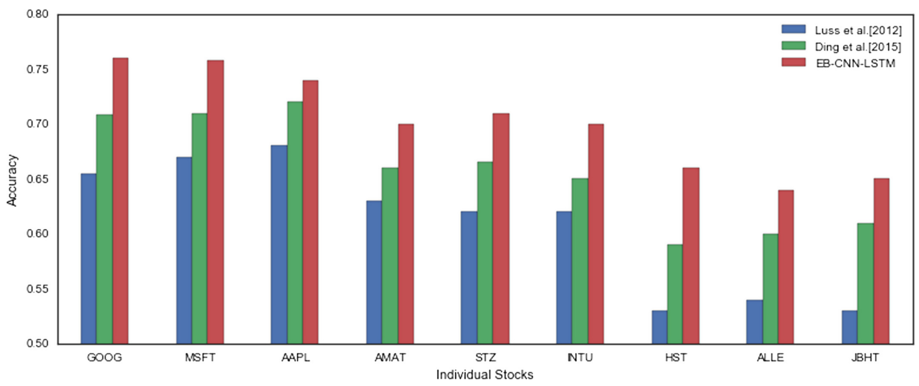


Fig. 7. Comparisons on individual stock prediction. Companies are named by ticker symbols.

From the figure, we can observe that our model achieves robust results in terms of the selected individual stocks. In addition, our model achieves relatively higher improvements on those lower fortune ranking companies, for which fewer pieces of news are available. For the baseline methods, the prediction results of low-ranking companies dramatically decrease. In contrast, our model achieves more stable performance. This is because our model uses the entity embedding layer to learn powerful distributed representations based on the news from these low-ranking companies. Hence, our model yields relatively high accuracy on prediction even without large amounts of daily news.

Diminishing Effects of the News

In order to detect diminishing effects of the news on stock market volatility, we label news in the next one day, next two day, and next three day, respectively. We train our model and the baseline models based on the different levels of labels, and report the experimental results in Fig. 8.

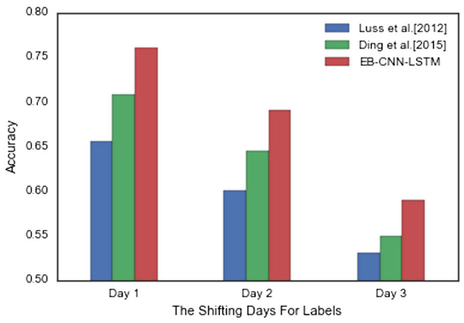


Fig. 8. Development results of different labels for the models

From the figure, we observe that our model achieves the best performance at different levels of labels compared to the baseline models. This finding exhibits the robustness and stability of our model. Besides, we observe that the effects of news on stock market prediction weakened over time, which indicates that daily prediction on stock market trend is necessary. We also use the news at a level of more than 3 days. The experimental results show that the influence of financial news is almost disappeared and useless for the prediction.

4 Conclusion and Future Work

In this paper, we propose a novel recurrent convolutional neural network model to predict the stock market trends based on financial news. In our model, we introduce an entity embedding layer to automatically learn distributed representation of financial entities without any handcrafted feature. We propose a recurrent convolutional neural network to extract the key information from financial news and model context-dependent relation for predicting stock market movements. The proposed network includes a convolutional layer and a long short-term memory layer for capturing abundant semantic information from the financial news. We conduct extensive experiments to evaluate the proposed model. Experimental results show that our model achieves significant improvement in terms of the prediction accuracy. In our future work, we will explore more effective model for predicting the stock market trend in consideration of temporal characteristics of news. We will also attempt to integrate external financial knowledge to optimize our model and improve the performance of stock trend prediction.

Acknowledgements. This work is partially supported by grant from the Natural Science Foundation of China (No. 61632011, 61572102, 61702080, 61602079, 61562080), State Education Ministry and The Research Fund for the Doctoral Program of Higher Education (No. 20090041110002), the Fundamental Research Funds for the Central Universities.

References

1. Huang, W., Nakamori, Y., Wang, S.Y.: Forecasting stock market movement direction with support vector machine. Comput. Oper. Res. **32**(10), 2513–2522 (2005)
2. Lee, M.C.: Using support vector machine with a hybrid feature selection method to the stock trend prediction. Expert Syst. Appl. **36**(8), 10896–10904 (2009)
3. Ni, L.P., Ni, Z.W., Gao, Y.Z.: Stock trend prediction based on fractal feature selection and support vector machine. Expert Syst. Appl. **38**(5), 5569–5576 (2011)
4. Yu, L., Wang, S., Lai, K.K.: Mining stock market tendency using GA-based support vector machines. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 336–345. Springer, Heidelberg (2005). https://doi.org/10.1007/11600930_33
5. Chai, J., Du, J., Lai, K.K., et al.: A hybrid least square support vector machine model with parameters optimization for stock forecasting. Math. Probl. Eng. **2015**, 1–7 (2015)

6. Marković, I., Stojanović, M., Božić, M., Stanković, J.: Stock market trend prediction based on the LS-SVM model update algorithm. In: Bogdanova, A.M., Gjorgjevikj, D. (eds.) ICT Innovations 2014. AISC, vol. 311, pp. 105–114. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-09879-1_11
7. Yu, L., Chen, H., Wang, S., et al.: Evolving least squares support vector machines for stock market trend mining. IEEE Trans. Evol. Comput. **13**(1), 87–102 (2009)
8. Crone, S.F., Kourentzes, N.: Feature selection for time series prediction – a combined filter and wrapper approach for neural networks. Neurocomputing **73**(10), 1923–1936 (2010)
9. Dai, W., Wu, J.Y., Lu, C.J.: Combining Nonlinear Independent Component Analysis and Neural Network for the Prediction of Asian Stock Market Indexes. Pergamon Press Inc., Tarrytown (2012)
10. Kara, Y., Acar Boyacioglu, M., Baykan, Ö.K.: Predicting direction of stock price index movement using artificial neural networks and support vector machines. Expert Syst. Appl. **38**(5), 5311–5319 (2011)
11. Kogan, S., Levin, D., Routledge, B.R., et al.: Predicting risk from financial reports with regression. In: North American Chapter of the Association for Computational Linguistics, pp. 272–280 (2009)
12. Schumaker, R.P., Chen, H.: Textual analysis of stock market prediction using financial news articles. In: Americas Conference on Information Systems (2006)
13. Hsieh, T.J., Hsiao, H.F., Yeh, W.C.: Forecasting stock markets using wavelet transforms and recurrent neural networks: an integrated system based on artificial bee colony algorithm. Appl. Soft Comput. **11**(2), 2510–2525 (2011)
14. Ding, X., Zhang, Y., Liu, T., et al.: Deep learning for event-driven stock prediction. In: Ijcai, pp. 2327–2333 2015
15. dos Santos, C.N., Gatti, M.: Deep convolutional neural networks for sentiment analysis of short texts. In: COLING, pp. 69–78 (2014)
16. Xie, B., Passonneau, R.J., Wu, L., Creamer, G.G.: Semantic frames to predict stock price movement. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp. 873–883 (2013)
17. Martin, L., Lars, K., Amy, L.: A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognit. Lett. **42**, 11–24 (2014)
18. Ding, X., Zhang, Y., Liu, T., Duan, J.: Using structured events to predict stock price movement: an empirical investigation. In: EMNLP, pp. 1415–1425 (2014)
19. Schmidhuber, J., Hochreiter, S.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
20. Guo, C., Berkhahn, F.: Entity embeddings of categorical variables. arXiv preprint arXiv:1604.06737 (2016)
21. Tetlock, P.C., Saar Tsechansky, M., Macskassy, S.: More than words: quantifying language to measure firms' fundamentals. J. Finance **63**(3), 1437–1467 (2008)
22. Radinsky, K., Davidovich, S., Markovitch, S.: Learning causality for news events prediction. In: Proceedings of the 21st International Conference on World Wide Web, pp. 909–918. ACM (2012)
23. Luss, R., d'Aspremont, A.: Predicting abnormal returns from news using text classification. Quant. Finance **15**(6), 999–1012 (2015)
24. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: Conference on Artificial Intelligence (No. EPFL-CONF-192344) (2011)

Ensemble of Binary Classification for the Emotion Detection in Code-Switching Text

Xinghua Zhang[1(✉)], Chunyue Zhang[2(✉)], and Huaxing Shi[2(✉)]

[1] Harbin Institute of Technology, Harbin, China
xing_hua_zhang@126.com
[2] DeepIntell Co. Ltd., Harbin, China
{cyzhang,shihuaxing}@deepintell.cn

Abstract. This paper describes the methods for the DeepIntell who participated the task1 in the NLPCC2018. The task1 is to label the emotion in a code-switching text. Note that, there may be more than one emotion in a post in this task. Hence, the assessment task is a multi-label classification task. At the same time, the post contains more than one language, and the emotion can be expressed by either monolingual or bilingual form. In this paper, we propose a novel method of converting multi-label classification into binary classification task and ensemble learning for code-switching text with sampling and emotion lexicon. Experiments show that the proposed method has achieved better performance in the code-switching text task.

Keywords: Multi-label classification · Binary classification
Sampling · Emotion lexicon · Ensemble learning

1 Introduction

With the widespread popularity of social media, such as Weibo, WeChat, Twitter, etc., the analysis of the content of user articles has played a pivotal role in the field of natural language processing. Most of the previous emotion classification problems were conducted in monolingual corpora. However, with the diversification of culture, people usually publish some multilingual articles or comments. [E1-E4] are four examples of code-switching text on evaluation data that contain both Chinese and English words. In order to detect the emotions better in the text, it is necessary to consider the emotional expression in all of them. The assessment task mainly includes automatic classification of emotion (*happiness*, *sadness*, *anger*, *fear*, *surprise*) in code-switching text. Different from monolingual emotion detection, the emotion in code-switching text can be expressed in either monolingual or bilingual forms. In this task, we focus on Chinese and English mixed code-switching text. Although Chinese is the major language, it has been shown that English words are critical for emotion expression.

[E1] 这个 show 真**好看**, 今天感觉很 **happy**!
(*The show is really* ***nice****, I feel very* ***happy*** *today!*)
[E2] 上了一天的课了，嗓子 **hold 不住** 了啊。
(*I have been teaching the whole day, my throat* ***can't take it anymore****.*)
[E3] 我在 caffe in (中山南路店)。
(*I'm at caffe in (Zhongshan South Road).*)
[E4] 你今天真得**很棒**, **give me five**!
(*You are really* ***good*** *today,* ***give me five!***)

In fact, examples above are essentially different. E1 expresses the emotion by monolingual form, either Chinese or English. E2 has the bilingual form to express sadness. English words in E3 can't express any emotions. E4 expresses the emotion by English phrase. And the corpus data can be roughly divided into four categories above. However, if we want to fully uncover the emotions in the text, using an emotion lexicon as an aid seems like a good choice. Thus, we have collected the emotion lexicon in Chinese, English and mixture to explore the emotional information better in the code-switching text.

We adopt the multi-label classification method RCNN and the binary classification Fasttext [21], RCNN, and CNN [20] for the automatic classification of emotion. The experiment results show that the methods above have different advantages and disadvantages for different emotions, so ensemble learning technique may achieve better results. In addition, pre-processing (sampling) of the training data will get better scores when using the binary classification method for emotion recognition.

In fact, the evaluation is essentially a multi-label classification task. To detect the emotion better in the code-switching text, we propose a method of binary classification and ensemble learning with sampling and emotion lexicon(BCEL). For the five emotions of this evaluation: *happiness*, *sadness*, *anger*, *fear*, and *surprise*, we generated five data sets of the same size as the original training set. However, sentences in each sets are labelled differently. Take *happiness* as an example, if a sentence contains this emotion, it is marked as *happiness*, otherwise it is marked as *non*. Then we adopt the binary classification methods to train five training sets respectively to obtain the optimal model. At the end, we adopt ensemble learning technique and emotion lexicon for the five optimal models to obtain the final result.

The rest of the paper is organized as follows. In Sect. 2, we give an overview on the related work. In Sect. 3, we introduce our methods that we have tried. In Sect. 4, we describe our BCEL method. Experiment results and discussion are reported in Sect. 5. Finally, we draw some conclusions and give the future works.

2 Related Work

In this section, we discuss related works on emotion classification and code-switching text.

2.1 Emotion Classification

With the increasing number of texts with subjective assessment on the Internet, text emotion classification has gradually become a research hotspot in the field of natural language processing. Mostafa [3] used a pre-defined vocabulary of approximately 6,800 adjectives to perform emotion analysis of microblogs from multiple company consumers and found that consumers had positive emotions for well-known brands. Pang Lei et al. (2012) used emotional words and facial expressions as learning knowledge, using support vector machines, naive Bayes and maximum entropy to propose an unsupervised emotion classification method based on Sina Weibo.

A major related research in the emotion classification task is the creation of emotional resources, such as the creation of emotion lexicons. Xu et al. [8] apply a graph-based algorithm and multiple types of resources to create a Chinese emotion lexicon. Volkova et al. [9] introduced a dictionary for exploring language colors, concepts, and emotions. Moreover, most of the relevant studies have focused on supervised learning methods. Alm et al. [10] achieved text-based emotion prediction using machine learning methods. Aman and Szpakowicz [11] implemented sentence-level fine-grained emotion recognition through a knowledge-based approach. Chen et al. [7] detected emotion-induced events by analyzing the language architecture. Purver and Battersby [12] trained a fully supervised classifier using auto-labeled data to achieve multiple types of emotion prediction without manual intervention. Lin et al. [6] first described the emotion classification tasks of readers in news texts, and then applied some standard machine learning methods to train a classifier that recognizes readers' emotions.

2.2 Code-Switching Text

Code-switching text has received considerable attention in the NLP community. Several studies have focused on identification and analysis. Ling et al. [13] presented a novel method for translation in code-switched documents. Solorio and Liu [14] predicted potential code-switching points in Spanish-English. Lignos and Marcus [15] tried to identify code-switched tokens and Li et al. [16] added code-switched support to language models. Peng et al. [17] learned poly-lingual topic models from code-switching text. Lee et al. [1] proposed a multiple-classifier-based automatic detection approach to detect emotion in the code-switching corpus for evaluating the effectiveness of both Chinese and English texts. Wang et al. [2] proposed a Bilingual Attention Network (BAN) model to aggregate the monolingual and bilingual informative words to form vectors from the document representation, and integrate the attention vectors to predict the emotion.

3 Methods

3.1 Fasttext

Fasttext [21] is a text classification tool developed by Facebook. It provides a simple but efficient method for text representation and text classification. For the text classification part, it only has one hidden layer in the architecture so that the classification process is relatively fast. Fasttext classification function is similar to word2vec's continue bag of words (CBOW) algorithm. Firstly, the feature vector combined with word sequence is linearly projected to middle hidden layer. Secondly, there is a non-linear activation function which projects middle hidden layer to the categorization label. The difference between Fasttext and CBOW is that Fasttext predicts labels while CBOW predicts middle terms.

We adopt Fasttext to train one classifier for each emotion. And then we will get five classifiers and finally perform five different emotion predictions.

3.2 RCNN(Multi-label)

The structure of RCNN model is shown in Fig. 1. RCNN has a two-layer bidirectional LSTM to extract the context information of each word. It also uses Embedding to obtain the word information directly. Then the output of LSTM and Embedding is concatenated, and the concatenating result is performed with two-layer convolution operation after K-MaxPooling that extracts the local features further. Finally, the features are continuously input into a classification network composed of two-layer fully connected networks.

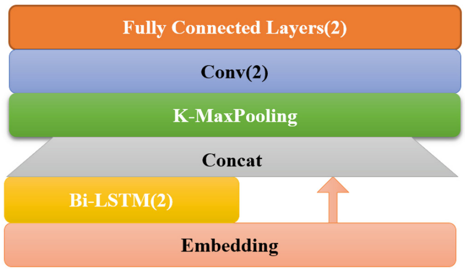


Fig. 1. The structure of Recurrent Convolutional Neural Networks model

3.3 RCNN(Binary)

The RCNN model adopted in this method is the same as the model used in the second method, except that the output of model is changed to the binary classification (idea of the first method). That is, using the RCNN to train five classifiers according to the emotion category, namely *happiness*, *sadness*, *anger*, *fear*, and *surprise*.

3.4 CNN(Binary)

The Convolutional Neural Networks(CNN) [20] model is shown in Fig. 2. It uses the word vectors (Word2Vec embedding) to obtain the word information directly.

Then, the model adopts four-layer convolution and two pooling layers (MaxPooling, AveragePooling) to extract features, in which the MaxPooling is used after the first two convolution layers, AveragePooling is adopted at the end. After extracting features using AveragePooling, the features are input into the fully connected layer continuously for classification.

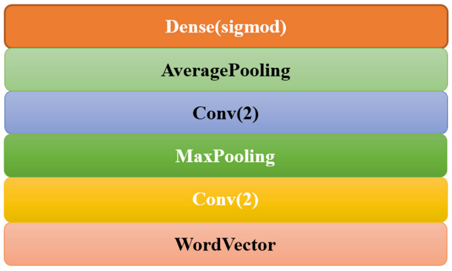


Fig. 2. The structure of Convolutional Neural Networks model

4 Binary Classification and Ensemble Learning with Sampling and Emotion Lexicon

Although research on text classification has been carried out for many years, most of the current studies assume a balanced distribution of samples of various categories. However, the reality is often not the case. In the actual collection of corpus, whether it is the product review text or microblogging text, Twitter text, etc., multi-class classification or binary classification, the distribution of samples in each category is often very unbalanced. What's more, the unbalanced distribution of samples will make the classification results obtained by applying the neural network classification method heavily biased towards classes with a large number of samples, thereby greatly reducing the classification performance. For the data of this evaluation, regardless of multi-label classification or binary classification, the distribution of samples in the classification is unbalanced. If the neural network model is adopted only, the performance will be affected.

To address the above challenges, we propose a binary classification and ensemble learning with sampling and emotion lexicon(BCEL) method. We firstly take sample on training set and perform ensemble learning with emotion lexicon finally.

4.1 Sampling

The current mainstream imbalance classification method is based on undersampling and oversampling machine learning classification methods. The main idea of the method is to use the undersampling or oversampling technique to obtain a balanced sample, and then classify the sample by a machine learning classification method. However, only a part of samples in training set can be used when adopting the undersampling technique, thus losing many samples that may be helpful for classification. Therefore, if the training set is too small, the undersampling method is not advocated. In order to make full use of the existing samples, we adopt the oversampling method (based on the undersampling technique) so that the number of positive and negative samples is basically the same. The basic model is shown in Fig. 3.

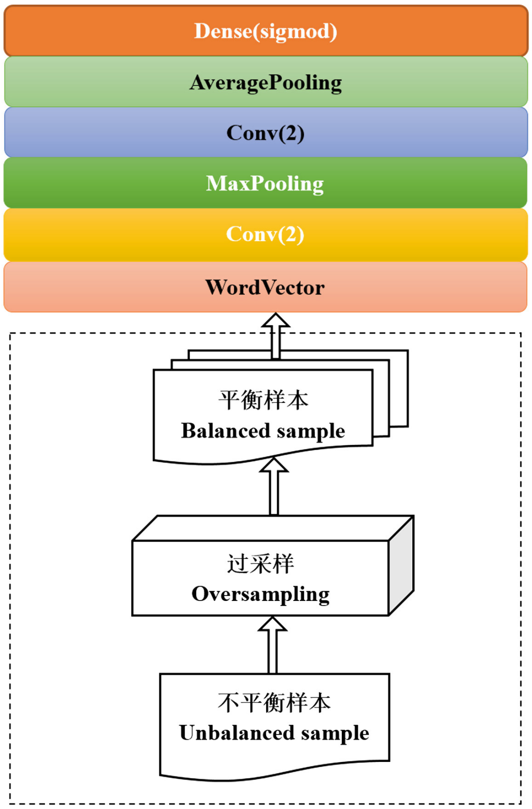


Fig. 3. The structure of Convolutional Neural Networks model with sampling

The oversampling technique we adopt is essentially based on undersampling, and the specific process is as follows: We first use the random undersampling method to perform undersampling n times for each category of samples (positive and negative samples for binary classifications). The number of samples we choose is equal to category which has minimum quantity (In this evaluation, it is the number of samples with a certain type of emotion). In the end, we get n sets of balanced samples, and then merge the n sets of balanced samples into a training set sample for training, as shown in Fig. 4.

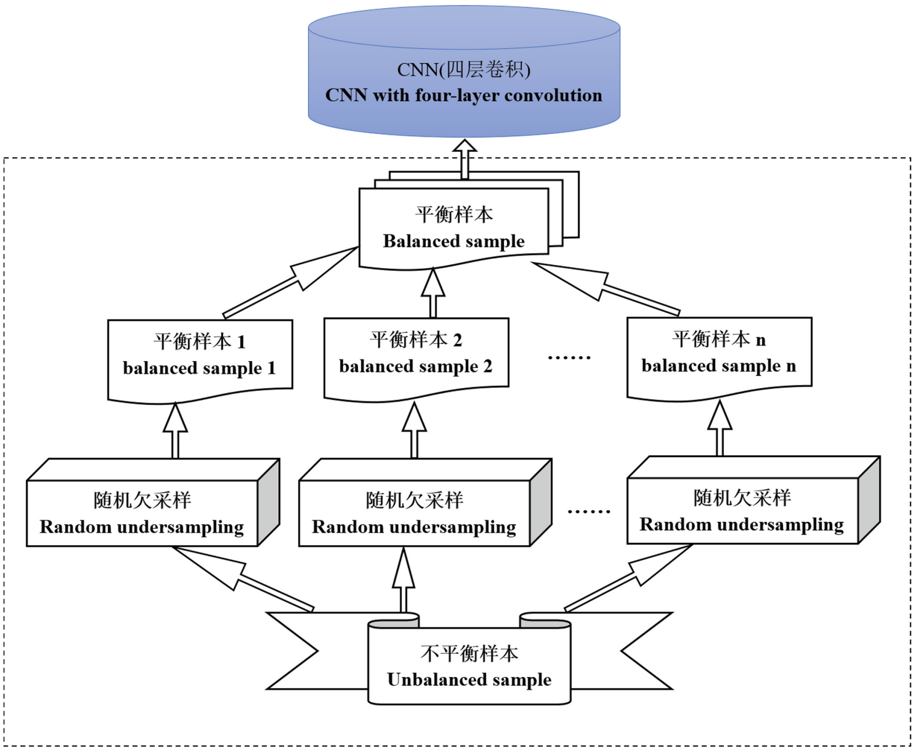


Fig. 4. The oversampling technique based on undersampling

4.2 Ensemble Learning

The idea of ensemble learning is to integrate several individual classifiers when classifying new instances, and to determine the final classification through a combination of the classification results of multiple classifiers to obtain better performance than a single classifier. If a single classifier is compared to a decision maker, the ensemble learning approach is equivalent to multiple decision makers making a common decision. Figure 5 shows the basic idea of integrative learning.

After getting multiple classifiers, the last step which is also the most important is to adopt combination strategies. Since there are few training data sets, we use a simple voting method. The specific process is as follows:

1. Using the classifier model with the highest accuracy (P-value) (we'll call it H) as the standard, whose prediction is right we think.
2. If the prediction of classifier model H is *non* (that is, there is no emotion), we look at the prediction results of other classifier models above, and take the emotion with the largest number of predictions (that is, other classifiers except H adopt the principle of the minority obeying the majority).
3. Because there are more posts containing *happiness* or *sadness*, therefore we introduce emotion lexicon for *happiness* and *sadness* classification.

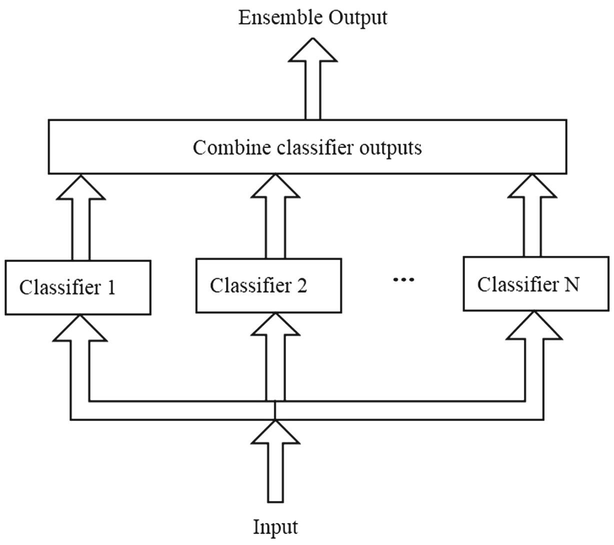


Fig. 5. The basic idea of ensemble learning

5 Experiments

5.1 Datasets

The number of training set is 6000, development set is 728, and the test set has 1200 posts in this shared task. There may be more than one emotion in a post. For each post, five basic emotions were annotated, namely *happiness*, *sadness*, *fear*, *anger* and *surprise*. The distribution of different emotions in training, development and test data is shown in Table 1.

Table 1. The distribution of different emotions.

	Train	Dev	Test
Happiness	0.304	0.302	0.408
Sadness	0.181	0.165	0.247
Anger	0.095	0.115	0.093
Fear	0.108	0.117	**0.031**
Surprise	0.109	0.126	**0.053**

In this shared task, we strictly use the given training dataset and construct the emotion lexicon by the emotion ontology of Dalian University of Technology (DLUT) and some English lexicons from github[1].

[1] https://github.com/timjurka/sentiment/tree/master/sentiment.

5.2 Experiment Settings

In our experiments, the input is a 300-dimensional vector denotes word embedding. We use the longest word-level (60) length of post in the data to unfold the BiLSTM networks. The learning rate is initialized to 0.001, and decay rate per 128 training steps is 0.9. We use a fixed number of epochs and always save the model with the best F1 score of the development set. The specific model parameters are set as shown in Table 2.

Table 2. Parameter configurations of our model.

Parameters	Configurations
Word embedding dimension	300
Learning rate	0.001
Loss function	Softmax,binary-crossentropy
PretrainedVectors	Yes
Number of epochs	20,40
Number of negatives sampled	3,20
Number of convolution	64,128,256
Size of kernel	3
Pooling Size	3
Decay rate	0.9
Batch size	128
Vocabulary size	213543

5.3 Results

According to official evaluation requirements, we compute precision (P), recall (R), and F1-Score for each emotion separately, and calculate the macro averaged P, R and F1 with all emotions. The official scoring metric is macro-averaged F1-Score.

We experimented with the mentioned methods above to get a comparison of the performance of the model on the development set, as shown in Table 3.

Table 3. The F1-Score of each model.

$F1-Score$ Model / Emotion	$Fasttext$	$RCNN(B)$	$RCNN(M)$	CNN	$CNN(S)$
happiness	0.600	0.573	0.611	0.564	**0.615**
sadness	**0.491**	0.411	0.364	0.487	0.487
anger	0.489	0.454	0.397	0.482	**0.606**
fear	0.350	0.395	0.206	0.368	**0.429**
surprise	**0.383**	0.322	0.100	0.263	0.378

As shown in Table 3, Fasttext and CNN model are binary classification. RCNN(B) is binary while RCNN(M) is multi-label classification. CNN(S) is binary classification and adopts sampling technique.

To achieve better scores, we introduce ensemble learning with emotion lexicon to determine the final result of the whole development samples by sub-models voting. The precision (P), recall (R), and F1-Score for each emotion on the development set, are shown in Table 4. Table 5 shows the test set scores.

Table 4. The final result by BCEL method on the development set.

	P	R	F1
Happiness	0.609756	0.681818	0.643776
Sadness	0.460122	0.616666	0.527016
Anger	0.501905	0.741379	0.598579
Fear	0.445455	0.502941	0.472456
Surprise	0.289412	0.603478	0.391210

Table 5. The final result by BCEL method on the test set.

	P	R	F1
Happiness	0.776765	0.695918	0.734123
Sadness	0.683128	0.560811	0.615955
Anger	0.613636	0.486486	0.542714
Fear	0.222222	0.324324	0.263736
Surprise	0.366667	0.485294	0.417722

5.4 Discussion

For emotion *happiness* and *sadness*, the difference between the positive and negative examples in quantity is not disparate. Therefore, the data sampling preprocessing does not significantly improve their performance. In order to improve the performance of the model for happiness and sadness, we tried to merge the training results of other models and the regular methods of the lexicon (Chinese and English). We fused the ensemble learning results of the above five models with the lexicon rule method. And the lexicon rule threshold k is set to 2, that is, if the number of certain emotion words in a sentence is greater than or equal to 2 or the ensemble learning result contains the emotion, then the post has that emotion. After that, the final results of development and test data are shown in Tables 4 and 5.

6 Conclusion

This paper presents an effective approach for code-switching text based on binary classification and ensemble learning. Firstly, we adopt the method of converting multi-label classification into binary classification task. At the end, the output of multiple classifiers is combined to form the final prediction result. However, the proportion of positive and negative examples in training data is very different, especially for the three emotions of *anger*, *fear*, and *surprise*. Therefore, we use the oversampling technique to balance the data and obtain relatively balanced training samples. However, for emotion *happiness* and *sadness*, the difference between the positive and negative examples in quantity is not disparate, so we introduced the English and Chinese emotion lexicon to fully explore the emotion of *happiness* and *sadness*. Next we will focus on the interaction between emotions and the construction of the Chinese-English hybrid dictionary will be an important aspect in the future.

References

1. Lee, S., Wang, Z.: Emotion in code-switching texts: corpus construction and analysis. In: Proceeding of SIGHAN-2015 (2015)
2. Wang, Z., Zhang, Y., Lee, S., Li, S., Zhou, G.: A bilingual attention network for code-switched emotion prediction. In: Proceeding of COLING-2016 (2016)
3. Mostafa, M.M.: More than words: social networks' text mining for consumer brand sentiments. Expert Syst. Appl. **40**(10), 4241–4251 (2013)
4. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995). https://doi.org/10.1007/978-1-4757-3264-1
5. Gao, W., Li, S., Lee, S.Y.M., Zhou, G., Huang, C.R.: Joint learning on sentiment and emotion classification. In: Proceedings of CIKM 2013 (2013)
6. Lin, K., Yang, C., Chen, H.: Emotion classification of online news articles from the reader's perspective. In: Proceeding of the International Conference on Web Intelligence and Intelligent Agent Technology, pp. 220–226 (2008)
7. Chen, Y., Lee, S., Li, S., Huang, C.: Emotion cause detection with linguistic constructions. In: Proceedings of COLING-10, pp. 179–187 (2010)
8. Xu, G., Meng, X., Wang, H.: Build Chinese emotion lexicons using a graph-based algorithm and multiple resources. In: Proceeding of COLING-2010, pp. 1209–1217 (2010)
9. Volkova, S., Dolan, W., Wilson, T.: CLex: a lexicon for exploring color, concept and emotion associations in language. In: Proceedings of EACL 2012, pp. 306–314 (2012)
10. Alm, C., Roth, D., Sproat, R.: Emotions from text: machine learning for text-based emotion prediction. In: Proceedings of EMNLP, pp. 579–586 (2005)
11. Aman, S., Szpakowicz, S.: Identifying expressions of emotion in text. In: Matoušek, V., Mautner, P. (eds.) TSD 2007. LNCS (LNAI), vol. 4629, pp. 196–205. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74628-7_27
12. Purver, M., Lee, S., Li, S., Huang, C.: Emotion cause detection with linguistic constructions. In: Proceedings of COLING-2010, pp. 179–187 (2010)
13. Ling, W., Xiang, G., Dyer, C., Black, A., Trancoso, I.: Microblogs as parallel corpora. In: Proceedings of ACL-2013 (2013)

14. Solorio, T., Liu, Y.: Learning to predict code-switching points. In: Proceedings of EMNLP 2008 (2008)
15. Lignos, C., Marcus, M.: Toward web-scale analysis of codeswitching. In: Proceedings of Annual Meeting of the Linguistic Society of America (2013)
16. Li, Y., Fung, P.: Code-switch language model with inversion constraints for mixed language speech recognition. In: Proceedings of COLING-2012 (2012)
17. Peng N., Wang, Y., Dredze, M.: Learning polylingual topic models from code-switched social media documents. In: Proceedings of ACL14 (2014)
18. Yan, Y., Liu, Y., Shyu, M.L., et al.: Utilizing concept correlations for effective imbalanced data classification. In: IEEE International Conference on Information Reuse and Integration, pp. 561–568. IEEE (2014)
19. Wang, Z., Lee, S.Y.M., Li, S., Zhou, G.: Emotion detection in code-switching texts via bilingual and sentimental information. In: Proceeding of ACL-2015, short paper, pp. 763–768 (2015)
20. Kim, Y.: Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv (2014)
21. Joulin, A., Grave, E., Bojanowski, P., et al.: Bag of tricks for efficient text classification, pp. 427–431 (2016)

A Multi-emotion Classification Method Based on BLSTM-MC in Code-Switching Text

Tingwei Wang[1], Xiaohua Yang[1](✉), Chunping Ouyang[1], Aodong Guo[2], Yongbin Liu[1], and Zhixing Li[3]

[1] School of Computer, University of South China, Hengyang 421001, China
xiaohua1963@usc.edu.cn
[2] College of Information Engineering, Xinhua University, Hefei 230031, China
[3] Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Abstract. Most of the previous emotion classifications are based on binary or ternary classifications, and the final emotion classification results contain only one type of emotion. There is little research on multi-emotional coexistence, which has certain limitations on the restoration of human's true emotions. Aiming at these deficiencies, this paper proposes a Bidirectional Long-Short Term Memory Multiple Classifiers (BLSTM-MC) model to study the five classification problems in code-switching text, and obtains text contextual relations through BLSTM-MC model. It fully considers the relationship between different emotions in a single post, at the same time, the Attention mechanism is introduced to find the importance of different features and predict all emotions expressed by each post. The model achieved third place in all submissions in the conference NLP&&CC_task1 2018.

Keywords: Multiple emotion classification · Code-switching texts
Attention mechanism · BLSTM multiple classifiers

1 Introduction

Emotion classification refers to mapping the information to be classified into a predefined emotional category system. Which is widely used in recommendation and public opinion analysis. According to different emotional granularity, emotion classification can be divided into binary classification (subjective, objective), ternary classification (allegative, derogatory, neutral), or multivariate classification. Among them, multivariate classification can classify the emotions more close to human real emotions. In 2001, Parrott divided human social psychology into Happiness, Sadness, Anger, Fear, and Surprise in the results of research on human social psychological and emotional expression [1].

The code-switching text contains five kinds of emotions (happiness, sadness, anger, fear, surprise).Each post contains both English and Chinese. Emotions can be expressed individually or mixed both Chinese and English. Therefore, there are four forms for expression of emotions: none, Chinese, English and both. None means this post does not contain any corresponding emotions (E1). Chinese or English means that emotions are expressed only by Chinese or English (E2, E3). Both mean that emotions are expressed in Chinese and English (E4). A single post may also contain multiple emotions (E5), so it is very different from monolingual and bilingual texts.

E1. 年底party 比较多啊，我在泰晤士小镇“美食玩家”大聚会正在准备中 ......

(At the end of the year, there are more parties, and I'm preparing for the "Gastro Gamer" party in Thames Small Town…)

E2. 心情极度不爽啊,为什么会这样, why? why? why?
(Extremely bad mood, why does this happen, why? why? why?)

E3. I'm so happy!虽然天空还飘着小雨~
(I'm so happy! Although the sky is still floating ∼)

E4. 开学以来,浮躁的情绪。不安稳的心态。确实该自己检讨一下了。。。Sig

(I have been grumpy and emotional since the first day of school, unstable mindset too. It's really time to self-evaluate…sigh.)

E5. 做了四张英语周报后发现还有两张真崩溃 i hate english。
(After making four English Weekly Reports, I discovered that there are two other real collapses. i hate english)

In recent years, deep learning technology has gradually replaced traditional machine learning methods and has become the mainstream model of emotional classification [2]; Socher et al. [3] used the recursive neural network (RNN) model to perform emotional classification on the Film review data. Kim [4] uses convolution neural network (CNN) model to classify emotion. Literature [5] uses Long-Short Term Memory (LSTM) model to comment sentences into word sequences for emotion classification; Cheng Yu et al. [6] used emotion-based LSTM model to classify Chinese product reviews.

The above literature has studied the classification of emotion carefully, but there are two shortcomings: (1) most of the studies are based on the binary or ternary classification; (2) there is no consideration for the existence of a variety of emotions at the same time in a single post. Therefore, there is a certain limitation on the restoration of human true feelings. Aiming at these two points, this paper proposes a BLSTM-MC (Multiple classifiers) model for multiple emotional classification of code-switching texts. By creating multiple classifiers of BLSTM, the model is associated with different emotional semantic information, At the same time, the Attention mechanism is introduced to different words with different text weights. The experiments use the dataset provided evaluation by the conference NLP&&CC_task1 2018. the results show that, the proposed model get third place in all submission results.

2 Related Work

High-quality Word Embedding is one of the important factors for the deep learning model. The traditional document representation are mostly based on the Bag of Words (BOW) method. It discards the word order information and the resulting text has sparseness and high dimensionality. Mikolov, Benkiv, et al. [7] expressed the text through the neural network training word vector, which solved the above problems well. Using word2vec to represent text and combining deep learning models such as convolution neural networks (CNN) [4, 8, 9], recurrent neural networks (RNN) [10, 11] and so on, emotion classification can be achieved better results than traditional Machine learning methods.

When sentence-level semantic features are modeled by word vectors, sequence models such as RNNs are widely used in emotion classification because of sequence structures in sentences or documents. In 1990, Elman [12] proposed a recurrent neural network that keeps the nodes in the hidden layers connected, but RNNs have long-distance dependence and gradient disappearance problems. In 1999, the LSTM proposed by Gers and Schmidhuber et al. [13] solved these problems by interacting with the information of the memory cells through the design of three sophisticated gating units. Literature [5] proposed using LSTM to model comment sentences into word sequences for emotion classification. However, the LSTM training process will lead to the deviation of weights; the bidirectional LSTM integrates the context information through the convolutional layer, and connects the two LSTM networks with opposite timings to the same output at the same time to improve the accuracy of the model. Cheng Lu [6] used the bidirectional LSTM model based on the attention mechanism to do emotion classify Chinese of product reviews and achieved good results in both the two-category and three-category tasks. However, the emotional classification result contains only one kind of emotion, and the emotional semantic information is lose, resulting in a single emotional outcome.

To solve the above problems, this paper proposes the LSTM-MC model, constructing five BLSTM classifiers to integrate different emotional semantic information, fully excavate the phenomenon of user multiple emotion coexistence and introduce the Attention mechanism to express the importance of different features.

3 BLSTM-MC Model

The BLSTM-MC model (see Fig. 1). First, the model enhances context semantic information by creating five BLSTM classifiers and introducing Attention mechanisms, and gets deeper features, then returns all the emotional predictions of all posts by Softmax.

3.1 Word Embedding

This paper uses the Skip-gram model to predict the words in its context window using the current word. First use the training document to construct a vocabulary, and then perform a one-hot encoding on the word. The value of each dimension in the one-hot

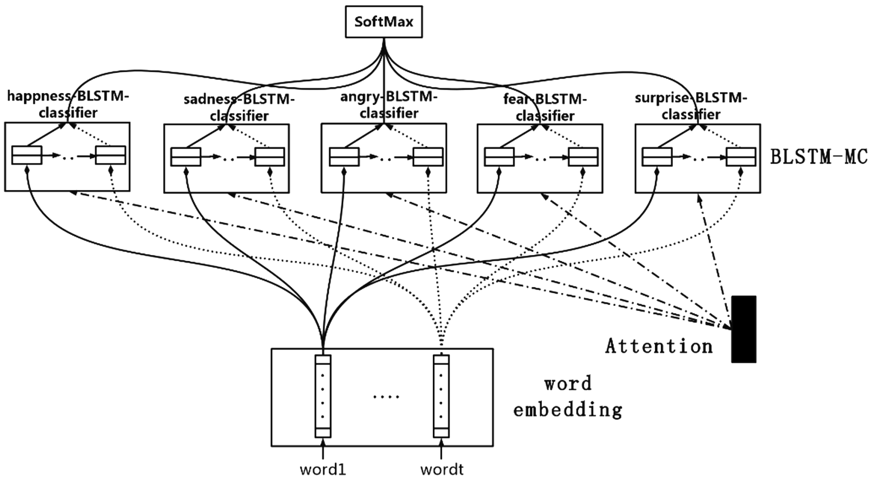


Fig. 1. BLSTM-MC model

encoding is only 0 or 1. The t-th word in the text is expressed as a vector of words: $w_t \in R^d$, where d is the dimension of the word vector. If the text length is T, the input text is represented as:

$$S = [w_1; w_2; \cdots; w_T] \in R^{T*d} \tag{1}$$

3.2 BLSTM Model

LSTM is applied to the processing of time series tasks (see Fig. 2). The BLSTM is constructed using the LSTM described by Zaremba et al. [12], and then the BLSTM model is used to integrate the context information to obtain text features.

Among them, f_t, i_t, o_t and c represent three kinds of gate mechanisms, the forget gate, the input gate, and the output gate, respectively, which control the read, write and lose operations of memory cell.

The input of this LSTM is a phrase representing the sequence $F = (F_1, \ldots, F_{l-w+1})$, the mechanism of which can be described by the following mapping.

Three gates of information flow input:

$$f_t = \sigma(w_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$

$$o_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o) \tag{4}$$

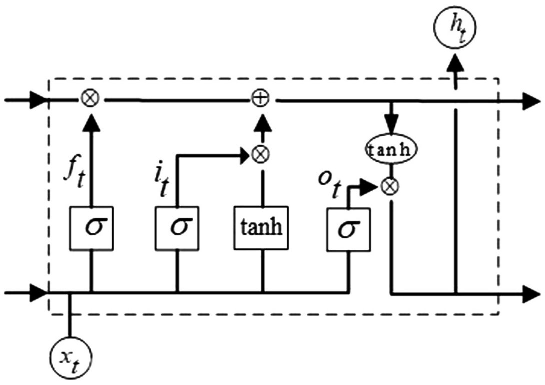


Fig. 2. LSTM structure diagram

Memory unit update:

$$\overline{c_t} = \tanh(w_c \cdot [h_{t-1}, x_t] + bc) \tag{5}$$

$$c_t = f_t * c_{t-1} + i_t * \overline{c_t} \tag{6}$$

Hidden layer unit update:

$$h_t = o_t * \tanh(c_t) \tag{7}$$

The timing t takes values $\{1, \ldots, l - w + 1\}, h_t, c_t \in R^n$, which are the hidden state and memory state at the time t, σ and tanh are the sigmoid and the hyperbolic tangent activation functions, respectively; i_t, f_t, o_t, $\overline{c_t}$ are input gates, forgetting gates, output gates, and new candidate memory states at time t, whose dimensions are equal to the hidden state dimension; * represents one by one element.

BLSTM includes forward $\overrightarrow{LSTM}$ and backward $\overrightarrow{LSTM}$ and these two parts share parameters. The forward LSTM reads F_1 to F_{l-w+1} sequentially from the phrase representation sequence, and the backward $\overrightarrow{LSTM}$ reads F_{l-w+1} to F_1 in turn, and its functions are shown in Eqs. (8) and (9).

$$\overrightarrow{h_t}, \overrightarrow{c_t} = \overrightarrow{LSTM}(F_t, \overrightarrow{h_{t-1}}, \overrightarrow{c_{t-1}}), t \in \{1, \cdots, l - w + 1\} \tag{8}$$

$$\overleftarrow{h_t}, \overleftarrow{c_t} = \overleftarrow{LSTM}(F_t, \overleftarrow{h_{t+1}}, \overleftarrow{c_{t+1}}), t \in \{l - w + 1, \cdots, 1\} \tag{9}$$

Among them, $\overrightarrow{h_0}, \overrightarrow{c_0}$ and $\overleftarrow{h_{l-w+2}}, \overleftarrow{c_{l-w+2}}$ are initialized to zero vectors. $\overrightarrow{h_t}$ is the phrase feature F_t fused with the above information representation, $\overleftarrow{h_t}$ is the phrase feature F_t fusion representation of the following information, and $h_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$ obtained by concatenating the two is the phrase representation of the fusion context information. Through the BLSTM layer, the resulting phrase sequence of fusion context information is represented by Eq. (10).

$$H = (h_1, \cdots, \mathrm{h}_{l-w+1}) \in R^{2\mathrm{n}*(l-w+1)} \tag{10}$$

3.3 Attention Mechanism

The core idea of the attention mechanism is to learn the weights of words in a word sequence to assign different attention to different content. The Attention mechanism has been widely used in image recognition [14], image annotation [15] and natural language processing [16]. In the Attention mechanism:

$$u_t = \tanh(w_w H_t + b_w) \tag{11}$$

$$a_t = soft\max(u_t^T, u_w) \tag{12}$$

$$v = \sum\nolimits_t a_t H_t \tag{13}$$

Among them, $\mathrm{u_t}$ is the hidden unit of $\mathrm{H_{t,}}$ $\mathrm{a_t}$ is the attention vector, v is the output vector after processing by the Attention mechanism. $\mathrm{u_w}$ is the context vector, initialize randomly at the beginning of the experiment, and continues to improve during the learning process.

3.4 BLSTM-MC Emotion Classification

As shown in the BLSTM-MC layer in Fig. 1, the BLSTM classifier is constructed for five categories, respectively: Happiness classifier, Sadness classifier, Anger classifier, Fear classifier and Surprise classifier. In the Happiness classifier, the code-switching text expressed by the word vector is used as the input of the classifier, and the BLSTM will combine contextual contexts to capture textual features. The Attention mechanism gives the text feature weight to the deep feature vector v, and uses Softmax to regress the final distribution of the emotion prediction probability and $\mathrm{P_i}$ to express the text. The probability of an emotional i.

$$P_i = soft\max(w_c v + b) \tag{14}$$

For example, According to the probability, we can see whether the post belongs to Happiness. The other four classifiers have the same principle as Happiness. The final emotion prediction will fuse all the emotion prediction results of BLSTM-MC and correlate different emotion semantic information.

4 Experiment

4.1 Date Set

In order to verify the validity of the model, this paper uses the data set provided by the conference NLP&&CC_task1 2018, and the labels of data sets are divided into five categories, and is the form of Code-Switching Text. The training data set is divided into training set and validation set according to the 8:2 ratio. Of which there are 1824 posts in the happiness class, 1086 in sadness, 570 in anger, 648 in fear, 651 in surprise, a

Table 1. Training data set sample.

Posts	Happiness	Sadness	Anger	Fear	Surprise
翻译真的差很多耶,是这样吗? so lovely～ (Translation is really bad, is this? so lovely～)	T	F	F	F	T
做了四张英语周报后发现还有两张真崩溃 i hate english。 (After making four English Weekly Reports, I discovered that there are two other real collapses. i hate english.)	F	T	T	F	F

total of 4104 sentences with emotions. However, 675 of the 4104 Posts contain a variety of emotions, so the total emotion contained in the corpus is 4779, and the remaining 1896 unmarked emotions. From the data, it is found that the posting of sentimental coexistence accounts for 16.4% of the total number of emotional corpus, indicating that the situation of feeling coexistence occupies a large proportion in the task of emotional classification. The sample of the data set (see Table 1).

4.2 Data Preprocessing and Parameter Setting

Since the data set is a form of code-switching text, this paper uses Google translate API to translate the data set into Chinese text, and then preprocesses the data set. Then use the Skip-gram model of the word2vec tool to train the word vector, and the word vector parameter setting (see Table 2), finally gets a word vector list, the words that do not appear in the word vector list, random initialization of the word vector, and the dynamic update of the word vector during the training process.

Table 2. Word2vec parameter setting.

Model	Skip-gram
Window_size	7
Word vector dimension	100
Sampling	Negative sampling
Word frequency threshold	10

This model selects sigmoid as an activation function, the learning rate is 0.01, Adam is used as an optimizer, Dropout is used to prevent over fitting, and set value to 0.5, the cross entropy is used as the loss function, the batch size (batch_size) is 32, and the number of training times (n_epochs) is 1000.

4.3 Experiment and Analysis

F1-Score is compute calculated for each emotion separately, and compute the macro averaged F1 with all emotions, we use the conference NLP&&CC_task1 2018 scoring metric Marco-F1 as the evaluation standard. The BLSTM-MC model is compared with the model previously tested on the dataset, including BLSTM and the BLSTM based on Attention mechanism model, The results (see Table 3)

Table 3. Multi-model classification F1 value.

Model	Happiness	Sadness	Anger	Fear	Surprise	Marco-F1
BLSTM	0.691	0.413	0.543	0.164	0.256	0.413
BLSTM based on Attention mechanism	0.695	0.634	0.528	0.289	0.136	0.456
BLSTM-MC	**0.710**	**0.652**	**0.540**	**0.292**	**0.139**	**0.467**

1. BLSTM: Improved model of RNN proposed by Schmidhuber et al. [13].
2. BLSTM based on Attention mechanism: The emotion analysis model proposed in the literature [6] for Chinese product reviews.

It can be seen from Table 3 that: Compared with the BLSTM model based on Attention mechanism, the Marco-F1 value of BLSTM model is reduced by 43%, which indicates that after joining the Attention mechanism, it can not only capture the impact of input nodes on output nodes, but also enrich the semantic information and reduce the information loss in the process of feature extraction.

The Marco-F1 value of the BLSTM-MC model proposed in this paper is higher 54% than that of the BLSTM model's Marco-F1.which is higher 11% than the Macro-F1 value of the BLSTM based on the Attention mechanism. It is proved that the proposed BLSTM-MC model considers the relationship between text emotions well, solves the problem of the loss of emotional semantic information and improves the Marco-F1 value of the model.

The results of the BLSTM-MC model experiment were compared with those of two terms DeepIntell and DUTIR_938. DeepIntell is a team name, which achieved the best result in the conference NLP&&CC_Task1 2018, DUTIR_938 is a term name, which achieved second place, we also compared with the median results, which is named baseline in the table. The results of the experiment (see Table 4).

It can be seen from Table 4 that: The Marco-F1 value of the BLSTM-MC model used in this article is 0.467, only lower 48% than that of DeepIntell, lower 1% than that of DUTIR_938, but the values of Sadness and Fear are higher than DeepIntell and DUTIR_938, the values of each class are also significantly higher than those of the

Table 4. Participation in NLP&CC2018_Task1 teams performance comparison.

Team	Happiness	Sadness	Anger	Fear	Surprise	Marco-F1
DeepIntell	0.734	0.616	0.543	0.264	0.418	0.515
DUTIR_938	0.715	0.521	0.541	0.166	0.396	0.468
Our Team	**0.710**	**0.652**	**0.540**	**0.292**	**0.139**	**0.467**
Baseline	0.587	0.500	0.390	0.108	0.128	0.342

Baseline. Among them, the F1 values of Happiness, Anger and Surprise corresponding to emotion are slightly lower than those of the other two teams. This paper calls Google translation to translate them into monolingual texts, because the texts in social media are relatively informal and require high quality of translation.

5 Conclusions

This paper proposes a BLSTM-MC model for the multi-emotion classification of code-switching texts. The code-switching text is converted to the word vector with the Skip-gram model, and the context information is fused by the multi classifiers of BLSTM. Take full account of the fact that a single post has multiple emotion s at the same time, mine the importance of different features, and to predict all emotions expressed by each posts, In code-switching text, each post contains a variety of languages, such as Chinese and English, and Chinese also contains Cantonese and other forms of language, which is more challenging than monolingual or bilingual texts.

Acknowledgements. This research work is supported by National Natural Science Foundation of China (No. 61402220, No. 61502221), the Philosophy and Social Science Foundation of Hunan Province (No. 16YBA323), the Double First Class Construct Program of USC (2017SYL16), scientific and technological research program of Chongqing municipal education commission (No. KJ1500438), basic and frontier research project of Chongqing, China (No. cstc2015jcyjA40018).

References

1. Parrott, W.: Emotions in Social Psychology: Essential Readings, pp. 1–392. Psychology Press, Philadelphia (2001)
2. Tang, D., Qin, B., Liu, T.: Deep Learning for sentiment analysis: successful approaches and future challenges. Wiley Interdisc. Rev. Data Min. Knowl. Discov. **5**(6), 292–303 (2015)
3. Socher, R., Huval, B., Manning, C.D., et al.: Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1201–1211. MIT Press, Cambridge (2012)
4. Kim, Y.: Convolutional neural networks for sentence classification (2014). arXiv:1408.5882
5. Zhu, X., Sobihani, P., Guo, H.: Long short-term memory over recursive structures. In: Proceedings of the International Conference on Machine Learning, pp. 1604–1612. ACM, New York (2015)

6. Cheng, J.: Research on two-dimensional LSTM model based on attention mechanism in sentiment classification of Chinese commodity reviews. Softw. Eng. **20**(11), 4–6 (2017)
7. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: Proceedings of NIPS, pp. 3111–3119 (2013)
8. Collobert, R., Weston, J., Bottou, L., et al.: Natural language processing (Almost) from scratch. J. Mach. Learn. Res. **12**(1), 2493–2537 (2011)
9. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences, p. 1. Eprint Arxiv (2014)
10. Lai, S., Xu, L., Liu, K., et al.: Recurrent convolutional neural network for text classification. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2267–2273 (2015)
11. Teng, Z., Vo, D.T., Zhang, Y.: Context-sensitive lexicon features for neural sentiment analysis. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1629–1638 (2016)
12. Elman, J.L.: Finding structure in time. Cogn. Sci. **14**(2), 179–211 (1990)
13. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
14. Mnih, V., Heess, N., Graves, A., et al.: Recurrent models of visual attention. In: Advances in Neural Information Processing Systems 27 (NIPS), pp. 2204–2212 (2014)
15. Xu, K., Ba, J., Kiros, R., et al.: Show, attend and tell: neural image caption generational with visual attention. In: Proceedings of the 32nd International Conference on Machine Learning (ICML), pp. 2048–2057 (2015)
16. Zheng, X., Ding, L., Wan, R.: Hierarchical BGRU model based on user and product attention mechanism. Comput. Eng. Appl., 27 May 2017
17. Zhang, Y., Jiang, Q.: Textual sentiment analysis based on two LSTM structures. Software **1**, 116–120 (2018)
18. Liu, Y., Ouyang, C., Li, J.: Ensemble method to joint inference for knowledge extraction. Expert Syst. Appl. **83**, 114–121 (2017)

Short Papers

Improved Character-Based Chinese Dependency Parsing by Using Stack-Tree LSTM

Hang Liu, Mingtong Liu, Yujie Zhang(✉), Jinan Xu, and Yufeng Chen

School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
{16120389,16112075,yjzhang,jaxu,chenyf}@bjtu.edu.cn

Abstract. Almost all the state-of-the-art methods for Character-based Chinese dependency parsing ignore the complete dependency subtree information built during the parsing process, which is crucial for parsing the rest part of the sentence. In this paper, we introduce a novel neural network architecture to capture dependency subtree feature. We extend and improve recent works in neural joint model for Chinese word segmentation, POS tagging and dependency parsing, and adopt bidirectional LSTM to learn n-gram feature representation and context information. The neural network and bidirectional LSTMs are trained jointly with the parser objective, resulting in very effective feature extractors for parsing. Finally, we conduct experiments on Penn Chinese Treebank 5, and demonstrate the effectiveness of the approach by applying it to a greedy transition-based parser. The results show that our model outperforms the state-of-the-art neural joint models in Chinese word segmentation, POS tagging and dependency parsing.

Keywords: Chinese word segmentation
POS tagging and dependency parsing · Dependency subtree
Neural network architecture

1 Introduction

Transition-based parsers [1–4] have been shown to be both fast and efficient for dependency parsing. These dependency parsers can be very accurate for languages that have natural separators such as blanks between words, but for Chinese that do not contain natural separators, these parsers maybe get worse.

One reason for the lower accuracy of Chinese dependency parser is error propagation: Chinese dependency parsing requires word segmentation and POS tagging as pre-processing steps; once the pipeline model makes an error in word segmentation, more errors are likely to follow. In order to address the issue, transition-based Chinese word segmentation, POS tagging and dependency parsing joint model are proposed, jointly learning the three tasks [5–8]. Modern approaches to joint model can be broadly categorized into feature engineering joint model and neural joint model. The feature engineering joint model [5–7] needs to manually define a large number of feature templates, and extracts the features from feature templates. The neural joint model [8] automatically extracts features by neural network such as RNN or LSTM, and then uses a small number of feature templates for model parsing and decision. These models perform better than pipeline models, but they ignore the complete dependency subtree feature, which has been proven to be an effective information for improving model performance in previous works [9–11].

In this paper, to improve Character-based dependency parsing, we extend the work of Kurita [8] using bidirectional LSTMs to learn n-gram feature and context information, and introduce a novel neural network architecture to encode the built dependency subtrees information, which use richer information and avoiding feature engineering. The neural network architecture is a stack structure combined with LSTM cell and Tree LSTM cell, called Stack-Tree LSTM, which can capture all the built dependency subtrees information. Then the subtree feature and n-gram feature are fed into a neural network classifier to make parsing decisions within a transition-based dependency parsing.

In the experiments, we evaluate our parser on the CTB-5 dataset and experimental results show that F1 scores of the Chinese word segmentation, POS tagging and dependency parsing reach 97.78%, 93.51% and 79.66% respectively, which are better than the baseline model in each task.

2 Related Work

In Chinese, the character-based dependency parsing solution was first proposed in Hatori [5]. He assumed that there was a dependency between the characters in the internal words, and unified the three tasks in one framework. The benefit of the solutions is that it can start with the character-level, and Chinese word segmentation, POS tagging and dependency parsing can be done in a joint framework, which improves the accuracies of three tasks and does not suffer from the error propagation. Zhang [6] studied the character-based Chinese dependency parsing by using pseudo and annotated word structures, and obtained better accuracies on three tasks. Moreover, they further analyzed some important factors for intra-word dependencies and demonstrated that intra-word dependencies can improve the performance of the three tasks. Guo [7] proposed a solution to transform the conventional word-based dependency tree into character-based dependency tree by using the internal structure of words and proposed a semi-supervised joint model for exploiting 2-g string feature and 2-g dependency subtree feature.

These methods achieve high accuracy on three tasks but rely heavily on feature engineering, which requires a lot of expertise and is usually incomplete. In addition, these methods are not able to learn the context information of the sentence being parsing. Recently, Kurita [8] proposed the first neural joint parsing model and explored the neural network with few features using n-gram bidirectional LSTMs avoiding the detailed feature engineering. Our model is similar to theirs. However, these methods are lacking in that it cannot capture all word dependencies in a subtree and all dependency subtrees. To provide richer information, we consider all word dependencies by using subtree distributional representation.

Specific to the subtree representation, Dyer [9] developed the Stack Long Short-Term Memory (Stack LSTM) architecture, which does incorporate recursive neural network and look-ahead, and yields high accuracy on the word-level dependency parsing. However, the architecture has a risk of suffering from gradient vanishing when dealing with deep subtrees. In this paper, we solve the issue by using Tree LSTM [10] in the Stack LSTM, which has similar gates to LSTM cell and has the capability to memorize important information and forget unimportant one.

3 Character-Level Neural Network Parser

In this section, we describe the architecture of our model and its main components, which is summarized in Fig. 1. Our model is clearly inspired by and based on the work of Kurita [8], which uses four bidirectional LSTMs to capture N-gram feature. There are a few structural differences: (1) we use Stack-Tree LSTM to capture dependency subtrees feature, (2) we use the POS tags to participate in actions decision.

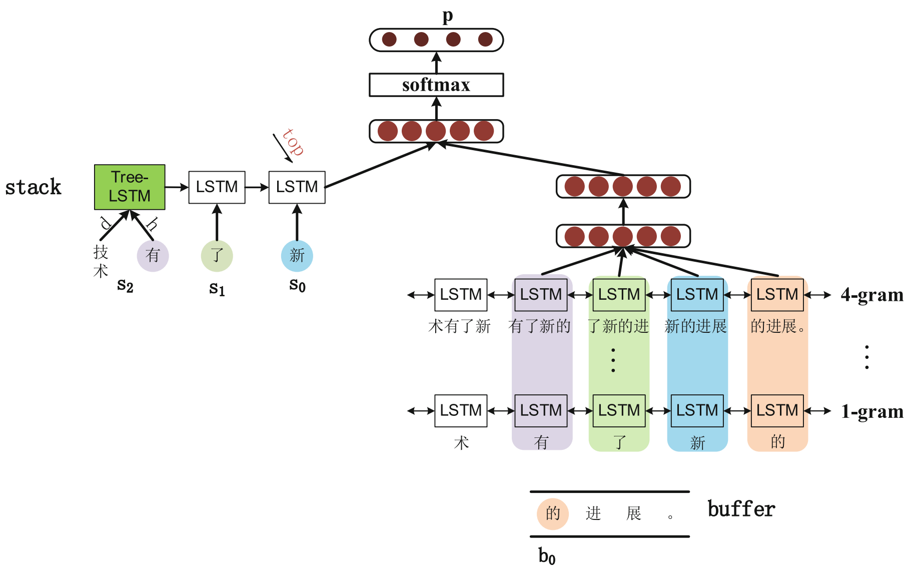


Fig. 1. The neural joint model for Chinese word segmentation, POS tagging and dependency parsing. The model consists of four bidirectional LSTM for extracting n-gram feature, Stack Tree LSTM for extracting subtree feature and MLP for predicting the possible action. Parser state computation encountered while parsing the sentence "技术有了新的进展". s_i represents the i + 1th element of the top of the stack; b_0 represents the first element of the buffer.

3.1 Transition System

Transition-based dependency parsing scan an input sentence from left to right, and perform a sequence of transition actions to predict its parse tree. The input sentence is put into a buffer and partially built tree fragments are organized by a stack. Paring starts with an empty stack and a buffer consisting of the whole input sentence. As the basis of our parser, we employ the arc-standard system [2] for dependency parsing, one of the popular transition systems, and follow Hatori [5] to modify the system for character-level transition, for which the actions are:

- SH (t) (shift): Move the front character b_0 from the buffer onto the top of the stack as a new word, and the POS tag t is attached to the word.
- AP (append): Append the front character b_0 of the buffer to the end of the top word of the stack.
- RR (reduce-right): Add a right arc between the top element s_0 and the second top element s_1 on the stack ($s_1 \rightarrow s_0$), and remove s_0 from the stack.
- RL (reduce-left): Add a left arc between the top element s_0 and the second top element s_1 on the stack ($s_1 \leftarrow s_0$), and remove s_1 from the stack.

In the character-level transition system, while the goal of SH(t) and AP operations is to construct a new word, where each word is initialized by the action SH(t) whereas AP makes the word longer by adding one character, the goal of RR and RL operations is to construct a dependency subtree. In this paper, we examine only greedy parsing, and this class of parsers is of great interest because of their efficiency. At each time step, a transition action is taken to consume the characters from the buffer and build the dependency tree.

3.2 Subtree Feature Layer

Inspired by Dyer [9], we propose a novel neural network (Stack-Tree LSTM) architecture that integrates Stack LSTM and Tree LSTM, improving the representational capacity of Stack LSTM and solving the problem of Tree LSTM that all built subtrees information cannot be captured at the same time. The neural network architecture is presented in Fig. 2. When the input of the stack node is a subtree, the LSTM cell is replaced by Tree LSTM cell in the stack. And the inputs of the Tree LSTM cell are the right child representation and left child representation encoded by the Tree LSTM.

Specifically, when SH operation is performed, the top item on the stack provides previous time state and a new LSTM cell is pushed into the stack. The new state is computed by the LSTM cell and the cell's input is the character vector to be shifted. Different with SH operation, AP operation updates the stack state by using appended character string vector.

When RL or RR operation is performed, the representations of top two items are popped off of the stack and fed into a Tree LSTM cell. Intuitively, the Tree LSTM cell combines two vectors representations from the stack into another vector, which represents a new dependency subtree and historical information. For example, in Fig. 1(d), a new dependency tree is built, where w_4 is the head node and w_5 is the dependency node, and the result of the Tree LSTM cell, namely new state, are computed as follows:

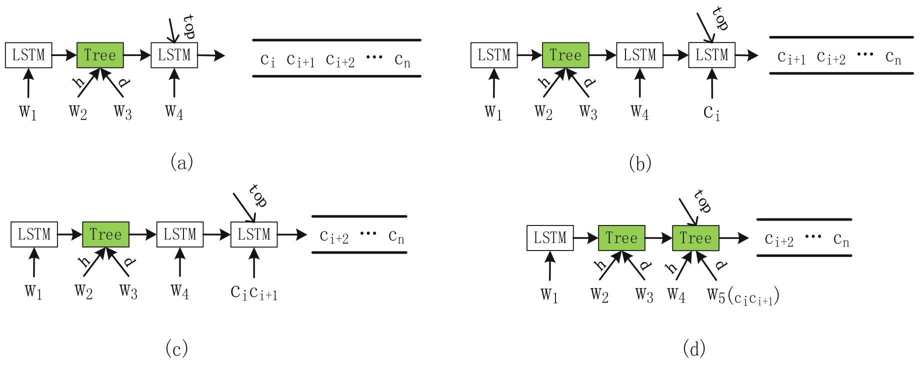


Fig. 2. The Stack Tree LSTM consists of LSTM cell and Tree LSTM cell. The figures show four configurations: (a) a stack with the input of two words w_1, w_4 and a subtree, (b) the result of a SH operation to this, (c) the result of a AP operation to (b), and (d) the result of applying a RR operation. The *top* pointer is used to access the output of the network.

$$\begin{bmatrix} i_t \\ f_{head} \\ f_{dep} \\ o_t \\ \tilde{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \left(W \begin{bmatrix} h_{head} \\ h_{dep} \\ h_{t-1} \end{bmatrix} + b \right) \tag{1}$$

$$c_t = f_{head} * c_{head} + f_{dep} * c_{dep} + i_t * \tilde{c}_t \tag{2}$$

$$h_t = o_t * tanh(c_t) \tag{3}$$

Where σ is the sigmoid activation function, * is the elementwise product. The resulting vector embeds the subtree in the same space as the words and other subtrees.

At each time step, all built subtrees are encoded by Stack-Tree LSTM, which integrates all items information to the top item of the stack. By querying the d-dimensional vector s_{top} of the top item, a continuous-space embedding of the contents of the current stack state is available. Then the s_{top} is fed into multi-layer perceptron (MLP) to predict the next possible transition action. When a predicted transition action is performed, the state of the stack will be updated and the output at the top of the stack will represent the new stack state.

3.3 N-Gram Feature Layer

A k-dimensional n-gram feature representation of each character is learned in this layer. Given n-characters input sentence s with characters c_1, …, c_n, we extract the uni-gram features c_i, bi-gram character string c_ic_{i+1}, tri-gram character string $c_ic_{i+1}c_{i+2}$ and four-gram character string $c_ic_{i+1}c_{i+2}c_{i+3}$, and create four sequences of input vectors $uni_{1:n}$, $bi_{1:n}$, $tri_{1:n}$, and $four_{1:n}$, in which all n-gram embeddings are given by the embedding of words and characters or the dynamically generated embedding of character strings.

And then these four sequences are fed as input to four bidirectional LSTMs respectively. These bidirectional LSTMs capture the input element with their contexts, which learn a good feature representation for parsing.

In this paper, a simple feature function $\phi(c)$ is used to extract four atomic features from a parsing configuration, which consists of the top 3 items on the stack and the first item on the buffer. Given a parse configuration q = $(\ldots|s_2|s_1|s_0,\ b_0|\ldots)$, the feature function is defined as:

$$\phi(c) = \upsilon_{s_2} \circ \upsilon_{s_1} \circ \upsilon_{s_0} \circ \upsilon_{b_0} \tag{4}$$

$$v_i = biLSTM_{uni}(i) \circ biLSTM_{bi}(i) \circ biLSTM_{tri}(i) \circ biLSTM_{four}(i) \tag{5}$$

Where $\circ$ is the concatenate operation.

3.4 Actions Decision Layer

This layer learns a classifier to predict the correct transition actions, based on n-gram features and subtree features extracted from the configuration itself. We implement three hidden layers composed h rectified linear units (Relu).

First, two feed-forward neural layers with Relu activation function project the n-gram layer's output from *4* k-dimensional vector space into a h-dimensional vector space. The purpose of the two layers is to fine-tune the n-gram feature embedding, which helps the model to capture deeper n-gram feature and more effective global information. Next, the resulting embedding h_2 is concatenated with the output s_{top} of Stack-Tree LSTM, and fed into the last hidden layer with Relu activation function.

$$h_3 = max\{0, W_{\text{com}}(h_2 \circ s_{top}) + b_{\text{com}}\} \tag{6}$$

Where $W_{com} \in \mathbb{R}^{h \times (h+d)}$ is the learned parameter matrix, $b_{com} \in \mathbb{R}^h$ is bias term.

Finally, h_3 is mapped into a softmax layer that outputs class probabilities for each possible transition operation:

$$p = softmax(Wh_3) \tag{7}$$

Where $W \in \mathbb{R}^{m \times h}$ and m is the number of transition actions.

3.5 Training

Given a set of training examples, the training objective of the greedy neural joint parser is to minimize the cross-entropy loss, plus a l_2-regularization term:

$$\mathcal{L}(\theta) = -\sum\nolimits_{i \epsilon A} logp_i + \frac{\lambda}{2}\|\theta\|^2 \tag{8}$$

A is the set of all gold actions in the training data and θ is the set of all parameters. The parameters are learned by minimizing the loss on the training data via the Adam optimizer [12]. The initial learning rate is 0.001. To avoid overfitting, we use dropout [13] with the rate of 0.5 for regularization, which is applied to all feedforward connections.

4 Word and String Representation

In n-gram layer, we prepare the same training corpus with the segmented word files and the segmented character files. Both files are concatenated and learned by word2vec [14]. The characters and words are embedded in the same vector space during pre-training. The embedding of the unknown character string, consisting of character c_1, c_2, …, c_n, is obtained by the mean of each character embedding $v(c_i)$ it contains. Embeddings of words, characters and character strings have the same dimension.

In subtree feature layer, when a character string becomes a word, the embedding of the word is queried in the pre-trained embeddings, if not, the same way encoding for unknown character string is adopted to get the word embedding. Different with Kurita [8], we use the predicted POS tags in our model, provided as auxiliary input to the parser. Specifically, the predicted POS tag embedding t of the word is concatenated with the word embedding w. A linear map is applied to the resulting vector and passed through a component-wise Relu.

$$v = max\{0, W_{word}[w;t] + b_{word}\} \tag{9}$$

The POS embedding and word embedding are learned together with the model.

5 Experiments

5.1 Experimental Settings

In this section, we evaluate our parsing model on the Penn Chinese Treebank 5.1 (CTB-5), splitting the corpora into training, development and test sets, following the splitting of [15]. The development set is used for parameter tuning. Pre-trained word and characters embeddings are learned from the Gigaword corpus and word2vec [14], as segmented by the Stanford Chinese Segmenter [16].

We use standard measures of word-level precision, recall, and F1 score to evaluate model performance on three tasks, following previous works [5–8]. Dependency parsing task is evaluated with the unlabeled attachment scores excluding punctuations. The POS tags and dependencies cannot be correct unless the corresponding words are segmented correctly.

Dimensionality. Our model sets dimensionalities as follows. Bidirectional LSTM and Stack Tree LSTM hidden states are of size 200. Embeddings of POS tags used in Stack Tree LSTM have 32 dimensions. Pre-trained word and character embeddings have 200 dimensions and three hidden layers in classifier have 400 dimensions.

5.2 Experimental Results and Analysis

Effects of Different Composition Methods in Joint Model. We conducted experiments to verify the capability of the Stack-Tree LSTM (ST-LSTM) on the dependency tree representations. We compare our ST-LSTM with three popular composition methods: Stack LSTM [9], recursive convolutional neural network [11], and a composition function based on bidirectional LSTM [17]. Table 1 show the comparison of F1 scores on three tasks. Clearly, our model is superior in terms of POS tagging and dependency parsing. However, we notice that the composition function of recursive convolutional neural network outperforms our model on Chinese word segmentation tasks. A likely reason for the close performance with our model may be the feature of relative distance between words. In future work, we also try to use distance feature to improve model performance.

Table 1. Experimental results for different composition functions. S-LSTM, B-LSTM, and RCNN denote Stack LSTM, bidirectional LSTM and recursive convolutional neural network respectively. ST-LSTM denotes Stack-Tree LSTM.

	Seg	POS	Dep
S-LSTM	97.71	93.36	79.21
B-LSTM	96.69	92.10	79.02
RCNN	**98.03**	93.35	79.58
ST-LSTM	97.78	**93.51**	**79.66**

Effects of POS Tags in Joint Model. Furthermore, we also conducted experiments to test the effectiveness of the predicted POS tagging on each task. We implemented two model: ST-LSTM and ST-LSTM model without POS tags (–POS). Concretely, we use predicted POS tagging and pre-trained embedding as word representations in ST-LSTM, but we only use pre-trained embedding as word representations in –POS. As shown in Table 3, performance of model without POS tags is weaker than the basic model in word segmentation and dependency parsing. In contrast, the basic model with POS tags gives a 0.21% accuracy improvement in dependency parsing.

Final Results. Table 2 shows the final test results of our parser for Chinese word segmentation, POS tagging and dependency parsing. Considering that the model proposed can extract the information of children's nodes, we only implement the feature function of four features. We also include in the table results from the first joint parser of Hatori [5], the using inter-word dependencies and intra-word dependencies parser of Guo [7], the arc-eager model of Zhang [6], the feature based parser of Kurita [8], and the n-gram bidirectional LSTM greedy model with four and eight features of Kurita [8].

Overall, our parser substantially outperforms the four features n-gram bidirectional LSTM model of Kurita [8], both in the full configuration and in the –POS conditions we report. Moreover, we find that our model can learn better dependency tree representations and achieve higher accuracies in each task than other composition function. And we note that this is a significant improvement in dependency parsing only after

Table 2. Final results. * denotes the feature engineering.

Model	Method	Seg	POS	Dep
Hatori12*	Beam	97.75	94.33	81.56
Guo14*	Beam	97.52	93.93	79.55
Zhang14*	Beam	97.67	94.28	**81.63**
Kurita17*	Greedy	**98.24**	**94.49**	80.15
Kurita17(4feat)	Greedy	97.72	93.12	79.03
Kurita17(8feat)	Greedy	97.70	93.37	79.38
ST-LSTM	Greedy	**97.78**	93.51	**79.66**
-POS	Greedy	97.74	**93.53**	79.45

using predicted POS information in word representation. Our model performs slightly worse than these joint models using large feature set, but we do not rely on feature engineering.

6 Conclusion

In this paper, we introduced Stack-Tree LSTM, a novel composition function to encode dependency subtrees from characters and words for Chinese word segmentation, POS tagging and dependency parsing. Our model only relies on effectively feature function and architecture design, and is able to automatically learn these useful features for making decision. Through a series of experiments, we demonstrated that our approach provides substantial improvement over the baseline methods, by capturing the subtree nodes information and more dependency structures.

In the future, we will expand the scale of the experiment and further verify the effectiveness of the proposed method. In addition, we further explore better way to learning dependency trees representations.

Acknowledgments. The authors are supported by the National Nature Science Foundation of China (Contract 61370130 and 61473294), and the Beijing Municipal Natural Science Foundation (4172047).

References

1. Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: International Workshop on Parsing Technologies 2003, Nancy, France, pp. 195—206 (2003)
2. Nivre, J.: Incrementality in deterministic dependency parsing. In: Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together, pp. 50–57. Association for Computational Linguistics (2004)
3. Zhang, Y., Clark, S.: A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam search. In: Proceedings of EMNLP, Hawaii, USA (2008)

4. Huang, L., Sagae, K.: Dynamic programming for linear-time incremental parsing. In: Proceedings of ACL, Uppsala, Sweden, pp. 1077–1086, July 2010
5. Hatori, J., Matsuzaki, T., Miyao, Y., Tsujii, J.I.: Incremental joint approach to word segmentation, pos tagging, and dependency parsing in Chinese. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. Long Papers, vol. 1, pp. 1045–1053. Association for Computational Linguistics (2012)
6. Zhang, M., Zhang, Y., Che, W., Liu, T.: Character-level Chinese dependency parsing. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics. Long Papers, vol. 1, pp. 1326–1336. Association for Computational Linguistics (2014)
7. Guo, Z., Zhang, Y., Su, C., Xu, J.: Character-level dependency model for joint word segmentation, POS tagging, and dependency parsing in Chinese. J. Chin. Inf. Process. **E99.D**(1), 257–264 (2014)
8. Kurita, S., Kawahara, D., Kurohashi, S.: Neural joint model for transition-based chinese syntactic analysis. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Long Papers, vol. 1, pp. 1204–1214. Association for Computational Linguistics (2017)
9. Dyer, C., Ballesteros, M., Ling, W., Matthews, A., Smith, N.A.: Transition-based dependency parsing with stack long short-term memory. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Long Papers, vol. 1, pp. 334–343. Association for Computational Linguistics (2015)
10. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Long Papers, vol. 1, pp. 1556–1566, Beijing, China. Association for Computational Linguistics (2015)
11. Zhu, C., Qiu, X., Chen, X., Huang, X.: A re-ranking model for dependency parser with recursive convolutional neural network. Comput. Sci. (2015)
12. Kingma, D.P., Adam, J.B.: A method for stochastic optimization. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Long Papers, vol. 1 (2015)
13. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Volume abs/1301.3781 (2013)
15. Jiang, W., Huang, L., Liu, Q., Lu, Y.: A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In: Proceedings of ACL-2008: HLT, pp. 897–904. Association for Computational Linguistics (2008)
16. Tseng, H., Chang, P., Andrew, G., Jurafsky, D., Manning, C.: A conditional random field word segmenter for SIGHAN bakeoff 2005. In: Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing (2005)
17. Dyer, C., Kuncoro, A., Ballesteros, M., Smith, N.A.: Recurrent Neural Network Grammars, pp. 199–209. The North American Chapter of the Association for Computational Linguistics (2016)

Neural Question Generation with Semantics of Question Type

Xiaozheng Dong, Yu Hong(✉), Xin Chen, Weikang Li, Min Zhang, and Qiaoming Zhu

School of Computer Science and Technology of Jiangsu Province, Soochow University, Suzhou 215006, Jiangsu, China
{xzdong,xchen121,wkli88}@stu.suda.edu.cn
{hongy,minzhang,qmzhu}@suda.edu.cn

Abstract. This paper focuses on automatic question generation (QG) that transforms a narrative sentence into an interrogative sentence. Recently, neural networks have been used in this task due to its extraordinary ability of semantics encoding and decoding. We propose an approach which incorporates semantics of the possible question type. We utilize the Convolutional Neural Network (CNN) for predicting question type of the answer phrases in the narrative sentence. In order to incorporate the question type semantics into the generating process, we classify the question type which the answer phrases refer to. In addition, We use Bidirectional Long Short Term Memory (Bi-LSTM) to construct the question generating model. The experiment results show that our method outperforms the baseline system with the improvement of 1.7% on BLEU-4 score and beyonds the state-of-the-art.

Keywords: Question generation · Question type · Answer phrases

1 Introduction

The goal of automatic question generation is to create natural questions from answer phrases in the narrative sentence, where the generated questions can be answered by them. Normally, the answer phrases are short texts in the sentence. Listed below are two questions generated by the same narrative sentence, where the S_{nar} is an original narrative sentence and the S_{que} 1 and S_{que} 2 are the questions generated respectively based on the answer phrases AP 1 and AP 2.

(1) $\mathbf{S}_{nar}$: *maududi founded the jamaat−e−islami party in 1941 and remained its leader until 1972.*
$\mathbf{S}_{que}$ **1**: *when did maududi found the jamaat-e-islami party?*
AP 1: *in 1941 and remained its leader until 1972*
$\mathbf{S}_{que}$ **2**: *who found the jamaat-e-islami party?*
AP 2: *maududi*

Question generation system is widely applied to many areas, such as reading comprehension, healthcare administration, knowledge-based question answering (KB-QA), and so on. For example, we can build a QA knowledge base by the generated questions and the raw narrative sentences. Incorporating the existing information retrieval technique into the knowledge base, we can create a practical QA system easily. In this case, the S_{nar} can serve as the answer, while the S_{que} 1 or S_{que} 2 serve as the questions.

It is challenging to generate questions in an automatic way. Not only the narrative sentence and the generated question share similar semantics, but also the generated question type should be correct. At the same time, the generated question ought to be a natural quesiton. As we can see in the example 1), S_{nar} and S_{que} 1 share similar semantics, "a person creates a party at some time", but are represented in different ways. Furthermore, QG can boil down to a translation problem. Therefore, existing works usually apply the translation models to handle this task [1], due to their brilliant ability of semantic encoding and decoding, especially reordering the sentence.

In this paper, we propose to utilize the question type prediction for phrases to improve the existing translation model. Adding the question type to the encoding and decoding process provides extra information to specify which type of question to generate, thus improving the performance of the translation model. In the beginning, we predict the question type which those ground truth answer phrases refer to. As shown in example 1), the AP 1 refers to a when type question and the AP 2 refers to a who type question. Then the question type is incorporated into the translation model based on Bi-LSTM [11], with the aim to provide more information for the encoding and decoding process.

Experiments are conducted on the Stanford Question Answering Dataset (SQuAD) [16], and the results show that even using such a classification model, with a precision of about 67%, for question type prediction, our translation model outperforms the baseline by increasing 1.7% on BLEU-4.

In the section below we discuss related work (Sect. 2), the details of our approach (Sect. 3) and describe our experiment setup (Sect. 4). We analyze the results in Sect. 5. Lastly, we conclude the paper in Sect. 6.

2 Related Work

Question generation has attracted the attention of the natural language generation (NLG) community, since the work of Rus et al. [17]

Heilman et al. [10] use the drafting rules to transform the declarative sentences and reorder the generated questions through the logistic regression (LR) model. They rank the generated questions and obtain the former 20% as the generation results, which nearly doubles the percentage of the questions rated as acceptable by annotators up to 52%. In addition, Liu et al. [13] apply a similar method to generate Chinese questions.

Du et al. [8] attempt to apply a Bi-LSTM network to generate questions. They respectively generate questions for sentences and paragraphs and get the best performance in automatic evaluation method, and the acceptability is also higher than the rule-based method by human evaluation. Duan et al. [9] utilize neural network model based on CNN and Bidirectional Gated Recurrent Unit (Bi-GRU) [4] model to generate question templates and then transform them into questions. Furthermore they exploit the generated question to assist QA [19] and get a better answer. Zhou et al. [20] add the syntactic feature and part-of-speech feature to the model based on Bi-GRU.

3 Approach

This paper proposes a method which merges question type prediction model and neural translation model. The former one is a CNN model, which is designed to predict the possible question type of the answer phrase in the sentence. The later one is a sequence-to-sequence model based on Bi-LSTM that aims to generate target questions. The structure of the entire system is shown in Fig. 1. Two modules are kept intact within a single pipeline stage. AP is an answer phrase in a sentence and also the focus of the artificial question. We replace the answer phrase with its interrogative pronoun that fetched from the question type prediction. After processing, a new sentence is used as the input of generate question model, whose structure is shown in Fig. 2.

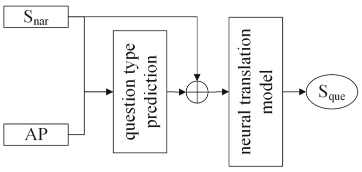


Fig. 1. The structure of the entire system

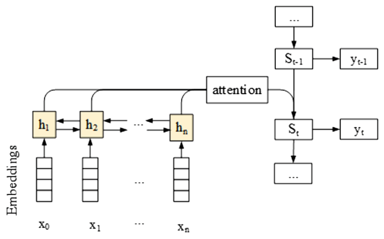


Fig. 2. Neural translation model

As shown in example 1), a "who" label can be assigned for AP 2. Then we replace AP 2 with "who" to form a new sentence $S_{nar}*$ for the S_{nar}, which is listed below. $S_{nar}*$ contains the semantics of question type and will be used for question generation.

$S_{nar}*$: ***who*** *founded the jamaat−e−islami party in 1941 and remained its leader until 1972.*

3.1 Question Type Prediction

As shown in Fig. 3, the question type prediction model is a slight variant of the CNN architecture. The input of the model contains a sentence S_{nar} and an answer phrase AP. The output is one of the following 13 labels, including

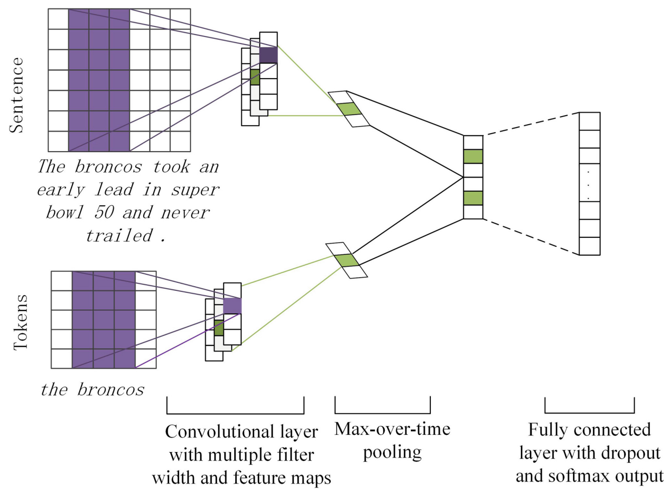

Sentence
The broncos took an early lead in super bowl 50 and never trailed .
Tokens
the broncos
Convolutional layer with multiple filter width and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Fig. 3. The question type prediction model.

"how much, how many, what, how long, which, where, how often, when, why, whose, who, how, other.". Not only the meaning of the answer phrase, but also the effect of the sentence needs to be considered. Given a sentence with n words $\mathrm{S}_{nar} = [x_1^c, x_2^c, ..., x_n^c]$ and an answer phrase with m words AP $= [t_1, t_2, ..., t_m]$, AP $\in \mathrm{S}_{nar}$. We use x_i^c and t_j to denote embedded vector of i-th word in S_{nar} and j-th word in AP.

In general, let $x_{i:i+h-1}^c$ refer to the concatenation of word embeddings x_i^c, x_{i+1}^c, ..., x_{i+h-1}^c and $t_{j:j+l-1}$ refer to the concatenation of word embeddings t_j,t_{j+1},...,t_{j+l-1}. A convolution operation involves two filters $W_1 \in \mathbb{R}^{hk}$ and $W_2 \in \mathbb{R}^{lk}$. W_1 is applied to a window of h words in a sentence and W_2 is applied to a window of l words in an answer phrase. c_i^x and c_i^t are generated by:

$$c_i^x = f(W_1 * x_{i:i+h-1}^c + b_x) \;\; and \;\; c_i^t = f(W_1 * t_{i:i+l-1} + b_t) \tag{1}$$

here $b_x \in \mathbb{R}$ and $b_t \in \mathbb{R}$ are bias term and f is non-linear function such as the hyperbolic tangent. W_1 filter is applied to produce a feature map. Similarly, the answer phrase is also manipulated by the filter W_2.

$$c_1 = [c_1^x, c_2^x, ..., c_{n-h+1}^x] \;\; and \;\; c_2 = [c_1^t, c_2^t, ..., c_{m-l+1}^t] \tag{2}$$

We then apply a max-over-time pooling operation [5] over the feature maps and take the maximum value $c = [max\ c_1; max\ c_2]$ as the feature corresponding to particular filters. The process is to capture the most feature, one with the highest value, for each feature map. The pooling scheme naturally deals with variable lengths of sentence and answer phrases. Upon the hidden layer, we stack a *softmax* layer for interrogative pronoun determination:

$$y_{label} = g(W * c + b) \tag{3}$$

where g is a *softmax* function, W is a parameter matrix and b is a bias term.

3.2 Neural Translation Model

The neural translation model is constructed based on Bi-LSTM. The encoder reads a word sequence of an input sentence $S_{nar}* = \{x_1, x_2, ..., x_s\}$, which contains the semantics of the possible question type. Let x_i refers to i-th word in a narrative sentence. The decoder predicts a word sequence of an output question $S_{que} = \{y_1, y_2, ..., y_q\}$, let y_i refer to i-th word in a question. The attention mechanism used in this model is adopted from Du et al.'s [8] work. The probability of generating a question Q in the decoder is computed as:

$$P(S_{que}) = \prod_{i=1}^{|S_{que}|} P(y_i|y_{<i}, c_i) \tag{4}$$

$$P(y_i|y_{<i}, c_i) = softmax(W_s tanh(W_i[h_i; c_i])) \tag{5}$$

the $softmax$ denotes a non-linear function that outputs the probability of generating y_i. h_i is computed as:

$$h_i = LSTM(y_{i-1}, h_{i-1}) \tag{6}$$

here, $LSTM$ [11] generates the new state h_i by the representation of previously generated word y_{i-1} (obtained from a word look-up table), and previous state h_{i-1}. c_i denotes the context vector, which is computed as:

$$c_i = \sum_{i=1,...,|x|} a_{i,t} b_i \quad and \quad a_{i,t} = \frac{exp({v_a}^T W_b b_i)}{\sum_j exp({v_a}^T W_b b_j)} \tag{7}$$

where ${v_a}^T$ and W_b are weights. b_i denotes the i^{th} hidden state of the encoder, which is the concatenation of the forward hidden state $\overrightarrow{b_i} = \overrightarrow{LSTM}(x_i, b_{i-1})$ and the back forward state $\overleftarrow{b_i} = \overleftarrow{LSTM}(x_i, b_{i+1})$.

4 Experimental Setup

Our method is experimented on the processed SQuAD dataset. In this section, we firstly describe the corpus. Then we give implementation details of our processing and the baselines to compare.

4.1 Dataset and Evaluation Methods

The SQuAD corpus is annotated by crowd-workers, we train the prediction model and the translation model through the processed data. Our data division refers to Du et al. [8]. Table 1 provides some statistics on the processed dataset.

The SQuAD corpus are used for training question type prediction model and neural translation model and testing. For the former, we can determine the interrogative labels of answer phrases.

We use simple and useful rules to construct the data set for the prediction model. In order to fetch the ground truth question type labels which the answer phrases refer to, we detect the interrogatives in the questions of the SQuAD to determine whether they contains the former 12 interrogative labels in Table 1. The matching order is from left to right. Once the question contains a label, we will commit the label to the answer phrase and terminate the matching. If not, we will set "*other*" label.

Table 1. Dataset (processed) statistic

# pairs(Train)	70484
# pairs(Dev)	10570
# pairs(Test)	11877
S_{nar}: avg.tokens	32.9
S_{que}: avg.tokens	11.3
AP: avg tokens	3.4

For the neural translation model, we utilize the ground truth question type labels for changing the raw sentence in training process (Sect. 3). While in the test process, we use the question type labels produced by the CNN classifier. The target is still an artificial question.

We adopt the micro-averaged precision (P), recall (R) and F1 score to evaluate the performance of the prediction model. The evaluation package released by Chen et al. [3] serves as the evaluation measures for question generation, which was originally used to score image captions in the generation task. The package includes BLEU-1, BLEU-2, BLEU-3, BLEU-4 [14], METEOR [7] and ROUGE$_L$ [12] evaluation scripts.

4.2 Implementation Details

We will describe the experimental parameters of the prediction model and the translation model. The output of the prediction model is a label while that of the translation model is a natural question. We use 300 dimensional word embedding pre-trained by the glove.840B.300d [15] for initialization, and fix the word representations during training. The experimental parameters of the prediction model and the translation model will be described respectively, and the parameters of two models are shown in Table 2.

Table 2. Hyperparameters used in our experiments.

Question Type Prediction Model				Neural Translation Model			
Parameters	Values	Parameters	Values	Parameters	Values	Parameters	Values
S_{nar} filter size	3	S_{nar} length	100	sentence max-length	100	dropout rate	0.3
Ap filter size	3	AP length	50	source vocabulary	40k	learning rate	0.5
dropout rate	0.5	batch size	64	target vocabulary	28k	hidden size	600
hidden size	100	-	-	batch size	64	layers	2

For the prediction model, the loss function is categorical-crossentropy [6], the optimizer is Ada [18]. For the translation model, the number of LSTM layer is 2 for both encoder and decoder. It uses SGD [2] for optimization, with an initial learning rate of 1.0. We start halving the learning rate at epoch 8, and fix the gradient as 5 when it beyonds 5. During decoding process, we do beam search with a size of 5. Finally, the decoding process stops when every beam in the stack generates the EOS token.

All hyperparameters of our models are tuned in the development set. The results are reported on the test set.

4.3 Experiment Setup

To prove the effectiveness of our method, we compare it with several competitive systems. Now we briefly introduce their approaches.

DirectIn is an intuitive yet meaningful baseline in which the longest sub-sentence is *d*irectly taken as predicted question. To split the sentence into sub-sentences, we use a set of splitters, *i.e.*, {"?", "!", ",", ".", ";"}.

H&S [10] is a rule-based overgenerate-and-rank system. When running the system, we set the parameter *just*-wh "*false*" and set max-length equal the longest sentence in training set. We take the top question in the ranked list.

NQG-LSTM [8] is a basic encoder-decoder learning system for question generation. Bi-LSTM is used for the encoder and LSTM is used for decoder. The system uses the raw question-sentence pairs.

NQG-GRU makes a slight change in NQG-LSTM model.In the model, we replace LSTM network with GRU network for question generation.

NQG++ [20] is different with NQG-GRU. The copy mechanism is added the model, and the encoder and decoder share the pre-train vectors. In addition, we only report the paper's score without model.

5 Result and Analysis

The experiment report contains the results of the question type prediction model and the neural translation model. The performance of the former has a direct impact on the later. We select the best model on the development set.

The prediction model achieves score with 67.78% P, 66.80% R and 60.58% F1 on the test set. According to performance, the labels determined by the model are used to replace the answer phrases in the sentences, and new sentences are produced as the source input for the neural translation model to generate questions. In order to verify the impact of the performance of question type prediction, we use the same question generation model based on correct labels. We name the generation system using correct label as "CL-QG". Table 3 shows the results of our method and some comparative systems.

According to the results, our performance achieves the state-of-the-art Comparing with the rule-based system H&S and DirectIn. The BLEU-4 score is increased about 2.6%, and the ROUGE$_L$ and METEOR value are respectively 17.96% and 54.74%.

Furthermore, the experimental score of NQG-GRU is the lower than NQG-LSTM, because the representation of sentence is Inadequate. We adopt Bi-LSTM in neural translation model for question generation. In these methods using neural translation model, our method performs better than the NQG++ system

Table 3. Results of generating questions. (n/a: the paper didn't list results of this task)

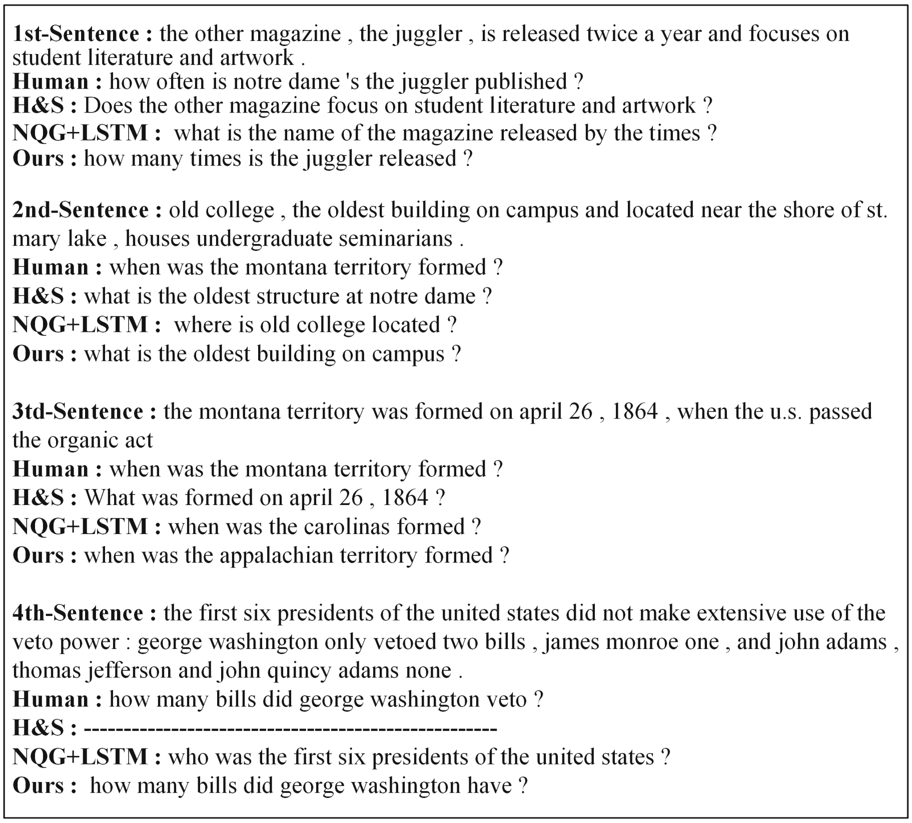
Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGE$_L$
DirectIn	0.3171	0.2118	0.1511	0.1120	0.1495	0.2247
H&S	0.3850	0.2280	0.1552	0.1118	0.1595	0.3098
NQG-GRU	0.2563	0.0990	0.0518	0.0310	0.0779	0.2846
NQG-LSTM	0.4288	0.2570	0.1728	0.1210	0.1644	0.3967
NQG++	n/a	n/a	n/a	0.1329	n/a	n/a
ours	**0.4572**	**0.2826**	**0.1934**	**0.1376**	**0.1796**	**0.4245**
CL-QG	0.4837	0.3079	0.2151	0.1556	0.1934	0.4574

with the highest performance and far exceeds the baseline NQG-LSTM system. This shows the question type prediction is helpful for question generation.

Although the recall rate of the prediction model is only about 67%, the promotion has a significant effect on the question generation performance. Comparing with "CL-QG", there is still room for growth in our method. The performance of the NQG-GRU system is lower, and NQG++ model are unknown here. So the generation results are shown in Fig. 4. For our qualitative analysis, we examine the sample outputs generated by H&S and our method. There exists a large gap between our results and H&S's. In the first two samples, the H&S only performs some syntactic transform over the input without paraphrasing, but our generated questions are "*wh*"-question and have higher reasoning. In the third sample, our model can successfully pay attention in "*april* 26, 1864". For the last sentence, the H&S system can not generate a question in that the sentence's length beyond its ability. These show that our method is better than rule-based system.

Even though NQG-LSTM utilizes the semantics of the sentence, the generated question labels are hardly similar to that of the ground truth questions. Such as the second sample, NQG-LSTM produces a question of "*where*" rather than "*what*". Our method easily detects the correct question type. Furthermore, NQG-LSTM model creates a good question by focusing on the answer phrase "*george washington*" in the fourth sentence, but it is wrong. We assign "*how many*" label to "*two bills*" through the prediction model and generate more similar question to artificial question.

Our system has the following advantages in generating questions: (1) Different from the systems based on rules, the generated questions are more reasonable in ours method. (2) Compared with some models based on neural network model, our method generated questions which are more fitting to the artificial questions. (3) our method outperforms the state-of-the-art.



1st-Sentence : the other magazine , the juggler , is released twice a year and focuses on student literature and artwork .
Human : how often is notre dame 's the juggler published ?
H&S : Does the other magazine focus on student literature and artwork ?
NQG+LSTM : what is the name of the magazine released by the times ?
Ours : how many times is the juggler released ?

2nd-Sentence : old college , the oldest building on campus and located near the shore of st. mary lake , houses undergraduate seminarians .
Human : when was the montana territory formed ?
H&S : what is the oldest structure at notre dame ?
NQG+LSTM : where is old college located ?
Ours : what is the oldest building on campus ?

3td-Sentence : the montana territory was formed on april 26 , 1864 , when the u.s. passed the organic act
Human : when was the montana territory formed ?
H&S : What was formed on april 26 , 1864 ?
NQG+LSTM : when was the carolinas formed ?
Ours : when was the appalachian territory formed ?

4th-Sentence : the first six presidents of the united states did not make extensive use of the veto power : george washington only vetoed two bills , james monroe one , and john adams , thomas jefferson and john quincy adams none .
Human : how many bills did george washington veto ?
H&S : --
NQG+LSTM : who was the first six presidents of the united states ?
Ours : how many bills did george washington have ?

Fig. 4. Sample output questions generated by human, our system, NQG-LSTM and H&S system.

6 Conclusion

In the paper, we propose a novel method for question generation which integrates the question type into the generating process. Only in this way can we acquire the representation of sentence which contains the semantic of question types. This makes question generation model preform better in that it accesses more semantic information. In the future, we will improve the performance of the question type prediction to generate better questions.

Acknowledgements. This work was supported by the national Natural Science Foundation of China via Nos. 2017YFB1002104, 61672368 and 61672367.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Lechevallier, Y., Saporta, G., (eds.) Proceedings of COMPSTAT 2010, pp. 177–186. Springer (2010). https://doi.org/10.1007/978-3-7908-2604-3_16
3. Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollár, P., Zitnick, C.L.: Microsoft coco captions: data collection and evaluation server. arXiv preprint arXiv:1504.00325 (2015)
4. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
5. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(Aug), 2493–2537 (2011)
6. De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Annal. Oper. Res. **134**(1), 19–67 (2005)
7. Denkowski, M., Lavie, A.: Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the Ninth Workshop on Statistical Machine Translation, pp. 376–380 (2014)
8. Du, X., Shao, J., Cardie, C.: Learning to ask: Neural question generation for reading comprehension. arXiv preprint arXiv:1705.00106 (2017)
9. Duan, N., Tang, D., Chen, P., Zhou, M.: Question generation for question answering. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 866–874 (2017)
10. Heilman, M., Smith, N.A.: Good question! statistical ranking for question generation. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 609–617. Association for Computational Linguistics (2010)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out (2004)
13. Liu, M., Rus, V., Liu, L.: Automatic chinese factual question generation. IEEE Trans. Learn. Technol. **10**(2), 194–204 (2017)
14. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, pp. 311–318. Association for Computational Linguistics (2002)
15. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
16. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)
17. Rus, V., Wyse, B., Piwek, P., Lintean, M., Stoyanchev, S., Moldovan, C.: The first question generation shared task evaluation challenge. In: Proceedings of the 6th International Natural Language Generation Conference, pp. 251–257. Association for Computational Linguistics (2010)
18. Schwarz, B., Kirchgässner, W., Landwehr, R.: An optimizer for ADA-design, experiences and results. In: ACM SIGPLAN Notices, vol. 23, pp. 175–184. ACM (1988)

19. Voorhees, E.M., et al.: The TREC-8 question answering track report. In: Trec, vol. 99, pp. 77–82 (1999)
20. Zhou, Q., Yang, N., Wei, F., Tan, C., Bao, H., Zhou, M.: Neural question generation from text: a preliminary study. In: Huang, X., Jiang, J., Zhao, D., Feng, Y., Hong, Y. (eds.) NLPCC 2017. LNCS (LNAI), vol. 10619, pp. 662–671. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73618-1_56

A Feature-Enriched Method for User Intent Classification by Leveraging Semantic Tag Expansion

Wenxiu Xie[1], Dongfa Gao[1](✉), Ruoyao Ding[1], and Tianyong Hao[2](✉)

[1] School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou, China
vasiliky@outlook.com, gaodf@gdufs.edu.cn, ruoyaoding@outlook.com

[2] School of Computer Science, South China Normal University, Guangzhou, China
haoty@126.com

Abstract. User intent identification and classification has become a vital topic of query understanding in human-computer dialogue applications. The identification of users' intent is especially crucial for assisting system to understand users' queries so as to classify the queries accurately to improve users' satisfaction. Since the posted queries are usually short and lack of context, conventional methods heavily relying on query n-grams or other common features are not sufficient enough. This paper proposes a compact yet effective user intention classification method named as ST-UIC based on a constructed semantic tag repository. The method proposes to use a combination of four kinds of features including characters, non-key-noun part-of-speech tags, target words, and semantic tags. The experiments are based on a widely applied dataset provided by the First Evaluation of Chinese Human-Computer Dialogue Technology. The result shows that the method achieved a F1 score of 0.945, exceeding a list of baseline methods and demonstrating its effectiveness in user intent classification.

Keywords: User intent · Classification · Target Words · Semantic Tag

1 Introduction

The research of human-computer dialogue has become a hot topic in both academia and industry in recent year [1]. A spoken dialogue system enables users to access information over the Internet using spoken languages as the medium of interaction [2]. As a vital component of spoken dialog systems, Spoken Language Understanding (SLU) aims to identify the domain and intent of users as expressed in natural language speech automatically. SLU is widely available and commonly used in a variety of application areas, such as mobile phone-based personal assistants like Siri [3, 4]. SLU typically involves three steps: domain classification, user intent determination and semantic tagging [5]. In such process, classifying user intents of input queries into a specific domain is the first and vital step of query semantic analysis in SLU [1, 3, 6, 7].

Since user intents are usually domain specific, predicting the labels of intents and domains can be treated as a single classification problem.

Detecting and classifying user intents is a growing research area not only in SLU but also in the research of search engines [8–11]. Besides, understanding the intent behind a user's query may help prune out irrelevant information and personalize answers, thus improving user satisfaction [8, 9, 12]. For both SLU and Web search engines, the most challenge of user intent detection and classification is how to understand the short, lacking of context, and noise-contained queries posted by users [5]. According to the research in [13], 93.15% of user queries for search engines are no more than 3 words and the average length is 1.85 words only. Furthermore, spoken sentences of queries in SLU usually do not follow formal language grammar and exhibit self-corrections, hesitations, repetitions and other irregular phenomena [14]. Therefore, how to detect user intent accurately and expand the semantic features of a query to assist user intent classification is an essential step.

According to the existing research [15, 16], the target words of a query can potentially represent the user intent in a SLU process. To that end, this paper presents a Semantic Tag-empowered User Intent Classification method named as ST-UIC for user intent detection and classification. In addition, ST-UIC integrates 4 kinds of features: character, non-key-noun part-of-speech (POS) tags, target words, and semantic tags, named as CKTS. A strategy is proposed to identify and extract query target words. Moreover, a sematic tag repository is automatically constructed for feature expansion purpose. Based on a publically available dataset, the F1 score on testing dataset reaches 94.5% and outperforms five baseline methods, demonstrating the effectiveness of the proposed method in user intent classification.

2 Related Work

The user intent classification task defined by Zhang et al. [1] is that, given an input message, classify the user intent into a specific domain category. With respect to that user intent is semantic constraint on the sought-after answers, how to effectively identify user intents to prune out irrelevant information that may mislead the classification process is a crucial problem. As a vital task of spoken language understanding, there has been existing representative research with respect to user intent classification (also known as domain classification) in last decades. One pioneering work was AT&T's "How May I Help You?" [17], where the users' fluently spoken utterances were classified into a number of predefined intent categories. Later, a variety of practical goal-oriented spoken dialog systems [5, 6, 18] were built and the research of SLU ranges from determining phrases via grammars, extracting predefined named entities, detecting users' intents for classification, to combinations of users' intent and named entities [18]. With the increasing use of web search, user search query logs were applied as a valuable source of unlabeled information for user intent classification [10, 19, 20]. Hakkani-Tür et al. [19] exploited search queries from search engine query logs to improve query intent classification in SLU. They assumed that clicked URL categories could be assigned as the domain labels of user queries. For example, label "hotels" was assigned to the user query "Holiday Inn and Suites" when the user clicked the URL "http://www.hotels.com". Celikyilmaz et al. [20]

utilized unlabeled queries collected from internet search engine click logs for SLU intent detection. Those ideas of utilizing query click logs for improving query intent classification were similar to the proposed semantic tags expansion in the paper. The difference was that the domain labels were sentence-level expansion of the query and the proposed method was word-level.

Besides, most of the previous research in SLU applied manually generated features, such as grammar information and predefined phrases. Gupta et al. [18] presented a SLU system which utilized a statistical classifier for intent determination and a rule-based fixed grammars for named entity extraction. Hernández et al. [21] proposed a simple model for user intent classification which leveraging only the text including in the query. The feature they extracted from the query text, including entity names, query length, transactional terms, interrogative terms and stop words, was verified as simple yet effective. In Ganti et al. [22] research, co-occurrence between the query keyword and the tags that associated with the retrieved search results were leveraged as tag ratio features for avoiding the sparse feature spaces issue in query intent classification task.

Deep learning [3, 7, 23, 24] recently achieved good performance on user intent classification task in SLU, due to their ability to learn compact and discriminative features. Experiments presented that proposed methods improved performances over baseline techniques on a large context-sensitive SLU dataset. Tur et al. [3] presented an application of deep convex networks (DCNs) for semantic utterance classification. The results showed that the proposed DCN-based method was effective on a domain classification task for spoken language understanding. Deng et al. [23] also applied deep learning techniques kernel version DCN (K-DCN) on a intent classification task of SLU, which yielded a good classification performance. Although the study of deep learning has already led to impressive theoretical results, several problems lie ahead such as requiring large training datasets, time-consuming for parameter tuning, and involving a difficult optimization [25].

3 The User Intent Classification Method

A Semantic Tag-empowered User Intent Classification (ST-UIC) method is proposed to identify user intent and expand semantic features for user intent classification. The ST-UIC method contains five major steps: preprocessing, target word extraction, non-key-noun POS expansion, semantic tag expansion, and intent classification. The framework of the method is shown as Fig. 1.

The preprocessing step includes query segmentation and transformation. All numerics in Chinese character are also converted into Arabic format. Then, a dependency relation-based strategy is proposed for target word extraction. After preprocessing, four proposed features are expanded for classification, i.e. character, non-key-noun part-of-speech (non-key-noun POS) tags, target words, and semantic tags, named as CKTS. Both Character and target word features are extracted for maintaining contextual information and representing the user intent of original queries. For the queries that none target words are identified, key noun words are extracted instead. Then, key noun words are further expanded with semantic tags to enrich the semantic information of the query. The non-key-noun POS feature is proposed as a supplementary strategy to enrich the

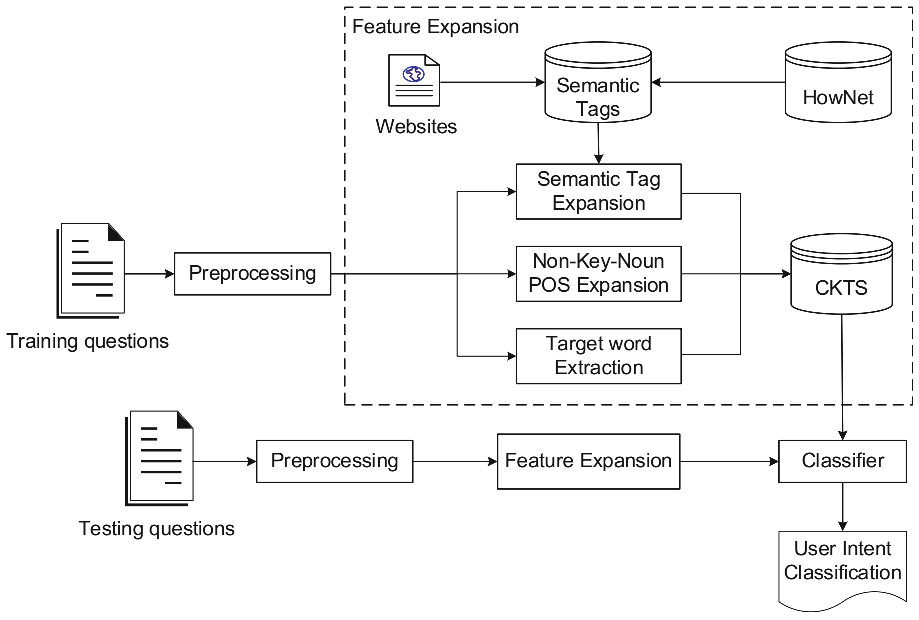


Fig. 1. The framework of the proposed ST-UIC method for user intent classification.

syntactic features of non-key-noun words. Finally, the expanded features are sent to a trained classifier to obtain user intent categories.

3.1 Target Word Extraction

Based on an existing research [15], the target words of a query substantially represent the intent of corresponding asker. For instance, the target word of the query "帮我链接到新浪网" ('*help me access to the Sina homepage*') is "新浪网" ('*the Sina homepage*'). This target word represents user intent to some extent. Through expanding the semantic tags "<网站\门户网站>"('*website*') of the target word "新浪网", the query can be directly linked to the intent category "website". As the queries come from a human-computer chit-chat and task-oriented dialogue, the queries are always short and verb-centered. For example, the same query contains a verb "链接"('*access*') and an object "新浪网", and these two words can well represent the intent of the query.

By leveraging Chinese Language Technology Platform (LTP) for dependency relation analysis, we design a strategy for extracting target words from queries. After analyzing the dependency relations of queries, we found that the target word of the verb-center query frequently contained in a "*VOB*" relation. We thus further represented the dependency relations of a query as "$word_1 \leftarrow pos$*: relation: pos*$\rightarrow word_2$". For all the dependency relations of a query, we extract the $word_2$ from the relation that is "*VOB*". Therefore, "新浪网"is extracted as target word of this query.

In addition to the typical relations, there are also some other cases that are not verb-centered, e.g., the query “颈椎病用什么药物治疗?” (‘*What is the medication treatment of cervical spondylosis?*’). Therefore, a key noun word extraction strategy is further proposed to deal with the non-verb-center queries for further feature expansion. We applied jieba, a Chinese text segmentation tool, for keyword noun words extraction. We use jieba TextRank keyword extraction function with a constraint that extract words with which POSs are included. After analyzing the queries, the POSs we set are {N(noun), NS(toponym), NT(organization/group name), NZ(other proper noun), NL (noun phrase)}. For the same example, the key noun words extracted of the query are “颈椎病”(‘*cervical spondylosis*’) and “药物”(‘*medication*’). Then, the extracted key noun words set KN_w will be expanded with semantic or syntactic features.

3.2 Semantic Tag Expansion

The input queries are always short and lack of context. We thus propose to expand the semantic information of queries. HowNet, a bilingual general knowledge base that describing relations between concepts as well as relations between concept attributes, is utilized for key noun words’ semantic expansion in this process. Each word in a HowNet taxonomy has an upper concepts in word’s definition. For instance, the upper concepts of word “糖尿病”(‘*diabetes*’) in query “糖尿病注意什么?”(‘*what does diabetes need to pay attention to*?’) is “disease|疾病”. Therefore, after upper concepts expansion, the query can directly link to the category “health”. We further reconstruct the HowNet corpus into a repository only containing words which POS is “Noun” and the corresponding upper concepts as semantic tags. Yet, HowNet is not large enough to covering all word concepts especially new words, such as “新浪网”and “颈椎病”. Therefore, a new semantic tag repository is constructed. Words and corresponding semantic tags are extracted from the websites such as Hudong Wikipedia. For instance, word “新 浪 网”and corresponding semantic tags “< 中 国 网 站 \ 互 联 网>”(‘*Internet*’) are extracted from the websites. All the words and semantic tags are structured as a tuple, i.e. (word,<semantic tags>). The newly constructed repository contains 255,824 words and their associated semantic tags. For all key noun words extracted from the process, ST-UIC match words in the repository to retrieve matched semantic tags as features.

$$f(w) = \begin{cases} semantic\,tag, & if\ w\ in\ KN_w \\ POS\,tag, & otherwise \end{cases} \tag{1}$$

Yet, not all queries are containing target words or key noun words. For example, queries of chit-chat category such as “你在干啥呢” (‘*what are you doing*’) may not contain nouns. For this circumstance, the non-key-noun POS feature is proposed as a supplementary strategy to enrich the syntactic features. For those that are not key noun words, the corresponding POS tags are expanded as features. Therefore, the feature expansion function *f*(*w*) is as Eq. (1), here *w* is the word in key noun word set KN_w extracted by previous process.

4 Evaluation and Results

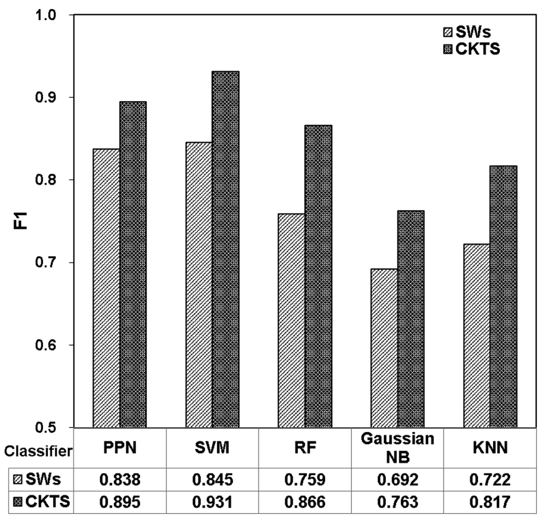
A standard dataset provided by iFLYTEK Corporation and the First Evaluation of Chinese Human-Computer Dialogue Technology (ECDT) [1] is used. All the data are mapped to a taxonomy containing 2 coarse grained categories and 31 fine grained categories. In this paper, the original train and develop data are used as **Training dataset A**, containing 3068 labeled queries. The original test data is used as **Testing dataset A**, containing 667 labeled queries.

To evaluate the stability of the proposed method, five experiments are conducted. The evaluation measures are the four widely used statistical classification measures: Accuracy, Precision, Recall and F1 score (F1).

The first experiment is to test the effectiveness of the proposed feature and to find out the optimal combination of features for the user intent classification. The proposed feature are Target Words (TWs), Semantic Tags (STs) and non-Key-noun POS tags (nK-POS). In the contrast, the commonly used feature Characters (C), Segment words (SWs) and Part-of-Speech tags (POS) are adopted as baseline features. Training on the Training dataset A with the same typical Logistic Regression (LR) classifier, the performance on the Testing dataset A are calculated and presented in Table 1. From the results, Comparing to use C or SWs alone, the F1 score of using C + SWs increased from 0.864 and 0.904 to 0.918. However, when adding SWs to the best result #11, the performances on all metrics decrease. The results of #1 and #2 indicate that the character feature is more useful and contains more semantic information than the segment word feature. As shown in the result #10, #11, #12 and #13, #14, #15, the proposed non-key-noun POS produce better performance than the commonly used POS. And comparing the result #7, #8, #9 with #13, #14, #15, adding the POS feature, the performance turn to be decreased. The proposed combination features C + nK-POS + TWs + STs (CKTS)

Table 1. The performance comparison of using different features on user intent classification.

#	Features	Accuracy	Precision	Recall	F1
1	SWs	0.855	0.932	0.826	0.864
2	C	0.888	0.934	0.887	0.904
3	C + SWs	0.910	0.945	0.900	0.918
4	C + nK-POS	0.901	0.936	0.893	0.911
5	SWs + nK-POS	0.870	0.932	0.846	0.878
6	C + SWs + nK-POS	0.904	0.939	0.895	0.913
7	SWs + TWs + STs	0.901	0.942	0.877	0.901
8	C + TWs + STs	0.907	0.946	0.904	0.920
9	C + SWs + TWs + STs	0.930	0.961	0.917	0.936
10	SWs + nK-POS + TWs + STs	0.897	0.937	0.880	0.902
11	C + nK-POS + TWs + STs	**0.936**	**0.962**	**0.932**	**0.945**
12	C + SWs + nK-POS + TWs + STs	0.931	0.958	0.921	0.937
13	SWs + POS + TWs + STs	0.892	0.931	0.865	0.890
14	C + POS + TWs + STs	0.919	0.956	0.903	0.926
15	C + SWs + POS + TWs + STs	0.918	0.955	0.895	0.919



Classifier	PPN	SVM	RF	Gaussian NB	KNN
▨ SWs	0.838	0.845	0.759	0.692	0.722
▩ CKTS	0.895	0.931	0.866	0.763	0.817

Fig. 2. The performance comparison 5 different classifiers on fine-grained classification using SWs and CKTS.

outperform the other types of feature combinations, demonstrating the effectiveness of proposed feature on user intent classification.

To better evaluate the robustness of proposed feature combination, both SWs and CKTS are applied to 5 commonly used classifiers including Support Vector Machine (SVM), Perceptron (PPN), Random Forest (RF), Gaussian Naive Bayes (GaussianNB) and *k*-Nearest Neighbor (KNN). The experiment results on fine-grained classification are shown in Fig. 2, where the proposed CKTS contributes every classifier than SWs, demonstrating the usefulness of the proposed features on user intent classification task.

In the third experiment, the stability of the proposed method is tested with different sizes of training data. The training dataset A is randomly divided into 5 training subsets containing 600, 1200, 1800, 2400, and 3000 queries respectively. The results are measured in accuracy, precision, recall and F1. As illustrated in Fig. 3, our method receives a stable performance on all evaluation metrics. Moreover, when only 600 queries are used, which are less than testing queries (667), the F1 of user intent classification still achieves 0.854, only 0.01 less than the baseline 0.864 in Table 2. This also verifies the effectiveness of proposed method.

To verify the strength and weakness of proposed method on different categories, we conduct the fourth experiment. The results, as shown in Table 2, presented that the proposed method achieves a precision of 1.000 on 20 of 31 fine-grained categories. Moreover, for the categories such as “bus”, “calc”, “contacts” etc., the proposed method gain the precision and F1 of 1.000 as well. For the categories “app”, “music”, “epg”, and “radio”, the proposed method obtain lower recall of 0.667, 0.864, 0.806, and 0.875.

The last experiment presents the comparisons of our method with existing user intent classification methods as baselines. The baselines are top five methods won the challenge in the shared task organized by the official ECDT website[1]. Using the same training and testing datasets, we compare the performance on all the fine-grained

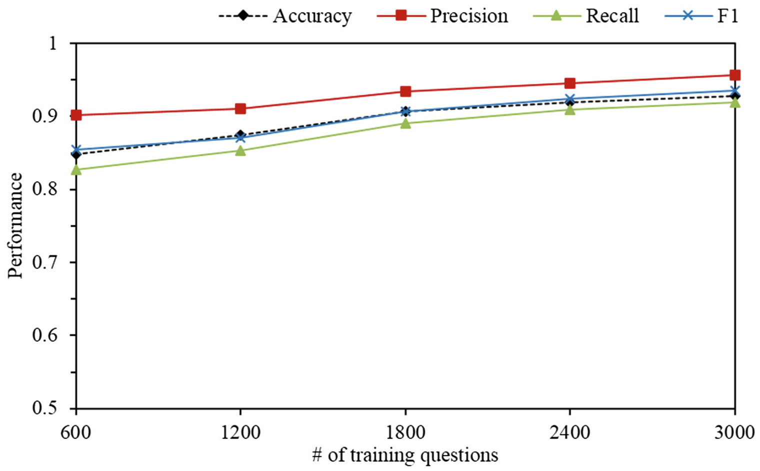


Fig. 3. The performance of our method with the increasing size of training datasets.

Table 2. The performance on fine grained categories of proposed method

Fine-grained categories	Precision	Recall	F1	Fine-grained categories	Precision	Recall	F1
App	0.857	0.667	0.750	Music	0.905	0.864	0.884
Bus	1.000	1.000	1.000	News	1.000	1.000	1.000
Calc	1.000	1.000	1.000	Novel	1.000	0.875	0.933
Chat	0.847	0.980	0.909	Poetry	1.000	0.882	0.938
Cinemas	0.778	0.875	0.824	Radio	1.000	0.875	0.933
Contacts	1.000	1.000	1.000	Riddle	1.000	1.000	1.000
Cookbook	0.978	0.989	0.983	Schedule	1.000	0.900	0.947
Datetime	1.000	0.833	0.909	Stock	1.000	0.875	0.933
Email	1.000	1.000	1.000	Telephone	0.952	0.952	0.952
Epg	0.967	0.806	0.879	Train	1.000	1.000	1.000
Flight	1.000	0.952	0.976	Translation	1.000	1.000	1.000
Health	1.000	0.944	0.971	TVchannel	0.885	0.958	0.920
Lottery	1.000	1.000	1.000	Video	0.750	0.934	0.832
Map	0.917	0.957	0.936	Weather	1.000	1.000	1.000
Match	1.000	1.000	1.000	Website	1.000	0.778	0.875
Message	1.000	1.000	1.000				

[1] http://ir.hit.edu.cn/SMP2017-ECDT-RANK.

Table 3. The performance comparison of the proposed method with 5 baselines.

Methods	Developed by	F1
LSTM + domain dictionary	Spoken Dialogue System Lab, SCAU (SIGSDS)	0.941
CNN + Ensemble Learning	DeepBrain Corporation	0.929
CNN + rules	Institute of automation, Chinese Academy of Sciences	0.926
Lib-SVM + unigram + bigram + trigram + 4-gram	School of Computer & Information Technology, Shanxi University	0.912
CNN + domain dictionary	Boyan Information Technology Company Limited	0.899
LR + CKTS	ST-UIC	**0.945**

categories in F1. From the result, as presented in Table 3, our method achieves a best F1 of 0.945. As reported on ECDT website, SIGSDS and Whisper apply a deep learning method (Long Short-Term Memory, LSTM, and Convolutional Neural Network, CNN) with manually constructed domain dictionaries. Our method obtains a comparable performance by adopting a traditional Logistic Regression with feature expansion.

5 Conclusions

Aiming for user intent identification and classification, this paper proposed a method called ST-UIC based on dependency relation analysis for target word extraction. Moreover, a semantic tag repository, containing 255,824 words and corresponding semantic tags, was automatically constructed for feature expansion. Using a publicly available dataset, five experiments were conducted for evaluating the effectiveness of ST-UIC method through a comparison with five baseline methods. The results presented that ST-UIC achieved the best performance in the comparison, demonstrating its effectiveness for user intent classification tasks.

Acknowledgements. This work was supported by National Natural Science Foundation of China (No.61772146) and Innovative School Project in Higher Education of Guangdong Province (No.YQ2015062). Guangzhou Science Technology and Innovation Commission (No. 201803010063).

References

1. Zhang, W.-N., Chen, Z., Che, W., Hu, G., Liu, T.: The First Evaluation of Chinese Human-Computer Dialogue Technology. arXiv preprint arXiv:1709.10217 (2017)
2. Zue, V., Seneff, S.: Spoken dialogue systems. Synth. Lect. Hum. Lang. Technol. **2**, 1–151 (2009)

3. Tur, G., Deng, L., Hakkani-Tür, D., He, X.: Towards deeper understanding: deep convex networks for semantic utterance classification. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5045–5048 (2012)
4. Zhang, J., Yang, T.Z., Hazen, T.J.: Large-scale word representation features for improved spoken language understanding. In: International Conference on Acoustics, Speech and Signal Processing, pp. 5306–5310 (2015)
5. Liu, J., Pasupat, P., Wang, Y., Cyphers, S., Glass, J.: Query understanding enhanced by hierarchical parsing structures. In: IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 72–77 (2013)
6. Liu, B., Lane, I.: Multi-Domain Adversarial Learning for Slot Filling in Spoken Language Understanding. arXiv preprint arXiv:1711.11310. pp. 1–6 (2017)
7. Xu, P., Sarikaya, R.: Contextual domain classification in spoken language understanding systems using recurrent neural network. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3–7 (2014)
8. Ashkan, A., Clarke, C.L.A., Agichtein, E., Guo, Q.: Classifying and characterizing query intent. In: Boughanem, M., Berrut, C., Mothe, J., Soule-Dupuy, C. (eds.) ECIR 2009. LNCS, vol. 5478, pp. 578–586. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00958-7_53
9. Hu, J., Wang, G., Lochovsky, F., Sun, J., Chen, Z.: Understanding user's query intent with Wikipedia. In: International Conference on World Wide Web, pp. 471–480. ACM (2009)
10. Park, K., Jee, H., Lee, T., Jung, S., Lim, H.: Automatic extraction of user's search intention from web search logs. Multimed. Tools Appl. **61**, 145–162 (2012)
11. Jansen, B.J., Booth, D.L., Spink, A.: Determining the user intent of web search engine queries. In: International Conference on World Wide Web, pp. 1149–1150. ACM (2007)
12. Zhang, S., Wang, B.: A survey of web search query intention classification. J. Chin. Inf. Process. **22**(4), 75–82 (2008)
13. Yu, H., Liu, Y., Zhang, M., Ru, L., Ma, S.: Research in search engine user behavior based on log analysis. J. Chin. Inf. Process. **21**, 109–114 (2007)
14. De Mori, R., Béchet, F., Hakkani-Tür, D., McTear, M., Riccardi, G., Tur, G.: Spoken language understanding: a survey. In: Automatic Speech Recognition and Understanding Workshop, pp. 365–376 (2007)
15. Hao, T., Xie, W., Wu, Q., Weng, H., Qu, Y.: Leveraging question target word features through semantic relation expansion for answer type classification. Knowl. Based Syst. **133**, 43–52 (2017)
16. Hao, T., Xie, W., Xu, F.: A wordnet expansion-based approach for question targets identification and classification. In: Sun, M., Liu, Z., Zhang, M., Liu, Y. (eds.) CCL 2015. LNCS (LNAI), vol. 9427, pp. 333–344. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25816-4_27
17. Gorin, A., Riccardi, G., Wright, J.: How may I help you? Speech Commun. **23**, 113–127 (1997)
18. Gupta, N., Tur, G., Hakkani-Tur, D., Bangalore, S., Riccardi, G., Gilbert, M.: The AT&T spoken language understanding system. IEEE Trans. Audio Speech Lang. Process. **14**(1), 213–222 (2006)
19. Hakkani-Tür, D., Heck, L., Tur, G.: Exploiting query click logs for utterance domain detection in spoken language understanding. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5636–5639 (2011)
20. Celikyilmaz, A., Hakkani-Tür, D., Tur, G.: Leveraging web query logs to learn user intent via bayesian discrete latent variable model. In: International Conference on Machine Learning (2011)

21. Hernández, I., Gupta, D., Rosso, P., Rocha, M.: A simple model for classifying web queries by user intent. In: Spanish Conference Information Retrieval, pp. 235–240 (2012)
22. Ganti, V., König, A.C., Li, X.: Precomputing search features for fast and accurate query classification. In: ACM International Conference on Web Search and Data Mining, pp. 61–70 (2010)
23. Deng, L., Tur, G., He, X., Hakkani-Tur, D.: Use of kernel deep convex networks and end-to-end learning for spoken language understanding. In: IEEE Workshop on Spoken Language Technology, pp. 210–215 (2012)
24. Shi, Y., Yao, K., Chen, H., Pan, Y.-C.Y., Hwang, M.-Y., Peng, B.: Contextual spoken language understanding using recurrent neural networks. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5271–5275 (2015)
25. Bengio, Y.: Deep learning of representations: looking forward. In: Dediu, A.-H., Martín-Vide, C., Mitkov, R., Truthe, B. (eds.) SLSP 2013. LNCS (LNAI), vol. 7978, pp. 1–37. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39593-2_1

Event Detection via Recurrent Neural Network and Argument Prediction

Wentao Wu(✉), Xiaoxu Zhu, Jiaming Tao, and Peifeng Li

School of Computer Science and Technology,
Soochow University, Suzhou, China
{wtwu, jmtao}@stu.suda.edu.cn, {xxzhu, pfli}@suda.edu.cn

Abstract. This paper tackles the task of event detection, which involves identifying and categorizing the events. Currently event detection remains a challenging task due to the difficulty at encoding the event semantics in complicate contexts. The core semantics of an event may derive from its trigger and arguments. However, most of previous studies failed to capture the argument semantics in event detection. To address this issue, this paper first provides a rule-based method to predict candidate arguments on the event types of possibilities, and then proposes a recurrent neural network model RNN-ARG with the attention mechanism for event detection to capture meaningful semantic regularities form these predicted candidate arguments. The experimental results on the ACE 2005 English corpus show that our approach achieves competitive results compared with previous work.

Keywords: Event detection · Argument prediction · Recurrent neural network

1 Introduction

Event extraction is divided into two subtasks, event detection (or trigger extraction) and argument extraction. The former focuses on identifying event triggers and categorizing their event types, while the latter aims to extract various arguments of a specific event type and assign them roles. Commonly, event triggers are single verbs or nominalizations that evoke some real-world events, while event arguments are composed of entity instances and play a certain role in an event. For example, in the sentence "**He** <u>*died*</u> in the wave of kidnappings in **Iraq**", event detection should recognize the token "died" as the trigger of the event type *died* and argument extraction should identify the entities "He" and "Iraq" as the arguments of this *died* event, and assign them the roles *Victim* and *Place*, respectively. This paper focuses on event detection because it is still the bottleneck of event extraction for its low performance.

Pipeline models are widely used in previous studies (Liao and Grishman [1]; Hong et al. [2]), where argument extraction is the subsequent stage of event detection. Therefore, argument information cannot be applied to event detection directly. However, event arguments are very effective for event detection. Take the following two sentences as examples:

S1: *Iraqis have* ***fired*** *sand missiles in this war.*
S2: MSNBC *has* ***fired*** *Phil Donahue.*

The sentences S1 and S2 have the same trigger word "fired". With the argument information, it is easy to identify S1 as an *Attack* event due to the entity "missiles" (*Weapon*) and S2 as an *End-Position* event due to the entity "Phil Donahue" (*Person*).

Unfortunately, during the stage of event detection, we do not know which entities act as the arguments of events. Most previous methods (e.g., Ji and Grishman [3]; Liao and Grishman [1]; Hong et al. [2]) approximatively used the either syntactically or physically nearest entities to the trigger as argument features. However, many arguments are far from their triggers in either syntactically or physically distance. Besides, neural network models (Nguyen and Grishman [4, 5], Chen et al. [6]; Sha et al. [7]) were applied to event detection, most of them focused on sequence and chunk information from specific contexts, ignoring the effect of argument information.

Arguments are capable of providing significant clues to event detection, how to provide accurate argument information to event detection is vital to the performance of event detection. To tackle this issue, we first propose a method to predict candidate arguments on the event types of possibilities and then apply them to a recurrent neural network to detect events. The experimental results on the ACE 2005[1] English dataset show that our model outperforms the state-of-the-art baselines.

2 Related Work

Various methods have been proposed for event detection. Early research has primarily focused on local-sentence representations, such as the lexical features (e.g., full word, POS), syntactic features (e.g., dependency features) and external knowledge features (WordNet) (Ahn [8]). Currently, global inference and joint model are widely used in event detection. Ji and Grishman [3] combined global evidence from related documents with local decisions. Gupta and Ji [9], Liao and Grishman [1] and Hong et al. [2] proposed cross-event and cross-entity inference for the event extraction task.

Representation-based approaches have been introduced into event detection very recently, which represent candidate event mentions by embeddings and fed them into neural networks (Chen et al. [6]; Nguyen and Grishman [4, 5]). Furthermore, Nguyen et al. [10] employed bidirectional RNN, which could jointly extract event trigger and arguments. However, joint model only makes remarkable improvements to argument extraction, but insignificant to event detection. Sha et al. [7] employed a dependency bridge recurrent neural network (dbRNN) for event extraction, which simultaneously applying tree structure and sequence structure in RNN to capture sequence and syntax information from specific context. Unfortunately, it totally ignored importance of arguments for event detection. Liu et al. [11] proposed a three-layer Artificial Neural Networks (ANNs) model on annotated arguments and the words around them to model

[1] https://catalog.ldc.upenn.edu/LDC2006T06.

the event detection task. It exploits argument information explicitly for event detection via supervised attention mechanisms, which construct gold attention for each trigger candidate based on annotated arguments in the training procedure.

3 Approach

We first propose a rule-based method to predict candidate arguments and then introduce RNN-ARG (Recurrent Neural Networks with Arguments) to detect events. In our model, we incorporate a Bi-LSTM (Bi-directional Long Short-Term Memory) to model the preceding and following information of a word and use another Bi-LSTM model with the attention mechanism to learn the representation of trigger and arguments.

3.1 Argument Prediction

When we cannot recognize event type from the semantics of trigger directly, it is very important to consider argument semantics. Most of previous studies adopted the syntactically or physically nearest entities to the trigger to represent argument semantics. Due to upstream errors from the syntax parsing and the diversity of sentence expression, this method always introduces many pseudo arguments to vent detection and then harms the precision. Another method is to consider all of the entities in the event sentence as arguments. Obviously, it will introduce more pseudo arguments to event detection. In this paper, we propose an event arguments prediction method to predict based on the event types of possibilities and the corresponding entity types, which can act as event roles following the definitions of event types.

In an event, the event type dominates its argument numbers and argument types, i.e., roles. Hence, we must detect the event type firstly and then select candidate arguments following the definition of the event type. Unfortunately, we do not know the event type before event detection. However, we can enumerate all possible types of a trigger word according to the annotation training data. In particular, 85.6% of the trigger words in the training set only refer to one event type, 11.7% of them belong to two distinct event types, and the rest (2.7%) has three or more event types.

Firstly, we enumerate all possible types of a candidate trigger w. If w does not appear in the training set, we first calculate the similarities between w and each annotated trigger in training set using WordNet similarity. Then we find a trigger *tri* in the training set, who has the highest similarity with w. For instance, in S3 the candidate trigger “discussed” does not appear in the training set and its most similar trigger is “talked”.

S3: *<u>Chretien</u>/PER said that <u>he</u>/PER and <u>Bush</u>/PER, <u>who</u>/PER had not spoken since late <u>February</u>/TIME,* ***discussed*** *issues including <u>Iraq</u>/GPE and aid to <u>Africa</u>/GPE.*

Secondly, we select top two high frequency event types of the candidate trigger w (if w does not occur in the training set, we use *tri* to replace it) according to the statistics on the training set. Because “discussed” does not occur in the training set, we use its similar trigger “talked” to find event types. The trigger word “talked” belongs to the event type *Phone-Write* (60%) and *Meet* (40%). Hence, we choose the top two event types *Phone-Write* and *Meet* for the trigger "discussed".

Finally, we extract the candidate arguments on entity type matching. For an event type *i*, we first extract all entity types, who can act as a role (we do not consider the role *Place* and *Time* because they do not have obvious event type discrimination) of this event type following the event definition, and store them to the list $EntType_i$. Then from the list of the annotated entities in the event mention, we extract the entities whose types belong to $EntType_i$, as candidate arguments. For example, the event type *Phone-Write* only has one role *Entity*, which can fill entities whose type are PER/ORG. Hence the list of entities [Chretien, he, Bush, who] can act as candidate arguments of *Phone-Write* event, due to their entity types PER or ORG.

3.2 RNN-ARG Model

The RNN-ARG model is showed in Fig. 1. The model contains two Bi-LSTM neural networks. Specifically, we first use a Bi-LSTM to encode semantics of each word with its preceding and following information. Then we add an attention-based Bi-LSTM neural network to capture the semantics of trigger and arguments.

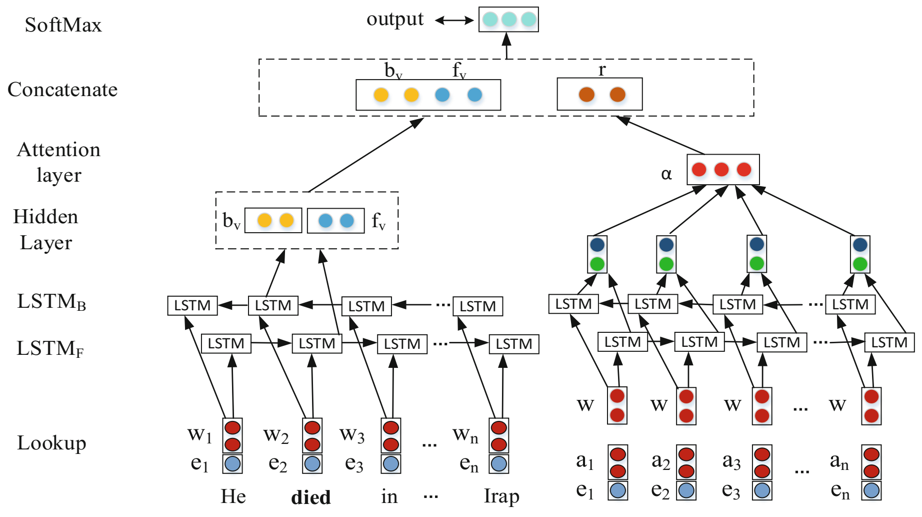


Fig. 1. The architecture of the model RNN-ARG (here the trigger candidate is "died")

- **Bi-LSTM**

In the sentence encoding phase, we take all the words of the whole sentence as the input and each token w_i into a real-valued vector x_i using the concatenation of the following two vectors:

(1) **Word Embedding Table**: Word embeddings are able to capture the meaningful semantic resularities (Bengio et al. [12]) and we train the word embeddings using Skip-Gram[2] (Mikolov et al. [13]) algorithm on the NYT corpus[3].

(2) **Entity Type Embedding Table**: Following existing work (Li et al. [14]; Chen et al. [6]; Nguyen and Grishman [4]), we exploit the annotated entity information as additional features. We randomly initialize embedding vectors for each entity type (including "None" which refers to an undefined entity type) and update it in the training procedure.

The transformation from the token w_i to the vector x_i essentially converts the input sentence W into a sequence of real-valued vectors $\mathrm{X} = (x_1, x_2, \ldots, x_n)$, to be used by recurrent neural networks to learn a more effective representation.

At each step t, the LSTM accepts current input $\boldsymbol{x}_t$ and previous hidden state $\boldsymbol{h}_{t\text{-}1}$ to compute hidden state. We run $LSTM_F$ from the beginning to the end of sequence, and run $LSTM_B$ from the end to the beginning of the sequence, while producing sequences of vectors for forward propagation $\vec{\boldsymbol{h}}_t = LSTM_F\left(\boldsymbol{x}_t, \vec{\boldsymbol{h}}_{t-1}\right)$ and another for the backward propagation $\overleftarrow{\boldsymbol{h}}_t = LSTM_B\left(\boldsymbol{x}_t, \overleftarrow{\boldsymbol{h}}_{t+1}\right)$ respectively. Afterwards we concatenate the vectors $\boldsymbol{h}_t = \left[\vec{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t\right]$ for each time step, and $\boldsymbol{h}_t$ is reduced into a single vector as the current state. The LSTM holds a state representative as a continuous vector passed to the subsequent time step, and it is capable of modeling long-range dependencies due to its gated memory. Afterwards, we concatenate of the hidden states f_v of the $LSTM_F$ and b_v of the $LSTM_B$ as the final output of Bi-LSTM, instead of averaging the last hidden vectors of the $LSTM_F$ and $LSTM_B$.

- **Bi-LSTM with Attention**

Different from sentence encoding which used the entire sentence as input of Bi-LSTM. In this encoding phase, we take the candidate trigger and its predicted arguments as the input of an event. We use w to denote the current candidate trigger, $[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n]$ to denote the candidate arguments/entities in the list, which is extracted by argument prediction method, and $[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n]$ to denote the entity types of the candidate arguments in the list. The element at position i of the input sequence is resolved by a vector $\boldsymbol{v}_i$ as follows:

$$\boldsymbol{v}_i = \boldsymbol{w} \oplus \boldsymbol{a}_i \oplus \boldsymbol{e}_i \tag{1}$$

[2] https://code.google.com/p/word2vec/.

[3] https://catalog.ldc.upenn.edu/LDC2008T19.

where $\oplus$ is the concatenation operator. Note that, both $\boldsymbol{w}$, $\boldsymbol{a}_i$ and $\boldsymbol{e}_i$ are originally in symbolic representation. Before entering the recurrent neural network, we transform them into real-valued vector by two look-up tables, Word Embedding Table and Entity Type Embedding Table. For the recurrent setup, we use a layer of LSTM networks in a bidirectional manner. The forward and backward LSTMs traverse the sequence v_i, producing sequences of vectors $\vec{\boldsymbol{h}}_t$ and $\overleftarrow{\boldsymbol{h}}_t$ respectively. Then the output at time t is $\boldsymbol{h}_t = \left[\vec{\boldsymbol{h}}_t; \overleftarrow{\boldsymbol{h}}_t\right]$. Finally, this model acquires weighted sum over $\boldsymbol{h}_t$ by using attention, and calculates the final vectors of the argument semantics. The attention mechanism lets the model decide the importance of each predict argument by weighing them when constructing the representation of the sequence. The attention layer contains the trainable vector $\boldsymbol{\omega}$ (of the same dimensionality as vectors $\boldsymbol{h}_t$) which is used to dynamically produce a weight vector $\boldsymbol{\alpha}$ over time steps t as follows.

$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{\omega}^{\mathrm{T}}\tanh(\boldsymbol{H})) \tag{2}$$

where $\boldsymbol{H}$ is a matrix consisting of vectors $\boldsymbol{h}_t$. The output layer $\boldsymbol{r}$ is the weighted sum of vectors in $\boldsymbol{H}$:

$$\boldsymbol{r} = \boldsymbol{H}\boldsymbol{\alpha}^{\mathrm{T}} \tag{3}$$

This representation vector $\boldsymbol{r}$ obtained from the attention layer is a high-level encoding of the trigger and arguments, which is used as input to the final softmax layer for the classification.

- **Final Classification**

Finally, we concatenate the sequence feature $\boldsymbol{b}_v$ and $\boldsymbol{f}_v$ which are learned from the Bi-LSTM, and argument semantic $\boldsymbol{r}$, which is the output of attention based Bi-LSTM, as a single vector $\boldsymbol{F} = [\boldsymbol{b}_v, \boldsymbol{f}_v, \boldsymbol{r}]$. To compute the confidence of each event type, the feature vector $\boldsymbol{F} \in \mathbb{R}^{4d}$, where d is the dimension of hidden state vector, is fed into a classifier. We exploit a softmax approach to identify trigger candidates and classify each trigger candidate as a specific event type as follows.

$$\boldsymbol{O} = \boldsymbol{W}_s\boldsymbol{F} + \boldsymbol{b}_s \tag{4}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{n\times(4d)}$ is the transformation matrix and $\boldsymbol{O} \in \mathbb{R}^n$ is the final output of the network, n is equal to the number of the event type including the "None" label for the candidate trigger which do not belong to any event type. For regularization, we also employ dropout (Kim [15]) on the penultimate layer.

3.3 Model Training

We define all of the parameters for the stage of trigger classification to be trained as $\theta = (\boldsymbol{E}, \boldsymbol{ET}, \boldsymbol{lf}, \boldsymbol{lb}, \boldsymbol{rf}, \boldsymbol{rb}, \omega, \boldsymbol{W}_s, \boldsymbol{b}_s)$. Specifically, $\boldsymbol{E}$ is the word embedding, $\boldsymbol{ET}$ is the embedding of the entity type, $\boldsymbol{lf}$ and $\boldsymbol{lb}$ are parameters of forward-LSTM and backward-LSTM, $\boldsymbol{rf}$ and $\boldsymbol{rb}$ are parameters of another forward-LSTM and backward-LSTM. $\boldsymbol{\omega}$ is parameters of attention layer, $\boldsymbol{W}_s$ and $\boldsymbol{b}_s$ are all of the parameters of the output layer.

Given an input example $\mathbf{x}$, the network with parameter θ outputs the vector $\mathbf{O}$, where the i-th value o_i of $\mathbf{O}$ is the confident score for classifying $\mathbf{x}$ to the i-th event type. To obtain the conditional probability $p(i|\mathbf{x}, \theta)$, we apply a softmax operation over all event types:

$$p(i|\mathbf{x}, \theta) = \frac{e^{o_i}}{\sum_{k=1}^{m} e^{o_k}} \tag{5}$$

The model can be trained in an end-to-end way by back propagation, where the objective function (loss function) is the cross-entropy loss. Given all of our (suppose T) training examples $(\mathbf{x}_i; y_i)$, we can then define the negative log-likelihood loss function as follows:

$$J(\theta) = -\sum_{i=1}^{T} \log p(y^{(i)}|\mathbf{x}^{(i)}, \theta) \tag{6}$$

To compute the network parameter θ, we train the model by stochastic gradient descent over shuffled mini-batches with Adam (Kingma and Ba [16]) rule.

4 Experiments

We first introduce the experimental setting and then report the experimental results and analysis.

4.1 Experimental Setting

We evaluate our model on the ACE 2005 English corpus and use the same data split and annotated entity mention as the previous work (Liao and Grishman [1]; Hong et al. [2]; Li et al. [14]; Nguyen and Grishman [4]). This data split includes 40 newswire documents for the test set, 30 other documents for the development set and remaining 529 documents as the training set. Besides, we report the micro-average Precision (P), Recall (R) and F1-score (F1), following the standards defined in (Ji and Grishman [3]). In our evaluation, we extract all annotated trigger words in the training set and use them to find the candidate triggers in the test set. Finally, 85.3% of trigger mentions are selected as candidates.

Hyper-parameters are tuned on the development set. We set 300, 50 dimensions for the word embeddings, the entity type embeddings, respectively. The hidden layer vector dimension of LSTM is 128. To prevent overfitting, we inherit the values for the other parameters from (Kim [15]), the dropout rate is set to 0.5, the mini-batch size is 50.

4.2 Experiments Results

We compare our model with the following baselines: (1) **CNN** (Chen et al. [6]), which exploits a dynamic multi-pooling convolutional neural network for event detection;

(2) **JointM** (Nguyen et al. [10]), which employs a bi-directional RNN to jointly extract event triggers and arguments; (3) **dbRNN** (Lei et al. [7]), which proposes a novel dependency bridge recurrent neural network (dbRNN) for event extraction; (4) **ANN-ATT** (Liu et al. [11]), which leverages additional arguments information for event detection.

Table 1 shows the results of the above five models and RNN-ARG achieves the comparably F1-score. Compared with two simple neural networks models CNN and JointM, our model RNN-ARG significantly improves the F1-score on event detection (trigger classification) by 2.5 and 2.3, respectively. This verifies that combined neural networks model is an appreciate model for the task of event extraction to capture more event semantics, and the attention layer is effective for capture more valuable information to extract an event when using recurrent neural networks.

Table 1. Comparison of event detection models on ACE.

Model	Trigger identification			Trigger classification		
	P	R	F	P	R	F
CNN	80.4	67.7	73.5	75.6	63.6	69.1
JointM	68.5	75.7	71.9	66.0	73.0	69.3
dbRNN	N/A	N/A	N/A	74.1	69.8	71.9
ANN-ATT	N/A	N/A	N/A	78.0	66.3	71.7
RNN-ARG	75.3	71.5	73.4	73.6	**69.8**	**71.6**

Compared with dbRNN, which carry syntactically related information when modeling each word by enhancing Bi-LSTM with dependency bridges. Our model only used two simple recurrent neural networks, but it still performs comparably with the enhanced model. In particularly, two models have achieved the same highest recall. This result justifies the effectiveness of the argument semantics and our RNN-ARG to detect events. With the additional predicted argument information, our RNN-ARG can learn more vital information from triggers, arguments and their combination.

Compared with ANN-ATT, which also introduces argument semantics to their model, RNN-ARG has achieved a very close F1-score on event detection, with the higher recall (+3.5%). ANN-ATT used annotated arguments to train attention mechanism and it only captured existed combination of trigger and arguments, ignoring other possible combination of trigger and arguments. However, our argument prediction can enumerate all possible arguments and provide more argument semantic information. Its disadvantage is it will introduce lots of pseudo arguments to our RNN-ARG to reduce the precision.

4.3 Analysis

To analyze the effectiveness of the argument semantics, we conduct the following models for comparison: (1) **ALLENT**, which regards all entities in the event mention as candidate arguments; (2) **ARG-1**, which predicts arguments based on only top one high frequency event type on the training set; (3) **ARG-1 w/o Att**, which refers to **ARG-1** without the attention mechanism; (4) **RNN-ARG**, our model which predicts arguments based on top two high frequency event types on the training set; (5) **ALL-TYP**, which predicts arguments based on all possible event types on the training set.

Table 2 shows the results of the above five models and RNN-ARG achieves the highest F1-score. In the training set, most trigger words only refer to 1–2 distinct event types. 85.6% of the trigger words refer to one event type, 11.7% of the trigger words belong to two distinct event types. These figures also justify that ARG-1 can achieve a relatively high F1-score.

Table 2. Experimental results of the variants of RNN-ARG on ACE.

Model	Trigger identification			Trigger classification		
	P	R	F	P	R	F
ALLENT	77.9	67.4	72.3	74.8	64.8	69.4
ARG-1(w/o Att)	78.5	67.6	72.6	76.2	65.3	70.3
ARG-1	75.2	71.0	73.1	72.8	68.7	70.7
RNN-ARG	75.3	71.5	73.4	73.6	**69.8**	**71.6**
ALLTYP	77.9	68.5	72.9	75.4	66.4	70.6

The statistics on the trigger words belonging to two distinct event types shows that 63.1% of the distribution on event type is larger than 3:7. This figure ensures that RNN-ARG outperforms ARG-1 in F1-score. Besides, the F1-score of ALLTYP is smaller than that of RNN-ARG, because it will introduce many pseudo arguments to the model.

ALLENT takes all the entities in the sentence containing the event mention as predicted arguments, and it obviously introduces many pseudo arguments to the model and then lead to ambiguity. Thus, its F1-score is lower than those of the other four models. Besides, ARG-1 outperforms ARG-1 w/o Att in F1-score and this result shows that the attention mechanism can optimize the final output vector, and mine richer semantic information from triggers and arguments.

Moreover, we also analyze the error results in our model RNN-ARG. Table 1 shows that 21.6% of pseudo instances are identified as event mentions by mistake. The main reason is that a trigger word may have more than one tense (especially support verbs, such as “sent”, “go”). Annotation ambiguity is also a problem and many event mentions are not annotated in the ACE corpus. For example, the trigger word “shot” in the sentence “she was shot herself”, which actually contains an *Attack* event and a *Die* event, only be assigned one event type for the annotation rule: each trigger mention only has one event type.

16.7% of trigger mentions in the test set belong to unknown trigger words (never appear in the training set), these mentions cannot be identified due to our candidate selection mechanism mentioned in Subsect 4.1. Otherwise, those nominal triggers

(5.8%) (such as "tours" and "missions") are hard to be recognized, because they lack enough event arguments to indicate their event type. Finally, some trigger words may belong to more than one event type, and they are easy to be identified wrongly.

5 Conclusion

Arguments are capable of providing significant clues to event detection. This paper proposes a novel RNN-ARG model with the attention mechanism and predicted arguments to detect events. The experimental results show the effectiveness of our model. Our future work will focus on extracting accurate arguments and giving effective representation to detect events.

Acknowledgments. The authors would like to thank three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant Nos. 61772354, 61773276 and 61472265, and was also supported by the Strategic Pioneer Research Projects of Defense Science and Technology under Grant No. 17-ZLXDXX-02-06-02-04.

References

1. Liao, S., Grishman, R.: Using document level cross-event inference to improve event extraction. In: ACL 2010, pp. 789–797 (2010)
2. Hong, Y., et al.: Using cross-entity inference to improve event extraction. In: ACL 2011, pp. 1127–1136 (2011)
3. Ji, H., Grishman, R.: Refining event extraction through cross-document inference. In: ACL-HLT 2008, pp. 254–262 (2008)
4. Nguyen, H.T., Grishman, R.: Event detection and domain adaptation with convolutional neural networks. In: ACL 2015, pp. 365–371 (2015)
5. Nguyen, H.T., Grishman, R.: Modeling skip-grams for event detection with convolutional neural networks. In: EMNLP 2016, pp. 886–891 (2016)
6. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks. In: ACL 2015, pp. 167–176 (2015)
7. Sha, L., Qian, F., Chang, B., Sui, Z.: Jointly extraction event trigger and arguments by dependency-bridge RNN and tensor-based argument interaction. In: AAAI 2018 (2018)
8. Ahn, D.: The stages of event extraction. In: Proceedings of the ACL 2006, pp. 1–8 (2006)
9. Gupta, P., Ji, H.: Predicting unknown time arguments based on cross event propagation. In: ACL-IJCNLP 2009, pp. 369–372 (2009)
10. Nguyen, H.T., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: ACL 2016, pp. 300–309 (2016)
11. Liu, S., Chen, Y., Liu, K., Zhao, J.: Exploiting argument Information to improve event detection via supervised attention mechanisms. In: ACL-2017, pp. 1789–1798 (2017)
12. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS 2013, UK, pp. 3111–3119 (2013)

14. Li, Q., Ji, H., Huang L.: Joint event extraction via structured prediction with global features. In: ACL 2013, pp. 73–82 (2013)
15. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP 2014, pp. 1746–1751 (2014)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

Employing Multiple Decomposable Attention Networks to Resolve Event Coreference

Jie Fang, Peifeng Li(✉), and Guodong Zhou

School of Computer Science and Technology,
Soochow University, Suzhou, China
jfang@stu.suda.edu.cn, {pfli,gdzhou}@suda.edu.cn

Abstract. Event coreference resolution is a challenging NLP task due to this task needs to understand the semantics of events. Different with most previous studies used probability-based or graph-based models, this paper introduces a novel neural network, MDAN (Multiple Decomposable Attention Networks), to resolve document-level event coreference from different views, i.e., event mention, event arguments and trigger context. Moreover, it applies a document-level global inference mechanism to further resolve the coreference chains. The experimental results on two popular datasets ACE and TAC-KBP illustrate that our model outperforms the two state-of-the-art baselines.

Keywords: Event coreference · Decomposable Attention Network
Global inference

1 Introduction

Event coreference resolution is vital for many NLP applications, such as topic detection (Allan et al. [1]), information Extraction (Li et al. [2]) and question answering (Narayanan and Harabagiu [3]). It is to determine which event mentions in texts refer to the same real-world event and then cluster them to a unique coreferential event chain. Take the following two event mentions as samples:

S1: A Cuban patrol boat with four heavily armed men **landed** on American shores.
S2: These bozos let four armed Cubans **land** on our shores.

The event mention in S1, whose event trigger is "landed", and the mention in S2 with trigger "land" refer to the same real-world Movement event, and are coreferential event mentions.

This paper focuses on document-level event coreference resolution. Document-level event coreference chains are challenging to resolve. Sometimes, coreferential event mentions in the same document can look very dissimilar ("killed/VB" and "murder/NN"), have event arguments partially or entirely omitted, or appear in distinct contexts compared to their antecedent event mentions, partially to avoid repetitions.

To capture the semantic information hiding in event trigger, event argument and the structure among the trigger and its arguments, this paper introduces a novel neural network, MDAN (Multiple Decomposable Attention Network) (Parikh et al. [4]), to resolve document-level event coreference from different views, i.e., event mention, event arguments and trigger context. This model can capture the different features from different views to better represent the event semantics. To resolve conflicts between different event mention pairs, this paper applies a document-level global inference mechanism to further resolve the coreference chains. The experimental results illustrate that our model outperforms the two state-of-the-art baselines on two popular datasets, the ACE 2005 corpus and the TAC KBP 2015 corpus. The contributions of this paper are as follows:

It constructs a novel neural network model MDAN for document-level event coreference resolution to capture different event semantics from multiple views.

It introduces event arguments to MDAN to better represent event semantics.

It applies a document-level global inference mechanism to further resolve the coreference chains.

The rest of this paper is organized as follows. Section 2 overviews the related work. Section 3 describes our MDAN model for event coreference resolution. Section 4 evaluates our approach and shows its effectiveness over two baselines. Section 5 concludes the paper with future work.

2 Related Work

Event coreference is much less studied in comparison to the large number of work on entity coreference. The studies on event coreference resolution are usually divided into within-document level and cross-document level.

Early work on document-level event coreference resolution mostly built on insights gained from the entity coreference literature (Cybulska and Vossen [5], Bejan and Harabagiu [6], Ng and Cardie [7]). Recent approaches focused on exploiting event specific structure and resolution model. Chen and Ji [8] modeled event coreference resolution as a spectral graph clustering problem that optimizes the normalized-cut criterion. Liu et al. [9] introduced a rich-features method with a large amount of features for propagating information between events and their arguments. Lu et al. [10] proposed a joint inference model based Markov logic networks to correct the mistakes from the pairwise event coreference resolver. Currently, neural networks are widely used in many NLP applications. To our knowledge, there is only one study employed neural networks for document-level event coreference. Krause et al. [11] introduced the Convolutional Neural Network (CNN) to event coreference. It is divided into two parts. The first part gives a representation for a single event mention and the second part is fed with two such event mention representations plus a number of pairwise features for the input event-mention pair, and calculates a coreference score.

3 MDAN for Event Coreference Resolution

The architecture of the model MDAN for event coreference resolution is shown in Fig. 1 and our model MDAN contains four parts, i.e., multi-similarity module, pairwise module, classifier module and global inference module.

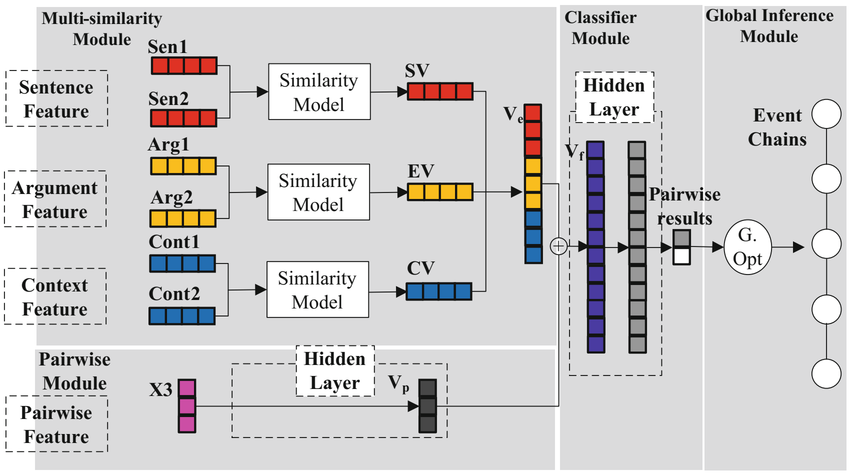


Fig. 1. The architecture of the MDAN model

Two event mentions are coreferential when they are similar in tokens, event structures, triggers, arguments, context of trigger, etc. The multi-similarity module first compute the similarity vectors of two event mentions from multiple views (i.e., event mention view, argument view and trigger context view), and then concatenate them into a vector to represent their final similarity. The advantage of multi-similarity module is that it can capture different semantic information from different views to represent different similarities of an event pair. The details of a single similarity module are shown in Fig. 2. Inspired by Parikh et al. [4], we introduce Decomposable Attention Network (DAN) as our similarity module. DAN outperforms several similarity models in our experiments, such as Siamese CNN Network, etc. Its advantage is that the soft attention in DAN can capture important hiding features and avoid noise. This similarity module containing the soft attention is suitable to learn the similarity of two event mentions in our experiments.

The pairwise module fed with pairwise features between two event mentions and maps them to a vector. Pairwise features reveal the similarities on various kind attributes (e.g., trigger and event type) of event mentions. These attributes can be regarded as the auxiliary of the multi-similarity module. Both the multi-similarity module and the pairwise module are the kernel components of MDAN, and we use them to capture the hidden features inside event mention.

The classifier module is to classify an event mention pair to coreference or not. The input of this module is the combination of the event similarity vector from the multi-similarity module and the pairwise vector from the pairwise module.

The global inference module is to optimize the results from the classifier module to form a more complete event chain, based on the merging and cutting rules.

In our model MDAN, the inputs of the multi-similarity module and the pairwise module are the extracted features from an event mention pair, while the output of the classifier module is the confidence score that two event mentions are coreferential. The

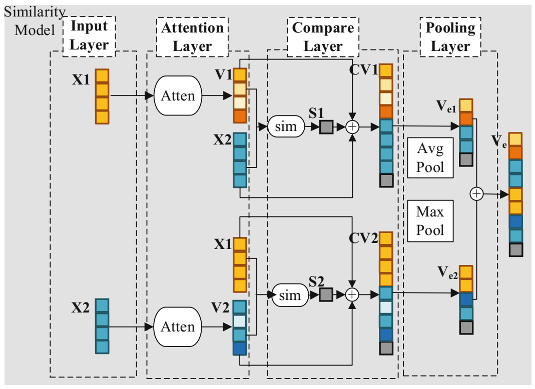


Fig. 2. The architecture of the DAN-based similarity model

input of the global inference module are all the confidence scores of two event mentions in a document and its output is the optimized results of event chains.

3.1 Input

Following Krause et al. [11], the input of the model MDAN is two event mentions e1 and e2 with annotated trigger, event type/subtype, event arguments, and event attributes (e.g., modality), etc. We extract the features from these two event mentions and the features used in Krause et al. [11] are employed in our model as follows.

Event features: (Multi-similarity module)

- Sentential features: words in sentences (event mentions) (F1); relative positions of words based on triggers (e.g., that of the word "shores" in S1 is 3) (F2)
- Context features: context around trigger (the windows size is set to 5, e.g., "with four heavily armed men landed on American shores" in S1) (F3).

Pairwise features: (Pairwise module)

- Event type and subtype is the same or not (F4)
- Distance between event mentions (numeric values) (F5)
- Event modality is the same or not (F6)
- Overlap in arguments or not (F7).

To further capture the semantic information in sentence structures and arguments (Haghighi and Klein [12]), we provide the additional features as follows:

Event features: (Multi-similarity module)

- Sentential features: POS of words in sentences (event mentions) tagged by NLTK tools (F8)
- Argument features: arguments in sentences (F9) and their entity types (F10).

For each event mention, we first embed the features (F1–F3 and F8–F10) in the sets of the sentential features, context features and argument features to six vectors, where the features F1, F3 and F9 are embedded by word2vec and the others are embedded randomly. Then we concatenate the vectors from F1, F2 and F8 into the vector **Sen**, the vectors from F9 and F10 into the vector **Arg**. Besides, the vector **Cont** is the vector from F3. In Fig. 1, **Sen1**, **Cont1** and **Arg1** are the vectors of the sentential features, context features and argument features for event mention e1, while **Sen2**, **Cont2** and **Arg2** are the vectors for event mention e_2.

3.2 Multi-similarity Module

Multi-similarity module contains three DAN-based similarity models showed in Fig. 2. Each DAN-based similarity model compute the similarity between two event mentions e_1 and e_2 on three different views, i.e., sentence, context and argument views, by using pairwise vector ***Sen1-Sen2***, ***Cont1-Con2*** and ***Arg1-Arg2*** as input, respectively.

In each similarity model, we first employ the attention mechanism to calculate the weights of input vectors ***X1*** and ***X2*** (i.e., ***Sen1-Sen2***, or ***Cont1-Con2***, or ***Arg1-Arg2***), for extracting important information from two given event mentions. Our soft alignment layer computes the attention weights w_{ij} as the similarity of words in the tuple <***X1, X2***> as Eq. (1) where function $\boldsymbol{F}$ is a feed-forward neural network, and $\boldsymbol{X1}_i$ is the vector of ith word in the vector ***X1***:

$$w_{ij} = F(\boldsymbol{X1}_i)^T \cdot F\left(\boldsymbol{X2}_j\right) \tag{1}$$

Then we use softmax to compute the weight of vectors. Vectors ***X1*** and ***X2*** are normalized as ***V1*** and ***V2*** as follows, where ℓ_{X1} and ℓ_{X2} is the length of ***X1*** and ***X2***, $\boldsymbol{V1}_i$ is ith word of ***X*** after adding attention weight values:

$$\begin{aligned}\mathbf{V1}_\mathrm{i} &= \sum\nolimits_{\mathrm{j}=1}^{\ell_{\mathrm{X1}}} \frac{\exp\left(\mathrm{w_{ij}}\right)}{\sum\nolimits_{\mathrm{k}=1}^{\ell_{\mathrm{X1}}} \exp(\mathrm{w_{ik}})} \mathbf{X1}_\mathrm{j}\ \forall \mathrm{i} \in [1, \ldots, \ell_{\mathrm{X1}}] \\ \mathbf{V2}_\mathrm{i} &= \sum\nolimits_{\mathrm{j}=1}^{\ell_{\mathrm{X2}}} \frac{\exp\left(\mathrm{w_{ij}}\right)}{\sum\nolimits_{\mathrm{k}=1}^{\ell_{\mathrm{X2}}} \exp(\mathrm{w_{ik}})} \mathbf{X2}_\mathrm{j}\ \forall \mathrm{i} \in [1, \ldots, \ell_{\mathrm{X2}}]\end{aligned} \tag{2}$$

Equation (3), which can be viewed as a noisy channel, is to compute the cosine distance of two vectors, so we can get the similarity scores of two vectors S1 and S2 as follows:

$$\mathbf{S1} = \mathrm{sim}(\mathbf{X2}, \mathbf{V1}) = \mathbf{X2}^\mathrm{T} \cdot \mathbf{V1},\ \mathbf{S2} = \mathrm{sim}(\mathbf{X1}, \mathbf{V2}) = \mathbf{X1}^\mathrm{T} \cdot \mathbf{V2} \tag{3}$$

Then we concatenate *X2*, *V1* and *S1* into a comparison vector *CV1*, and the same for comparison vector *CV2*:

$$\boldsymbol{CV1} = [\boldsymbol{X2}, \boldsymbol{V1}, \boldsymbol{S1}],\ \boldsymbol{CV2} = [\boldsymbol{X1}, \boldsymbol{V2}, \boldsymbol{S2}] \tag{4}$$

Next, we use pooling to reduce the complexity of representation. Maximum Pooling (MaxPool) and Averaging Pooling (AvgPool) are two main approaches of pooling. However, AvgPool may weaken strong activation values, and MaxPool may lead to overfitting. So we compute both average pooling (i.e., $PV1_{avg}$ and $PV2_{avg}$) and max pooling (i.e., $PV1_{max}$ and $PV2_{max}$), and concatenate all of them to produce the vector V_e:

$$\boldsymbol{PV1_{avg}} = \sum_{i=1}^{\ell_{CV1}} \frac{\boldsymbol{CV1_i}}{\ell_{CV1}}, \quad \boldsymbol{PV1_{max}} = \boldsymbol{max}_{i=1}^{\ell_{CV1}} \boldsymbol{CV1_i} \tag{5}$$

$$\boldsymbol{PV2_{avg}} = \sum_{j=1}^{\ell_{CV2}} \frac{\boldsymbol{CV2_j}}{\ell_{CV2}}, \quad \boldsymbol{PV2_{max}} = \boldsymbol{max}_{j=1}^{\ell_{CV2}} \boldsymbol{CV2_j} \tag{6}$$

$$\mathbf{V}_e = \left[\boldsymbol{PV}1_{avg}, \boldsymbol{PV}1_{max}, \boldsymbol{PV}2_{avg}, \boldsymbol{PV}2_{max}\right] \tag{7}$$

3.3 Pairwise Module

The pairwise module is to judge the similarity between two event mentions on the pairwise features. This model is very simple. We first transfer the numeric values of to the vector $X3$, and then feed them into a feed-forward network to get the vector V_p for extracting features hiding in pairwise features.

3.4 Classifier Module

We first concatenate V_e from the multi-similarity module and V_p from the pairwise model into the vector V_f to represent the final semantic relation between two event mentions.

$$\boldsymbol{V}_f = \left[\boldsymbol{V}_e, \boldsymbol{V}_p\right] \tag{8}$$

The final vector V_f is fed into a final multilayer perceptron (MLP) classifier, which has three hidden layers with RELU activation, as following:

$$\boldsymbol{V}_{\mathrm{h}} = \alpha\left(\boldsymbol{W}_h * \boldsymbol{V}_f + \boldsymbol{b}\right) \tag{9}$$

where α is the activation function, W_h and b are parameters. Finally, we can get the coreference score with the output of sigmoid layer:

$$Score = sigmoid(\boldsymbol{W}_{out} * \boldsymbol{V}_h + \boldsymbol{b}_{out}) \tag{10}$$

The objective function of model is set as following:

$$\mathcal{T}(\theta) = -\log\prod_{i=1}^{N} p\left(y^{(i)}/x^{(i)}, \theta^{(i)}\right) + \frac{\lambda}{2}\|\theta\|^2 \tag{11}$$

where $\theta = \{\boldsymbol{W}_{X1}, \boldsymbol{W}_{X2}, \boldsymbol{W}_{V1}, \boldsymbol{W}_{V2}, \boldsymbol{W}_{CV1}, \boldsymbol{W}_{CV2}, \boldsymbol{W}_e, \boldsymbol{W}_p, \boldsymbol{W}_f, \boldsymbol{W}_h, \boldsymbol{W}_{out}, \boldsymbol{b}_{out}\}$. To prevent overfitting, we utilize dropout and batch normalization.

3.5 Global Inference Module

To ensure consistent outputs on a document, we propose a document-level global inference module to solve the conflicting decisions in MDAN. We transfer the pairwise results to form coreference chains. The coreferential events are characterized as reflexive, symmetric and transitive. We utilize the transitive property in coreferential events following the merging and cutting rules.

Merging: if both event mention pairs (e_i, e_k) and (e_k, e_j) are coreferential, coreferential relation must hold between e_i and e_j, with event mention e_k as a bridge to link e_i and e_j.

Cutting: if event mention pairs (e_i, e_k) are coreferential and (e_k, e_j) are not coreferential, the pair (e_k, e_j) are not coreferential. If this constraint is in conflict with the merging rule, the merging rule is prior to this cutting rule.

To avoid the conflicts of the above four rules, we count the numbers of coreference and not coreference judgements, respectively, and make final decisions with Eq. (12) as follows.

$$\mathrm{argmax}_x\left(x * \mathrm{CR}\left(e_i, e_j\right) + (1 - \mathrm{x}) * \mathrm{CUR}\left(e_i, e_j\right)\right) \tag{12}$$

where x is a binary indicator. If x equals to 1, the event mention pair (e_i, e_j) is coreferential; otherwise, they are not coreferential. CR(e_i, e_j) and CUR(e_i, e_j) are the count of the results Coref(e_i, e_j) and Uncoref(e_i, e_j) infer by above four rules.

4 Experiments

In this section, we first introduce the experimental setting and then evaluate our model MDAN on two corpora to justify its effectiveness and report the experimental results. Finally, we give the analysis on the experimental results.

4.1 Experimental Setting

In our experiments, we mainly evaluate our MDAN model on the ACE 2005 English corpus, following most previous studies on document-level event coreference resolution. This corpus contains 599 documents in six genres. This corpus annotated events with 8 event types and 33 event subtypes. In our evaluation, we use the same training and test set as Krause et al. [11][1]. Every event mention is paired with every event mention in the text. Besides, we also report the results of our MDAN on another widely used corpus, the TAC KBP 2015 English corpus which is annotated with event nuggets that fall into 38 types and coreference relations between event mentions. Table 1 shows the statistics on the above two corpora.

In the evaluation, we set the dimensions of the POS, entity type, and relative position embeddings as 50 and $\lambda = 10^{-4}$, which parameters of embedding matrix are randomly initialized. We initialize word embeddings via pre-trained embeddings of

[1] https://git.io/vwEEP.

Table 1. Statistics on the ACE and TAC KBP corpora.

Corpus	#Documents	#Sentences	#Event mentions	# Event chains
ACE 2005	599	15494	5268	4046
TACKBP 2015	360	15824	12976	7415

GloVe and set the dimensions as $d_0 = 50$. Besides, we employ mini-batch SGD algorithm to optimize our models and the model training is run for 15 epochs, after which the best model on the valid dataset is selected.

We compare all systems using four standard F1 metrics in previous work: a link-level metric MUC, a mention-level metric B3, an entity-level metric CEAFe and an average pairwise-positive and pairwise-negative F1-score metric BLANC. (Vilain et al. [13]) We also use the average scores (AVG) of the above metrics as comparison metric.

4.2 Experimental Results

To evaluate the performance of our MDAN model on document-level event coreference resolution, we compare it with two strong baselines: a state-of-the-art classifier model (Liu et al. [9]) with more than 100 features and a state-of-the-art neural network model (Krause et al. [11]). Table 2 illustrates the performance comparison on three models based on annotated event mentions.

Table 2. Performance of the model MDAN & competitors on ACE corpus.

System	BLANC			B^3			MUC			AVG
	P	R	F1	P	R	F1	P	R	F1	F1
Liu	70.01	70.88	70.43	88.86	89.90	89.38	48.75	53.42	50.98	70.26
Krause	71.80	75.16	73.31	86.12	90.52	88.26	45.16	61.54	52.09	71.22
MDAN	**80.87**	**86.28**	**83.29**	**89.34**	**90.71**	**90.02**	**65.69**	**65.21**	**65.45**	**79.58**

The results in Table 2 show that our model MDAN outperforms two baselines on all three metrics and their averages, with an average gain of 9.32 and 8.36 in F1-scores, respectively. Compared to baseline Krause, our MDAN improves the F1-scores on three metrics BLANC, B^3 and MUC by 9.98, 1.76 and 13.36, respectively. These results confirm that our decomposable attention network and the global inference mechanism are better than their CNN model and rules on transitive closure.

We also evaluate our MDAN model on another popular event coreference corpus, the TAC KBP 2015 English corpus. Table 3 shows the F1-scores of our MDAN and the top system (TAC-TOP) [12] in the 2015 TAC KBP Event Nugget Evaluation Task. Due to this corpus did not annotate argument tags, we use PractNLPTools[2] to extract event arguments automatically.

[2] https://github.com/biplab-iitb/practNLPTools.

Table 3. F1-scores of MDAN and top system (TAC-TOP) on the 2015 TAC-KBP event nugget evaluation task. (Lu and Ng [15])

System	BLANC	B^3	MUC	$CEAF_e$	AVG
TAC-TOP	76.91	82.29	68.08	74.12	75.35
MDAN	**76.97**	**82.36**	**69.88**	**76.65**	**76.46**

Table 3 shows that our model MDAN outperforms TAC-TOP in all metrics and this result further ensures the effectiveness of our MDAN. Compared with the work of TAC-TOP (Mitamura et al. [14]), which used additional semantic resources and additional annotated datasets, we did not use any external resources.

4.3 Analysis

Compared to the baselines, our improvements mainly derive from three aspects: (1) argument information, (2) MDAN model and its attention mechanism, and (3) global inference mechanism. Table 4 shows the performance when we remove the argument information, or the attention mechanism, or the global inference mechanism from MDAN, respectively.

Table 4. Performance of our MDAN without argument information (Arg)/attention mechanism (Att)/global optimization method (Opt).

System	BLANC			B^3			MUC			AVG
	P	R	F1	P	R	F1	P	R	F1	F1
MDAN	**80.87**	**86.28**	**83.29**	**89.34**	**90.71**	**90.02**	**65.69**	**65.21**	**65.45**	79.58
w/o Arg	73.66	77.44	75.36	88.2	88.98	88.59	61.31	60.86	61.09	75.01
w/o Att	64.61	80.51	67.91	73.12	**92.76**	81.78	45.37	**74.63**	56.43	68.7
w/o Opt	68.76	79.5	72.37	77.81	91.61	84.15	49.5	71.73	58.57	71.69

If we remove the argument information from MDAN, Table 4 shows that the F1-scores on the metrics BLANC, B3 and MUC are reduced −7.93, −1.43 and −4.36, respectively. These results prove that argument information is helpful to identify event mentions and their coreference, because event semantics not only derives from trigger semantics, but entity semantics.

Table 4 also shows that the attention mechanism is helpful to prevent the interference from the uncorrelated features and improves the F1-scores on the metrics BLANC, B^3, and MUC significantly. The principle of our attention mechanism is to weight different input information. Compared with MDAN w/o Opt, We found our MDAN's recall decreased, because the vectors with low attention weight values will be ignored.

We also use the Krause's rules to replace our global inference mechanism, and the results are shown in Table 4 (MDAN w/o Opt). The results show that our global inference mechanism outperforms Krause's rules on all metrics. The reason is that our global inference mechanism applies both merging and cutting rules to optimize the results of the neural network model and then can balance the output results.

5 Conclusion

This paper introduces a novel neural network MDAN to resolve document-level event coreference from different views. Moreover, it applies a document-level global inference mechanism to further resolve the coreference chains. The experimental results illustrate that our model outperforms the two state-of-the-art baselines on two popular datasets ACE and TAC-KBP. Our future work is to expand our model to cross-document and multi-language event coreference resolution.

Acknowledgments. The authors would like to thank three anonymous reviewers for their comments on this paper. This research was supported by the National Natural Science Foundation of China under Grant Nos. 61772354, 61773276 and 61472265, and was also supported by the Strategic Pioneer Research Projects of Defense Science and Technology under Grant No. 17-ZLXDXX-02-06-02-04.

References

1. Allan, J., Carbonell, J.G., Doddington, G., Yamron, J., Yang, Y.: Topic detection and tracking pilot study: final report. In: Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, pp. 194–218 (1998)
2. Li, P., Zhu, Q., Zhou, G.: Argument inference from relevant event mentions in chinese argument extraction. In: Proceedings of ACL 2013, pp. 1477–1487 (2013)
3. Narayanan, S., Harabagiu, S.: Question answering based on semantic structures. In: Proceedings of COLING 2004, pp. 693–702 (2016)
4. Parikh, A.P., Tackstrom, O., Uszkoreit, J.: A decomposable attention model for natural language inference. In: Proceedings of EMNLP 2016, pp. 2249–2255 (2016)
5. Cybulska, A., Vossen, P.: Using a sledgehammer to crack a nut? lexical diversity and event coreference resolution. In: Proceedings of LREC 2014, pp. 4545–4552 (2014)
6. Bejan, C.A., Harabagiu, S.: Unsupervised event coreference resolution. Comput. Linguist. **40**(2), 311–347 (2014)
7. Ng, V., Cardie, C.: Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In: Proceedings of COLING 2002, pp. 1–7 (2002)
8. Chen, Z., Ji, H.: Graph-based event coreference resolution. In: Proceedings of TextGraphs 4, pp. 54–57 (2009)
9. Liu, Z., Araki, J., Hovy, E., Mitamura, T.: Supervised within-document event coreference using information propagation. In: LREC 2014, pp. 4539–4544 (2014)
10. Lu, J., Ng, V.: Joint learning for event coreference resolution. In: Proceedings of ACL 2017, pp. 90–101 (2017)
11. Krause, S., Xu, F., Uszkoreit, H., Weissenborn, D.: Event linking with sentential features from convolutional neural networks. In: Proceedings of CoNLL 2016, pp. 239–249 (2016)
12. Haghighi, A., Klein, D.: Simple coreference resolution with rich syntactic and semantic features. In: Proceedings of EMNLP 2009, pp. 1152–1161 (2009)

13. Vilain, M., Burger, J., Aberdeen, J., Connolly, D., Hirschman, L.: A model-theoretic coreference scoring scheme. In: Proceedings of MUC-6, pp. 45–52 (1995)
14. Mitamura, T., Liu, Z., Hovy, E.: Overview of TAC-KBP 2015 event nugget track. In: Proceedings of TAC 2015 (2015)
15. Lu, J., Ng, V.: UTD's event nugget detection and coreference system at KBP 2016. In: Proceedings of TAC 2016 (2016)

Cross-Scenario Inference Based Event-Event Relation Detection

Yu Hong, Jingli Zhang, Rui Song, and Jianmin Yao(✉)

Computer Information Processing Technology of Soochow University, Suzhou 215006, Jiangsu, China
tianxianer@gmail.com, jlzhang05@gmail.com, cnsr27@gmail.com, jyao@suda.edu.cn

Abstract. Event-Event Relation Detection (RD_{2e}) aims to detect the relations between a pair of news events, such as `Causal` relation between `Criminal` and `Penal` events. In general, RD_{2e} is a challenging task due to the lack of explicit linguistic feature signaling the relations. We propose a cross-scenario inference method for RD_{2e}. By utilizing conceptualized scenario expression and graph-based semantic distance perception, we retrieve semantically similar historical events from Gigaword. Based on explicit relations of historical events, we infer implicit relations of target events by means of transfer learning. Experiments on 10 relation types show that our method outperforms the supervised models.

Keywords: Relation detection · Cross scenario · Semantic distance

1 Introduction

Event relation refers to the way in which an event exerts an influence on the other, such as `Conditionality`, `Causality`, `Concession`, etc. For example, the event "*Snowden was trained as a secret agent*" is the necessary `condition` of the event "*he successfully escaped scrutiny*".

The goal of a RD_{2e} system is to automatically detect the implicit relations between event mentions, some of which occur in a single document while others different documents (cross-document relations). In this paper, we limit our discussion to the ground-truth event mentions that have been manually extracted from news articles. As defined in ACE and KBP (Ji and Grishman, 2008; Liao and Grishman, 2010), an event mention is a text span that contains a trigger and the closely related arguments.

In this paper, we propose a cross-scenario inference approach, which performs with minimal supervision. The fundamental behind the inference approach is that if there are some historical events similar to the target events, and the relations between the historical events are explicit, thus the implicit relations of the target can be inferred accordingly using those explicit relations. For example, we can determine the unknown relation in (1) as conditionality, because such a relation is held between the similar historical events in (2) and has been explicitly signalled by the conjunction "*thus*".

(1) **Target:** *Edward Snowden was trained as a secret agent. The certification would have given him some of the skills he needed to escape scrutiny.* (in 2013)
T_1 : Snowden was trained as a secret agent. T_2 : Snowden escaped scrutiny.
S_{T1} : Person AND Education-teaching AND People-by-vocation
S_{T2} : Person AND Avoiding AND Scrutiny

(2) **Historical:** *Edward Howard, a CIA case officer, was trained as a spy and* ***thus*** *eluded FBI surveillance.* (in 1993)
H_1 : Howard was trained as a spy. H_2 : Howard eluded surveillance.
S_{H1} : Person AND Education-teaching AND People-by-vocation
S_{H2} : Person AND Avoiding AND Scrutiny

Methodologically, we leverage FrameNet, a frame-level semantic dictionary, to the description of event scenarios, transforming the words in an event mention to the semantic frame tags by looking-up. This allows the similarity computing at the level of semantics, and makes it easy to match semantically similar event scenarios (see the frame tags in $S_{T1\&T2}$ and $S_{H1\&H2}$). In addition, we utilize the labelled conjunctions (e.g., "*thus*") in the corpus of Penn Discourse Tree Bank (PDTB), so as to reinforce the explicit-to-implicit relation inference.

The rest of the paper is organized as below: Sect. 2 overview the related work; Sect. 3 presents the inference approach; Sect. 4 gives a refined scenario model; Sect. 5 provides the test results; we conclude the paper in Sect. 6.

2 Related Work

Girju et al. [8] use lexico-syntactic patterns of noun phrases (NPs) and verbs, <`cause`-NP, `causal` verb, `effect`-NP>, to automatically extract the `Causal` relation within one sentence. Soon thereafter, Chang and Choi [5] revise Girju et al. (2002)'s pattern, using lexical pair (LP) and cue phrase (e.g., *due to*) to generate the new version, <LP, cue, LP>. The LP and cue-phrase probabilities are estimated in raw corpus by EM procedure, and jointly used in a naive Bayes classifier. Abe et al. [1] go further by using co-occurrence probability.

Sufficient attentions have been given to the optimization of `causal` relation recognition from different aspects: fine-grained `causal` relation classification [10], feature selection [3], syntactic, graphical and sophisticated patterns [11], use of discourse structure [6], rule generation [21], and domain adaptation [22].

The pilot study on event-oriented temporal relation derives from Mani et al. [14] and Lapata and Lascarides [12], both of whom focus on machine learning of temporal relations. In the past decade, the SemEval [20] has promoted a great deal of experimental study, including on the grammatical, syntactic, semantic and ordering feature-based temporal relation classification [4], and the validate of sequence labeling and Markov Logic [25]. SemEval-2015 [16] defines cross-sentence and cross-document event ordering tasks, both of which go a step further than the previous challenges. Most recently, deep neural networks have shown promising results. Santo et al. [24] proposed the Ranking CNN(CR-CNN) model with a class embedding matrix. Miwa and Bansal [17] similarly observed that LSTM-based RNNs are outperformed by models using CNN due to limited linguistic structure captured in the network architecture.

3 Cross-Scenario Inference

3.1 Scenario Expression and Conceptualization

We model the scenario as a vector of scene elements. Given an event mention, we use the content words (nouns, verbs, adjectives and adverbs) in the mention as the representation of the scene elements. They are able to cover most types of scene elements, such as objects (e.g., participants and attributes), activities (event triggers) and status (time, locations, surroundings and other conditions): **Mention-** *Edward fled in 1983.* **Scenario-** <*Edward*, *fled*, *1983*>; **Concept-** <`PER`, `Fleeing`, `19830000`>.

We conceptualize a scenario by transforming scene elements (content words) into their semantic frames. We brief the definition of frame semantics in the next section. In addition, we conceptualize named entities with the labels of entity types, including `Person`, `Location` and `Organization`. We generalize time expressions with the pattern `yyyy\mm\dd`. The granularity is set as `day`. See a conceptualized scenario in the above example.

3.2 Frame Semantics (FrameNet)

Semantic frame is defined as the concept of lexical units (words or phrases) that share similar semantic context [7]. It helps to identify the homogeneous lexical units. It is noteworthy that such units aren't definitely synonymous. For example, both the word *boat* and *plane* comply with the frame `Vehicle` but are different in sense.

Scenario conceptualization by semantic frame facilitates the discovery of semantically similar event mentions. See the following examples, which adhere to the same scenario at the level of semantics: **Mention1-** *He took a plane and ran away.* **Mention2-** *He fled on a boat.* **Concept-** <`PER`, `Fleeing`, `Vehicle`>.

We obtain the frames of words from FrameNet. FrameNet is a machine-readable lexical database of English [23]. It alphabetically indexes more than 10,000 frames. Each frame tag (`Nam`) corresponds to a cluster of lexical units (`Lus`) that evoke the frame, along with the related frames (also named as frame-to-frame relation, abbr., `Ffr`). Listed in Table 1 are the components of the frame `Avoiding`. The key behavioral elements in (1) and (2), i.e., *escape* and *elude*, both are the `Lus` of `Avoiding`.

3.3 Explicit-To-Implicit (ETI) Relation Assignment

Let the scenarios of a pair of target events be $< e_{t1}, e_{t2} >$. We assume that the relation of the events is determined by their scenarios, i.e., $R^t_{imp} \Leftarrow < e_{t1}, e_{t2} >$,

Table 1. Semantic frame (*e.g., scrutiny).*

Nam	Avoiding
Def	An agent avoids an undesirable situation under certain circumstances
Ffr	Inherits from Intentionally_act; Is inherited by Dodging and Evading
Lus	*avoid* (v), *avoidance* (n), ***escape* (v)**, ***elude* (v)**, *keep away* (v), etc.

where R^t_{imp} denotes an unknown (implicit) relation. For a pair of historical events that contain an known (explicit) relation R^h_{exp}, similarly, we have that $R^h_{exp} \Leftarrow < e_{h1}, e_{h2} >$. If the scenarios of the historical events are conceptually similar to that of the target, we can conclude that R^t_{imp} is very likely to be the same as R^h_{exp}. So, we produce the `ETI` relation assignment:

$$R^t_{imp} \Leftarrow R^h_{exp}, \text{if} < \mathrm{e_{t1}}, \mathrm{e_{t2}} > \Leftrightarrow < \mathrm{e_{h1}}, \mathrm{e_{h2}} >$$

We name the above assignment process as cross-scenario inference. Figure 1 shows an inference process from the historical defection events H_1 and H_2 to the targets T_1 and T_2 (The full contents of the events can be accessed in (1)).

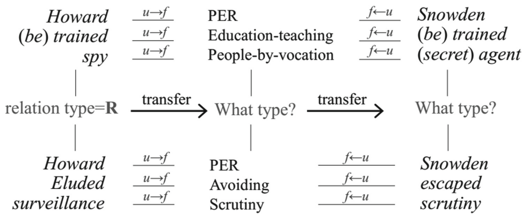


Fig. 1. Cross-scenario relation inference. Either "u→f" or "f←u" refers to a translation process from a lexical unit (word or phrase) to its semantic frame in a specific context. The left column shows historical events, the middle column shows conceptual scenario and the right column shows the targets.

It is strictly required by cross-scenario inference to obtain historical events that contain an explicit relation. The relation is used as prior knowledge to support ETI relation assignment, assigning explicit relation of reliable homogeneous historical events to the target.

3.4 Connective-Relation (C-R) Alignment

If two event mentions is connected by a connective, or included in the scope of the connective [27], they contain an explicit relation. We detect the explicit relation type by a connective-relation alignment approach. The connective "*thus*", for example, aligns with (i.e., signals) the `conditional` relation.

Table 2. Partial relation-connective mapping table

`Causal`- *because*	`Competitive`- *unlike*
`Temporal`- *before*	`Concessive`- *however*
`Subevent`- *involving*	`Disjunctive`- *except*
`Conjunctive`- *in addition*	`Conditional`- *in order to*

We aligns a connective with its relation type by searching it in a one-to-many relation-connective mapping table (See a part of the table in Table 2).

We initialize the table based on the ground-truth connectives and relation samples in PDTB 2.0 corpus[1] [19]. We filter the ambiguous connectives. For example, the connective "*since*" is ambiguous because it may signal a `causal` relation (meaning"*in view of the fact that*") or `temporal` ("*from a past time until now*"). We collected all pairs of events that contain an explicit relation beforehand from 8.2M documents in the 2003 English Gigaword[2]. They're employed as the only data available for the acquisition of the desired similar historical events.

3.5 Scenario Similarity Calculation

We acquire similar historical events based on scenario similarity to the target. We use a frame-based vector space to model event scenarios, in which each dimension indicates a semantic frame. We instantiate the vector by weighting the frames with 1 and 0. Given a frame, it will be weighted by 1 if the word of the frame occurs in the event mention, otherwise 0. We measure the similarity by the cosine metric. Then, we measure the joint similarity between a pair of historical events and the target based on maximum likelihood principle:

$$tuple(e_{hi}, e_{tj}) = argmaxS(e_{hi}, e_{tj}), i, j \in 1, 2 \tag{1}$$

$$S(e_{h1}, e_{h2}, e_{t1}, e_{t2}) = \frac{S(e_{hi}, e_{tj})S(\overline{e_{hi}}, \overline{e_{tj}})}{exp(S(e_{hi}, e_{tj}) - S(\overline{e_{hi}}, \overline{e_{tj}}))} \tag{2}$$

where, e_{hi} and e_{tj} are, respectively, the historical and target events that yield the maximum S, while $\overline{e_{hi}}$ and $\overline{e_{tj}}$ are the other two. The exponential function, i.e., $exp(*)$, ensures that the value of the denominator is nonzero.

3.6 Confidence Level Computing

We search similar events from the data source mentioned in Sect. 3.4, and rank them by scenario similarity to the target. We use top n search results as eligible **R**eference **S**ource (RS, i.e., set of similar historical events) for cross-scenario inference.

[1] https://www.seas.upenn.edu/~pdtb/.
[2] https://catalog.ldc.upenn.edu/LDC2003T05.

For the case that there is only one type of relation R_{exp} occurred in RS, we directly assign it to R_{imp} of the target events as the RD_{2e} result. If there are multiple types occurred, we will evaluate their confidence, and select the most confident one as the RD_{2e} result. Given a type of relation, we measure its confidence level based on the probability it occurred in RS and the ratings:

$$C(\Re) = \frac{f(\Re)}{\prod_{i \in n} rank(\Re)} \tag{3}$$

where, $\Re$ is a relation type, $f(\Re)$ denotes the frequency of $\Re$ occurred in RS. For a reference sample that contains $\Re$, $rank(\Re)$ denotes its rating in the similarity-based ranking list of RS.

4 Graph Based Scenario Modeling

It was found that some lexical units's senses adhere to a semantic frame at different levels, some strictly, others loosely. In the cases, the scenario modeling mentioned in section 2.5 fails to identify the difference, assigning the same weight to the units at the dimension of the frame, either 1 if they occur in event mentions, otherwise 0. This causes biases in similarity calculation among scenarios. In this section, we use a lexicon-frame graph to improve scenario modeling.

4.1 Hybrid Lexicon-Frame Graph

Given a lexical unit u, if its sense conforms to a frame f and the semantic distance is d, the weight of f can be measured by the reciprocal of d, i.e., $w_u(f) = 1/d$.

In order to facilitate the measurement of semantic distance, we built a hybrid lexicon-frame graph based on `Ffr`. `Ffr` is a connection between frames, indicating a semantic relation. There are totally 8 `Ffr` types defined in FrameNet, including `Inheritance`, `Subframe`, `Precedence`, `Perspective`, `Inchoation`, etc. By tracing along the `Ffrs` in an end-to-end manner, we can find out new semantically related unit-frame, unit-unit or frame-frame pairs. They are useful for the generation of a hybrid directed lexicon-frame graph. In practice, we only obtained a collection of subgraphs because the manufactured `Ffr` edges are incompleted. We brought the subgraphs into use in our experiments. See an example in Fig. 2.

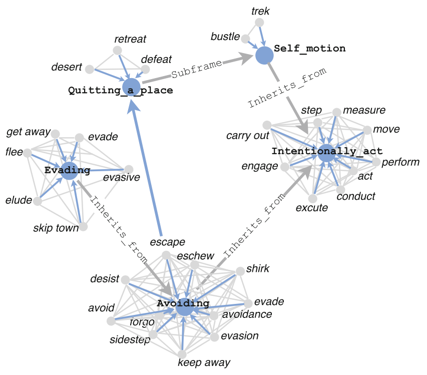


Fig. 2. Hybrid lexicon-frame subgraph

For two nodes in the graph, we specify the length of the shortest path between the nodes as their semantic distance. The length is measured by the number of edges between the nodes. We take into consideration all frames that are directly or indirectly connected with a lexical unit in the graph, and use them as semantically related to the unit. Based on the semantic distance, the refined scenario model will assign a weight $w_u(f)$ to every frame f related to a unit u, for every unit occurred in the event mention. Still, it assigns 0 to other frames. *escape* → `Avoiding` → `Intentionally_act`.

4.2 Probabilistic Scenario Model

We employ a probabilistic matrix as a substitute for VSM. Each matrix row is an uneven projection on all semantic frames, denoting the probability distribution of a specific lexical unit over the frames. The probability is estimated by semantic distance in the same way as the weighting method $w_u(f)$, except that it is normalized by the sum of the weights:

$$p_u(f) = \frac{w_u(f)}{\sum_f w_u(f)} \tag{4}$$

Accordingly an event scenario is represented as a lexicon-frame matrix, where each row indicates the probability model $p_u(f)$ of a lexical unit u over all frames. If a unit doesn't occur in the mention, the probability $p_u(f)$ over the frames in the corresponding row will all be set as 0. Thus, the similarity between events can be estimated by the agreement of their lexicon-frame probability matrixes.

Technically we use the Kullback-Liebler (KL) divergence [15] as a measure of the agreement A(*,*). A significant KL divergence is equivalent to little agreement. The agreement between matrixes is fully determined by that between all their rows. In addition, we involve the standard deviation in the measurement of partial agreement between a row in a matrix and all in another. The deviation is employed to reduce the effect of general rows on the measure of agreement. A general row corresponds to a unit who has similar $p_u(f)$ over the usual frames. We measure the full agreement between scenarios by:

$$S(e_h, e_t) \propto A(P_h, P_t) = \frac{G}{D_{KL}(P_h, P_t)}, \qquad g(u) = \sqrt{\frac{\sum_{f \in F}(p(f) - \overline{p(f)})^2}{n-1}} \tag{5}$$

$$D_{KL}(P_h, P_t) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{f \in F} p_{u_i,h}(f)\ log\frac{p_{u_i,h}(f)}{p_{u_j,t}(f)}, \qquad G = \sum_{i=1}^{n}\sum_{j=1}^{n} g_h(u_i)g_t(u_j) \tag{6}$$

5 Experiments

5.1 Corpus and Evaluation Metric

We follow Yu et al. [9] relation schema (see Table 3). We collected 828 event mentions from 24 expository texts in the American National Corpus (ANC)[3].

[3] https://framenet.icsi.berkeley.edu/fndrupal/fulltextIndex.

There are 968 pairs of events found related to each other, in which 569 are structurally adjacent, 330 cross-sentence and 69 cross-document. Three students who major in computational linguistics were employed to label the relations. The agreement among them is fairly good (kappa = 0.69 at worst).

In addition, we annotate the relations for 303 pairs of related events in 159 ACE2005 news stories . They were used for the purpose of evaluating the cross-domain stability of our RD_{2e} system. We employ Precision (P), Recall (R) and F-score (F) as the evaluation metrics.

5.2 Main Results

We test our basic RD_{2e} on ANC. The frame-based VSM was used. Table 4 shows the performance for each existing relation type in ANC. The performance for the two temporal relation types (*Bef.* & *Eql.*) is different from the true state. Annotators were unable to detect temporal relations for most samples in ANC due to lack of visible evidence, except the 80 ones shown in Table 3. The scores of *Bef.* & *Eql.*, hence, are yielded only for them not all. For uncommon relation types (Table 3), our RD_{2e} shows poor recall (Table 4). We suggest that if a relation type is uncommon, there is little explicit and definitive evidence in neither our mind nor an electronic database available for inferring it.

Table 3. Relational Schema

Relation(*REL.*)	Num
Before(*Bef.*)	29
Equal(*Eql.*)	51
Opposite(*Opp.*)	91
Variance(*Var.*)	31
Cause(*Cas.*)	131
Condition(*Con.*)	197
Conjunction(*Coj.*)	203
Concession(*Coc.*)	43
Coreference(*Cor.*)	89
Sub-event(*Sub.*)	103

Table 4. Performance for all types

REL.	# P	# R	# F
Bef.	0.61	0.31	0.41
Eql.	0.82	0.25	0.38
Opp.	0.51	0.67	0.58
Var.	0.65	0.30	0.41
Cas.	0.79	0.66	0.72
Con.	0.61	0.43	0.50
Coj.	0.39	0.73	0.51
Coc.	0.85	0.16	0.27
Cor.	0.57	0.29	0.38
Sub.	0.31	0.69	0.43

We built RD_{2e} systems by using different models proposed in the paper, including word-level and frame-level scenario description along with VSM and probability (PRO) matrix models. We evaluate them on both ANC and ACE. See performance in Table 5. It is illustrated that the frame based scenario conceptualization and the generalized probability model are both conducive to cross-scenario inference. Moreover, they show better stability than the word-level model in different domains.

Table 5. Performance (F) on ANC and ACE

System construction	# ANC	# ACE
Lexical Unit + VSM	0.45	0.41
Frame + VSM	0.51	0.50
Frame + PRO	0.56	0.53

We reproduce some state-of-the-art relation detection methods. Some of them were employed in determining discourse-oriented structural relations between adjacent text spans, such as the supervised relation classification (CLA) based on syntactic constituent and dependency [13] or multiple types of linguistic features [18], as well as language model (LG) based connective and relation prediction [26]. Others focus on event-event relation problems, such as the use of coocurrence (Coo) information of pairwise phrases or pattens for relation analysis [2]. We evaluate the methods on the samples of different structures in ANC.

Table 6. F-measure for event pairs in adjacent (ADJ), CroS and CroD structures

System construction	# ADJ	# CroS	# CroD
Syntactic Features (CLA)	0.50	N/A	N/A
All features (CLA)	0.61	N/A	N/A
Connective (LG)	0.51	0.30	0.33
Phrase (Coo)	0.49	0.39	0.35
Pattern (Coo)	0.56	0.41	0.40
Our best	**0.55**	**0.59**	**0.53**

Table 6 shows their performance and our best. The classifiers didn't yield any performance for both the structurally cross-sentence (CroS) and cross-document (CroD) test samples, as in the cases, the syntactic features are non-existent and therefore unavailable. Beyond that, we can see that our unsupervised cross-scenario inference method yields insignificant performance loss for the cross-sentence and cross-document samples. It means the method is more credible than others when provided few available structural-level linguistic features.

6 Conclusion

We propose a cross-scenario inference method for event-event relation detection. It outperforms the state of the art in low resource settings. This can facilitate the fast detection of relations for the newly occurred related events, although the detection results are not quite exact (pseudo relations). Encouraged by this observation, we will focus on the refinement of pseudo relations by using semi-supervised learning method.

Acknowledgements. This work was supported by the national Natural Science Foundation of China via Nos. 2017YFB1002104, 61672368 and 61672367.

References

1. Abe, S., Inui, K., Matsumoto, Y.: Acquiring event relation knowledge by learning cooccurrence patterns and fertilizing cooccurrence samples with verbal nouns. In: IJCNLP, pp. 497–504 (2008)
2. Abe, S., Inui, K., Matsumoto, Y.: Two-phased event relation acquisition: coupling the relation-oriented and argument-oriented approaches. In: COLING, pp. 1–8 (2008)
3. Blanco, E., Castell, N., Moldovan, D.I.: Causal relation extraction. In: LREC (2008)
4. Caselli, T., Fokkens, A., Morante, R., Vossen, P.: Spinoza vu: an NLP pipeline for cross document timelines. In: SemEval-2015 p. 787 (2015)
5. Chang, D.-S., Choi, K.-S.: Causal relation extraction using cue phrase and lexical pair probabilities. In: Su, K.-Y., Tsujii, J., Lee, J.-H., Kwong, O.Y. (eds.) IJCNLP 2004. LNCS (LNAI), vol. 3248, pp. 61–70. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30211-7_7
6. Do, Q.X., Chan, Y.S., Roth, D.: Minimally supervised event causality identification. In: EMNLP, pp. 294–303 (2011)
7. Gildea, D., Palmer, M.: The necessity of parsing for predicate argument recognition. In: ACL, pp. 239–246 (2002)
8. Girju, R., Moldovan, D.I., et al.: Text mining for causal relations. In: FLAIRS, pp. 360–364 (2002)
9. Hong, Y., Zhang, T., O'Gorman, T., Horowit-Hendler, S., Ji, H., Palmer, M.: Building a cross-document event-event relation corpus. In: LAW X, p. 1 (2016)
10. Inui, T., Inui, K., Matsumoto, Y.: Acquiring causal knowledge from text using the connective marker tame. TALIP **4**(4), 435–474 (2005)
11. Ittoo, A., Bouma, G.: Extracting explicit and implicit causal relations from sparse, domain-specific texts. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) NLDB 2011. LNCS, vol. 6716, pp. 52–63. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22327-3_6
12. Lapata, M., Lascarides, A.: Learning sentence-internal temporal relations. J. Artif. Intell. Res. (JAIR) **27**, 85–117 (2006)
13. Lin, Z., Ng, H.T., Kan, M.Y.: A PDTB-styled end-to-end discourse parser. Nat. Lang. Eng. **20**(02), 151–184 (2014)
14. Mani, I., Verhagen, M., Wellner, B., Lee, C.M., Pustejovsky, J.: Machine learning of temporal relations. In: COLING and ACL, pp. 753–760 (2006)
15. Migon, H.S., Gamerman, D., Louzada, F.: Statistical inference: an integrated approach. CRC Press, Boca Raton (2014)
16. Minard, A.L., et al.: Semeval-2015 task 4: Timeline: Cross-document event ordering. In: SemEval, pp. 778–786 (2015)
17. Miwa, M., Bansal, M.: End-to-end relation extraction using lstms on sequences and tree structures (2016). arXiv preprint. arXiv:1601.00770
18. Pitler, E., Louis, A., Nenkova, A.: Automatic sense prediction for implicit discourse relations in text. In: ACL and AFNLP, pp. 683–691 (2009)
19. Prasad, R., et al.: The penn discourse treebank 2.0. In: LREC (2008)

20. Pustejovsky, J., Verhagen, M.: Semeval-2010 task 13: evaluating events, time expressions, and temporal relations (tempeval-2). In: Workshop on Semantic Evaluations, pp. 112–116 (2009)
21. Radinsky, K., Davidovich, S., Markovitch, S.: Learning causality for news events prediction. In: WWW, pp. 909–918 (2012)
22. Raja, K., Subramani, S., Natarajan, J.: Ppinterfinder–a mining tool for extracting causal relations on human proteins from literature. Database 2013, bas052 (2013)
23. Rugemalira, J.M.: What is a symmetrical language? multiple object constructions in Bantu. In: Berkeley Linguistics Society, vol. 17 (2012)
24. Santos, C.N.d., Xiang, B., Zhou, B.: Classifying relations by ranking with convolutional neural networks (2015). arXiv preprint arXiv:1504.06580
25. Velupillai, S., Mowery, D.L., Abdelrahman, S., Christensen, L., Chapman, W.W.: Blulab: Temporal information extraction for the 2015 clinical tempeval challenge. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pp. 815-819 (2015)
26. Zhou, Z.M., Xu, Y., Niu, Z.Y., Lan, M., Su, J., Tan, C.L.: Predicting discourse connectives for implicit discourse relation recognition. In: COLING, pp. 1507–1514 (2010)
27. Zou, B., Zhou, G., Zhu, Q.: Tree kernel-based negation and speculation scope detection with structured syntactic parse features. In: EMNLP, pp. 968–976 (2013)

SeRI: A Dataset for Sub-event Relation Inference from an Encyclopedia

Tao Ge[1,2](✉), Lei Cui[2], Baobao Chang[1], Zhifang Sui[1], Furu Wei[2], and Ming Zhou[2]

[1] School of EECS, Peking University, Beijing, China
{chbb,szf}@pku.edu.cn
[2] Microsoft Research Asia, Beijing, China
{tage,lecu,fuwei,mingzhou}@microsoft.com

Abstract. Mining sub-event relations of major events is an important research problem, which is useful for building event taxonomy, event knowledge base construction, and natural language understanding. To advance the study of this problem, this paper presents a novel dataset called SeRI (**S**ub-**e**vent **R**elation **I**nference). SeRI includes 3,917 event articles from English Wikipedia and the annotations of their sub-events. It can be used for training or evaluating a model that mines sub-event relation from encyclopedia-style texts. Based on this dataset, we formally define the task of sub-event relation inference from an encyclopedia, propose an experimental setting and evaluation metrics and evaluate some baseline approaches' performance on this dataset.

1 Introduction

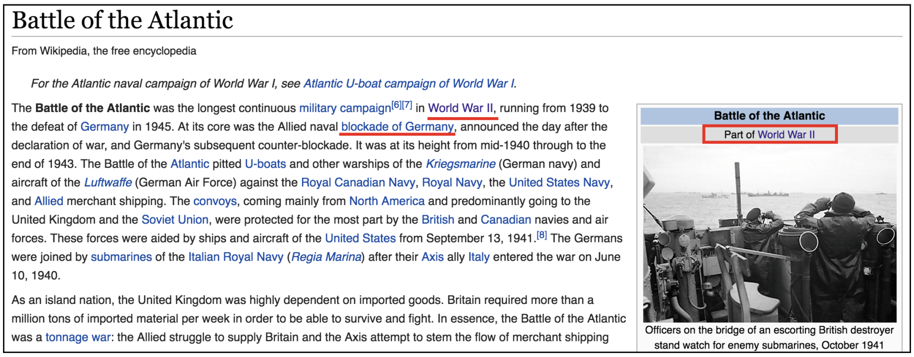
Event relation inference is an important research topic because event relations are not only indispensable for constructing event knowledge bases but also helpful for natural language understanding and knowledge inference. As one of the most important event relations, sub-event relation knowledge has been proved to be useful in many applications [2,6,14,19–23]. For example, the sub-event knowledge *sub*(*Battle of the Atlantic*, *World War II*) can greatly help textual entailment (e.g. (1)) and knowledge inference (e.g., (2)) task.

(1) He died in the *Battle of the Atlantic.* → He died in *World War II.*
(2) time(*World War II*, 1939-1945) → time(*Battle of the Atlantic*, years during 1939-1945)

Despite the significance of general sub-event knowledge, there is little work on mining the sub-event knowledge from the web, which may be due largely to a lack of datasets. To advance the research of this problem, this paper presents a dataset called SeRI (**S**ub-**e**vent **R**elation **I**nference). SeRI contains 3,917 event articles[1] from English Wikipedia, and the annotations for their sub-events, which

[1] Event articles refer to the articles that describe a major event in Wikipedia, like Fig. 1.

© Springer Nature Switzerland AG 2018
M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 268–277, 2018.
https://doi.org/10.1007/978-3-319-99501-4_23



Battle of the Atlantic

From Wikipedia, the free encyclopedia

For the Atlantic naval campaign of World War I, see Atlantic U-boat campaign of World War I.

The **Battle of the Atlantic** was the longest continuous military campaign[6][7] in World War II, running from 1939 to the defeat of Germany in 1945. At its core was the Allied naval blockade of Germany, announced the day after the declaration of war, and Germany's subsequent counter-blockade. It was at its height from mid-1940 through to the end of 1943. The Battle of the Atlantic pitted U-boats and other warships of the *Kriegsmarine* (German navy) and aircraft of the *Luftwaffe* (German Air Force) against the Royal Canadian Navy, Royal Navy, the United States Navy, and Allied merchant shipping. The convoys, coming mainly from North America and predominantly going to the United Kingdom and the Soviet Union, were protected for the most part by the British and Canadian navies and air forces. These forces were aided by ships and aircraft of the United States from September 13, 1941.[8] The Germans were joined by submarines of the Italian Royal Navy (*Regia Marina*) after their Axis ally Italy entered the war on June 10, 1940.

As an island nation, the United Kingdom was highly dependent on imported goods. Britain required more than a million tons of imported material per week in order to be able to survive and fight. In essence, the Battle of the Atlantic was a tonnage war: the Allied struggle to supply Britain and the Axis attempt to stem the flow of merchant shipping

Battle of the Atlantic

Part of World War II

Officers on the bridge of an escorting British destroyer stand watch for enemy submarines, October 1941

Fig. 1. The event article of *Battle of the Atlantic* event in Wikipedia. The red rectangle provides with the annotation information and the underlined phrases are mentions of other events, linking to the corresponding event articles. (Color figure online)

can be used for training and evaluating a model for mining sub-event relations from an encyclopedia. Based on this dataset, we formally define the task of sub-event relation inference from an encyclopedia, propose an experimental setting and evaluation metrics and evaluate some baseline approaches' performance on this dataset.

The main contributions of this paper are:

– We release a dataset[2] for studying sub-event relation inference from an encyclopedia.
– We formally define the task of sub-event relation inference from an encyclopedia and propose an experimental setting and evaluation metrics.
– We evaluate some baseline approaches' performance on this dataset.

2 SeRI Dataset

SeRI is a dataset we propose for studying sub-event relation inference. For constructing SeRI, we first need to find event articles on Wikipedia. In this paper, we use those 21,275 event articles identified by EventWiki [8] as our research targets. We keep the events with "*partof*" relation annotation in their infoboxes (shown in Fig. 1) and use the annotation as their sub-event relation annotations.

It should be noted that the *partof* annotations in infoboxes are mainly for direct sub-event relations. However, sub-event relations are transitional and there are many indirect sub-event relations that are not annotated in EventWiki. Therefore, we add the annotations of indirect sub-event relations through transitivity rules. The step is necessary to annotate some long-distance sub-event relations that are often not explicitly expressed in Wikipedia. For example, given the annotation from EventWiki that *sub*(*Battle of Netherlands*, *Battle of France*),

[2] Please contact the first author to request the access to the dataset.

sub(*Battle of France*, *Western Front*) and *sub*(*Western Front*, *World War II*), according to the transitivity, long-distance sub-event relations such as *sub*(*Battle of Netherlands*, *World War II*) will be added to our annotation.

The resulting SeRI dataset has 3,917 Wikipedia articles (i.e., 3,917 events) and their sub-event annotations.

3 Task Overview

If an event e_i is a part of event e_j, we say e_i is e_j's *sub-event* and e_j is e_i's *sup-event*. There are many event articles (e.g., *World War II*) with reference links in an encyclopedia, as Fig. 2 depicts. The goal of this task is to harvest as much sub-event knowledge as possible from an encyclopedia. Strictly speaking, for an event e_i, we should identify all the other events in an encyclopedia to see if they are the sub-events or sup-events of e_i. In this sense, we must infer the sub-event relation over all the possible event pairs, which results in expensive cost of computation. Instead of the brutal-force solution, we propose a heuristic assumption that if an event e_i is a sub-event or sup-event of another event e_j, then e_i should be mentioned by the article of e_j or e_j should be mentioned by the article of e_i. This assumption is reasonable because if the sub-event relation holds between e_i and e_j then e_i and e_j should be strongly related and either of them should be mentioned by the other. Based on the assumption, we identify sub-event relations over the event pairs in which one event is mentioned (i.e., linked) by the article of the other event. For simplicity, we call such event pairs *candidate event pairs*.

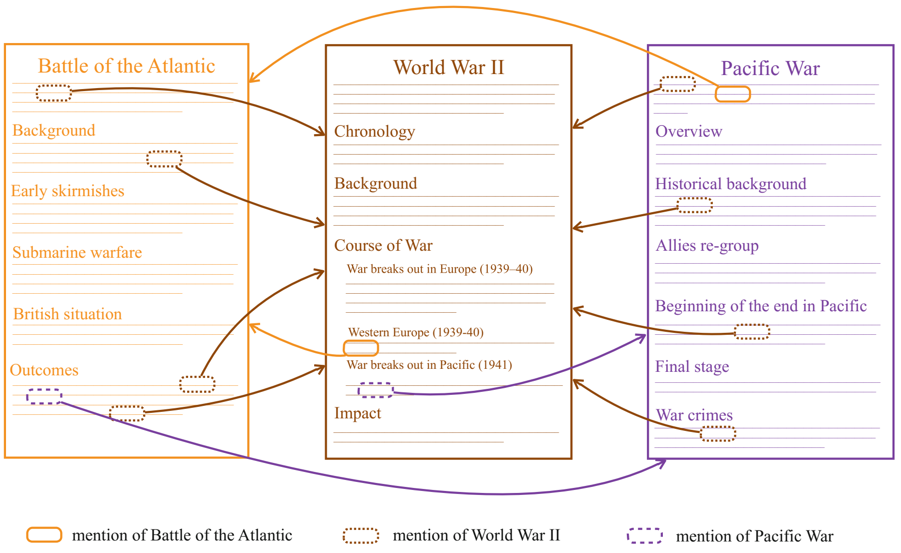


Fig. 2. An illustration of the basic structure of an encyclopedia in which articles are interconnected by the reference links.

We give the formal definition of the sub-event inference task: given a set of candidate event pairs $\mathcal{S} = \{\langle e_i, e_j \rangle\}$ in an encyclopedia, the goal is to identify all sub-event pairs $\langle e_i^*, e_j^* \rangle$ from $\mathcal{S}$ so that e_i^* is a sub-event or sup-event of e_j^*. In other words, for each pair $\langle e_i, e_j \rangle \in \mathcal{S}$, a model should be able to correctly predict its label $r \in \{none, sub, sup\}$ that indicates the sub-event relation between e_i and e_j. In our SeRI dataset, there are totally 7,373 candidate event pairs with relation label annotation $r \in \{none, sub, sup\}$.

4 Baseline Models

For the task of harvesting sub-event knowledge from an encyclopedia, how an event article links to (i.e., refers to) another is an important clue for inferring the sub-event relation between them. For example, it can be easily inferred that the *Battle of the Atlantic* is a sub-event of the *World War II* through analyzing the sentence corresponding to a reference link from *Battle of the Atlantic* to *World War II*, as Fig. 3 shows.

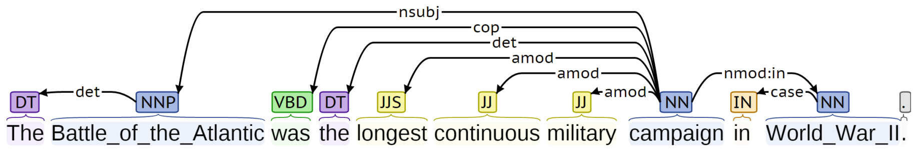

nsubj
cop
det
amod
amod
amod
nmod:in
det
case
DT
NNP
VBD
DT
JJS
JJ
JJ
NN
IN
NN
.
The Battle_of_the_Atlantic was the longest continuous military campaign in World_War_II.

Fig. 3. The sentence that corresponds to the first link from *Battle of the Atlantic* to *World War II* in Fig. 2.

As Fig. 2 depicts, there may be multiple reference links from both directions between an event pair. Given the value of the reference links for sub-event relation inference, we propose a link-based classification model that predicts an event pair's sub-event relation through the reference links between them.

For an event pair $\langle e_i, e_j \rangle$ (i.e., an instance), we first find all the reference links between them. Then, we consider each link as a sub-instance and describe it using various features.

4.1 Features

As discussed previously, each link $e_i \xrightarrow{l} e_j$, which denotes the l^{th} link from the article of e_i to e_j, corresponds to a sentence that mentions e_j in the article of e_i. We extract the following features to describe the link:

- Context words of e_j: we extract the context words (i.e., last 5 words and next 5 words) surrounding the mention of e_j in the article of e_i as features. We denote this feature as $\boldsymbol{f_c}$.

- N-grams that start or end with the mention of e_j: we extract 2-, 3- and 4-grams that start or end with e_j as features. For generalization, e_j is replaced with *"TARGET"* in feature strings. We denote this feature as $\boldsymbol{f_g}$.
- Dependency parse: We extract all dependency arcs that contain e_j as features. In addition, we also use the part-of-speech tag of the word in a dependency arc as features. We denote this feature as $\boldsymbol{f_p}$.
- Section name: In addition to the sentence-level features that are commonly used by traditional relation extraction models, we propose to use the section name feature. For example, for the reference link from *World War II* to *Battle of the Atlantic* in Fig. 2, one can observe that the mention of *Battle of the Atlantic* is in the section called *Course of war* in the article of *World War II*. In this case, we extract "course of war" as the section name features. We denote this feature as $\boldsymbol{f_s}$.

After extracting the features of a reference link between an event pair $\langle e_i, e_j \rangle$, we can represent the features as a tuple to represent the link:

$$\boldsymbol{f} = (\boldsymbol{f_c}, \boldsymbol{f_g}, \boldsymbol{f_p}, \boldsymbol{f_s}, d) \tag{1}$$

where the last element $d \in \{\rightarrow, \leftarrow\}$ indicates the direction of the link.

We represent an event pair instance $\langle e_i, e_j \rangle$ by concatenating the features of all the links between the event articles in the pair to train the link-based classifier and infer the sub-event relation in the following way:

$$y^* = \arg\max_y P(y|\textsc{ConCat}_{l=1}^{L}(\boldsymbol{f}^{(l)})) \tag{2}$$

where l is the l-th link between e_i and e_j, $\boldsymbol{f}^{(l)}$ is the features of l-th link between e_i and e_j, and ConCat is the operation that concatenates the features of all the links between e_i and e_j.

4.2 Bi-directional Training Instance Generation

Although we do not consider the order of the items in a pair (i.e., we consider $\langle e_i, e_j \rangle$ and $\langle e_j, e_i \rangle$ are the identical pair), it is necessary to specify the order of an pair when generating training and test instances for a classifier because the label of the instance depends on the order[3]. For example, if the label of a pair instance (e_i,e_j) is *sub*, then the label of a pair instance (e_j,e_i) should be *sup*.

Therefore, for an event pair $\langle e_i, e_j \rangle$, we specify the order of the items to generate training and test instances. Specifically, for training instance generation, we generate bi-directional pair instances (i.e., 2 pair instances (e_i,e_j) and (e_j,e_i)). There are two reasons for the bi-directional training instance generation: First, the bi-directional training instances will provide different views for the model to learn to predict *sub* and *sup*, which is important because the test

[3] To distinguish from a *pair* which is denoted as $\langle e_i, e_j \rangle$ for which we do not consider the order of items, we say a *pair instance* denoted as (e_i,e_j)) when we consider the order of items in a pair.

instance could be in any order. Bi-directional training instances allow the model to handle the test instances with any order; Second, the bi-directional training instance generation can alleviate the imbalance of labels because the number of the training instances with *sub* and *sup* will be identical.

5 Experiments

5.1 Experimental Setting

As we mentioned before, SeRI has 3,917 Wikipedia articles (i.e., 3,917 events) and 7,373 candidate event pairs in total. Each candidate event pair has a label $r \in \{none, sup, sub\}$ indicating the sub-event relation. In our experiments, we use 80% of candidate event pairs as the training set and test on the remaining 20% of the pairs. Note that in the split, we split event pairs instead of event articles because it is very common that an event article's sub-event relation is not complete and it might have sub-event relation with new added event articles. Therefore, splitting event pairs is a more practical setting than splitting event articles. The label distribution is shown in Table 1.

Table 1. Label distribution in training and test set

	Train		Test	
Label	none	sub+sup	none	sub+sup
Number of instances	3,824	2,112	893	544
Ratio	64.4%	35.6%	62.1%	37.9%

In our experimental setting, we do not use any structured information (i.e., infoboxes) since we want a model to be general so that it can applied to any encyclopedia articles regardless of the existence of the infoboxes for discovering the sub-event knowledge that has not been explicitly expressed in an encyclopedia.

As traditional relation extraction tasks, we use Precision, Recall and F-score for evaluation:

$$Precision = \frac{\left|\{\langle e_i, e_j\rangle \mid \hat{r_{i,j}} = r^*_{i,j}\}\right|}{|\{\langle e_i, e_j\rangle | \hat{r_{i,j}} \in \{sup, sub\}\}|}$$

$$Recall = \frac{\left|\{\langle e_i, e_j\rangle \mid \hat{r_{i,j}} = r^*_{i,j}\}\right|}{\left|\{\langle e_i, e_j\rangle | r^*_{i,j} \in \{sup, sub\}\}\right|}$$

$$F\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where $\hat{r_{i,j}}$ and $r^*_{i,j}$ are the prediction and gold standard relation of $\langle e_i, e_j\rangle$ respectively.

5.2 Experimental Results

We conduct experiments test the baseline approaches mainly for answering the following three questions:

– Whether are the features used in the link-based classifier effective?
– Whether is generating bi-directional training instances necessary?
– How well a baseline model can perform?

We use Stanford CoreNLP [16] to do POS tagging and dependency parsing and maximum entropy classifier as the link-based classification model.

Table 2. Performance of baseline models. Uni- and Bi-directional indicate if the training instances are generated using bi-directional generation strategy.

Model		Uni-directional			Bi-directional		
		P	R	F	P	R	F
Link-based classifier	(1): Context	51.8	65.8	58.0	68.4	61.4	64.7
	(2): (1)+N-grams	52.6	72.4	61.0	70.6	63.1	66.6
	(3): (2)+Dependency parse	52.7	71.1	60.5	69.7	65.3	67.4
	(4): (3)+section name	63.0	76.8	69.3	77.8	73.7	75.7

Table 2 shows the performance of various approaches. For the link-based classifier with bi-directional training instance, the proposed features are all effective in inferring the sub-event relations, especially the section name features. The reason of the significant improvement (8.3% F-score gain) brought by the section name features is that they provide totally different views from the sentence-level features such as n-grams and can nicely address the cases where n-gram and parse features cannot help.

By comparing uni-directional and bi-directional training instance generation, we can easily observe the superiority of the bi-directional training generation strategy. For the models using the same feature set, the model adopting bi-directional training generation strategy outperforms the uni-directional counterpart by approximately 6.0% F-score gain. As we discussed before, the improvement is mainly owing to that bi-directional training instances provide more information for the model and alleviate the label imbalance problem.

According to Table 2, it is observed that the best baseline model can achieve a good performance of 75.7% F-score. Given this is a preliminary study on this task, we expect to see more results of various approaches by future work that studies on this dataset.

At last, we show the example of sub-event knowledge discovered by our model on the test set in Fig. 4. As observed, the results can be used for building an event hierarchy tree, which will be useful for event knowledge base construction and management as well as many other applications like knowledge inference and question answering.

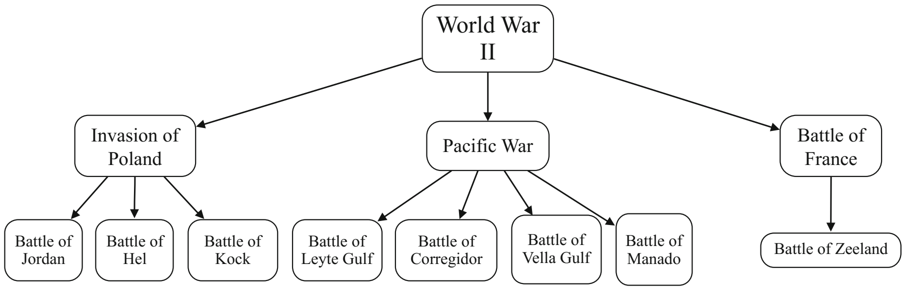


Fig. 4. An event tree derived from the predictions of our model on the test data.

6 Related Work

There is some previous work on extracting temporal and causal relation between events [1,3–5,7,9–11,17,18,24]. As for the studies regarding sub-events, most of them study either fine-grained events in a sentence or a document [2] or hierarchies of events (topics) in a text stream [12,13,23]. Even though they are very useful for understanding a document or the relation of events in a text stream, they do not focus on harvesting knowledge and the relations identified by their models are less likely to be used as general knowledge. In contrast, this paper mainly studies the acquisition of sub-event knowledge which can be used as world knowledge for a variety of applications.

Another research branch related to our task is hypernymy detection (e.g., [15]), whose goal is to discover the hypernymy between general words. In contrast to hypernymy detection, our targets are events which contain much richer information and sub-event relations that are more complicated because it is common that there is no co-occurrence of the mentions of two events in a sentence.

7 Conclusion and Future Work

In this paper, we study a novel dataset SeRI and define a new task – harvesting sub-event knowledge from an encyclopedia. We propose a link-based classification baseline model with features of encyclopedia-style documents, which achieves decent results on our dataset.

Following the preliminary study on event relations, we plan to study advanced approaches for this task and do more empirical studies on this dataset, and make an effort to keep enlarging the dataset. We expect more research could be conducted regarding this task, which would contribute to event knowledge discovery and event knowledge base construction.

Acknowledgments. The research work is supported by the National Science Foundation of China under Grant No. 61772040. The contact author is Zhifang Sui.

References

1. Abe, S., Inui, K., Matsumoto, Y.: Two-phased event relation acquisition: coupling the relation-oriented and argument-oriented approaches. In: Proceedings of the 22nd International Conference on Computational Linguistics, vol. 1, pp. 1–8. Association for Computational Linguistics (2008)
2. Araki, J., Liu, Z., Hovy, E.H., Mitamura, T.: Detecting subevent structure for event coreference resolution. In: LREC, pp. 4553–4558 (2014)
3. Chambers, N., Cassidy, T., McDowell, B., Bethard, S.: Dense event ordering with a multi-pass architecture. Trans. Assoc. Comput. Linguist. **2**, 273–284 (2014)
4. Chambers, N., Jurafsky, D.: Unsupervised learning of narrative schemas and their participants. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, vol. 2, pp. 602–610. Association for Computational Linguistics (2009)
5. Chambers, N., Jurafsky, D.: Unsupervised learning of narrative event chains. In: ACL, vol. 94305, pp. 789–797. Citeseer (2008)
6. Daniel, N., Radev, D., Allison, T.: Sub-event based multi-document summarization. In: Proceedings of the HLT-NAACL 03 on Text Summarization Workshop, vol. 5, pp. 9–16. Association for Computational Linguistics (2003)
7. Do, Q.X., Chan, Y.S., Roth, D.: Minimally supervised event causality identification. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 294–303. Association for Computational Linguistics (2011)
8. Ge, T., Cui, L., Chang, B., Sui, Z., Wei, F., Zhou, M.: Eventwiki: a knowledge base of major events. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018) (2018)
9. Ge, T., Cui, L., Ji, H., Chang, B., Sui, Z.: Discovering concept-level event associations from a text stream. In: Lin, C.-Y., Xue, N., Zhao, D., Huang, X., Feng, Y. (eds.) ICCPOL/NLPCC -2016. LNCS (LNAI), vol. 10102, pp. 413–424. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50496-4_34
10. Hashimoto, C., Torisawa, K., Kloetzer, J., Oh, J.H.: Generating event causality hypotheses through semantic relations. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
11. Hashimoto, C., et al.: Toward future scenario generation: extracting event causality exploiting semantic relation, context, and association features. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, USA. Association for Computational Linguistics, June 2014
12. Hu, L., Li, J., Zhang, J., Shao, C.: o-HETM: an online hierarchical entity topic model for news streams. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS (LNAI), vol. 9077, pp. 696–707. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18038-0_54
13. Hu, L., et al.: Learning topic hierarchies for wikipedia categories. In: ACL (2015)
14. Im, S., Pustejovsky, J.: Annotating lexically entailed subevents for textual inference tasks. In: Twenty-Third International Flairs Conference (2010)
15. Levy, O., Remus, S., Biemann, C., Dagan, I., Ramat-Gan, I.: Do supervised distributional methods really learn lexical inference relations? In: HLT-NAACL, pp. 970–976 (2015)
16. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J.R., Bethard, S., McClosky, D.: The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations) (2014)

17. Mirza, P.: Extracting temporal and causal relations between events. In: ACL (Student Research Workshop), pp. 10–17 (2014)
18. Mirza, P., Tonelli, S.: An analysis of causality between events and its relation to temporal information. In: COLING, pp. 2097–2106 (2014)
19. Mulkar-Mehta, R., Welty, C., Hoobs, J.R., Hovy, E.: Using granularity concepts for discovering causal relations. In: Proceedings of the FLAIRS Conference (2011)
20. Pohl, D., Bouchachia, A., Hellwagner, H.: Automatic sub-event detection in emergency management using social media. In: Proceedings of the 21st International Conference Companion on World Wide Web, pp. 683–686. ACM (2012)
21. Shen, C., Liu, F., Weng, F., Li, T.: A participant-based approach for event summarization using twitter streams. In: HLT-NAACL, pp. 1152–1162 (2013)
22. Unankard, S., Li, X., Sharaf, M., Zhong, J., Li, X.: Predicting elections from social networks based on sub-event detection and sentiment analysis. In: Benatallah, B., Bestavros, A., Manolopoulos, Y., Vakali, A., Zhang, Y. (eds.) WISE 2014. LNCS, vol. 8787, pp. 1–16. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11746-1_1
23. Xing, C., Wang, Y., Liu, J., Huang, Y., Ma, W.Y.: Hashtag-based sub-event discovery using mutually generative lda in twitter. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
24. Yoshikawa, K., Riedel, S., Asahara, M., Matsumoto, Y.: Jointly identifying temporal relations with markov logic. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, vol. 1, pp. 405–413. Association for Computational Linguistics (2009)

Densely Connected Bidirectional LSTM with Applications to Sentence Classification

Zixiang Ding[1], Rui Xia[1](✉), Jianfei Yu[2], Xiang Li[1], and Jian Yang[1]

[1] School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
{dingzixiang,rxia,xiang.li.implus,csjyang}@njust.edu.cn

[2] School of Information Systems, Singapore Management University, Singapore, Singapore
jfyu.2014@phdis.smu.edu.sg

Abstract. Deep neural networks have recently been shown to achieve highly competitive performance in many computer vision tasks due to their abilities of exploring in a much larger hypothesis space. However, since most deep architectures like stacked RNNs tend to suffer from the vanishing-gradient and overfitting problems, their effects are still understudied in many NLP tasks. Inspired by this, we propose a novel multi-layer RNN model called densely connected bidirectional long short-term memory (DC-Bi-LSTM) in this paper, which essentially represents each layer by the concatenation of its hidden state and all preceding layers hidden states, followed by recursively passing each layers representation to all subsequent layers. We evaluate our proposed model on five benchmark datasets of sentence classification. DC-Bi-LSTM with depth up to 20 can be successfully trained and obtain significant improvements over the traditional Bi-LSTM with the same or even fewer parameters. Moreover, our model has promising performance compared with the state-of-the-art approaches.

Keywords: Sentence classification · Densely connected
Stacked RNNs

1 Introduction

With the recent trend of deep learning, various kinds of deep neural architectures have been proposed for many tasks in speech recognition [2], computer vision [13] and natural language processing (NLP) [6], which have been shown to achieve better performance than both traditional methods and shallow architectures. However, since conventional deep architectures often suffer from the well-known vanishing-gradient and overfitting problems, most of them are not easy to train and therefore cannot achieve very satisfactory performance.

To address these problems, different approaches have been recently proposed for various computer vision tasks, including Highway Networks [16], ResNet [3]

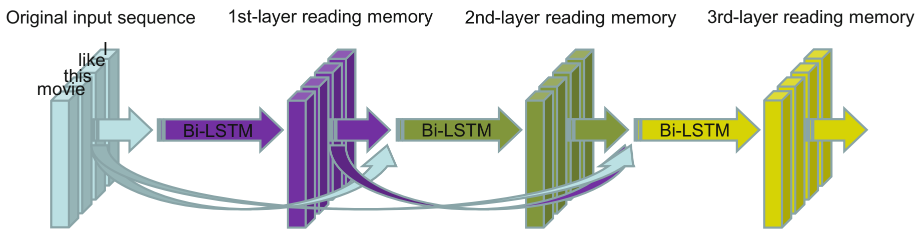


Fig. 1. The architecture of DC-Bi-LSTM. We obtain first-layer reading memory based on original input sequence, and second-layer reading memory based on the position-aligned concatenation of original input sequence and first-layer reading memory, and so on. Finally, we get the n-th-layer reading memory and take it as the final feature representation for classification.

and GoogLeNet [17, 18]. One of the representative work among them is the recently proposed Dense Convolutional Networks (DenseNet) [5]. Different from previous work, to strengthen information flow between layers and reduce the number of parameters, DenseNet proposes to directly connect all layers in a feed-forward fashion and encourages feature reuse through representing each layer by concatenating the feature-maps of all preceding layers as input. Owing to this well-designed densely connected architecture, DenseNet obtains significant improvements over the state-of-the-art results on four highly competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet).

Motivated by these successes in computer vision, some deep architectures have also been recently applied in many NLP applications. Since recurrent neural networks (RNNs) are effective to capture the flexible context information contained in texts, most of these deep models are based on the variants of RNNs. Specifically, on basis of Highway Networks, Zhang *et al.* [23] proposed Highway LSTM to extend stacked LSTM by introducing gated direct connections between memory cells in adjacent layers. Inspired by ResNet, Yu *et al.* [21] further proposed a hierarchical LSTM enhanced by residual learning for relation detection task. However, to the best of our knowledge, the application of DenseNet to RNN has not been explored in any NLP task before, which is the motivation of our work.

Therefore, in this paper, we propose a novel multi-layer RNN model called Densely Connected Bidirectional LSTM (DC-Bi-LSTM) for sentence classification. In DC-Bi-LSTM, we use Bi-LSTM to encode the input sequence, and regard the sequence of hidden states as reading memory for each layer. The architecture of DC-Bi-LSTM is shown in Fig. 1. We evaluate our proposed architecture on five sentence classification datasets, including Movie Review Data [11] and Stanford Sentiment Tree-bank [15] for fine-grained and polarity sentiment classifications, TREC dataset [9] for question type classification and subjectivity classification dataset [10]. DC-Bi-LSTM with depth up to 20 can be successfully trained and significantly outperform the traditional Bi-LSTM with the same or even fewer parameters. Moreover, our model achieves indistinguishable performance in comparison with the state-of-the-art approaches.

2 Model

In this section, we describe the architecture of our proposed Densely Connected Bidirectional LSTM (DC-Bi-LSTM) model for sentence classification.

2.1 Deep Stacked Bi-LSTM

Given an arbitrary-length input sentence $S = \{w_1, w_2, \ldots, w_s\}$, Long Short-Term Memory (LSTM) [4] computes the hidden states $h = \{h_1, h_2, \ldots, h_s\}$ by iterating the following equations:

$$h_t = \text{lstm}(h_{t-1}, e(w_t)). \tag{1}$$

where $e(w_t) \in R^m$ is the word embedding of w_t.

As shown in Fig. 2(a), deep stacked Bi-LSTM [1,14] uses multiple Bi-LSTMs with different parameters in a stacking way. The hidden state of l-layer Bi-LSTM can be represented as h_t^l , which is the concatenation of forward hidden state $\overrightarrow{h_t^l}$

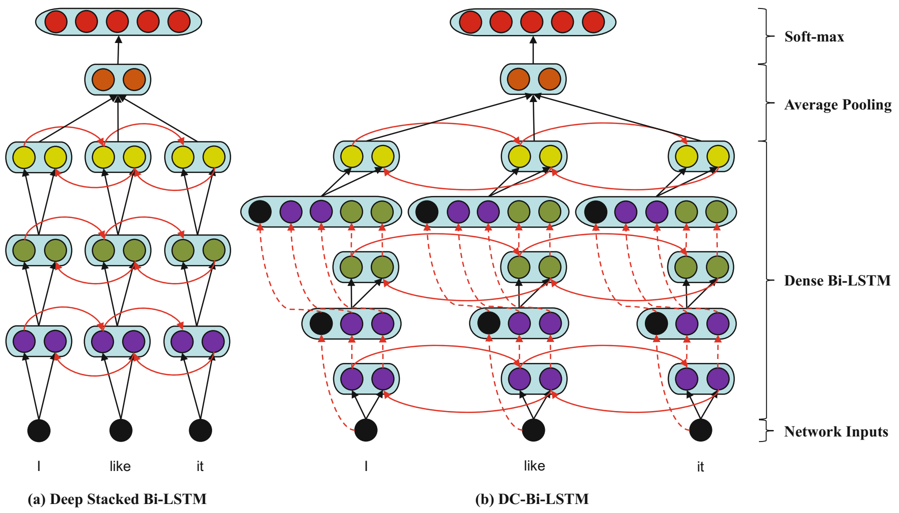


Fig. 2. Illustration of (a) Deep Stacked Bi-LSTM and (b) DC-Bi-LSTM. Each black node denotes an input layer. Purple, green, and yellow nodes denote hidden layers. Orange nodes denote average pooling of forward or backward hidden layers. Each red node denotes a class. Ellipse represents the concatenation of its internal nodes. Solid lines denote the connections of two layers. Finally, dotted lines indicate the operation of copying. (Color figure online)

and backward hidden state $\overleftarrow{h_t^l}$. The calculation of h_t^l is as follows:

$$h_t^l = [\overrightarrow{h_t^l}; \overleftarrow{h_t^l}], \text{ specially, } h_t^0 = e(w_t), \tag{2}$$

$$\overrightarrow{h_t^l} = \text{lstm}(\overrightarrow{h_{t-1}^l}, h_t^{l-1}), \tag{3}$$

$$\overleftarrow{h_t^l} = \text{lstm}(\overleftarrow{h_{t+1}^l}, h_t^{l-1}). \tag{4}$$

2.2 Densely Connected Bi-LSTM

As shown in Fig. 2(b), Densely Connected Bi-LSTM (DC-Bi-LSTM) consists of four modules: network inputs, dense Bi-LSTM, average pooling and soft-max layer.

(1) Network Inputs
The input of our model is a variable-length sentence, which can be represented as $S = \{w_1, w_2, \ldots, w_s\}$. Like other deep learning models, each word is represented as a dense vector extracted from a word embedding matrix. Finally, a sequence of word vectors $\{e(w_1), e(w_2), \ldots, e(w_s)\}$ is sent to the dense Bi-LSTM module as inputs.

(2) Dense Bi-LSTM
This module consists of multiple Bi-LSTM layers. For the first Bi-LSTM layer, the input is a word vector sequence $\{e(w_1), e(w_2), \ldots, e(w_s)\}$, and the output is $h^1 = \{h_1^1, h_2^1, \ldots, h_s^1\}$, in which $h_t^1 = [\overrightarrow{h_t^1}; \overleftarrow{h_t^1}]$ as described in Sect. 3.2. For the second Bi-LSTM layer, the input is not the sequence $\{h_1^1, h_2^1, \ldots, h_s^1\}$ (the way stacked RNNs use), but the concatenation of all previous outputs, formulated as $\{[e(w_1); h_1^1], [e(w_2); h_2^1], \ldots, [e(w_s); h_s^1]\}$, and the output is $h^2 = \{h_1^2, h_2^2, \ldots, h_s^2\}$. For the third layer, whose input is $\{[e(w_1); h_1^1; h_1^2], [e(w_2); h_2^1; h_2^2], \ldots, [e(w_s); h_s^1; h_s^2]\}$, like the second layer does. The rest layers process similarly and omitted for brevity. The above process is formulated as follows:

$$h_t^l = [\overrightarrow{h_t^l}; \overleftarrow{h_t^l}], \text{ specially, } h_t^0 = e(w_t), \tag{5}$$

$$\overrightarrow{h_t^l} = \text{lstm}(\overrightarrow{h_{t-1}^l}, M_t^{l-1}), \tag{6}$$

$$\overleftarrow{h_t^l} = \text{lstm}(\overleftarrow{h_{t+1}^l}, M_t^{l-1}), \tag{7}$$

$$M_t^{l-1} = [h_t^0; h_t^1; ...; h_t^{l-1}]. \tag{8}$$

(3) Average Pooling
For a L layer Dense Bi-LSTM, the output is $h^L = \{h_1^L, h_2^L, \ldots, h_s^L\}$. Average pooling module reads in h^L and calculate the average value of these vectors, the computation can be formulated as $h^* = \text{average}(h_1^L, h_2^L, \ldots, h_s^L)$.

(4) Soft-max Layer
This module is a simple soft-max classifier, which takes h^* as features and generates predicted probability distribution over all sentence labels.

2.3 Potential Application Scenario

From a semantic perspective, the dense Bi-LSTM module adds multi-read context information of each word into their original word vector in a concatenation way: h^1 is the first reading memory based on the input sentence S, h^2 is the second reading memory based on S and h^1, h^k is the k-th reading memory based on S and all previous reading memory. Since the word vector for each word is completely preserved, this module is harmless and can be easily added to other models that use RNN. For example, in the task of machine translation and dialog system, the Bi-LSTM encoder can be replaced by dense Bi-LSTM module and may bring improvements.

3 Experiments

3.1 Dataset

DC-Bi-LSTM are evaluated on several benchmark datasets. Movie Review Data(MR) is a popular sentiment classification dataset proposed by Pang and Lee 2005 [11]. Stanford Sentiment Treebank(SST-1) is an extension of MR [15]. And each review has fine-grained labels, moreover, phrase-level annotations on all inner nodes are provided. SST-2 is the same dataset as SST-1 but used in binary mode without neutral sentences. Subjectivity dataset(Subj) is from Pang and Lee 2004 [10], where the task is to classify a sentence as being subjective or objective. TREC is a dataset for question type classification task [9]. The sentences are questions from 6 classes.

3.2 Implementation Details

In the experiments, we use publicly available 300-dimensional Glove vectors, the number of hidden units of top Bi-LSTM (the last Bi-LSTM layer in dense Bi-LSTM module) is 100, for the rest layers of dense Bi-LSTM module, the number of hidden units and layers are 13 and 15 respectively.

For training details, we use the stochastic gradient descent (SGD) algorithm and Adam update rule with shuffled mini-batch. Batch size and learning rate are set to 200 and 0.005, respectively. As for regularization, dropout is applied for word embeddings and the output of average pooling, besides, we perform L2 constraints over the soft-max parameters.

3.3 Results

Results of DC-Bi-LSTM and other state-of-the-art models on five benchmark datasets are listed in Table 1. Performance is measured in accuracy. We can see that DC-Bi-LSTM gets consistently better results over other methods, specifically, DC-Bi-LSTM achieves new state-of-the-art results on three datasets (MR, SST-2 and Subj) and slightly lower accuracy than BLSTM-2DCNN on TREC and SST-1. In addition, we have the following observations:

- Although DC-Bi-LSTM is a simple sequence model, but it defeats Recursive Neural Networks models and Tree-LSTM, which relies on parsers to build tree-structured neural models.
- DC-Bi-LSTM obtains significant improvement over the counterparts (Bi-LSTM) and variant (LR-Bi-LSTM) that uses linguistic resources.
- DC-Bi-LSTM defeats all CNN models in all datasets.

Above observations demonstrate that DC-Bi-LSTM is quite effective compared with other models.

Table 1. Classification accuracy of DC-Bi-LSTM against other state-of-the-art models. The best result of each dataset is highlighted in **bold**. There are mainly five blocks: (i) traditional machine learning methods; (ii) Recursive Neural Networks models; (iii) Recurrent Neural Networks models; (iv) Convolutional Neural Net-works models; v) a collection of other models. **SVM**: Support Vector Machines with unigram features [15] **NB**: Na-ive Bayes with unigram features [15] **Standard-RNN**: Standard Recursive Neural Network [15] **RNTN**: Recursive Neural Tensor Network [15] **DRNN**: Deep Recursive Neural Network [6] **LSTM**: Standard Long Short-Term Memory Network [19] **Bi-LSTM**: Bidirectional LSTM [19] **Tree-LSTM**: Tree-Structured LSTM [19] **LR-Bi-LSTM**: Bidirectional LSTM with linguistically regularization [12] **CNN-MC**: Convolutional Neural Network with two channels [8] **DCNN**: Dynamic Convolutional Neural Network with k-max pooling [7] **MVCNN**: Multi-channel Variable-Size Convolution Neural Network [20] **DSCNN**: Dependency Sensitive Convolutional Neural Networks that use CNN to obtain the sentence representation based on the context representations from LSTM [22] **BLSTM-2DCNN**: Bidirectional LSTM with Two-dimensional Max Pooling [24].

Model	MR	SST-1	SST-2	Subj	TREC
SVM [15]	-	40.7	79.4	-	-
NB [15]	-	41.0	81.8	-	-
Standard-RNN [15]	-	43.2	82.4	-	-
RNTN [15]	-	45.7	85.4	-	-
DRNN [6]	-	49.8	86.6	-	-
LSTM [19]	-	46.4	84.9	-	-
Bi-LSTM [19]	81.8	49.1	87.5	93.0	93.6
Tree-LSTM [19]	-	51.0	88.0	-	-
LR-Bi-LSTM [12]	82.1	50.6	-	-	-
CNN-MC [8]	81.1	47.4	88.1	93.2	92.2
DCNN [7]	-	48.5	86.8	-	93.0
MVCNN [20]	-	49.6	89.4	93.9	-
DSCNN [22]	81.5	49.7	89.1	93.2	95.4
BLSTM-2DCNN [24]	82.3	**52.4**	89.5	94.0	**96.1**
DC-Bi-LSTM (**ours**)	**82.8**	51.9	**89.7**	**94.5**	95.6

3.4 Discussions

Moreover, we conducted some experiments to further explore DC-Bi-LSTM. For simplicity, we denote the number of hidden units of top Bi-LSTM (the last Bi-LSTM layer in dense Bi-LSTM module) as *th* , for the rest layers of dense Bi-LSTM module, the number of hidden units and layers are denoted as *dh* and *dl* respectively. We tried several variants of DC-Bi-LSTM with different *dh*, *dl* and *th*, The results are shown below.

(1) Better parameter efficiency

Better parameter efficiency means obtaining better performance with equal or fewer parameters. In order to verify DC-Bi-LSTM has better parameter efficiency than Bi-LSTM, we limit the number of parameters of all models at 1.44 million (1.44M) and conduct experiments on SST-1 and SST-2. The results are shown in Table 2.

Table 2. Classification accuracy of DC-Bi-LSTM with different hyper parameters. We limit the parameters of all models at 1.44M in order to verify DC-Bi-LSTM models have better parameter efficiency than Bi-LSTM.

dl	*dh*	*th*	Params	SST-1	SST-2
0	10	300	1.44M	49.2	87.2
5	40	100	1.44M	49.6	88.4
10	20	100	1.44M	51.0	88.5
15	13	100	1.40M	**51.9**	**89.7**
20	10	100	1.44M	50.2	88.8

The first model in Table 2 is actually Bi-LSTM with 300 hidden units, which is used as the baseline model, and the results are consistent with the paper [19]. Based on the results of Table 2, we get the following conclusions:

- DC-Bi-LSTM improves parameter efficiency. Pay attention to the second to the fifth model, compared with baseline model, the increase on SST-1(SST-2) are 0.4% (1.2%), 1.8% (1.3%), 2.7% (2.5%) and 1% (1.6%), respectively, with the parameters not increased, which demonstrates that DC-Bi-LSTM models have better parameter efficiency than base-line model
- DC-Bi-LSTM models are easy to train even when the they are very deep. We can see that DC-Bi-LSTM with depth of 20 (the fifth model in Table 3) can be successfully trained and gets better results than baseline model. In contrast, we trained deep stacked LSTM on SST-1, when depth reached more than five, the performance (For example, 30% when the depth is 8, which drops 19.2% compared with baseline model) drastically decreased.
- The fifth model performs worse than the fourth model, which indicates that too many layers will bring side effects when limiting the number of

parameters. One possible reason is that more layer lead to fewer hidden units (to ensure the same number of parameters), impairing the ability of each Bi-LSTM layer to capture contextual information.

(2) Effects of increasing depth (*dl*)

In order to verify that increasing *dl* does improve performance of DC-Bi-LSTM models, we increase *dl* gradually and fix *dh* at 10 . The results on SST-1 and SST-2 are shown in Table 3.

Table 3. Classification accuracy of DC-Bi-LSTM with different hyper parameters. We increase *dl* gradually and fix *dh* at 10 in order to verify that increasing *dl* does improve performance of DC-Bi-LSTM models.

dl	*dh*	*th*	Params	SST-1	SST-2
0	10	100	0.32M	48.5	87.5
5	10	100	0.54M	49.4	88.1
10	10	100	0.80M	49.5	88.4
15	10	100	1.10M	**50.6**	**88.8**
20	10	100	1.44M	50.2	**88.8**

The first model in Table 3 is actually Bi-LSTM with 100 hidden units, which is used as the baseline model. Based on the results of Table 3, we can get the following conclusions:

- It is obvious that the performance of DC-Bi-LSTM is positively related to *dl*. Compared with baseline model, DC-Bi-LSTM with *dl* equal to 5, 10, 15 and 20 get improvements on SST-1 (SST-2) by 0.9% (0.6%), 1.0% (0.9%), 2.1% (1.3%) and 1.7% (1.3%) respectively.
- Among all models, the model with *dl* equal to 15 works best. As *dl* continues to increase, the accuracy does not further improve, nevertheless, there is no significant decrease.

(3) Effects of adding hidden units (*dh*)

In this part, we explore the effect of *dh*. The number of layers in dense Bi-LSTM module (*dl*) is fixed at 10 while the number of hidden units (*dh*) is gradually increased. The results on SST-1 and SST-2 are shown in Table 4. Similarly, we use Bi-LSTM with 100 hidden units as baseline model (the first model in Table 4). Based on the results of Table 4, we can get the following conclusions:

- Comparing the first two models, we find that the second model outperforms baseline by 0.7% on SST-1 and 0.8% on SST-2, which shows that even if *dh* is equal to 5, DC-Bi-LSTM are still effective.
- As *dh* increases, the performance of DC-Bi-LSTM steadily increases. One possible reason is that the ability of each layer to capture contextual information is enhanced, which eventually leads to the improvement of classification accuracy.

Table 4. Classification accuracy of DC-Bi-LSTM with different hyper parameters. We increase *dh* gradually and fix *dl* at 10 in order to explore the effect of *dh* models.

dl	*dh*	*th*	Params	SST-1	SST-2
10	0	100	0.32M	48.5	87.5
10	5	100	0.54M	49.2	88.3
10	10	100	0.80M	49.5	88.4
10	15	100	1.10M	50.2	88.4
10	20	100	1.44M	**51.0**	**88.5**

4 Conclusion and Future Work

In this work, we propose a novel multi-layer RNN model called Densely Connected Bidirectional LSTM (DC-Bi-LSTM) for sentence classification tasks. DC-Bi-LSTM alleviates the problems of vanishing-gradient and overfitting and can be successfully trained when the networks are as deep as dozens of layers. We evaluate our proposed model on five benchmark datasets of sentence classification, experiments show that our model obtains significant improvements over the traditional Bi-LSTM and gets promising performance in comparison with the state-of-the-art approaches. As future work, we plan to apply DC-Bi-LSTM in the task of machine translation and dialog system to further improve their performance, for example, replace the Bi-LSTM encoder with dense Bi-LSTM module.

Acknowledgments. The work was supported by the Natural Science Foundation of China (No. 61672288), and the Natural Science Foundation of Jiangsu Province for Excellent Young Scholars (No. BK20160085).

References

1. El Hihi, S., Bengio, Y.: Hierarchical recurrent neural networks for long-term dependencies. In: NIPS (1996)
2. Graves, A., Mohamed, A.r., Hinton, G.: Speech recognition with deep recurrent neural networks. In: ICASSP (2013)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
5. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely connected convolutional networks. In: CVPR (2017)
6. Irsoy, O., Cardie, C.: Deep recursive neural networks for compositionality in language. In: NIPS (2014)
7. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
8. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)

9. Li, X., Roth, D.: Learning question classifiers. In: COLING (2002)
10. Pang, B., Lee, L.: A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In: ACL (2004)
11. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL (2005)
12. Qian, Q., Huang, M., Lei, J., Zhu, X.: Linguistically regularized LSTMs for sentiment classification. arXiv preprint arXiv:1611.03949 (2016)
13. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
14. Schmidhuber, J.: Learning complex, extended sequences using the principle of history compression. Neural Comput. **4**(2), 234–242 (1992)
15. Socher, R., et al.: Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP (2013)
16. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: NIPS (2015)
17. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR (2015)
18. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: CVPR (2016)
19. 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)
20. Yin, W., Schütze, H.: Multichannel variable-size convolution for sentence classification. arXiv preprint arXiv:1603.04513 (2016)
21. Yu, M., Yin, W., Hasan, K.S., dos Santos, C., Xiang, B., Zhou, B.: Improved neural relation detection for knowledge base question answering. arXiv preprint arXiv:1704.06194 (2017)
22. Zhang, R., Lee, H., Radev, D.: Dependency sensitive convolutional neural networks for modeling sentences and documents. arXiv preprint arXiv:1611.02361 (2016)
23. Zhang, Y., Chen, G., Yu, D., Yaco, K., Khudanpur, S., Glass, J.: Highway long short-term memory RNNs for distant speech recognition. In: ICASSP (2016)
24. Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B.: Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. arXiv preprint arXiv:1611.06639 (2016)

An End-to-End Scalable Iterative Sequence Tagging with Multi-Task Learning

Lin Gui[1,2], Jiachen Du[1], Zhishan Zhao[3], Yulan He[2], Ruifeng Xu[1(✉)], and Chuang Fan[1]

[1] Harbin Institute of Technology (Shenzhen), Shenzhen, China
xuruifeng@hit.edu.cn
[2] Aston University, Birmingham, UK
[3] Baidu Inc., Beijing, China

Abstract. Multi-task learning (MTL) models, which pool examples arisen out of several tasks, have achieved remarkable results in language processing. However, multi-task learning is not always effective when compared with the single-task methods in sequence tagging. One possible reason is that existing methods to multi-task sequence tagging often reply on lower layer parameter sharing to connect different tasks. The lack of interactions between different tasks results in limited performance improvement. In this paper, we propose a novel multi-task learning architecture which could iteratively utilize the prediction results of each task explicitly. We train our model for part-of-speech (POS) tagging, chunking and named entity recognition (NER) tasks simultaneously. Experimental results show that without any task-specific features, our model obtains the state-of-the-art performance on both chunking and NER.

Keywords: Multi-task learning · Interactions · Sequence tagging

1 Introduction

Sequence tagging is one of the most important topics in Natural Language Processing (NLP), encompassing tasks such as part-of-speech tagging (POS), chunking, and named entity recognition (NER). In recently years, neural network (NN) based models have achieved impressive results on various sequence tagging tasks, including POS tagging [1,2], chunking [3,4], and NER [5,6].

One of the challenges for sequence tagging tasks is that there is not enough training data to train a good model. Heavy handcrafted features and language-specific knowledge resources are costly to develop in new sequence tagging tasks [5]. To overcome this problem, multi-task learning (MTL) models have been proposed.

MTL is an important mechanism that aims to improve the generalization of model performance by learning a task together with other related tasks [7]. Several NN based MLT models have been applied to various sequence tagging tasks

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 288–298, 2018.
https://doi.org/10.1007/978-3-319-99501-4_25

[4,8,9]. These models use shared representations constructed by some lower layers, then stack several task-specific upper layers to learn task-relevant representations. However, representation sharing in lower NN layers only captures weak interactions between different tasks. That perhaps one reason of that multi-task learning is not always effective compared with single-task methods in sequence tagging [10].

To improve MTL performance, we propose a unified multi-task learning framework for sequence tagging which could iterative utilize predicted tags of each task explicitly. Essentially, our model predicts tags of all tasks with several iterations. The learned distributions of task-specific prediction in every iteration are merged with the shared representations for tag sequence prediction in the next iteration. The iterative training mechanism gives the model capability to refine prediction results of one task by simultaneously considering prediction results of other tasks.

In particular, we propose a CNNs-Highway-BLSTM as base model for sequence tagging. Character-level and word-level convolutional neural networks (CNNs) are used to capture morphological features and a Highway network is implemented to keep valuable features by an adaptive gating units. These features are fed into a Bidirectional Long Short-term Memory Network (BLSTM) for task-specific prediction. According to our iterative training mechanism, the distribution of prediction can be concatenated with shared representations from Highway layer output and sent into BLSTM in the next iteration. The experimental results show that our base model outperforms the state-of-the-art methods on Chunking and NER with iterative training process.
The main contributions of this paper are:

– We propose a new framework to explicitly make use of predicted tags from one task as additional features for other tasks and hence better capture interactions between different tasks in multi-task learning.
– We proposed a CNNs-Highway-BLSTM as a base model for sequence tagging.
– We evaluate our model on several benchmarking datasets and achieve the state-of-art performance on both chunking and NER.

2 Our Approach

In this section, we will first describe a general framework of multi-task learning for sequence tagging and introduce our idea of iterative training. We will then propose a base sequence tagging model which includes lower layers of character-level CNN, word-level-CNN, Highway network and BLSTM that shared across all tasks. At the end of this section, we will present the details of model training.

2.1 Multi-task Learning

Multi-task Learning (MTL) aims to improve model generalization by learning multiple tasks simultaneously from shared representations based on the assumption that features learned for each task can be helpful for other tasks. In addition, pooling samples of several tasks could also potentially generalize model well [7]. As we focus on sequence tagging here, both inputs and outputs are in the vector form. Given an input sequence $\boldsymbol{x}$, the MLT model outputs a tagging sequence $\overrightarrow{y^{(i)}}$ for different tasks. The model typically consists of two kinds of parameters: task-specific parameters $h^{(i)}$, which are in the upper layer of the architecture shown in Fig. 1, and the parameters $h^{(shared)}$ shared across all tasks, which are in the lower layer of the architecture shown in Fig. 1.

In MTL, it is reasonable to assume that tags predicted for one task can be useful features for other tasks. For instance, a word with POS tag "noun" is more likely to be a named entity compared to others tagged as cardinal numeral. Conventionally, lots of NLP systems use features obtained from the output of other preexisting NLP systems [4].

In order to explicitly use the tag prediction results from one task in others in MTL, we propose an iterative training procedure for MTL as shown in Fig. 1, in which steps enclosed in a dashed box are repeated for a number of iterations. The inputs to each iteration consist of:

1. $h^{(shared)}$, features extracted from data by a shared layer;
2. $\boldsymbol{y} = [\overrightarrow{y^{(1)}}, \overrightarrow{y^{(2)}}]$, concatenated prediction probabilities of all tasks from the previous iteration.

Both inputs are concatenated and fed to BLSTM. It is worth noting that BLSTM is shared across all tasks and it is separated from $h^{(shared)}$ because it participates in all iterations. Our proposed iterative training procedure (Fig. 1) allows the model to incorporate the predicted results of all tasks as additional features in the next iteration. With BLSTM, the model extends tags interaction to the sentence level. To ensure that the predicted results at each iteration is close to the true tag sequence, we define a cost function by taking into account the differences between the predicted and true tag sequences in all iterations and for all tasks:

$$cost = \frac{1}{T}\sum_{i=1}^{T} L(\overrightarrow{y_t}, \overrightarrow{y*}), \tag{1}$$

$$L(\overrightarrow{y_t}, \overrightarrow{y*}) = \frac{1}{M}\sum_{m=1}^{M} \alpha_m \tilde{L}(\overrightarrow{y_t^{(m)}}, \overrightarrow{y^{(m)*}}) \tag{2}$$

where $\overrightarrow{y_t^{(m)}}$ is the prediction probabilities sequence of task m at iteration t, $\overrightarrow{y^{(m)*}}$ is the ground-truth label sequence of task m, T is the number of iterations, M is the number of tasks, α_m is the weight of different tasks, $\tilde{L}$ is the cross entropy function. For task m, the final prediction result is a mean of predictions across all iterations:

$$\overrightarrow{y^{(m)}} = \frac{1}{T}\sum_{i=1}^{T} \overrightarrow{y_t^{(m)}} \tag{3}$$

2.2 Basic Sequence Tagging Model

In the previous subsection, we have introduced the iterative training procedure and the cost function used in MTL. Since the predicted tag sequence will be used as additional features in next iteration of model training, a strong base sequence tagging model is needed for obtaining better performance. In this section, we will propose a NN-based sequence tagging model, which is called CNNs-Highway-BLSTM. It starts with character-level and word-level CNNs to capture morphological and contextual features. Next, a Highway network is used to keep valuable features across word-level CNNs by using *transform* and *carry* gates. These features are input into a BLSTM for capturing sequence long-term dependencies. The output of BLSTM is fed into different task-specific output layers.

Word-Level CNN. The word-level CNN takes an input sentence, called *Word Representation*, as a sequence of words $x = [x_1, x_2, ..., x_n]$ where each word is represented as a d-dimensional vector, and returns another sequence $S = [s_1, s_2, ..., s_n]$ which represents local information about the sequence at every word of the input. A narrow convolution is applied between x and a kernel $W \in \mathbb{R}^{kd}$ of width k. $\lfloor \frac{k}{2} \rfloor$ and $\lfloor \frac{k-1}{2} \rfloor$ padding vectors are added to the head and tail of sequence to make sure the sequence length does not change after the convolution layer.

Character-Level CNN. Character-level CNNs have been shown to be an effective method to extract morphological features from characters of words [2,11]. Given a word, we first apply its character embedding to a CNN layer and then take the max-over-time pooling operation [4] to capture the most important features for each feature map. The vector output from the max-over-time pooling layer is the character-level representation of the word, called *Character Representation*. This representation is then concatenated with the word embedding as the input to word-level CNNs.

Highway Network. In our experiments, simply stacking multiplayer word-level CNNs makes the performance worse. Instead, we implement a highway network [12] after the CNN layer to keep valuable features across word-level CNNs. Highway layer allows part of s_i to be carried unchanged to the output while the reset to go through convolutional transformations.

BLSTM. In the next step, the output of highway network will be obtained for the input of a Long Short-term Memory Network (LSTM) [13], which was proposed to address this issue of learning long-term dependencies by maintains three multiplicative gates which control the information to forget and to pass on to the next step. In sequence tagging task, we apply a Bidirectional LSTM (BLSTM) to make use of past (left) and future (right) information by present each sequence forwards and backwards to two separate hidden states. The two hidden states are then concatenated to form the final output.

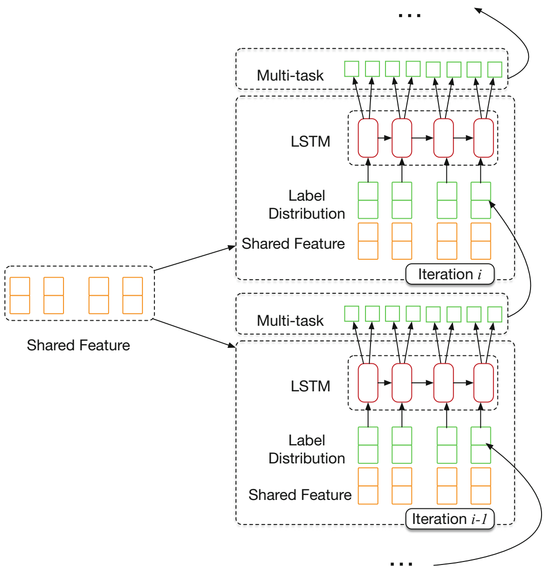


Fig. 1. MTL with recurrent iterative framework.

2.3 The Final Model

Our final MTL architecture is illustrated in Fig. 1. When iterative training begins, we initialize prediction probabilities of all tasks with zero vectors, as input of BLSTM for a unified form. It is worth noting that parameters in each iteration are not shared. Following previous work [4], our task-specific hidden layer $h^{(i)}$ only includes task-specific fully connected output layers.

3 Experiment

3.1 Experimental Setup

We test our model on three NLP sequence tagging tasks: POS tagging on Penn TreeBank (PTB), chunking on CoNLL 2000 and named entity recognition (NER) on CoNLL 2003.

POS assign a unique tag to each word, which indicate its syntactic role, such as noun, verb and so on. Chunking, or shallow parsing, assigns each word with its phrase type. NER labels each word into other or four categories: Person, Location, Organization, or Miscellaneous. We use the BIOES tagging scheme for chunking and NER tasks. For the preprocessing, we follows previous work [5].

In the training of model, we fine turn the parameter on the validation dataset. For the chunking task on CoNLL 2000 data, we split 10% training data for validation because the benchmark only provide training data. Since the test data for chunking in the CoNLL 2000 shared task is part of the validation data in the standard POS, we simply remove part of repeat samples from validation data of POS task. The size of training data we used in our multi-task learning is smaller than the standard task.

All the experimental result reported in our paper is with p-value less than 0.01 by t-test. For the metrics in evaluation, we follow the standard metrics on three benchmark respectively. In detail, for the POS tagging task, we use **Accuracy** to evaluate the performance. For the chunking and NER tasks, we evaluate the result by **F1-measure**.

3.2 Experimental Results

In this section, we first explore the effectiveness of recurrent iteration mechanism on several experiments. Then we report the results of our model on the benchmark and compare to the previously-reported state-of-art results.

Table 1. The impact of recurrent iteration.

Length	POS	Chunking	NER
T=0	97.54	95.31	90.52
T=1	**97.58**	95.47	91.16
T=2	97.56	95.54	**91.31**
T=3	97.56	**95.58**	91.30
T=4	97.55	**95.58**	91.26
T=5	97.55	95.57	91.24

Impact of Iterative Training. In this section, we first explore the effectiveness of our proposed iterative training procedure. Then we report the results of our model on the benchmarking datasets in comparison to previously reported state-of-the-art results.

Table1 shows how the results with the increasing number of iterations for the iterative training of our proposed base sequence tagging model. When iterative training just once (T=0), the model is common form MTL model as illustrate in Fig. 1. It can be observed that iterative training has little impact on POS tagging, but it improves the performance of both chunking and NER significantly, especially in the first two iteration. For computational efficiency, we set the number of iteration to 3 in the later experiment.

To understand the effect of the predicted tag sequences from one task on the others, we drop tag prediction results from one task at a time during the iterative

Table 2. The effect of excluding the predicted tag sequences from one task at a time during iterative training.

Models	POS	Chunking	NER
Full	**97.56**	**95.58**	**91.30**
Without POS	97.52	95.41	90.70
Without Chunking	97.53	95.51	90.93
Without NER	97.52	95.52	90.75

training process (i.e. the input of BLSTM at each iteration only includes the prediction results of two tasks and $h^{(shared)}$). From the results presented in Table 2, we find that POS tags have more noticeable influence on NER and chunking. Chunking tags also have some impact on NER, but do not influence much on POS tagging. NER tags do not contribute much to POS tagging or chunking. We also notice that NER tag sequences from previous training iteration are helpful for tagging prediction in the next iteration.

Comparison with Existing Systems. We first compare our method with the previous multi-task learning method [4], which used CNNs as base model, in Table 3. It can be observed that with our proposed base sequence tagging model and the iterative training process, our approach outperforms the method in [4] significantly on NER, noticeably on chunking and slightly on POS tagging.

Table 3. Compare with existing multi-task learning method.

Models	POS	Chunking	NER
Collobert et al., 2011	97.22	94.10	88.62
Our approach	**97.56**	**95.58**	**91.30**

We also compare our results with methods developed specifically for POS tagging, chunking and NER in Table 5, respectively. For POS tagging, the best performing model is the one proposed in [14] where an accuracy of 97.78% was achieved by using character-level and word-level BLSTM model. The word embeddings used in their model are trained by themselves and are not publicly available. [1] achieved an accuracy of 97.55% by using CNN as the character-level model and BLSTM-CRF as the word-level model. Our model slightly outperforms [1] demonstrating the effectiveness of using CNN-Highway for modeling word-level local features. Overall, our model outperforms all the other systems apart from [14], including the ones using handcrafted features. For chunking methods, [15] won the CoNLL2000 challenge with an F1-score of 93.48% by using SVMs. The previous state-of-the-art F1-score of 95.23% was reported in [16] by using a voting classifier scheme with carefully handcrafted features.

Table 4. Compare with existing POS tagging methods, Chunking Methods and NER Methods. (The methods with handcraft features have been marked with †.)

Methods	POS Tagging	Chunking	NER
Sha adn Pereira, 2003	-	94.30	-
Shen et al., 2005 †	-	95.23	-
Collobert et al., 2011	97.29	94.32	89.59
Lample et al., 2015	**97.78**	-	90.94
Ma et al., 2016	97.55	-	91.21
Huang et al., 2015	97.55	-	90.10
Our approach	97.56	**95.58**	**91.30**

Our multi-task learning model outperforms all the existing methods on chunking, partly attributing to the additional training data from POS tagging and NER tasks and also common features extracted by the shared lower layer.

For NER methods, with the same data pre-processing as ours, [5] obtained an F1-measure of 90.94% by using the character-level BLSTM and word-level BLSTM-CRF model. [17] achieved an F1-score of 91.20% using the joint NER and entity linking model with heavily handcrafted features. With our proposed iterative training process, our model improves upon [17] by 0.10% in F1-score. To the best of our knowledge, the previous best result of 91.21% was reported in [1] with a BLSTM-CNNs-CRF model. Our model further boosts the performance by 0.09% and achieves the state-of-art performance.

3.3 Case Study

To validate our assumption of iterative training, we choose an example in test set of chunking task to show the iterative tagging results in Table 5:
Ex.1 *The Canadian Wheat Board reported six ships loading*

In Table 6, the top row illustrates the ground truth of POS and Chunk labels, and the next three rows show the iterative tagging results in validation phase. We can obverse that, without iterative training, our model is not able to predict the right chunking label of word "*loading*". However, the POS label of "*loading*" is correctly predicted and it is used to help correct the label of chunking when T=1. In second iteration, by leveraging the sequential property of text, our model also replaces the chunking label of "*ships*" from "*E-NP*" to "*I-NP*". We also find that, with increasing number of iterations, the tagging results of our model just remain stable.

4 Related Work

In recently years, a number of neural architectures have been proposed for sequence tagging. [3] proposed a BLSTM-CRF model for sequence tagging

Table 5. Tagging results of different iterations

	Tag	Reported	Six	Ships	Loading
Ground	POS	VBD	CD	NNS	NN
	Chunk	S-VP	B-NP	I-NP	E-NP
T=0	POS	VBD	CD	NNS	NN
	Chunk	S-VP	B-NP	E-NP	S-VP
T=1	POS	VBD	CD	NNS	NN
	Chunk	S-VP	B-NP	E-NP	E-NP
T=2	POS	VBD	CD	NNS	NN
	Chunk	S-VP	B-NP	I-NP	E-NP
T=3	POS	VBD	CD	NNS	NN
	Chunk	S-VP	B-NP	I-NP	E-NP
T=4	POS	VBD	CD	NNS	NN
	Chunk	S-VP	B-NP	I-NP	E-NP

tasks. The experimental results on POS tagging, chunking and NER tasks have obtained impressive results. However, the handcrafted features they used make the model less flexible. [6] proposed a hybrid of BLSTM and CNNs for NER based on character-type, capitalization and lexicon features. To reduce handcrafted features, [2] proposed CharWNN, which stacks a convolutional layer to capture word morphology and shape features. It obtains the state-of-the-art accuracy on English POS tagging. More recently, [5] proposed a BLSTM-CRF model for NER based on character-level and word-level information. [1] proposed a LSTM-CNNs-CRF model for POS tagging and NER, which included character-level CNN layers and word-level LSTM-CRF layers. Both [1,5] utilized neural network that learns character-level representation of words instead of using handcrafted features. All models mentioned above are trained on a single task and can not leveraged datasets of related tasks.

More recently, multi-task learning has been applied to various sequence tagging tasks, including name error recognition [8], POS [18]. Most of these models use shared representation constructed by lower layers and task-specific representation constructed by upper layers. [4] utilized multi-task learning for POS, Chunking, and NER joint tagging. However, this work did not consider the predicted tagging sequences from each individual task and the base model is a simple CNNs, which limited the performance. [19] jointly trained co-reference resolution, entity linking, and NER using a single CRF model with added cross-task interaction factors, which also could capture interaction of tags between related tasks. However, jointly training a CRF model requires all tasks having fully labeled training data and the model can not leverage exiting partial labeled data for training.

5 Conclusion

In this paper, we have presented a new multi-task learning network architecture for sequence tagging. In this method, CNNs was used to model both character-level and word-level representations, and BLSTM was used to capture long range dependencies. We also utilize a Highway network to further improve the model performance. A main contribution of our proposed framework is that it can leverage the predicted tagging sequences between related tasks through the iterative training procedure. Our model achieves the state-of-the-art performance on chunking and NER, and performs comparably to the best performing model on POS tagging, without using any external knowledge or handcrafted features. Since our model does not require any domain- or task-specific knowledge, it can be applied to other sequence tagging tasks, which will be explored in our future work.

Acknowledgement. This work was supported by the National Natural Science Foundation of China U1636103, 61632011, Shenzhen Foundational Research Funding 20170307150024907, Key Technologies Research and Development Program of Shenzhen JSGG20170817140856618.

References

1. Ma, X., Hovy, E.H.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In: ACL, pp. 1064–1074 (2016)
2. dos Santos, C.N., Zadrozny, B.: Learning character-level representations for part-of-speech tagging. In: ICML, pp. 1818–1826 (2014)
3. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991 (2015)
4. Collobert, R., et al.: Natural language processing (almost) from scratch. JMLR **12**, 2493–2537 (2011)
5. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: NAACL, pp. 260–270 (2016)
6. Chiu, J.P.C., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNS. TACL **4**, 357–370 (2016)
7. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
8. Cheng, H., Fang, H., Ostendorf, M.: Open-domain name error detection using a multi-task RNN. In: EMNLP, pp. 737–746 (2015)
9. Søgaard, A., Goldberg, Y.: Deep multi-task learning with low level tasks supervised at lower layers. In: ACL(2), pp. 231–235 (2016)
10. Alonso, H.M., Plank, B.: When is multitask learning effective? Semantic sequence predictionunder varying data conditions. In: EACL, pp. 44–53 (2017)
11. Kim, Y., Jernite, Y., Sontag, D., Rush, A.M.: Character-aware neural language models. In: AAAI, pp. 2741–2749 (2016)
12. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: NIPS, pp. 2377–2385 (2015)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

14. Ling, W., Dyer, C., Black, A.W., Trancoso, I., Fermandez, R., Amir, S., Marujo, L., Luís, T.: Finding function in form: compositional character models for open vocabulary word representation. In: EMNLP, pp. 1520–1530 (2015)
15. Kudoh, T., Matsumoto, Y.: Use of support vector learning for chunk identification. In: CoNLL, pp. 142–144 (2000)
16. Shen, H., Sarkar, A.: Voting between multiple data representations for text chunking. In: Kégl, B., Lapalme, G. (eds.) AI 2005. LNCS (LNAI), vol. 3501, pp. 389–400. Springer, Heidelberg (2005). https://doi.org/10.1007/11424918_40
17. Luo, G., Huang, X., Lin, C.-Y., Nie, Z.: Joint entity recognition and disambiguation. In: EMNLP, pp. 879–888 (2015)
18. Plank, B., Søgaard, A., Goldberg, Y.: Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In: ACL(2), pp. 412–418 (2016)
19. Durrett, G., Klein, D.: A joint model for entity analysis: coreference, typing, and linking. TACL **2**, 477–490 (2014)

A Comparable Study on Model Averaging, Ensembling and Reranking in NMT

Yuchen Liu[1], Long Zhou[1], Yining Wang[1], Yang Zhao[1], Jiajun Zhang[1], and Chengqing Zong[1,2](✉)

[1] National Laboratory of Pattern Recognition, CASIA, University of Chinese Academy of Sciences, Beijing, China
[2] CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China
{yuchen.liu,long.zhou,yining.wang,yang.zhao,jjzhang,cqzong}@nlpr.ia.ac.cn

Abstract. Neural machine translation has become a benchmark method in machine translation. Many novel structures and methods have been proposed to improve the translation quality. However, it is difficult to train and turn parameters. In this paper, we focus on decoding techniques that boost translation performance by utilizing existing models. We address the problem from three aspects—parameter, word and sentence level, corresponding to checkpoint averaging, model ensembling and candidates reranking which all do not need to retrain the model. Experimental results have shown that the proposed decoding approaches can significantly improve the performance over baseline model.

1 Introduction

Neural machine translation (NMT) has significantly improved the quality of machine translation in recent years, which has shown promising results on multiple language pairs [1,4,7,18,21]. It builds upon a single and large neural network directly mapping source sentence to associated target sentence. Recently relying entirely on an attention mechanism, the transformer model introduced by Vaswani et al. [21] achieved state-of-the-art results for machine translation.

However, designing a novel and good translation model is a tough work. Training model is also time-consuming and occupies massive computing resources. For example, despite its remarkable success, transformer requires 3.5 days with 8 GPUs on training for a big model. Thus instead of modifying model structure, how to make effective use of the existing models to improve the translation performance is well worth considering.

In this paper, we investigate and practice decoding approaches to boost translation performance without training model again. As shown in Fig. 1, we address the problem from three aspects—parameter, word and sentence level. First, we adapt the checkpoint averaging to get a more robust set of parameters, which can utilize multiple checkpoints saved at different timesteps in a single model.

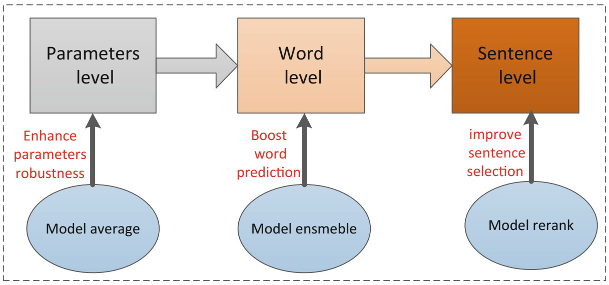


Fig. 1. Our framework for decoding approaches from three aspects: parameters, word and sentence level.

For word level, we introduce three ensembling strategies to boost word prediction, including checkpoint ensemble, independent ensemble and different ensemble. Finally, we observe that in validation set if each translation candidate with top sentence-level BLEU score is selected, we can obtain over 18 BLEU points improvement compared with outputs by beam search algorithm. Inspired by this observation, we attempt to select better candidates by reranking techniques, which includes linear regression, pairwise-rank method and minimum bayes risk (MBR) decoding.

We conducted massive experiments on large-scale English-to-Chinese translation. Experimental results have shown that decoding approaches can obtain significant BLEU score improvements over state-of-the-art NMT baseline.

2 Neural Machine Translation

The decoding approaches we discuss can be applied to any neural machine translation model. Here we choose Transformer [21] as baseline model for later experiments. In this section, we will give a brief introduction of Transformer encoder-decoder framework.

Given a set of bilingual data $D = \{(X^{(i)}, Y^{i)})\}_{i=1}^{N}$ where both X and Y are a sequence of tokens, the encoder maps a input sequence $X = (x_1, x_2, \cdots, x_n)$ to a sequence of continuous representations $z = (z_1, z_2, \cdots, z_n)$ whose size varies with respect to the source sentence length. The decoder generates an output sequence $Y = (y_1, y_2, \cdots, y_m)$ from the continuous representations. The encoder and decoder are trained jointly to maximize the conditional probability of target sequence given a source sequence:

$$P(Y|X;\theta) = \prod_{j=1}^{N} P(y_j|y_{<j}, x; \theta) \tag{1}$$

Transformer consists of N stacked encoder and decoder layers. Encoder layer consists of two blocks, which is self-attention block followed by a position-wise feed-forward block. Decoder layer has the same architecture as encoder layer except an extra encoder-decoder attention block. Residual connection and layer normalization are used around each block.

For self-attention and encoder-decoder attention, a multi-head attention block is used to obtain information from different representation subspaces at different positions. Each head corresponds to a scaled dot-product attention, which operates on a query Q, key K and a value V:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

where d_k is the dimension of the key.

For the sake of brevity, we refer readers to Vaswani et al. [21] for additional details regarding the architecture.

3 Methods Description

Many approaches have been proposed to generate better translation results in decoding [11–13,17,19]. In this section, we introduce three decoding techniques that boost translation performance by utilizing existing models, which includes checkpoint averaging, ensembling strategies and different methods of reranking.

3.1 Checkpoint Averaging

Checkpoint averaging is to average trainable parameters which are saved at last timesteps in a single model, when the model is near convergence. Since we use Stochastic Gradient Descent algorithm to optimize model, only a mini-batch of data are used during each step, causing that the parameters may over adapt to one mini-batch. We can get more robust parameters by checkpoint averaging. As illustrated in Fig. 2, for example, the value in the red circle of second checkpoint is 0.30, distinct from the corresponding values of other two checkpoints, which means that it may be a noise value. After checkpoint averaging it turns to 0.40, thus a more proper value can be obtained. Vaswani [21] suggested to average the last 20 checkpoints saved every 10 min. Here we will make experiments on how many checkpoints to be chosen can obtain the best result.

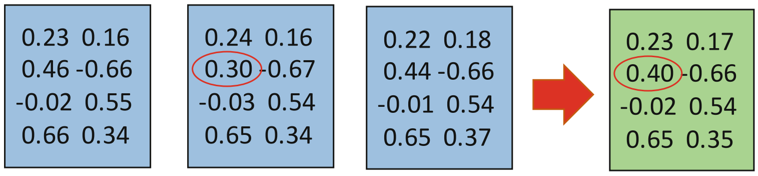


Fig. 2. Model averaging with the number of three (Color figure online)

3.2 Different Ensembling Strategies

Model ensembling is a method to integrate the probability distributions of multiple models before predicting next target word. It has been proved effective in neural machine translation [12,13,26]. We apply three different ensembling strategies as following:

- **checkpoint ensemble.** We use the checkpoints saved at different times in a single training model, which do not need to train several models from scratch. It is a cheap way to obtain an ensemble model.
- **independent ensemble.** This strategy needs to train N models independently with the same architecture but different initialization ways. Combining N models in different initialization ways as an ensemble model can help to avoid local optimization and obtain better results.
- **different ensemble.** Different ensemble trains N models with both different architectures and initialization ways, which is expensive but can yield better and more diverse result.

3.3 Different Reranking Strategies

Reranking is a long-term study in machine translation [11,15,17,19]. In this paper, we apply three reranking strategies to investigate how to better select candidates, which includes linear regression, pairwise-rank method and MBR decoding.

Linear Regression. Linear regression is usually used to model the relationship between a dependent variable and one or more explanatory variables. We denotes sentence-level BLEU score as dependent variable, sentence-level features as explanatory variables. More specifically, we use beam search algorithm to obtain a list of candidate translations. Since validation set has reference, the sentence-level BLEU score of each sentence can be calculated. We then use target side right-to-left model, target-to-source model, n-gram language model, neural language model and SMT model trained by Moses[1] to calculate each sentence's feature scores. We first fit a linear regression model for validation set, then adapt it to test set. For test sentences we estimate the BLEU scores of each candidate, then select the sentence with top score as final output.

Pairwise Rank. In fact, we do not need to know the exact BLEU score of each sentence. The only thing we concern is the order of translation candidates. One of approaches is to reduce the ranking problem as a classification problem by using pairwise sampling [6]. However, ranks may be unreliable for machine translation where candidates sometimes can not be strongly distinguished between each other. To alleviate this problem, we assume the difference among top r translation candidates is not obvious. Thus, we regard the top r of the n-best

[1] http://www.statmt.org/moses/.

candidates as good translations and the bottom k as bad translations, where $r + k \geq n$. Then a classification model is trained to split the good translations from the bad translations for each sentence. Besides, the following criteria are proposed:

- We assign a larger margin for the candidates whose ranks are far and a smaller margin for closer pairs. For example, $margin(\mathbf{e}_1, \mathbf{e}_{20}) > margin(\mathbf{e}_1, \mathbf{e}_{10})$
- For the same rank gap, the margin between a high rank and a low rank is larger than that between two low ranks. For example, $margin(\mathbf{e}_1, \mathbf{e}_{10}) > margin(\mathbf{e}_{21}, \mathbf{e}_{30})$. The reason is that the scoring function will be penalized if it can not separate former case, but not for the latter.

MBR Decoding. MBR decoding is a method to find a candidate with the least expected loss [19]. It measures the similarity of each candidate translation instead of the quality against reference. The Bayes risk of each candidate y is computed by:

$$R(y) = \sum_{y' \in E} \Delta(y, y') p(y'|x) \tag{3}$$

The term $\Delta(y, y')$ is calculated by $1 - BLEU(y, y')$, which denotes the discrepancy between candidate y and candidate y'. The term $p(y'|x)$ is the generating probability of each candidate given by a NMT model. The candidate with lowest Bayes risk means that it is similar to the most candidates in the evidence space.

4 Experiments

4.1 Dataset

We perform our experiments on corpus provided by AI Challenger—the English-Chinese Machine Translation track[2]. This corpus contains about 10 million parallel English-Chinese sentences which are collected from English learning websites and movie subtitles. In our experiments, we first filter the bilingual corpus according to the following criteria:

- Sentences which contain less than 3 words or more than 100 words are removed.
- We use fast_align toolkit to learn a word alignment of sentences pairs. Sentence pairs whose alignment ratio is lower than 0.3 are removed.
- We sort sentence pairs by perplexities and remove the bottom 5 percent sentence pairs.

After filtering we retain about 9 million pairs of training data. We also use monolingual sentences to train n-gram and neural language model for later use in reranking. The English side is preprocessed by tokenizer and lowercaser scripts in Moses. The Chinese side is segmented by our in-house toolkit. We learn a BPE [14] model with 80k merge operations for both English side and Chinese side, and extract 83k and 78k subwords as source and target vocabularies. The evaluation metric is BLEU as calculated by the `multi-blue.perl` script.

[2] https://challenger.ai/competition/translation.

4.2 Training Details

We adopt the Transformer model as our baseline model[3]. All hyper parameter settings are set the same as transformer_big_single_gpu if not specifically mentioned. The dimension of word embedding is set to 1024. The size of attention block is 1024 with 16 heads and the size of feed forward block is set to 4096.

We train the baseline model for a total of 300K steps with Adam optimizer [8] on three GPUs. We set the initial learning rate to 0.1 and apply decay method described in Vaswani et al. [21]. Dropout was applied on residual layer to avoid over-fitting, which is 0.1. At test time, we employ beam search with beam size 4. As for reranking, we set beam size as 50 to generate a list of candidates.

4.3 Results and Comparison on Different Approaches

In this section, we first separately compare different strategies in each approach. Then we report the final results by combining all these approaches. All of the first lines in every table are the baseline model without using any decoding approaches.

Effects on the Number of Checkpoints for Averaging. We first test how many checkpoints to be used can get better results. Figure 3 shows the effect on the number of checkpoints for averaging. We obverse that the BLEU scores by applying checkpoint averaging are all above baseline model. As the number of checkpoints increases, the set of parameters becomes more robust which brings more BLEU scores improvement. However too many checkpoints may contaminate parameters with scores decreasing. In our experiments, averaging 40 checkpoints saved in ten-minute interval obtains the best results. Compared with the time needed for training the whole model from scratch, checkpoint averaging only takes a few minutes before decoding, so it is a free way to boost.

Comparison of Different Ensembling Strategies. Table 1 shows the BLEU scores with different ensembling strategies. To be fair, all the three ensembling strategies use five models for ensembling. Specifically, checkpoint ensemble uses checkpoints saved at five different time during a single model training process; independent ensemble uses two models initialized by uniform algorithm, two by normal algorithm and one by orthogonal; different ensemble uses two 6-layer encoder-decoder structures with uniform and normal initializer, two 8-layer structures with 24 heads, 960 hidden size and 32 heads, 1024 hidden size respectively, and one 10-layer encoder with 6-layer decoder structure. Model ensembling can indeed boost translation result, while the improvements by three ensembling strategies are different. Among these strategies, different ensemble outperforms the other two strategies by about 1.2 and 0.6 points respectively. This strategy is expensive for training but more likely to yield better and more diverse results. This is consistent with the wisdom that there is no free lunch.

[3] https://github.com/tensorflow/tensor2tensor.

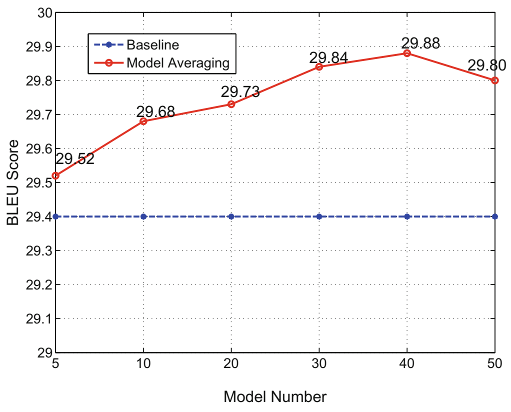


Fig. 3. Translation results (BLEU score) on checkpoint averaging.

Table 1. Translation results (BLEU score) on different ensembling strategies.

System	BLEU
Baseline	29.40
Checkpoint ensemble	29.55(+0.15)
Independent ensemble	30.18(+0.78)
Different ensemble	30.76(+1.36)

Comparison of Different Reranking Strategies. In our experiment, we first generate 50-list of translation candidates by beam search algorithm, then calculate sentence-level BLEU for each candidate on validation set. To our surprise, we observe a remarkable gap between the output generated by beam search algorithm and the candidate selected by sentence-level BLEU. As illustrated in Fig. 4, if we select each candidate with top 1 sentence-level BLEU, the corpus-level BLEU can reach 47.50; even if we select all the candidates with 20th sentence-level BLEU, we can also get a final BLEU of 29.77. However, the BLEU score of the output generated by beam search algorithm is only 29.40. Thus we think reranking has a great potential to further improve translation performance. We then attempt three reranking strategies. From Table 2, we observe that linear regression and MBR decoding can significantly improve the BLEU scores, while pairwise rank cannot well distinguish between the good translation candidate with the bad one. One possible reason is that the selected sentence-level features may not linearly separable. Though reranking by linear regression obtains an improvement, it heavily relies on the selected features which are expensive to extract. From this perspective, MBR decoding is the best strategy since all it needs is just a list of translation candidates. Although these strategies can improve the BLEU scores, it is still far away from the best result. We will further investigate this challenge in the future.

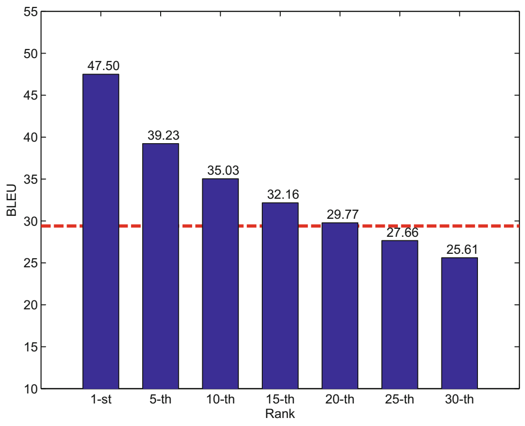


Fig. 4. Translation results (BLEU score) based on sentence-level BLEU.

Table 2. Translation results (BLEU score) on different reranking strategies.

System	BLEU
Baseline	29.40
Linear regression	30.10(+0.70)
Pairwise rank	29.88(+0.44)
MBR decoding	30.13(+0.73)

Results Combining Three Approaches. We choose the best strategy in each approach mentioned above, which are the averaging of 40 checkpoints, different ensemble and MBR decoding respectively. Combining all these approaches, we can obtain significant BLEU score improvements over baseline model. We list the BLEU scores of our proposed model in Table 3. Specifically, after checkpoints averaging, we can get an improvement of 0.54 BLEU points over baseline. We further obtain 0.99 BLEU points improvement with different ensembling strategies. By applying reranking method, another improvement of 0.69 BLEU scores can be achieved. It confirms the effectiveness of these decoding techniques.

Table 3. Translation results (BLEU score) for English-to-Chinese translation.

System	BLEU
Baseline	29.40
+checkpoint averaging	29.94(+0.54)
+model ensembling	30.93(+1.53)
+reranking	31.62(+2.22)

5 Related Work

Recently many novel structures and methods have been proposed to improve the translation quality in neural machine translation. Most of the existing approaches focus on designing better models [10,20,21], augmenting data with large-scale monolingual corpus [2,25], integrating SMT techniques [5,16,22]. In spite of modifying the model structure, our work mainly focuses on improving translation quality by using decoding techniques, which is somehow easier to implement.

Vaswani et al. [21] proposed to use checkpoint averaging method to obtain lower variance and more stable translation results. However, they did not explain how to choose checkpoints and how many checkpoints to be used can obtain better results. Sennrich et al. [13] first applied the method of checkpoint ensembling in WMT16, then they further tried independent ensembling in WMT17 [12], which achieved a significant improvement compared to the former strategy.

To get better final output, various reranking methods have been explored. Shen et al. [15] introduced two novel perceptron-inspired reranking algorithms that improve on the quality of machine translation. Kumar [19] presented MBR decoding for statistical machine translation aiming to minimize expected loss of translation errors under loss functions that measure translation performance. However, these methods are mainly applied to statistical machine translation. Here, we apply them to neural machine translation to explore how good the selected candidate can be by each reranking strategy.

6 Conclusion

In this work, we boost translation performance by making effective use of the existing models with three decoding techniques. Experiments have shown that these decoding techniques can obtain significant improvement over baseline. We point out that reranking has a great potential for improving translation performance, however, a wide gap between the oracle selection still exists. In the future, we will further investigate a better way to shrink this gap.

Acknowledgments. The research work descried in this paper has been supported by the National Key Research and Development Program of China under Grant No. 2016QY02D0303.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of ICLR (2015)
2. Cheng, Y., Xu, W., He, Z., He, W., Wu, H., Sun, M., Liu, Y.: Semi-supervised learning for neural machine translation. In: Proceedings of ACL (2016)
3. Chiang, D.: A hierarchical phrase-based model for statistical machine translation. In: Proceedings of ACL (2005)
4. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning (2017). arXiv preprint: arXiv:1705.03122

5. He, W., He, Z., Wu, H., Wang, H.: Improved neural machine translation with SMT features. In: Proceedings of AAAI (2016)
6. Herbrich, R.: Large margin rank boundaries for ordinal regression. In: Advances in Large Margin Classifiers (2000)
7. Kalchbrenner, N., Blunsom, P.: Recurrent continuous translation models. In: Proceedings of EMNLP (2013)
8. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of ICLR (2015)
9. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: Proceedings of ACL-NAACL (2003)
10. Mi, H., Sankaran, B., Wang, Z., Ittycheriah, A.: A coverage embedding model for neural machine translation (2016). arXiv preprint: arXiv:1605.03148
11. Och, F.J.: Minimum error rate training in statistical machine translation. In: Proceedings of ACL (2003)
12. Sennrich, R., Birch, A., Currey, A., Germann, U., Haddow, B., Heafield, K., Barone, A.V.M., Williams, P.: The University of Edinburgh's neural MT systems for WMT 2017 (2017). arXiv preprint: arXiv:1708.00726
13. Sennrich, R., Haddow, B., Birch, A.: Edinburgh neural machine translation systems for WMT 2016 (2016). arXiv preprint: arXiv:1606.02891
14. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of ACL (2016)
15. Shen, L., Sarkar, A., Och, F.J.: Discriminative reranking for machine translation. In: Proceedings of HLT-NAACL (2004)
16. Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., Liu, Y.: Minimum risk training for neural machine translation (2015)
17. Shu, R., Nakayama, H.: Later-stage Minimum Bayes-Risk Decoding for Neural Machine Translation (2017). arXiv preprint: arXiv:1704.03169
18. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceedings of NIPS (2014)
19. Tromble, R.W., Kumar, S., Och, F., Macherey, W.: Minimum Bayes-risk decoding for statistical machine translation. In: Proceedings of HLT-NAACL (2004)
20. Tu, Z., Lu, Z., Liu, Y., Liu, X., Li, H.: Modeling coverage for neural machine translation. In: Proceedings of ACL (2016)
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (2017)
22. Wang, X., Lu, Z., Tu, Z., Li, H., Xiong, D., Zhang, M.: Neural machine translation advised by statistical machine translation. In: Proceedings of AAAI (2017)
23. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M.: Google's neural machine translation system: bridging the gap between human and machine translation (2016). arXiv preprint: arXiv:1609.08144
24. Zhai, F., Zhang, J., Zhou, Y., Zong, C., et al.: Tree-based translation without using parse trees. In: Proceedings of COLING (2012)
25. Zhang, J., Zong, C.: Exploiting source-side monolingual data in neural machine translation. In: Proceedings of EMNLP (2016)
26. Zhou, L., Hu, W., Zhang, J., Zong, C.: Neural system combination for machine translation. In: Proceedings of ACL (2017)

Building Corpus with Emoticons for Sentiment Analysis

Changliang Li[1(✉)], Yongguan Wang[2], Changsong Li[3], Ji Qi[1], and Pengyuan Liu[2]

[1] Kingsoft AI Laboratory, 33, Xiaoying West Road, Beijing 100085, China
{lichangliang,qijil}@kingsoft.com
[2] Beijing Language and Culture University, 15, Xueyuan Road, Beijing 100083, China
yongguan1992@163.com, liupengyuan@blcu.edu.cn
[3] Peking University, 5, Yiheyuan Road, Beijing 100871, China
lichangsong@pku.edu.cn

Abstract. Corpus is an essential resource for data driven natural language processing systems, especially for sentiment analysis. In recent years, people increasingly use emoticons on social media to express their emotions, attitudes or preferences. We believe that emoticons are a non-negligible feature of sentiment analysis tasks. However, few existing works focused on sentiment analysis with emoticons. And there are few related corpora with emoticons. In this paper, we create a large scale Chinese Emoticon Sentiment Corpus of Movies (CESCM). Different to other corpora, there are a wide variety of emoticons in this corpus. In addition, we did some baseline sentiment analysis work on CESCM. Experimental results show that emoticons do play an important role in sentiment analysis. Our goal is to make the corpus widely available, and we believe that it will offer great support to sentiment analysis research and emoticon research.

Keywords: Emoticon · Sentiment analysis · Corpus

1 Introduction

Sentiment analysis, also called opinion mining, is the field of study that analyzes people's opinion and sentiments towards entities such as products, events and topics and so on [1].

The problem of sentiment analysis has been of great interest in the past decades because of its practical applicability. For example, consumers can seek advices about a product to make decisions in the consuming process. And vendors are paying more and more attention to online opinions about their products and services. Hence, sentiment analysis has attracted increasing attention from many research communities such as machine learning, data mining, and natural language processing.

Recently, people increasingly use emoticons on social media to express their feelings. The sentiment of a message is often affected by the emoticons that appear in the text. For example, given two movie reviews: "It's not Hollywood style :-)" and "It's not Hollywood style :-(", though the text information is the same, but both reviews represent different sentiment due to different emoticons used.

For further research, we need large scale corpora with emoticons. However, most existing corpora are based on texts but ignore the emoticon information. This restricts the research for sentiment analysis and emoticon. So we established a large scale Chinese Emoticon Sentiment Corpus of Movies (CESCM). On CESCM, we have done a lot of experiments, and the experiment results show that the emoticon information help to analyze sentiment better (see the result in Sect. 4). This indicates that emoticons have great influence on the emotion of the whole text.

The remaining paper is structured as follows. Section 2 briefly discusses related work about sentiment dataset and emoticon. Section 3 describes the process of creating the corpus and gives the statistics of the corpus. Section 4 gives the experiment results based on CESCM. Section 5 presents a conclusion.

2 Related Work

Movie reviews are popular resource for sentiment analysis research. For example, Cornell Movie Review Data [2, 3] (MRD) is a commonly used sentiment analysis corpus that includes three datasets: sentiment polarity datasets, sentiment scale datasets, and subjectivity datasets. The Stanford Sentiment Treebank, referred as SST-1, is the first corpus with fully labeled parse trees. Based on the SST-1, SST-2 is created simply with neutral reviews removed from SST-1. And SST-2 is used for the binary classification task [4–6]. Stanford's Large Movie Review Dataset (LMRD) is used for binary sentiment classification and contains more substantial data compared with previous benchmark datasets [8]. Reviews from IMDB and YELP have also been used for some related research. And there is the Chinese Opinion Analysis Evaluation (COAE) [7], which contains product, movie, and finance reviews. Another is Chinese Sentiment Treebank [8, 9].

At present, some people have noticed the role of emoticon and carried out a series of work about emoticon. Someone consider emoticons as noisy labels in their sentiment classification models [10–12]. [13, 14] exploit emoticons in lexicon-based polarity classification. As an important research direction, emoticons are attracting more and more attention. So it is critical to building an corpus with emoticons to support sentiment analysis research.

Despite many corpora have been created for sentiment research, there still lacks large enough corpus with emoticons. For improving sentiment analysis research, we established a large scale Chinese Emoticon Sentiment Corpus of Movies (CESCM). In the next part, we will describe the process of creating the corpus and gives the statistics and analysis of the corpus.

3 Corpus Construction

In this section, we introduce the process of creating CESCM. First of all, we collect a lot of review data. Then, we build a common emoticon set and filter the movie review data based on the set. Next, we clean the data to get rid of redundant movie review data. Finally, we successfully constructed the corpus and analyzed it. We will describe the four key steps in detail.

3.1 Data Collection

Quality and scale are two most important aspects of a corpus. We collect the review data from famous Chinese movie review websites www.douban.com, which is the largest and best movie review site in China. On this website, there are a lot of short reviews about each movie. Each review is made up of one or several sentences. And each review has a star label. The number of star ranges from 1 to 5. People express their attitudes and preferences through movie reviews and star ratings. In addition to using text, there are many emoticons used in the reviews, such as emoticon “:)” or “: (”. So we use crawler technology to collect the reviews and the corresponding labels from the website.

To study the usage of emoticon in text, we first constructed a common emoticon set. This emoticon set includes 510 emoticons that cover most of the popular emoticons on the web. We collected and collated these emoticons from the Internet by artificial methods and there are various styles of emoticons. For instance, emoticons in Western style have the eyes on the left, followed by nose and the mouth such as “:-)”, and emoticons in Japanese style such as “(*_*)” (Also known as “Kaomojis”). In addition, we also selected some of the emoticons from the Unicode character set such as ““😊””. Some examples of the emoticon set were shown in Table 1.

We build CESCM based on the emoticon set. After we collected a great deal of movie reviews, we use these emoticons as necessary conditions to extract all reviews containing emoticon. In the end, we obtained a large number of movie review original resources with rich emoticon information.

3.2 Data Cleaning

The original data contains many false, dirty or unusable data. It can also be called “noise data”. We utilize various strict rules for noise filtering to obtain high quality corpus data. There are several major rules below.

(1) We remove the reviews without star label or content.
(2) We remove the reviews unrelated to the movie. Such as website link, advertising and meaningless characters or gibberish.
(3) We limit reviews length from 3 to 50 words. Too long or too short will affect the quality of the corpus. Figure 1 gives the statistic of the movie reviews length after processing. We can see that the length of most reviews is between 5 and 20 words.

Note that we just list part of rules as example. There are more strict rules employed at this step.

Table 1. Review examples

ID	Emoticon	Description
1	= =	Awkward, indifferent, satiric
2	><	Annoyed, excited, uneasy, hesitant
3	=。=	Awkward, indifferent, tired
4	╮(╯▽╰)╭	Shrug
5	TAT	Sad, crying
6	XD	Laughing, big grin
7	T T	Sad, crying
8	:)	Smiley or happy face
9	O(∩_∩)O	Joyful, excited
10	╮(╯_╰)╭	Dissatisfied

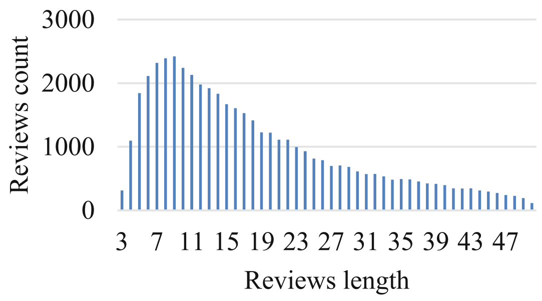


Fig. 1. The length distribution of reviews

3.3 Data Annotation

Table 2 shows some examples. To better understand the meaning, we translate the review into English.

In the CESCM, each review is labeled as stars, and the star number ranges from 1 to 5. These labels are given by people who have seen the movie and represent their evaluation of the movie. 1 means very negative; 2 means negative; 3 means neutral; 4 means positive; 5 means very positive.

Table 2. Review examples

Binary	class	Reviews
Negative	1	╮(╯▽╰)╭，没啥说的。 ╮(╯▽╰)╭, Have nothing to say.
	2	快睡着了，硬撑着看完的=。= Fast asleep, hard to finish watching =。=
	3	我还是适合看文艺片= = It seems that I am suitable for art films = =.
Positive	4	这不是大侦探 这是谢耳朵！聪明是 21 世纪的性感:) This isn't a detective, it's Sheldon! Smart is 21st century sexy :)
	5	将近 4 小时的长片(^o^) 配乐超赞，越来越喜欢罗伯特德尼罗了！ It's nearly four hours long(^o^), The soundtrack is excellent, more and more like Robert DE Niro.

Besides fine-grained (five classes) analysis, this corpus can also be used for binary analysis. For this purpose, we marked each review with a new label based on its star label, just positive and negative. Reviews with 4 and 5 stars belong to positive, and others belong to negative.

3.4 Statistics and Analysis

After the process above, we obtain 47,250 high quality movie review data finally, and each movie review contains at least one emoticon. To our best knowledge, CESCM is the first large scale Chinese sentiment corpus with emoticons. We do a comparison with other popular sentiment analysis corpus, which are widely employed in sentiment analysis. The statistical summary of these corpora is presented in Table 3.

Compared to others, CESCM is larger and contains abundant information of emoticons. And each review contains at least one emoticon, and some reviews contain multiple different emoticons.

In order to better understand CESCM, Table 4 gives the statistic of different emoticons in each movie review. We can see that most reviews contain only one emoticon. There are 3374 reviews contain two emoticons, but only a few number of reviews contain 3 or more emoticons.

Table 3. The size of corpora

Corpora	Total size
MR [2]	10,662
SST-1 [4]	11,855
SST-2 [4]	9,613
Sentiment-Li [8]	13,550
CESCM	47,250

Table 4. Statistic of the emoticon count in each review

#Emoticon/Review	Reviews count
1	43683
2	3374
3	183
4	9
5	1

Figure 2 gives the distribution of emoticons. There are in total 175 different emoticons appear in CESCM. Figure 2 shows the proportion of top 15 emoticons and other emoticons. We can see that the emoticon "= =" is used most. And we found that people were more likely to use this kind of emoticons with rich meaning in real-world.

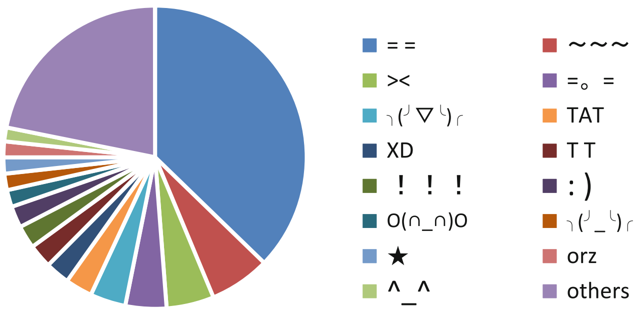


Fig. 2. Emoticons distribution in CESCM

Figure 3 gives an example of the distribution of emoticon sense. We chose a typical emoticon ":)" as illustration. From Fig. 3, we can see that ":)" is mostly employed as positive token. It should be noted that different emoticons may have different distribution of emoticon sense.

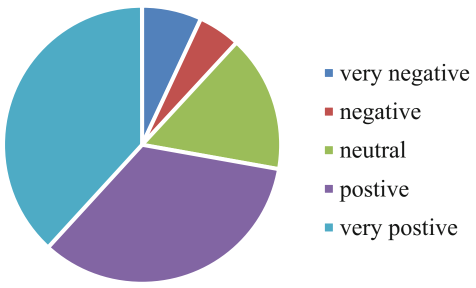


Fig. 3. Emoticon ":)" usage distribution

4 Experiment

In this section, we design experiment combination with our corpus. And we also introduce some baseline results on our corpus for other researchers to compare, as well as to understand the nature of the corpus. These approaches are widely used as baselines in sentiment analysis work, We report the experiment result on two tasks: fine-grained (5-class) and binary analysis (2-class).

4.1 Methods

Majority Method. It is a basic baseline method, which assigns the majority sentiment label in training set to each instance in the test set.

Feature Based Method
NB. We train a Naive Bayes classifier with TFIDF as text feature.
SVM. We train a SVM classifier with TFIDF as text feature.

Neural Network Method
Fast-text. Fast-text is a simple model for sentence representation and classification [15].
CNN. This approach utilizes convolutional neural networks (CNN) for sentiment classification just like work [16]. We employed max and kmax pooling respectively.
LSTM. LSTM based model is applied from the start to the end of a sentence. LSTM uses the last hidden vector as the sentence representation [17, 18]. LSTM_2 used 2-layer LSTM and Bi-LSTM represents bidirectional LSTM model.
Models with Emoticon Information. For baseline purpose, we just add the emoticon information on reviews simply as input fed to the models mentioned above.

4.2 Result

Table 5 gives experiment results for fine grained (5-class) and binary (2-class) sentiment predictions. The metric is accurate rate of predicted sentiment label.

Our experiment was divided into two groups, one group uses emoticons and the other does not.

Table 5. Experiment results of different approach

Information	Method	Fine-grained	Binary
Without emoticons	Majority	27.03%	50.88%
	NB	35.68%	67.09%
	SVM	34.98%	66.94%
	Fast-text	39.47%	77.86%
	CNN_max	40.99%	78.12%
	CNN_kmax	40.79%	78.16%
	LSTM	39.97%	78.62%
	LSTM_2	40.10%	78.29%
	Bi-LSTM	41.35%	78.31%
With emoticons	Majority	27.03%	50.88%
	NB	38.50%	70.20%
	SVM	37.30%	71.30%
	Fast-text	39.55%	79.09%
	CNN_max	42.39%	78.64%
	CNN_kmax	42.78%	78.62%
	LSTM	42.47%	79.01%
	LSTM_2	42.24%	80.04%
	Bi-LSTM	**43.60%**	**80.13%**

In both groups, majority performs worst, because it is just a simple statistic method without any semantic analysis. NB and SVM perform similar as a representative of machine learning methods. However, the performance of NB and SVM is poor compared with the method based on neural network. In the method of neural network, LSTM performs generally better than CNN and Fast-text.

From comparison between two groups, we can see that the methods in second group outperforms the corresponding method in group one. For CNN-based models, it usually increases by more than 1.4% in fine-grained and 0.4% in binary. For LSTM-based models, it usually increases by more than 2.1% in fine-grained and 0.3% in binary. This proves that emoticons play an important role in emotional analysis.

From the result, we can see that combination with the emoticon information boosts the performance of some simple methods. This not only proves the importance of emoticons, but also the potential of our corpus in both sentiment analysis research and emoticon analysis.

5 Conclusion

In this paper, we introduce a high quality and large sentiment corpus of Chinese movie reviews (CESCM). The corpus consists of 47,250 reviews with emoticon information. By explaining the data creation process and providing the results of the baseline algorithms, we hope to help researchers to better understand the nature of CESCM. We believe that the corpus described in this paper will offer great help to researchers for future research on emoticons and sentiment analysis.

References

1. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2007)
2. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 115–124 (2005)
3. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Empirical Methods in Natural Language Processing, pp. 79–86 (2002)
4. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642 (2013)
5. Socher, R., Huval, B., Manning, C.D., Ng, A.Y.: Semantic compositionality through Recursive matrix-vector spaces. In: Empirical Methods in Natural Language Processing, pp. 1201–1211 (2012)
6. Socher, R., Pennington, J., Huang, E., Ng, A.Y., Manning, C.D.: Semi-Supervised recursive autoencoders for predicting sentiment distributions. In: Empirical Methods in Natural Language Processing, pp. 151–161 (2011)
7. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 142–150. Association for Computational Linguistics (2011)
8. Li, C., Xu, B., Wu, G., He, S., Tian, G., Hao, H.: Recursive deep learning for sentiment analysis over social data. In: Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol. 2, pp. 180–185. IEEE Computer Society (2014)
9. Li, C., Xu, B., Wu, G., He, S., Tian, G., Zhou, Y.: Parallel recursive deep model for sentiment analysis. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015, Part II. LNCS (LNAI), vol. 9078, pp. 15–26. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18032-8_2
10. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. Technical report (2009)
11. Pak, A., Paroubek, P.: Twitter as a corpus for sentiment analysis and opinion mining. In: 7th Conference on International Language Resources and Evaluation (LREC 2010), pp. 1320–1326. European Language Resources Association (2010)
12. Liu, K.L., Li, W.J., Guo, M.: Emoticon smoothed language models for twitter sentiment analysis. In: AAAI Conference on Artificial Intelligence (2012)
13. Hogenboom, A., Bal, D., Frasincar, F., et al.: Exploiting emoticons in sentiment analysis. In: ACM Symposium on Applied Computing, pp. 703–710. ACM (2013)
14. Hogenboom, A., Bal, D., Frasincar, F., et al.: Exploiting emoticons in polarity classification of text. J. Web Eng. **14**(1–2), 22–40 (2015)
15. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification (2016). arXiv preprint: arXiv:1607.01759

16. Kim, Y.: Convolutional neural networks for sentence classification. In: Empirical Methods in Natural Language Processing, pp. 1746–1751 (2014)
17. Sundermeyer, M., Schlüter, R., Ney, H.: LSTM neural networks for language modeling. In: Interspeech, vol. 31, pp. 601–608 (2012)
18. Wang, Y., Feng, S., Wang, D., Zhang, Y., Yu, G.: Context-aware Chinese microblog sentiment classification with bidirectional LSTM. In: Li, F., Shim, K., Zheng, K., Liu, G. (eds.) APWeb 2016, Part I. LNCS, vol. 9931, pp. 594–606. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45814-4_48

Construction of a Multi-dimensional Vectorized Affective Lexicon

Yang Wang, Chong Feng(✉), and Qian Liu

Beijing Institute of Technology, Beijing, China
fengchong@bit.edu.cn

Abstract. Affective analysis has received growing attention from both research community and industry. However, previous works either cannot express the complex and compound states of human's feelings or rely heavily on manual intervention. In this paper, by adopting Plutchik's wheel of emotions, we propose a lowcost construction method that utilizes word embeddings and high-quality small seed-sets of affective words to generate multi-dimensional affective vector automatically. And a large-scale affective lexicon is constructed as a verification, which could map each word to a vector in the affective space. Meanwhile, the construction procedure uses little supervision or manual intervention, and could learn affective knowledge from huge amount of raw corpus automatically. Experimental results on affective classification task and contextual polarity disambiguation task demonstrate that the proposed affective lexicon outperforms other state-of-the-art affective lexicons.

Keywords: Affective analysis · Affective lexicon
Knowledge representation

1 Introduction

Affective analysis is a rapidly developing area of Natural Language Processing that has received growing attention from both research community and industry in recent years [18,21]. It helps companies to know what customers feel about their products, and it helps a political party or government to know what the voters feel about their actions and proposals. On the other hand, it helps customers or voters to choose wisely and in an informed way by knowing what their peers feel about a product or a political candidate. With this, affective analysis and opinion mining are of great importance for aiding economy and democracy.

Affective resource plays an important role in the analysis. In fact, researchers in related area can hardly progress much without a good pool of affective lexicon, although there really exist many available affective resource. Most of the current affective lexicons, e.g., NTUSD [11], only tells whether the given word is positive (+1) or negative (−1), and even some words are divided into the two parts simultaneously. However, human's affects are complex and compound states of feelings that result in physical and psychological reactions influencing both thought and behavior.

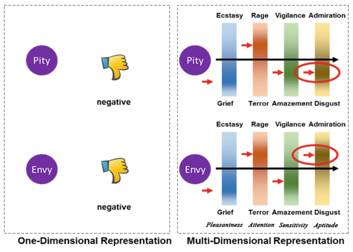


Fig. 1. Comparison between one-dimensional vs. multi-dimensional representation

Although there are many advantages for building a multi-dimensional emotional resource, traditional methods encountered many problems: (1) Substantial human labors consumption. Constructing dictionaries is a labor-intensive task. (2) High degree of subjective. The disagreement among annotators makes the quality of annotation varies significantly.

Moreover, most of the current affective lexical resources could not overcome the elusive nature of emotions and the ambiguity of natural language. E.g., traditional affective lexicon only tells "pity" and"envy" are both negative, and regards their affective information as the same (Fig. 1). But it is far from reality [16]. In Multi-Dimensional representation, Aptitude dimensionality of "envy" is positive implying the affirmation of one's abilities, while the same dimensionality of"pity" is inversely negative (Fig. 1). Only the fine-grained lexicon could tell the difference.

To solve this problem, we propose a construction method that utilizes word embeddings and high-quality small seed-sets of affective words to generate multi-dimensional affective vector automatically. As a test and verification, we construct a large-scale affective lexicon. Unlike existing affective lexicon, our lexicon is a multi-dimensional vectorized lexical resource, which is based on the psychological model of affect and grounded in a continuous affective space (denoted as $\phi_{\ senti}$).

Overall, the main contributions of this paper include: (i) By bridging the gap between the semantic space($\Psi_{\ sema}$) and the affective space ($\Phi_{\ senti}$) following Plutchik's wheel of emotions, our lexicon gains vectorized description ability of the fine-grained affective states. (ii) The construction of our lexicon uses little supervision or manual intervention, and could learn affective knowledge from huge amount of raw corpus automatically. (iii) Experimental results on several representative affective analysis tasks demonstrate that the proposed lexicon is efficient, and outperforms the best baseline affective lexicons.

2 Related Works

2.1 Psychological Models of Affect

For a long time before AI researchers' concern, the study of affect has been one of the most confused (and still open) chapters in the history of psychology.

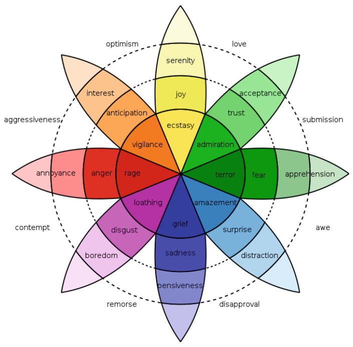


Fig. 2. Plutchik's wheel of emotions

Psychologists have developed many different affective models. Over the recent years, the adoption of psychological models of affect has become a common trend among researchers and engineers working in the sphere of affective analysis [14].

The well-known wheel of emotions is an affective categorization model developed starting from Plutchik's studies on human emotions [16]. The conceptual scheme of the wheel of emotions is illustrated in Fig. 2. In this model, Plutchik suggested eight basic bipolar emotions, whose different levels of activation make up the total emotional states of the mind (shown as Fig. 1). Actually, Plutchik's wheel of emotions is based on the idea that the mind is made of different independent dimensionalities and that emotional states result from setting these dimensionalities on different values.

Apparently, this model is particularly useful to recognize, understand and express affects in the context of HCI. Therefore, Plutchik's wheel of emotions is leveraged as the theoretic basis of our work.

2.2 Common Affective Lexical Resource

Generally, affective lexicon is important for HCI. [22] classified affective lexicons into three basic types. (i) The ones only containing affective words, such as the Never-Ending Language Learner (NELL) [5]. They can not able to tell whether the texts have positive or negative affects; (ii) The ones containing both affective words and affective polarities, such as National Taiwan University Sentiment Dictionary (NTUSD) [11] and HowNet [6]. They lack the semantic relationship among the words and cannot distinguish the extent of the affect expressed by the words; (iii) The ones containing words and relevant affective polarity values (i.e., affective polarity and degree), such as SentiWordNet [7], WordNet-Affect [20] and SenticNet [3].

Based on the theory of wheel of emotions, SenticNet is proposed as the state-of-the-art affective lexical resource for affective analysis [4]. However, their deficiency is concluded as follows: (i) They have a complicated process of construction. And it is difficult to expand to other languages; (ii) they did not fully utilize large-scale unlabelled data and can not unsupervisedly mine the affects implied from statistics. The proposed construction procedure as our lexicon demonstrated tries to overcome the above-mentioned drawbacks.

3 The Proposed Affective Lexical Resource

In this section, we present the construction method of the proposed Affective lexicon. The method is comprised of three base modules: (1) Distributional word representation learning for all words in the lexicon; (2) Affective seed-set construction of each basic affect defined in Plutchik's wheel of emotions; (3) Construction of vectorized affective representations. We mainly introduce the second step and the third step here.

3.1 Constructing the Affective Seed-Set

The affective seed-set plays an important role in the proposed lexicon to align the semantic space and the affective space. To achieve the complete description of the given basic affect, the seed-set should have full coverage of basic emotional state of the mind, and avoid incorporating the domain-specific affective words.

In Plutchik's model, affects are reorganized around 4 independent dimensionalities. We follow [2] to reinterpret the 4 dimensionalities as *Pleasantness*, *Attention*, *Sensitivity* and *Aptitude*. The set of these four dimensionalities is denoted as $\mathcal{D} = \{\ Plsn,\ Attn,\ Snst,\ Aptt\}$. Each dimensionality has 6 basic affects which determine the intensity of the perceived emotion. Afterwards, we aims at generating affective seed-set for each basic affect in Table 1.

Table 1. Basic affects in different affective dimensionalities used in the proposed lexicon.

Affective Dim	Pleasantness(Plsn)	Attention(Attn)	Sensitivity(Snst)	Aptitude(Aptt)
Basic Affect	Ecstasy(+1)	Vigilance(+1)	Rage(+1)	Admiration(+1)
	Joy(+0.6)	Anticipation(+0.6)	Anger(+0.6)	Trust(+0.6)
	Serenity(+0.2)	Interest(+0.2)	Annoyance(+0.2)	Acceptance(+0.2)
	Pensiveness(−0.2)	Distraction(−0.2)	Apprehension(−0.2)	Boredom(−0.2)
	Sadness(−0.6)	Surprise(−0.6)	Fear(−0.6)	Disgust(−0.6)
	Grief(−1)	Amazement(−1)	Terror(−1)	Loathing(−1)

We mainly utilize the following dictionaries to expand these basic affect by synonym expansion respectively, and construct our affective seed-sets, which will be used to generate the affective lexicon.

Youdao Dictionary[1]: As the first translation software based on search engine technology, it consists of a huge number of buzzwords in the Internet by its novel web interpretation function.

[1] http://www.youdao.com/.

Webster's International Dictionary[2]: As the synthesizer of the structural linguistics in U.S., it consists of More than 450 thousand words, and is reported as the largest single volume English dictionary in the word.

Thesaurus Synonym Dictionary[3]: It provides huge amount of the synonym relationships among words.

WordNet[4] : It assigns each synset with a positive score, a negative score and an objective score. The positive/negative score represents the extent to which the word expresses a positive/negative emotion.

With these dictionaries, affective seed-set is constructed as follows: given basic affect w, (i) obtain the synonyms of w, and then obtain the synonyms of the obtained synonyms; (ii) filter all of the synonyms manually by discarding the words whose affect orientation is wrong. Finally, we totally obtain 24 seed-sets for 24 basic affective words, and each seed-set consists of about 100 affective words. For example, "good spirits","rapture", "be on cloud nine","passion" , "well-being","melody" , "happiness" etc., belong to the seed-set of the basic affect $Ecstasy$.

3.2 Constructing the Lexicon

Based on the word embeddings for all the words in the vocabulary and the affective seed-sets constructed above, we aims at generating 4-dimensional affective vector for each word w in the vocabulary, as follows:

$$vector(w) = (Plsn(w), Attn(w), Snst(w), Aptt(w)) \tag{1}$$

Take $Pleasantness$ dimensionality as an example, we describe how to generate affective value of word w in this dimensionality, as follows:

Step 1. As discussed above, we have constructed 6 affective seed-sets for $Pleasantness$ dimensionality, namely $Ecstasy$, Joy, $Serenity$, $Pensiveness$, $Sadness$ and $Grief$. The vector of word w in a basic affect is calculated according to its most similar N words. Cosine distance is utilized to measure the similarity.

Step 2. The minimum cosine distance between the given word w and the average distance of the N words means the maximum correlation, which is denoted as maxCorrelation here, and the affective strength value of the corresponding basic affect is denoted as x ($x \in -1, -0.6, -0.2, 0.2, 0.6, 1]$ as shown in Table 1).

Step 3. For word w, the affective value in $Pleasantness$ dimensionality could be formulated as:

$$Plsn(w) = x * sigmoid(\alpha * (maxCorrelation - \phi)) \tag{2}$$

wherein, ϕ denotes the threshold determining whether the given word w is close to this affective dimensionality. If $maxCorrelation < \phi$, we think that the given word w does not belong to this affective dimensionality. In this case,

[2] http://www.merriam-webster.com/dictionary.
[3] http://www.thesaurus.com.
[4] http://wordnet.princeton.edu/wordnet/.

$sigmoid(\ \alpha *(maxCorrelation -\ \phi))$ returns 0, and we believe the given word w has no obvious affective orientation in Pleasantness dimensionality. Whereas, if $maxCorrelation > \phi$, we think that the given word w could be clustered into this basic affect. In our study, the value of ϕ is set of 0.29, which is the average distance among all word vectors in semantic space. Another parameter α decides whether assign the word w with this affective dimensionality, when $maxCorrelation$ is close to the average distance ϕ. The larger value of α is, the bigger the slope of $Plsn(w)$ becomes.

So far, the proposed affect lexicon has been generated completely. It totally consists of 62,101 affective vectors.

4 Experiments

In this section, we conduct experiments to evaluate the effectiveness of the proposed affective lexicon, by applying it in sentence-level and word-level affective analysis. Moreover, we carry out affective vector analysis on the affective space Φ_{senti} established by the lexicon to investigate whether it could adequately reflect affect diversity but not semantic difference.

4.1 Experiments Setup

Comparative Affective Lexical Resources. We compare the proposed lexicon with other widely-used affective lexical resources, including SenticNet and SentiWordNet. SenticNet [3] is a widely used affective lexical resource for opinion mining based on the hourglass model, which is derived from Plutchik's wheel of emotions. Similar with our lexicon, it describes each word with 4 independent dimensionalities. SentiWordNet [1] was developed based on WordNet [8]. It assigns different affective values to each of the synonyms under different parts-of-speech (POS). The statistics of all the alternative affective lexical resources are illustrated in Table 2.

Parameter Settings. In our experiments, the word embeddings are trained using CBOW model. The context window size is set to 8, the number of negative samples is set to 5, and the dimension of vectors is 300. We trained the word embeddings using the Wikipedia corpus with the size of 13.3 GB.

Table 2. The statistics of comparative affective lexical resources.

Lexicon	Format	Quantity
SenticNet	word,Pleasantness,Attention, Sensitivity,Aptitude	30,000
SentiWordNet	POS,ID,PosScore,NegScore, SynsetTerms,Gloss	117,659
Our lexicon	word,Pleasantness,Attention, Sensitivity,Aptitude	62,101

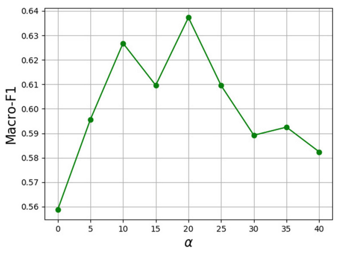


Fig. 3. The comparison of results with different values of parameter α on affective classification task

The size of selected seed words in each affective seed set (i.e. N) is set to 5. We tune the parameter α (in Eq. (2)), and study its influence on the performance of the lexicon in affective classification task on dataset **Comments_BBC**. As shown in Fig. 3, the parameter α is set as 20 to get the best experimental performance.

4.2 Sentence-Level: Affective Classification

We use the affective classification task [15] to evaluate the effectiveness of the pro-posed lexicon.

Datasets. Three datasets are utilized for experiments. The statistics are illustrated in Table 3.

Comments_BBC is collected from BBC News Reviews. **Tweets_STF** is a manual labeled tweet dataset from specific domains. **IMDB** provides a highly polar movie reviews sets.

Since the first two datasets only provide testing data, we apply the training set of the **IMDB** as their training set, and 5,000 of **IMDB**'s testing set as their development set. For the last datasets, we use 33% of the data as the development set. Besides, the neutral sentences are removed for all datasets.

Table 3. Datasets for affective classification task.

Datasets	#Positive	#Negative	#Total
Comments_BBC [23]	99	653	752
Tweets_STF [9]	182	177	359
IMDB [12]	25,000	25,000	50,000

Settings. We evaluate the effectiveness of the above lexicons by employing them as the lexical features. Following [19], we run a GRU network for affective classification. On the input layer in the GRU network, each word will be represented as a 4-dimensional vector. A GRU layer with a dropout (0.5) is followed by a dense layer with the sigmoid as the activation function. We use the Adam algorithm [10] to optimize the parameters. All the models are trained over 50 epochs with a batch size of 64. Additionally, we add four syntactic features (i.e., noun, verb, adjective and adverb) to connect with the lexical features. Macro-F1 is used here as the evaluation metric.

Table 4. The results of affective classification task.

F1	Comments_BBC	Tweets_STF	IMDB
SenticNet [17]	0.568	0.702	/
SentiWordNet	0.593	0.682	0.677
SenticNet	0.575	0.639	0.683
Our lexicon	**0.637**	**0.707**	**0.706**

Results. The results are given in Table 4. Since SenticNet is a closed paid software, we can't reproduce the original method proposed in [17]. Hence we refer the experimental results about SenticNet reported in [17]. We can observe that the proposed lexicon outperforms other lexicons: (i) on dataset **Comments_BBC**, it exceeds SenticNet by 6.9%, and exceeds the best baseline lexicon SentiWordNet by 4.4%; (ii) on dataset **Tweets_STF**, it exceeds SenticNet by 0.5%, and exceeds SentiWordNet by 2.5%; (iii) on dataset **IMDB**, it exceeds the best baseline lexicon SenticNet by 2.3%. We contribute the enhancement to the modeling ability of the lexicon, which could capture more expressive and discriminative affective information.

It is noted that IMDB is not utilized in Ribeiro's work and their work could not be reproduced. Thus the results of their method on IMDB is not presented in Table 4.

4.3 Word-Level: Contextual Polarity Disambiguation

We conduct the Contextual Polarity Disambiguation task on all lexicons, which is a perennial task in SemEval[5]. The task aims to determine whether a given word is positive or negative in its context (Table 5).

Table 5. Datasets for contextual polarity disambiguation task.

Dataset	#Positive	#Negative	#Total
SemEval2015-Task10-A	5,316	2,839	8,155

[5] http://alt.qcri.org/semeval2015/task10/.

Datasets. The official dataset of contextual polarity disambiguation task in SemEval2015 utilized. We also use 33% of the training data as the development set.

Settings. We follow [13] to extract features from the target word as well as from the context. Different from [13], we only use the lexical features to focus on lexicon evaluation and use GRU to implement the experiment similar as Sect. 4.2. All settings are in accordance with the experiments in Sect. 4.2, except the input form of target word and its context.

Results. Table 6 show the experimental results of contextual polarity disambiguation task.

Table 6. The results of contextual polarity disambiguation task in SemEval2015-Task10-A.

Macro-F1	SemEval2015-Task10-A
SentiWordNet	0.627
SenticNet	0.664
Our lexicon	**0.696**

From these tables, we could conclude that our lexicon exceeds the best baseline lexicon SenticNet by 3.2%, and exceeds SentiWordNet by 6.9% on SemEval2015. We conclude that our proposed multi-dimensional vectors could express potential affective states of the given word, covering all the possible affect which this word may imply, and hence does not have to change with context. It suggests the high description ability of the lexicon might alleviate the problem of word's ambiguous affective representation forms.

4.4 Affective Space (Φ_{senti}) Analysis

As discussed above, during the construction of the proposed lexicon, an affective space (Φ_{senti}) is generated. We would like to evaluate the words closed to each other in Φ_{senti} whether share the similar affective orientation and sense or not. We select four words as target words. To find their nearest words in Φ_{senti}, the cosine similarity is applied here.

From Table 7, it could be found that, given the target word, its closest words reveal the similar affective information. It demonstrates the word distribution in the affective space meets our common sense.

We also compare the affective space (Φ_{senti}) with the semantic space (Ψ_{sema}). Taking the word *happy* as an example: in semantic space (Ψ_{sema}) the closest words with *happy* are *sad*, *pleased*, *glad*, *delighted*, and *unhappy*. These words share similar context but inconsistent affect. It verifies that the proposed

Table 7. The results of word sentiment similarity in affective space Φ_{senti}

Target word		Words with minimum distance in Φ_{senti}
Adjective	Happy	glad, okay, excited, honestly, assured...
	Upset	dismayed, irritated, unnerved, unhappy, embarrassed...
Noun	Bliss	ageless, enchanted, lively, radiant, beauteous...
	Disaster	epidemic, famine, repressed, anarchy, oppressive...

lexicon has fulfilled our hypothesis in mapping the semantic space (Ψ_{sema}) to a new affective space (Φ_{senti}).

5 Conclusion and Future Works

This paper presented a novel method to construct the affective lexicon by bridging the gap between the semantic space (Ψ_{sema}) and the affective space (Φ_{senti}) following Plutchik's wheel of emotions. We constructed a affective lexicon which provide vectorized description ability of the fine-grained and compound affective states. It can be observed by experimental results that our lexicon outperforms other lexicons on affective classification task and contextual polarity disambiguation task.

The future research will focus on specific domain oriented affect transfer representing and constructing.

References

1. Baccianella, S., Esuli, A., Sebastiani, F.: SentiWordNet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In: LREC 2010, vol. 10, pp. 2200–2204 (2010)
2. Cambria, E., Livingstone, A., Hussain, A.: The hourglass of emotions. In: Esposito, A., Esposito, A.M., Vinciarelli, A., Hoffmann, R., Müller, V.C. (eds.) Cognitive Behavioural Systems. LNCS, vol. 7403, pp. 144–157. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34584-5_11
3. Cambria, E., Poria, S., Bajpai, R., Schuller, B.W.: SenticNet 4: a semantic resource for sentiment analysis based on conceptual primitives. In: COLING 2016, pp. 2666–2677 (2016)
4. Cambria, E., Speer, R., Havasi, C., Hussain, A.: SenticNet: a publicly available semantic resource for opinion mining. In: AAAI Fall Symposium: Commonsense Knowledge, vol. 10 (2010)
5. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI 2010. vol. 5, p. 3 (2010)
6. Dong, Z., Dong, Q., Hao, C.: HowNet and its computation of meaning. In: Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations, pp. 53–56 (2010)

7. Esuli, A., Sebastiani, F.: SentiWordNet: a high-coverage lexical resource for opinion mining. Evaluation **17**, 1–26 (2007)
8. Fellbaum, C.: WordNet. Wiley Online Library, New York (1998)
9. Go, A., Bhayani, R., Huang, L.: Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, vol. 1, no. 12 (2009)
10. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Ku, L.W., Liang, Y.T., Chen, H.H.: Opinion extraction, summarization and tracking in news and blog corpora. In: Proceedings of AAAI, pp. 100–107 (2006)
12. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, pp. 142–150 (2011)
13. Mohammad, S.M., Kiritchenko, S., Zhu, X.: NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. arXiv (2013)
14. Nozza, D., Fersini, E., Messina, E.: A multi-view sentiment corpus. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, vol. 1, pp. 273–280 (2017)
15. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing-Volume 10, pp. 79–86. Association for Computational Linguistics (2002)
16. Plutchik, R.: The nature of emotions human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. Am. Sci. **89**(4), 344–350 (2001)
17. Ribeiro, F.N., Araújo, M., Gonçalves, P., Gonçalves, M.A., Benevenuto, F.: SentiBench-a benchmark comparison of state-of-the-practice sentiment analysis methods. EPJ Data Sci. **5**(1), 1–29 (2016)
18. Schouten, K., Frasincar, F.: Survey on aspect-level sentiment analysis. TKDE **28**(3), 813–830 (2016)
19. Stojanovski, D., Strezoski, G., Madjarov, G., Dimitrovski, I.: Finki at SemEval-2016 task 4: deep learning architecture for Ttwitter sentiment analysis. In: SemEval 2016, pp. 149–154 (2016)
20. Strapparava, C., Valitutti, A., et al.: WordNet affect: an affective extension of WordNet. In: LREC, vol. 4, pp. 1083–1086 (2004)
21. Talavera, E., Radeva, P., Petkov, N.: Towards Egocentric Sentiment Analysis. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST 2017. LNCS, vol. 10672, pp. 297–305. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-74727-9_35
22. Tang, D., Wei, F., Qin, B., Yang, N., Liu, T., Zhou, M.: Sentiment embeddings with applications to sentiment analysis. TKDE **28**(2), 496–509 (2016)
23. Thelwall, M.: Heart and soul: sentiment strength detection in the social web with sentistrength. In:Cyberemotions: Collective emotions in cyberspace (2013)

Convolution Neural Network with Active Learning for Information Extraction of Enterprise Announcements

Lei Fu[1,2], Zhaoxia Yin[1], Yi Liu[2], and Jun Zhang[3](✉)

[1] Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education, Anhui University, Hefei 230601, People's Republic of China
[2] PKU Shenzhen Institute, Shenzhen, China
[3] Shenzhen Securities Information, Co., Ltd., Shenzhen, China
zhangjun@cninfo.com.cn

Abstract. We propose using convolution neural network (CNN) with active learning for information extraction of enterprise announcements. The training process of supervised deep learning model usually requires a large amount of training data with high-quality reference samples. Human production of such samples is tedious, and since inter-labeler agreement is low, very unreliable. Active learning helps assuage this problem by automatically selecting a small amount of unlabeled samples for humans to hand correct. Active learning chooses a selective set of samples to be labeled. Then the CNN is trained on the labeled data iteratively, until the expected experimental effect is achieved. We propose three sample selection methods based on certainty criterion. We also establish an enterprise announcements dataset for experiments, which contains 10410 samples totally. Our experiment results show that the amount of labeled data needed for a given extraction accuracy can be reduced by more than 45.79% compared to that without active learning.

Keywords: Text classification · Active learning
Convolutional neural networks · Enterprise announcements

1 Introduction

At present, information extraction has become an important branch of the NLP field. The task of information extraction is to obtain target information accurately and quickly from a large amount of data and improve the utilization of information. The information extractions of enterprise announcements help the users identify concerns quickly. Therefore, this paper addresses information extraction task for enterprise announcements which is a type of document that publicly informs the society of important issues and extracts the information about investment and other specifically. We extract the key information through text classification. There are many methods for text classification. Currently, the mainstream method implements text classification through deep learning.

There is no doubt that deep learning has ushered in amazing technological advances on natural language processing(NLP) researches, applied to various tasks such as text classification [1], machine translation [2], document summarization [3]. But in the deep learning environment, there is a bottleneck that is manual collection of sample labels is expensive and time-consuming. Convolutional neural networks (CNN) [4] is common model in deep learning and many natural language processing tasks and has great classification performance. In order to obtain great classification performance, a large amount of manually-labeled data is needed. Therefore, the low efficient manual labeling work has become a problem that restricts the development of text classification tasks. Active learning [5, 6] is motivated for solving the problem. Active learning is an iterative process of selecting useful samples as training data. And the size of the training data set should be as small as possible while maintaining classification performance. Therefore, the kernel of active learning is to choose a useful sample set. This paper proposes an active learning method to improve the effect of deep learning for text classification task.

Based on the deep learning architecture, this paper proposes a novel active learning algorithm, which is to judge the useful data according to the category probability strategy. It solves the problem that the applying deep learning to text classification require a lot of manual data annotation.

In this paper, Sect. 2 describes the related work of this study. Section 3 gives a description of the principles of CNN and active learning methods. Section 4 introduces the experimental results and analysis. Section 5 concludes the paper and outlines the future work.

2 Related Work

At present, there are a lot of research about active learning. Most of the early work can be found in the classical survey of Settles [7]. It covers acquisition functions such as information theoretical methods [8]. And Bayesian active learning method typically uses a non-parametric model like Gaussian process to estimate the expected improvement by each query [9] or the expected error after a set of queries [10]. [9] presented a discriminative probabilistic framework based on Gaussian Process priors and the Pyramid Match Kernel and introduced an active learning method for visual category recognition based on the uncertainty estimates provided by the GP-PMK. [10] presents an active learning method that directly optimizes expected future error. A recent approach by Gal & Ghahramani [11] shows an equivalence between dropout and approximate Bayesian inference enabling the application of Bayesian methods to deep learning. The Support Vector Machines [12] of active learning method presented two novel multi-label active learning strategies, a max-margin prediction uncertainty strategy and a label cardinality inconsistency strategy, and then integrate them into an adaptive framework of multi-label active learning. The Query by committee [13] active learning method is to train a committee of learners and query the labels of input points where the committee's predictions differ, thus minimizing the variance of the learner by training on input points where variance is largest. The Expectation-Maximization (EM) [14] with active learning (uses a modified QBC) for text classification modify the

Query-by-Committee (QBC) method of active learning to use the unlabeled pool for explicitly estimating document density when selecting examples for labeling. Then active learning is combined with Expectation-Maximization in order to “fill in” the class labels of those documents that remain unlabeled.

In our research work, the active learning is mainly applied to the image [15] and rarely applied to the text for the deep learning method. Proposed in this paper, the certainty criterion applies active learning to text classification tasks in deep learn fields.

3 Proposed Method

Figure 1 shows the framework diagram of CNN based on active learning. This section mainly introduces the classification method of CNN and the principle of active learning method. Convolution neural network can reduce the parameter quantity through the local connection and weight sharing to greatly reduce training complexity and over-fitting. Meanwhile weight sharing also gives the convolutional network tolerance for translation. Active learning methods can effectively reduce the number of training samples, therefore significantly reducing training time and manual tagging workload.

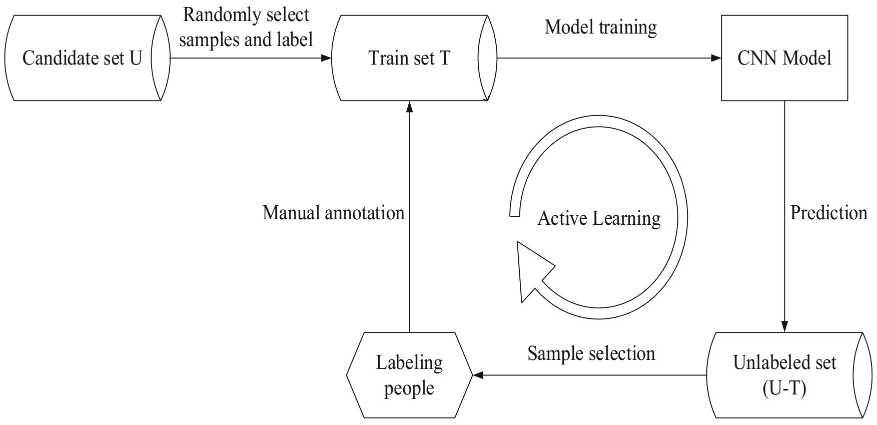


Fig. 1. CNN based on active learning

3.1 Convolutional Neural Network

CNN is a deep neural network and has made a major breakthrough in computer vision and speech recognition, and mainly contains the convolution layer and the pool layer. By convolutional operations at different scales are implemented on long text, more comprehensive features can be extracted.

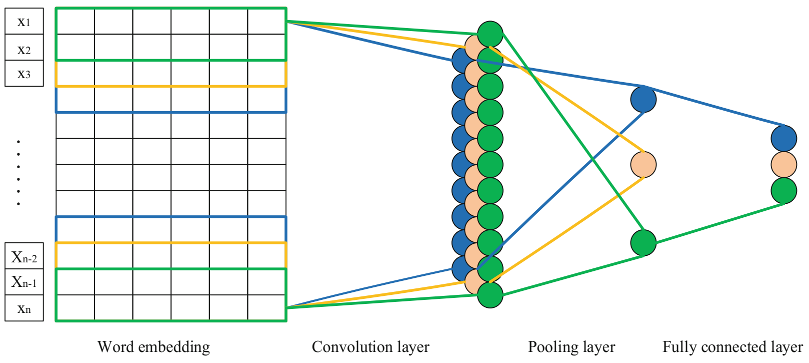


Fig. 2. CNN model diagram

Figure 2 shows the framework of the CNN. x_i indicates the k-dimensional word embedding of the *i-th* word of a text. A text $x_{1:n}$ which is length n is represented as:

$$x_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n \tag{1}$$

$\oplus$ indicates a concatenation operation. In general, let $x_{i:i+j}$ indicate $x_i, x_{i+1}, \cdots,$ x_{i+j}. A filter is a window of size h words and produces a new feature by a convolution operation, a convolution operation corresponding to a filter $w \in R^{hk}$. For example, feature c_i is generated from a window size h words $w_{i:i+k-1}$ by:

$$c_i = f(w \cdot x_{i:i+k-1} + b) \tag{2}$$

Here, b denotes the bias term, f denotes a nonlinear function such as rectifier or tanh. Applying a filter to each window $\{x_{1:h}, x_{2:h+1}, \cdots, x_{n-h+1:n}\}$ in the text to generate a set of feature maps:

$$c = [c_1, c_2, \cdots, c_{n-k+1}] \tag{3}$$

$c \in R^{n-h+1}$. Then the largest feature map of the group of feature maps c is selected to represent the group of features by the maximum pooling operation.

A filter extracts a feature. Therefore, multiple features are extracted by multiple filters (windows of different sizes). These features are connected in the full connection layer to get the text feature vector and sent it to Softmax layer to get its corresponding category. The model uses a cross-entropy function as a loss function, which measures the probability error of each independent classification task.

3.2 Active Learning

At present, the supervised algorithms require a lot of labeled training data, and the result of the experiment is affected the quality of the data in deep learning. This paper introduces active learning into the CNN to classify long text. In essence, active learning is an iterative training process of selecting a useful sample from unlabeled sample data and obtaining its label.

The three sample selection methods based on certainty criterion proposed in this paper chooses the next sample to be annotated from the unmarked sample ***U*** based on certainty criterion, which has three kinds of sample selection methods.

The First Method: Selecting the sample with probability range of $[1/C - a, 1/C + b]$. The probability of the sample belongs to this range is the uncertainty sample. And if the sample is wrongly predicted, it has a large impact on the classifier and meets the selection requirements.

The Second Method: Selecting the sample with probability range of $([0, c] \cup [d,1])$. The probability of the sample belongs to this range is the certainty sample. And if the sample is wrongly predicted, it has a large impact on the classifier and meets the selection requirements.

The Third Method: this method is to select $\propto$ of the first method and $(1\text{-}\propto)$ of the second method, where $\propto$ is obtained based on engineering experience.

The total number of categories is C, and the constants of a, b, c, and d are based on engineering experience. Using these three probabilistic selection methods, each sample selected is a useful sample data, so it also has the greatest influence on the classifier. The influence of the remaining samples on the classifier is gradually weakened.

The specific steps of active learning based on certainty criterion are as follows:

```
Probability Selection Strategy
    Input: Uncategorized the candidate sample set U.
    Output: Classifier fc.
    Begin:
    Step 1. Selecting i samples from the candidate sample set U and correctly
            labelling their categories to construct the initial training set T;
    Step 2. Using the training set T to train the classifier fc;
    Step 3. Using the classifier fc to classify the sample (U-T) remaining in the
            candidate set, and selecting the sample that meets the requirements
            (Within a specified range of probabilities and label errors predicted)
            for annotation;
    Step 4. Using labeled data and training set T to train a new classifier fc;
    Step 5. If the accuracy reaches $\beta$, the algorithm terminates and returns fc;
            Otherwise returns to step 2);
    End.
```

For active learning, the classifier doesn't passively accept the data provided by the user, but instead actively asks the user to label the sample that is selected by the current classifier. Through continuous selection of indistinct samples to iteratively train obtain a satisfactory classifier. Theoretically, active learning can significantly reduce the number of required samples compared to random selection with similar experimental results [16].

4 Experiment

For verifying the validity of our method, we conduct a series of experiments on our self-built enterprise announcements dataset. Note that we apply the CNN as the basic deep learning classification method.

4.1 Data Set

We crawl the enterprise announcements as our dataset on the Internet. And each enterprise announcement is segmented to form the corresponding text segment according to the title. The detail is listed in Table 1. In order to the convenience of experiment, the text segments are annotated manually in advance. In other words, the categories of the text segments are divided into Investment and Other labels. The corresponding meaning of the labels is listed in Table 2.

Table 1. Data set

Date set	Total (number)	Investment (number)	Other (number)
Train	8554	2669	5885
Valid	916	101	815
Test	940	120	820

Table 2. Corresponding meaning of the labels

Label	Corresponding meaning
Investment	The content of the text block is related to investment
Other	Other represents text blocks that have nothing to do with investment

4.2 Experimental Setting

The dimension of the CNN model is *Dim* = 64, the number of convolution kernel is 3, and the convolution kernel size are 2, 3 and 4. According to the observation of all enterprise announcements, their length is about 400 words. Therefore, all enterprise announcements are fixed to 400 words. If the length of sample is greater than 400

Table 3. Method and probability values

	Probability one	Probability two	Probability third
The first method	(0.1,0.9)	(0.125,0.875)	(0.15,0.85)
The second method	[0,0.1] ∪ [0.9,1]	[0,0.1] ∪ [0.95,1]	[0,0.05] ∪ [0.9,1]
The third method	(0.125,0.875) ∪ ([0,0.1] ∪ [0.95,1])	(0.125,0.875) ∪ ([0.0.05] ∪ [0.9,1])	(0.15,0.85) ∪ ([0,0.1] ∪ [0.95,1])

words, we select the foregoing 400 words. Otherwise, tag <pad> is added for complementation until achieving 400 words. Besides, for our dataset, this paper proposes three group values for each method in Table 3.

Through our empirical study, the parameters are set the appropriate value. The α of the third method is 0.5. When the learning rate of *lr* is set 0.01, the effect of model is best. Furthermore, the F1-measure don't increase significantly after 30 consecutive epochs and the model stopped updating iterations. In this experiment, Word2vec of Google Company is used to pre-train the unlabeled 700 MB Chinese Sogou corpus into a word embedding table of $Dim = 64$. Therefore, the trained words embedding can represent the words accurately.

4.3 Evaluation Index

General text classification evaluation criteria use accuracy (P), recall (R) and F1-measure (F) as an indicator. Accuracy rate is the correct rate of information related to the need to pay attention; recall rate is the proportion of information retrieved; the F1-measure can be thought of as a weighted average of model accuracy and recall, with a maximum of 1 and a minimum of 0. F1-measure is calculated as follows:

$$F_1 = \frac{2TP}{2TP + FN + FP} \tag{4}$$

In this experiment, TP indicates that the predicted sample label of model is Investment, and the actual label is Investment; FP indicates that the predicted sample label of model is Investment, and the actual label is other; FN indicates that the predicted sample label of model is other, and the actual label is Investment; TN indicates that the predicted sample label of model is other, and the actual label is other. In this paper, the F1-measure of experimental result is represented.

4.4 Experimental Results and Analysis

In the Tables 4, 5 and 6, the first row is the probability value and the first column is the number of iterations, the middle content is the F1 value (a used percentage of all data). x indicates the prediction probability of sample.

Table 4. Experimental results of first method

	0.1 < x < 0.9	0.125 < x < 0.875	0.15 < x < 0.85
Step 1	0.79 (42.66)	0.79 (42.66)	0.79 (42.66)
Step 2	0.77 (46.20)	0.78 (45.24)	0.78 (45.28)
Step 3	0.79 (51.43)	0.80 (47.99)	0.80 (50.15)
Step 4	0.80 (53.59)	0.82 (50.22)	0.81 (56.09)
Step 5	0.81 (56.04)	0.81 (52.26)	0.84 (58.02)
Step 6	0.83 (58.23)	0.83 (54.58)	**0.85 (59.40)**
Step 7	**0.85 (58.67)**	**0.85 (55.62)**	0.84 (60.97)
Step 8	0.84 (60.10)	0.85 (57.21)	0.84 (62.68)

Table 5. Experimental results of second method

	x <= 0.1 or x >= 0.95	x <= 0.05 or x >= 0.9	X <= 0.1 or x >= 0.9
Step 1	0.79 (42.66)	0.79 (42.66)	0.79 (42.66)
Step 2	0.78 (45.92)	0.77 (45.45)	0.77 (46.08)
Step 3	0.83 (47.99)	0.82 (48.01)	0.81 (48.19)
Step 4	**0.85 (55.70)**	**0.85 (56.18)**	**0.85 (55.94)**

Table 6. Experimental results of third method

	(x <= 0.1 or x >= 0.95) or 0.125 < x < 0.875	(x <= 0.05 or x >= 0.9) or 0.125 < x < 0.875	(x <= 0.1 or x >= 0.95) or 0.1 < x < 0.9
Step 1	0.79 (42.66)	0.79 (42.66)	0.79 (42.66)
Step 2	0.79 (44.54)	0.77 (44.41)	0.78 (45.56)
Step 3	0.80 (48.23)	0.80 (49.32)	0.81 (51.46)
Step 4	0.82 (50.41)	0.81 (51.18)	0.83 (58.23)
Step 5	0.83 (52.84)	0.83 (52.91)	0.84 (59.11)
Step 6	0.84 (54.72)	**0.85 (54.21)**	**0.85 (59.82)**
Step 7	**0.85 (55.41)**	0.83 (56.14)	0.84 (60.02)

When using all of the training data, the F1 value of results 0.85, which is compared with our methods as the baseline method. Tables 4, 5 and 6 are the experimental results of our three methods.

Table 4 shows the experimental results of our first method by using three different sets of probability range values. From Table 4, we can find that only 55.62% of the sample data can arrive the expected experimental result 0.85, when the selection probability range is (0.125, 0.875). The effect of this experiment is the best for the first method.

Table 5 shows the experimental results of our second method. And we also apply three different sets of probability range values. From Table 5, we can see that only 55.70% of the sample data can arrive the expected experimental result 0.85, when the selection probability range is $[0, 0.1] \cup [0.95, 1]$. The effect of this experiment is the best for the second method.

Table 6 is the experimental results of our third method, which select 50% of the first method and 50% of the second method. From Table 6, it can be observed that only 54.21% of the sample data can arrive the expected experimental result 0.85, when the selection probability range is $([0, 0.05] \cup [0.9, 1]) \cup (0.125, 0.875))$. The effect of this experiment is the best result of third method and also the best result of these three methods. The result of experiment shows that our active learning approach requires 45.49% less training data. When F1 reaches a certain value, F1 tends to a stable state. In the meantime, there are few samples taken through the category probability strategy each time. Furthermore, those experiment results indicate we can achieve the expected experimental results without using all the data. As these useful samples, selected for each time, have a large impact on the classifier, we can use a small amount of data to achieve the same effect as a large amount of data.

5 Conclusion

In this paper, we propose a certainty criterion under the framework of deep learning to solve the problem of requiring a lot of manual data annotation. The certainty criterion has three selection method based on certainty criterion of classification. Through these three selection methods, the convolutional neural network is trained on the selected data iteratively, so that the model can achieve the expected results. In the experiment, it can be found that 45.49% of the manually labeled work can be saved in the best result. In our further work, we will investigate more text classification algorithms combined with our proposed method and apply our proposed method to other areas.

Acknowledgments. This work is partially supported by Shenzhen Science & Research projects (No: JCYJ20160331104524983) and Key Technologies Research & Development Program of Shenzhen (No: JSGG20160229121006579). We thank the reviewers for the insightful comments.

References

1. Yogatama, D., et al.: Generative and discriminative text classification with recurrent neural networks (2017). arXiv preprint, arXiv:1703.01898
2. Tu, Z., et al.: Modeling coverage for neural machine translation (2016). arXiv preprint, arXiv:1601.04811
3. Cao, Z., et al.: Improving multi-document summarization via text classification. In: AAAI (2017)
4. Conneau, A., et al.: Very deep convolutional networks for natural language processing (2016). arXiv preprint, arXiv:1606.01781
5. Joshi, A.J. Porikli, A.J., Papanikolopoulos, N.: Multi-class active learning for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 2372–2379 (2009)
6. Tur, G., Hakkani, D.: Combining active and semi-supervised learning for spoken language understanding. Speech Commun. **45**(2), 171–186 (2005)
7. Settles, B.: Active learning literature survey. Univ. Wisconsin, Madison **52**(55–66), 11 (2010)
8. MacKay, David J.C.: Information-based objective functions for active data selection. Neural Comput. **4**(4), 590–604 (1992)
9. Kapoor, A., Grauman, K., Urtasun, R., Darrell, T.: Active learning with Gaussian processes for object categorization. In: ICCV (2007)
10. Roy, N., McCallum, A.: Toward optimal active learning through monte carlo estimation of error reduction. In: ICML (2001)
11. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning (2016)
12. Li, X., Guo, Y.: Active learning with multi-label SVM classification. In: IJCAI (2013)
13. Seung, H.S., Opper, M., Sompolinsky, H.: Query by committee. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. ACM (1992)

14. McCallumzy, A.K., Nigamy, K.: Employing EM and pool-based active learning for text classification. In: Proceedings of the International Conference on Machine Learning (ICML), pp.359–367 (1998). Citeseer
15. Wang, K., et al.: Cost-effective active learning for deep image classification. IEEE Trans. Circ. Syst. Video Technol. **27**(12), 2591–2600 (2017)
16. Dasgupta, S.: Coarse sample complexity bounds for active learning. In: Advances in Neural Information Processing Systems (2006)

Research on Construction Method of Chinese NT Clause Based on Attention-LSTM

Teng Mao, Yuyao Zhang, Yuru Jiang(✉), and Yangsen Zhang

Institute of Intelligent Information Processing,
Beijing Information Science and Technology University, Beijing, China
maoteng_mt@163.com, yurujiang@123.com

Abstract. The correct definition and recognition of sentences is the basis of NLP. For the characteristics of Chinese text structure, the theory of NT clause was proposed from the perspective of micro topics. Based on this theory, this paper proposes a novel method for construction NT clause. Firstly, this paper proposes a neural network model based on Attention and LSTM (Attention-LSTM), which can identify the location of the missing Naming, and uses manually annotated corpus to train the Attention-LSTM. Secondly, in the process of constructing NT clause, the trained Attention-LSTM is used to identify the location of the missing Naming. Then the NT clause can be constructed. The accuracy of the experimental result is 81.74% (+4.5%). This paper can provide support for the task of text understanding, such as Machine Translation, Information Extraction, Man-machine Dialogue.

Keywords: NT clause · Attention-LSTM · Text understanding

1 Introduction

In Natural Language Processing (NLP), many tasks centers on text analysis, such as Machine Translation, Information Extraction and so on. The basic unit of a text is *sentence*, so the correct definition and cognition of *sentence* is very important for NLP.

How to define a sentence? Bloomfield proposed "Any sentence is an independent form of language, and should not be included in any larger form of language by any grammatical structure. The sentence can be divided by this fact." [1]. He emphasized the independence of sentences and the maximum of inclusion. For Indo-European languages, the basic pattern of sentences can be summed up as "subject-predicate" by virtue of whose linguistic characteristics [2]. Relatively speaking, there is no formal rule defining a Chinese *sentence* yet. Despite the odds, scholars have been trying to study Chinese sentences.

Zhu [3] defined sentence in such a way: "There is a pause before and after the sentence, and intonation represented the relative complete meaning of the language form". In the process of sentence cognition, pause and intonation are partially verifiable, but "relatively complete meaning" lacks operational standards. When Xing [4] annotated complex sentences, the principle was that full stop, sign and question mark separated sentences. However, the use of these punctuations in real texts is often arbitrary and lacks the constraints of linguistic norms. And, such sentence is incomplete in structure and meaning.

Cao [5] proposed the concept of Topic Chain, and defined that a topic can be shared by more than one sub-clauses. He regarded topics and subjects as two coexisting elements in a text. So the concept of his topic is narrower, and the topic chain structure can't cover the full text of Chinese texts.

Song [6], from the microscopic topic level, put forward the concept of Naming and Telling, defined the concept of Naming Structure, and formed NT clause theory. The theory of NT clause reveals the organization form of Chinese text in micro topic level, and proved its high coverage and operability in a number of corpus.

Based on the theory of NT clause, this paper uses an attention-LSTM model to identify Naming of punctuation sentences to construct NT clause in the text, which provides support for further text understanding.

2 NT Clause Theory

2.1 Basic Concepts

Punctuation Sentence (PS): It is a string of words, which is separated by commas, semicolons, periods, exclamation marks and question marks from text.

Example 1（钱钟书《围城》Zhongshu Gao *Fortress Besieged*)

①高松年发奋办公，②亲兼教务长，③精明得真是睡觉还睁着眼睛，④戴着眼镜，⑥做梦都不含糊的。

(Songnian Gao works hard, who is the chief of the dean of teaching, and is so shrewd that he still opens his eyes, wears glasses and can't be vague when sleeping.)

According to the definition of PS, there are 6 PSs in Example 1. Although PSs are not necessarily independent in structure and meaning, it has pause and intonation, which is the basic element of the sentence. The structure of the sequence of PS is constrained by grammar, so PS has a certain grammatical meaning. PS can be applied to the full text without ambiguity, so it is suitable for machine processing. All in all, PC is the basic unit of NT clause.

Naming-Telling: In the context of a text, if one component in PS is mentioned by other PS, the former is called the latter's Naming, and the latter is the former's Telling. Each Naming governs one or more PS as its Telling. The structure of this Naming-Telling relation is called the Naming Structure (NS). Using newline-indent schema to represent the header, each PS occupies one line, and indents to the right, and shrinks to the right side of the upstream Naming. Example 1 can be represented as Fig. 1.

高松年发奋办公，
(Songnian Gao works hard,)
亲兼教务长，
(who is the chief of the dean of teaching)
精明得真是睡觉还睁着眼睛，
(who is so shrewd that he still opens his eyes
戴着眼镜，
wears glasses
做梦都不含糊的。
and can’t be vague when sleeping.)

Fig. 1. The representation of Example 1

NT Clause: In many PSs, the Naming is missing and the propositional elements are incomplete. After a PS has been supplemented to the component that appears in context and should be used as a Naming of this PS, it is called as Naming Sufficient Clause (NSC). NSC consists of Naming and Telling, therefore it is also called NT clause. If a PS’s head is not missing, it is called a Naming Structure Independent Sentence (NSIS), also referred to as an Independent Sentence (IS). In Example 1, ① is a IS. The NT clause of all PSs in Example 1 is shown as Fig. 2.

高松年发奋办公，
(Songnian Gao works hard,)
高松年亲兼教务长，
(Songnian Gao is the chief of the dean of teaching)
高松年精明得真是睡觉还睁着眼睛，
(Songnian Gao is so shrewd that he still opens his eyes when sleeping.)
高松年精明得真是睡觉还戴着眼镜，
(Songnian Gao is so shrewd that he still wears glasses when sleeping.)
高松年精明得真是做梦都不含糊的。
(Songnian Gao is so shrewd that he can’t be vague when dreaming.)

Fig. 2. NT clauses of all PSs in Example 1

2.2 Construction NT Clause

In the case of human understanding, they can recognize the missing words of each PS in turn, that is to say, they can obtain NT clauses from PSs. This process relies on the semantic relationship between words and PSs, and it is also restricted by formal patterns. How does the machine carry out the same process?

In previous studies, some traditional methods have been used to construct NT clause [7–10], and the accuracy of single PS is 77.24% [10], sequence PS is 67.37% [10]. In recent years, the neural network has developed and perfected gradually. It has strong representation ability, can obtain grammar and semantic information in the context and has good performance on many Natural Language Processing tasks. Long short-term memory (LSTM) networks is a special form of recurrent neural networks (RNNs) [11]. It can process sequence data and obtain context information of data. Neural Attention Model is first used for machine translation. Bahdanau et al. utilized an attention-based contextual encoder to constructs a representation based on the generation context [12].

The core goal of Attention is to select more critical information among many information sources for the goal of current task. Therefore, this paper constructs a neural network model based on Attention and LSTM to construct NT clause.

3 Method of Construction NT Clause Based on Attention-LSTM

In this paper, the Chinese word is represented as w, and the PS in the text is represented as $c = \{w_1, w_2, \ldots, w_i, \ldots, w_m\}$, where w_i is the i^{th} word in c and m is the number of words in c. The NT clause of c is represented as $t = \{w_1, w_2, \ldots, w_j, \ldots, w_l\}$, where w_j is the j^{th} word in t and l is the number of words in t. The PS sequence is represented as $C = \{c_1, c_2, \ldots, c_i, \ldots, c_n\}$, where c_i is the i^{th} PS in C and m is the number of words in C, it is defined that $c_{i,i}$ is the i^{th} word w_i of PS c_i in C, the default is c_1 is an Independent Sentence. The NT clause sequence corresponding to the PS sequence is represented as $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$, where the default is $c_1 = t_1$, and $t_{i,j}$ is the j^{th} word w_j of the NT clause t_i in T.

In order to construct NT clause, this paper proposes a model based on Attention-LSTM, which can identify the location of the missing Naming. This task can be divided into two tasks: the first task is to construct the NT clause of a single PS; the second task is the dynamic construction of the NT clause sequence for sequence PS. In the first task the trained Attention-LSTM is used to identify the location of the missing Naming. Then the NT clause of a single punctuation sentence can be constructed. This paper focuses on the first task.

3.1 Attention-LSTM Model

In this paper, the input of Attention-LSTM is the NT clause t_i of PS c_i and the next PS c_{i+1} in the text, in which i represents the position of the PS in text, the target output $index\_pred_i$ is the location of the missing Naming of the c_{i+1} in t_i, which can be represented as $AttentionLSTM(t_i, c_{i+1}) = index\_pred_i$. The framework of Attention - LSTM model constructed is shown in Fig. 3, which has four parts: Word Embedding, Contextual Embedding, Attention -LSTM Layer, Output Layer.

Word Embedding maps each word in t_i and c_{i+1} to a hight-dimensional vector space. This paper uses pre-trained word vectors to obtain the fixed word embedding for each word. The output of Word Embedding are two matrices: $T \in R^{m \times d}$ for t_i and $C \in R^{n \times d}$ for c_{i+1}, where d is the dimension of word vector.

Contextual Embedding uses a BiLSTM on top of Word Embedding to obtain contextual information from surrounding words, so that if refines the embedding of the words. Hence $H\_T \in R^{m \times 2d}$ can be obtained from $T \in R^{m \times d}$, and $H\_C \in R^{n \times 2d}$ from $C \in R^{n \times d}$, where column is $2d$-dimensional because of the concatenation of the outputs of the forward and backward BiLSTM, each with d-dimensional output. Note that in order to ensure consistent representation performance, the same BiLSTM is respectively used in T and C.

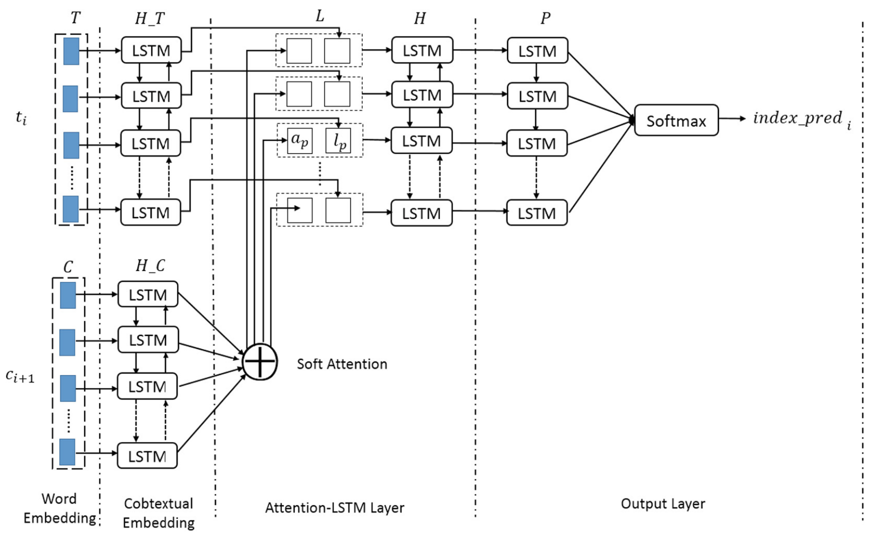


Fig. 3. The architecture of attention-LSTM model

Attention-LSTM Layer tries to match t_i against c_{i+1}. At position p of t_i, it first uses the standard word-by-word attention mechanism [13] to obtain attention weight vector: $a_p \in R^n$. a_{pq} indicates the degree of matching between the p^{th} token in t_i with the q^{th} token in c_{i+1}. Then this paper uses the attention weight vector: a_p to obtain a weighted version of H_C and combine it with the current token of t_i to form a vector $l_p \in R^{2d}$. After each token in t_i is processed, $L \in R^{m \times 2d}$ is obtained, and fed into a BiLSTM.

After this BiLSTM, $H \in R^{m \times 4d}$ is obtained, where column is $4d$-dimensional because of the concatenation of the outputs of the forward and backward BiLSTM, each with $2d$-dimensional output.

The input to Output Layer is H, which encodes the match information between t_i and c_{i+1}. H is fed into LSTM, and the expected output is the probability matrix of *index* at each location: $P \in R^m$, where p_k is the probability of *index* as k. Finally, P is fed into Softmax, the result is *index_pred*$_i$.

3.2 Construction of NT Clause for a Single PS

When constructing NT clause for a single PS, the input is the NT clause t_i of PS c_i and the next PS c_{i+1} in the text, in which i represents the position of the PS in text, the target output is the NT clause t_{i+1} of PS c_{i+1}. This paper first uses the Attention - LSTM model to predict position: *index_pred*$_i$, and then generates NT clause: $t\_pred_{i+1} = \left\{t_{i,1}, .., t_{i,index}, c_{i+1,1}, \ldots, c_{i+1,m}\right\}$. For example, Example 1 can be divided into 6 processes of constructing NT clauses for a single PS, such as t_1: {高松年发奋 办公(Songnian Gao works hard)} and c_2: {亲兼 教务长(who is the chief of the dean of teaching)} as input, target output is *index_pred*$_2$ = 1, t_pred_2 = {高松年亲兼 教务长(Songnian Gao is the chief of the dean of teaching)}.

4 Experiments

4.1 Dataset

The data used in this paper is a big sequence PSs with 11,790 PSs, which is derived from the Encyclopedia of fish. After manual annotation, it has already formed a sequence of NT clauses. When constructing NT clause, data can be represented in triples: $(t_i, c_{i+1}, index_i)$, where t_i is NT clause of c_i, c_{i+1} is next PS of c_i, *index* is align position. This paper transforms the original data into triples of this format, and a total of 11789 data are obtained, which is called All-DATA. Part of All-DATA is shown as follows.

	t_i	c_{i+1}	$index_i$
1	鮟鱇目 是 硬骨鱼纲 的 1 目 ，	有3 亚目 16 科 64 属 265 种 。	1
2	鮟鱇目 有 3 亚目 16 科 64 属 265 种 。	因 胸鳍 形成 假臂，	1
3	鮟鱇目 因 胸鳍 形成 假臂，	在 旧 的 系统 中 称为 柄鳍类 。	1
4	鮟鱇目 在 旧 的 系统 中 称为 柄鳍类 。	体 平扁 或 侧扁 ，	1
5	鮟鱇目 体 平扁 或 侧扁 ，	粗短 或 延长 。	2
6	鮟鱇目 体 粗短 或 延长 。	头 大 ，	1
7	鮟鱇目 头 大 ，	平扁 或 侧扁 。	2

In All-Data, this paper carries out a statistical analysis of the number of data corresponding to different *index*. The statistical results are shown in Fig. 4. It can be seen that the *index* range is 0–24 in All-Data. Note that $index = 0$ means that c_{i+1} is an Independent Sentence. The distribution of *index* is very uneven, of which $index = 1$ has the largest number of data, and *index* is mainly distributed between 0–10, accounting for 99.14% of the total data. Except for 0, with the increase of *index*, the number of data is getting smaller and smaller.

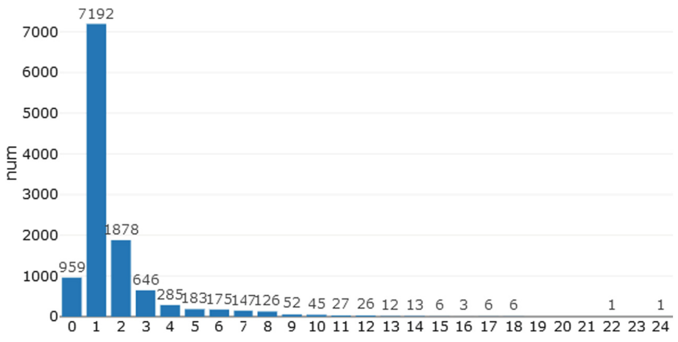


Fig. 4. Distribution of *index* in all-dataset

The training of Attention-LSTM requires a large number of data. Therefore, this paper divides the All-Data according to the proportion of 9:1. 9/10 of All-Data as ALSTM-Dataset, which is used to train Attention-LSTM model. 1/10 of All-Data as

Test-Dataset to do experiment on the construction of NT clause. In addition, the distribution ratios of different *index* in different dataset are calculated respectively, and the results are shown in Fig. 5.

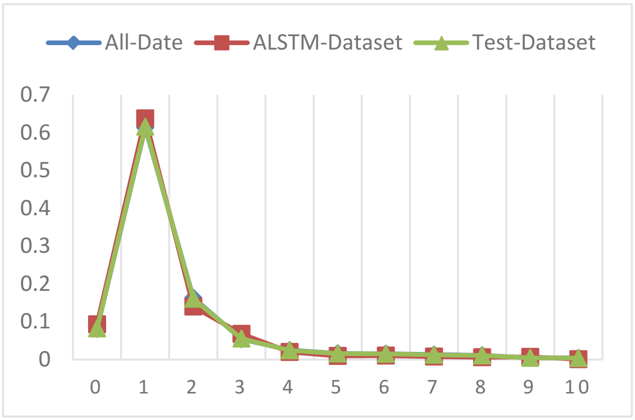


Fig. 5. Distribution of *index* in different dataset

It is can be seen that the lines of All-Data and ALSTM -Dataset almost overlap, and have little difference compared with Test-Dataset's. Therefore, the distribution ratio of *index* is almost the same in different datasets, and the three datasets can be regarded as undifferentiated datasets.

4.2 Attention-LSTM Model Details

All sequence in All-Dataset are tokenized by Jieba[1], a segmentation tool Pre-trained word vectors used in this paper is trained by word2vec module in gensim[2] from BaiduBaike croups[3], the dimension d is fix to 200. This paper uses the AdaDelta [14] optimizer, with a minibatch size of 64 and an initial learning rate of 0.5, for 30 epochs. A dropout [15] rate is 0.3 in LSTM and BiLSTM.

4.3 Training on Attention-LSTM

For training the Attention-LSTM model, this paper divides ALSTM-Dataset into Train-Data and Valid-Data according to the proportion of 9:1, in which Train-Data is used to train model, and Valid-Data is used to verify the model performance. In the training process, the result of each epoch is shown in Fig. 6, where line of Train-Acc almost overlap with Train-F1. In 0–5 epoch, Acc_{index} and F_1-*Score* of Train-Data and Valid-

[1] https://pypi.python.org/pypi/jieba/.

[2] https://radimrehurek.com/gensim/moels/word2vec.html.

[3] https://baike.baidu.com/.

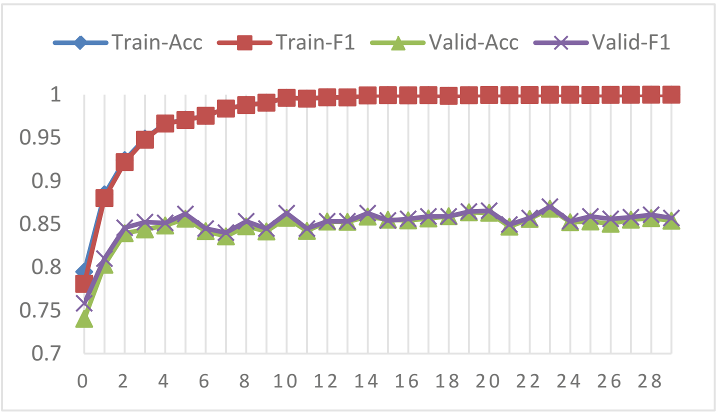


Fig. 6. Training process of model

Data rapidly increased. In 5–29 epoch, the results on the Train-Data is obviously better than Valid-Data, the phenomenon of over fitting appeared.

In order to verify the stability and reliability of the model, 10-fold cross-validation is carried out. The SNT-Dataset data is divided into 10 parts: 9 of them as training data, the remaining as the test data, then the mean value of the 10 results is regarded as the accuracy of the model. The results of 10-fold cross-validation are shown in Fig. 7.

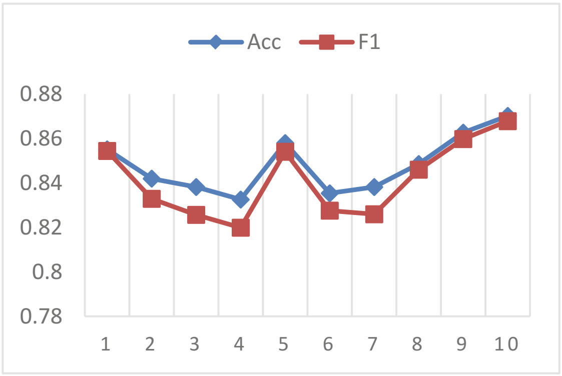


Fig. 7. Results of 10-fold cross-validation in first task

From Fig. 7, we can see that the maximum difference between results is 0.05%, and the fluctuation is small. Therefore the model has reliable stability.

4.4 Experiment on the Construction of NT Clause for a PS

In the process of constructing single NT clause for a PS, This paper separately uses 10 trained models, which were trained in the 10-fold cross-validation process to construct NT clause for 1,158 punctuation sentences in Test-Dataset. The results are shown in Fig. 8:

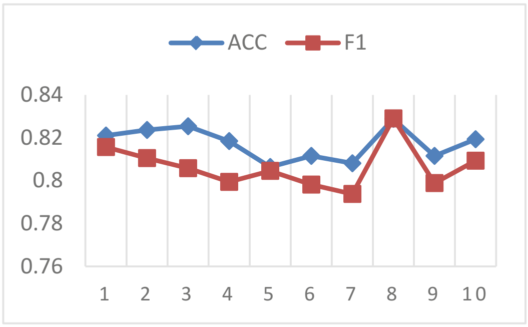


Fig. 8. Results of 10-fold cross-validation in first task

From Fig. 8, we can see that the maximum difference between results is 0.04%, and the fluctuation is small. Therefore the model has reliable stability. Finally, Acc_{index} is 81.74% and F_1-*Score* is 80.65%.Acc_{index} is improved by 4.5%.

This paper also analyzes the model performance in different *index* data, as shown in Fig. 9 (some *index* are not listed in the diagram because they did not appear in the prediction results).

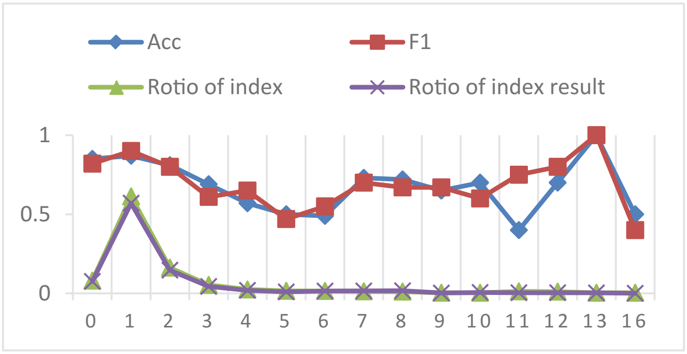


Fig. 9. Results in different *index* in first task

As can be seen from Fig. 9, *index* distribution of the result is basically the same as that in All-Data, which indicates that although *index* distribution is unbalanced, the effect on the model is less. The results of different *index* are closely related to the number of data. In general, the more data there is, the higher accuracy it shows, such as 0–9. However, when the data is small, the results are fluctuating and not reliable enough, such as 10–24.

5 Conclusion

This paper briefly describes the theory of NT clause and proposes a novel method based on Attention-LSTM model for construction NT clause. In the experiment, dataset is taken from Fish BaiduBaike corpus, and it is extracted after the data is manually labeled and processed, then it is used to train Attention-LSTM model. The results show that the method has certain advantages.Acc_{index} is 81.74%(+4.5). However, there are some deficiencies in this research, because the performance of neural network model depends on the quantity and quality of training data, and the number of different *index* data is different in this paper. Therefore, the performance of the model varies greatly on different indexes. Future work in this same direction of study is to add more features into the model, such as POS, grammar and so on, so as to reduce the difference on different *index*. All in all, NT clause construction based on NT clause theory can benefit other text understanding tasks, such as Machine Translation, Information Extraction, Man-machine Dialogue.

Acknowledgements. This work was supported by grants from National Nature Science Foundation of China (No. 61602044), National Nature Science Foundation of China (No. 61370139), Scientific Research Project of Beijing Educational Committee (No. KM201711232022).

References

1. Bloomfield, L.: Language. George Allen and Unwin, American (1933)
2. Jianming, L.: Characteristics of Chinese sentences. Chin. Lang. Learn. **1**, 1–6 (1993)
3. Dexi, Zh.: Grammar Lecture. The Commercial Press, China (2003)
4. Fuyi, X.: Research on Chinese Complex Sentences. The Commercial Press, China (2001)
5. Fengpu, C., Wang, J.: Sentence and Clause Structure in Chinese: A Functional Perspective. Beijing Language and Culture University Press, China (2005)
6. Rou, S.: Chinese Clause Complex and the Naming Structure. Empirical and Corpus Linguistic Frontiers. China Social Sciences Press, China. (To be published, 2018)
7. Yuru, J., Rou, S.: Topic clause identification based on generalized topic theory. J. Chin. Inf. Process. **26**(5), 114–119 (2012)
8. Yuru, J., Rou, S.: Topic Structure Identification of PClause Sequence Based on Generalized Topic Theory. Natural Language Processing and Chinese Computing, pp. 85–96 (2012)
9. Yuru, J., Rou, S.: Optimization of candidate topic clause evaluation function in topic clause identification. J. Beijing Univ. Technol. **40**(1), 43–48 (2014)
10. Yuru, J., Rou, S.: Topic clause identification method based on specific features. J. Comput. Appl. **34**(05), 1345–1349 (2014)
11. Sepp, H., Jürgen, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)

12. Dzmitry, B., Cho, K., Yoshua, B.: Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473 (2014)
13. Tim, R., Grefenstette, E., Hermann, K.M., et al.: Reasoning about entailment with neural attention (2015)
14. Zeiler, M.D.: AdaDelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
15. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR (2014)

The Research and Construction of Complaint Orders Classification Corpus in Mobile Customer Service

Junli Xu(✉), Jiangjiang Zhao, Ning Zhao, Chao Xue, Linbo Fan, Zechuan Qi, and Qiang Wei

IT System Department, China Mobile Online Services Company Limited, Zhengzhou, Henan, China
{xujunli_zb, zhaojiangjiang, zhaoning_zb, xuechao, fanlinbo, qizechuan, weiqiang}@coms.chinamobile.com

Abstract. Complaint orders in mobile customer service are the records of complaint description, which professional knowledge and information on customer's complaint intention are kept. Complaint orders classification is important and necessary to be established and completed for further mining, analysis and improve the quality of customer service. Constructed corpus is the basis of research. The lack of complaint orders classification corpus (COCC) in mobile customer service has limited the research of complaint orders classification. This paper first employs K-means algorithm and professional knowledge to determine complaint orders classification labels. Then we craft the annotation rules for complaint orders, and then construct complaint orders classification corpus. The corpus consists of 130044 complaint orders annotated. Finally, we statistically analyze the corpus constructed, and the agreement of each question class reaches over 91%. It indicates that the corpus constructed could provide a great support for complaint orders classification and specialized analysis.

Keywords: Mobile customer service · Complaint orders classification corpus K-means · Annotation rules

1 Introduction

Complaint orders are the description of customer's complaint and recorded by customer service staff, in order to do the mining work and analyze the complaint intention in future. Identifying categories of complaint orders is called complaint orders classification. Research on complaint orders classification in mobile customer service and capture the semantic information is significant for capturing customer's intention and specialized analysis. Therefore, it is necessary to construct complaint orders classification corpus (COCC) in mobile customer service.

Current modes of constructed corpus contain expert tagging, crowdsourcing tagging (such as Amazon Mechanical Turk[1], Crowd Flower[2]), group tagging. Considering

[1] Available at https://www.mturk.com/.
[2] Available at https://www.crowdflower.com.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 351–361, 2018.
https://doi.org/10.1007/978-3-319-99501-4_31

accuracy, reliability and economy, this paper focuses on constructing COCC in mobile customer service based on group tagging. An annotated complaint order is taken from COCC.

Complaint order 1:

******客户反映，本机上开通两个校讯通，收取3个校讯通功能费，
客户要求查明原因并回复说明，请处理。　费用问题

(******customer complains that two school newsletters have been opened, but functional fees for three school newsletters are charged. The customer requires to find out the cause and responds to the explanation, please handle it. cost question).

In complaint order 1, complaint order is separated into two parts by tabulator key, which are the content of complaint order and question category.

Recently, the corpus in public are appearing in dialogue [1], micro-blog [2–4], linguistic [5, 6], medical [7], etc. However, there is no public research on COCC in mobile customer service. The lack of COCC has limited the research of complaint orders classification.

This problem motivates us to construct COCC. Firstly, based on K-means algorithm and business knowledge, we divide complaint orders into 8 categories: marketing activities question, unknowing customization question, information security question, service question, cost question, business processing question, network question, and business use question. Secondly, business experts craft detailed annotation rules. Each complaint order is annotated by two skilled annotators respectively. Then, we repeatedly discuss and revise about bifurcation points with annotators, business experts without participation in annotating the corpus, aiming at determining inconsistent labels of complaint orders, then we construct the COCC. Finally, we statistically analyze the constructed corpus. As we know, COCC is the first publicly available and large dataset in mobile customer service field so far, which could provide a great support for classifying complaint orders, mining semantic information and conduct specialized analysis.

2 Related Work

Researches on corpus constructed have attracted considerable attention [8–12]. In customer service field, Yin et al. [13] identify 8 new metrics, named as tones, to describe emotional information by collecting and labeling online conversations of customer service on Twitter. They solve the problem that conventional metrics do not fit emotion analysis of online customer service conservation. Quan and Ren [14] introduce a blog emotion corpus for Chinese emotional expression analysis, which contains manual annotation of eight emotional categories (expect, joy, love, surprise, anxiety, sorrow, angry and hate), emotion intensity, etc. Chen and Nie [15] describe a parallel text mining system that finds parallel texts automatically on the Web, and generate Chinese-English parallel corpus for training a probabilistic translation model which translates queries for Chinese-English cross-language information retrieval. Feng et al. [16] propose Uyghur emotional words corpus construction based on CRFs.

Yang et al. [17] construct the corpus for named entities and entity relations on Chinese Electronic Medical Records, which comprises 992 medical text documents, and inter-annotator agreement of named entity annotations and entity relation annotations attain 0.922 and 0.895, respectively. However, due to the domain specificity, it is very difficult to be applied to the mobile customer service field. Moreover, there is no COCC in mobile customer service. In accordance with the characteristics of customer's complaint intention, we select data and construct COCC.

We select data from June, 2016 to May, 2017 comes from China Mobile Client Service System of 31 provinces. 60 million data are randomly selected, and the number of complaint orders from each province is roughly the same. To ensure privacy, we replace mobile phone number, address and other information in the complaint order with "******". Finally, we use 130044 complaint orders with more standardized format to construct COCC. Compared with other corpus, COCC takes full consideration of the structure, grammar and expression of complaint orders, such as regionalism, diversity and wide coverage, etc.

3 The Research and Construction of Complaint Orders Classification Corpus (COCC)

3.1 Classified Labels for Determining of Complaint Orders Based on K-Means and Professional Knowledge

The process of combining K-means and professional knowledge to determine classification labels for complaint orders is shown in Fig. 1. This process contains 5 phases: preprocessing, features extract, K-means cluster, classified label numbers for determining, classified label names for determining.

(1) In preprocessing phase, firstly, we process desensitization of mobile phone number, address and other information in the complaint orders. Then, the corpus is preprocessed with LTP segmentation toolkit[3]. Finally, we use Word2Vec toolkit[4] to train word embedding (the dimension of word embedding is 100) on the complaint orders corpus.
(2) In features extract phase, we statistically analyze the data and extract two features for comparison: (a) n-gram features (including unigram, bigram, trigram), we extract n-gram features and use tfidf for sentence representations. (b) sentence representations feature, we adopt the way of each word embedding summation in the order to obtain sentence representations.
(3) In K-means cluster phase, we use K-means algorithm ($5 \leq K \leq 12$) to cluster complaint orders based on unigram, bigram, trigram and sentence representations features respectively. From the cluster results, we can see that: (a) the effect based on unigram feature is not ideal, which may due to that unigram features do not utilize contextual information. (b) we vary K from 5 to 12 with an interval of 1, the

[3] Available at https://github.com/HIT-SCIR/ltp.
[4] Available at https://code.google.com/p/word2vec/.

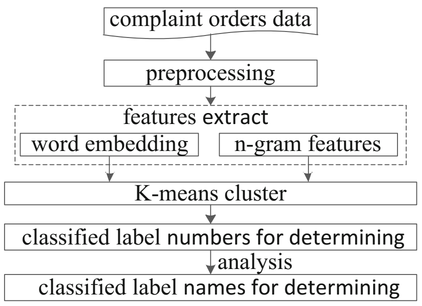


Fig. 1. The process of determining classification labels for complaint orders

performance based on trigram features is still not good. This may due to that trigram features are easy to generate data sparsity and cause probability distortion. (c) Clustering results based on sentence representations feature are also unsatisfactory, the reason is that: ① complaint orders are too long, which may bring noises to obtain sentence representations by the way of word embedding summation. ② templates of complaint orders are similar, which result in low discriminability of sentence representations. So this paper performs cluster experiments based on bigram features by varying K from 5 to 12 with an interval of 1.

(4) In classified label number for determining phase, we use error square sum of elbow method to determine the number of classified labels is 8. The relationship between error square sum and K is an elbow shape. The corresponding K value of elbow is the real clustering number of data.

(5) In classified label names for determining phase, 500 complaint orders from each class are randomly selected and given to the business experts for analysis based on bigram features and $K = 8$, aiming to determining the classification label names.

3.2 Annotation Rules

Making annotation rules of COCC is difficult, because it involves complicated professional knowledge. we craft the following annotation rules of each class based on the analysis of complaint orders and professional knowledge in mobile customer service.

(1) marketing activities question

Complaint orders that customer complains marketing activities are annotated “marketing activities question”. For example, customers are not satisfied with the rules of marketing activities, and complain that they can’t participate in marketing activities.

Annotation rule: the object of complaint order is about marketing activities, such as participation or withdrawal of marketing activities, dissatisfaction with rules about marketing activities, disagrees of propaganda and practical, without receiving gifts about marketing activities (including telephone fees/flow) on time, quality problems (including error telephone fees), etc.

(2) unknowing customization question

Without being aware of it, customer complains that some businesses are opened, cancel, change should be annotated unknowing customization question, such as unsubscribe.

Annotation rule: complaint orders that the customer clearly state that a product/business has been opened, changed or cancelled without being aware of it, are concluded “unknowing customization question”. The cost query arising from unknowing customization is classified as “unknowing customization question”.

(3) information security question

Complaint orders that the customer’s personal information or privacy information is got damage should be annotated “information security question”, such as password stolen.

Annotation rule: complaint orders about crank call/short messages, telephone fraud/short messages, adverse website report, subscriber information revealing, card replication, monitoring, telephone poisoning, etc., should be classified “information security question”.

(4) service question

Complaint orders about service channels, business related service attitude and service quality, etc., should be annotated “service question”, such as slow process for solving problem.

Annotation rule: complaint orders related to various service channels in mobile customer service should be labeled as “service question”, which contains: service attitude, service quality, service aging, service channels can’t provide normal service, etc. For service flow regulation of service channels, service opening time, discontent with service boundaries, service problem from official business channel, etc. For example, the adjustment of service time during Spring Festival, scope of service at night.

(5) cost question

Complaint orders about tariff and consumption situation should be annotated “cost question”, such as accounting query, charge after cancelling.

Annotation rule: complaint orders are annotated as cost question should meet either of two situations: (a) the customer have the accounting query for the use of tariff or some products/businesses, including the disagree of telephone fees, without using so much, charge after cancelling, having free resources but charging, etc. (b) the customer clearly state that he do not use flow but generate fees, such as the flow can’t access to the Internet.

(6) business processing question

Complaint orders about the failure to provide specific business/products for the mobile business and results in fault problems in the process of handling, are annotated “business processing question”. such as involving products or business, the failure to open, cancel, change, the suite is not successfully processed, the invoice can't be printed, the flow is not reached, the fixed tariff can't be cancelled, etc.

Annotation rule: When judging, confirm whether the object of complaint order is a product or business provided by Mobile Corporation. If so, we annotate complaint orders about products/business failed to open, change, cancel as "business processing question". Moreover, complaint orders about payment problems (such as fees recharged and fees reached is different) and unsuccessful registration of real-name system are also labeled "business processing question".

(7) network question

Complaint orders about going online, talking quality of speech, signal intensity, etc. are annotated "network question", such as no signal, weak signal.

Annotation rule: if complaint orders are about network and there are clear official notifications, which should be annotated "network question". If there is no reference to the surrounding people, it does not belong to the "network question". If the customer expresses the specific position in the complaint order, it is still classified as "network question", although it does not mention other people conditions.

(8) business use question

Complaint orders about a fault caused by the operation of the specific products/ business should be annotated "business use question", such as integral reduction without reason, display error 678 of broadband connection, broadband can't access to Internet.

Annotation rule: when judging, we need to confirm whether the object of complaint order is a product or business provided by the Mobile Corporation. If so, as long as the product/business can't provide the service normally, it should be annotated as "business use question". Moreover, complaint orders about broadband use problems, and integral use problems (including exchange telephone fare, call duration, short messages, the failure of virtual products to reach and logistics problems of exchanging integral for material object, etc.) also should be annotated "business use question".

3.3 Annotation Process

This paper takes complaint order as the annotated unit (no matter how many sentences are included, each complaint order has only one annotated label). Annotation process is shown in Fig. 2. Firstly, we analyze characteristics of complaint order, and craft detailed annotation rules. Secondly, rule makers explain annotation rules to annotators from the annotation team, and annotators carry out annotation on complaint orders. Finally, rule makers check annotated results, and we carry on the repeated discussion and revision about bifurcation points with annotators, business experts without participation in annotating the corpus, to determine inconsistent labels of complaint orders, then COCC is construct finally.

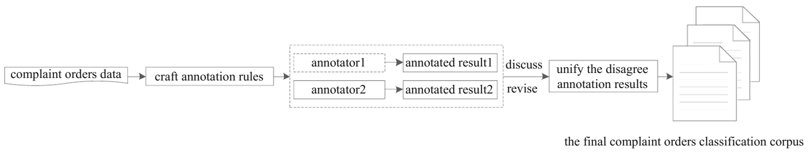


Fig. 2. Annotation process for the corpus

3.4 Basic Rules for Complaint Orders Classification Corpus

Understanding the customer's problem and taking customer's complaint intention as the main consideration is the premise of annotating the corpus. three kinds of complaint orders are discarded as follows:

- duplicate order: complaint order has no specific complaint content and only a number of the previous complaint order, which is directly marked as "duplicate order".
- template order: a single template built for the order, which has no free text information about customer's complaint intention, is labeled as "template order".
- invalid order: complaint order that customer's description is not clear, and it is hard to find the customer's complaint point, is labeled as "invalid order".

To ensure the accuracy and effectiveness of the annotated COCC, annotators must strictly obey the following annotation rules in annotation process.

(1) On the basis of customer requirement

When annotating, it is necessary to fill in the customer requirement from the perspective of customers. As long as the customer expresses the intention whether the requirement is reasonable or not, the intention of complaint order should be annotated.

(2) Abandon business experience

The purpose of annotation is to let the system learn the rules of COCC from massive data. As the system does not have business knowledge and reasoning ability, in annotation process, the business experience can't be considered and result annotated can't be obtained based on reasoning.

(3) Avoid speculation

When annotating complaint orders, annotators can't dig into the subjective conjecture, and can't speculate on customer's complaint intention beyond the intention expressed in the complaint order.

(4) Purely lean on literals

To ensure the effectiveness and accuracy of the annotated data, annotators should purely on literals in the complaint order, and do not make associations and experience judgments beyond literals.

3.5 Annotation Explanation of Bifurcation Points

Although clear labeling rules have been formulated, there are still bifurcation points due to some subjectivity in annotation process, and the flexibility and complexity of complaint orders recorded, which lead to the disagreement of annotation results. Bifurcation points are as follows:

- Whether the broadband problem is network question. We think that broadband is a product provided by Mobile Corporation, rather than a network service, so the broadband problem is annotated "business use question".
- Query fees problem generated by unknowing customization is prone to confusion. If the customer has clearly expressed that he did not subscribing the related business, and the fee is only caused by unknowing customization, the order should be annotated "unknowing customization question".
- Unknowing customization question and marketing activities question are easy to confuse. If we have verified that the customers subscribing the business from marketing activities, according to the special situation that "subscribing marketing activities without being aware of it should be labeled as marketing activities question" to unify them.
- Service question and business processing question are easy to confuse. Business processing question is only appearance of the customer's complaint. If the customer's final complaint intention is that some businesses fail to be processed because of the mistake made by the staff, this order should be annotated as "service question".

4 Statistics and Agreement Analysis

4.1 Statistics for Complaint Orders Classification Corpus (COCC)

COCC consists of 130044 complaint orders annotated[5]. Table 1 shows the statistical information about COCC, it indicates that:

(1) The three highest proportion of complaint orders are business use question, marketing activities question and cost question. Analyzing the Top3 questions of complaint orders are of great significance for quickly locating the categories of customer complaints and improving the quality of service. Moreover, information security question is only 2.29%, the main reason is that: the specialty of the mobile customer, the Mobile Corporation is strict with the management of customer's privacy information, which leads to the number of complaint orders belonging to information security question is very less.

(2) The average length of complaint orders from 8 categories is between 200 and 800 characters, among them the average length of the complaint orders from information security question is the longest, reaching 772.31 characters. This may due

[5] Available at https://github.com/zhng1200/COCC.

Table 1. Statistical information about COCC

Classified label names	#complaint orders	%complaint orders	Avg. length of complaint orders
Network question	10262	7.89	413.13
Cost question	17901	13.77	274.14
Marketing activities question	22702	17.46	418.38
Business use question	46438	35.71	452.63
Business processing question	11831	9.10	329.49
Service question	10004	7.69	383.91
Unknowing customization question	7925	6.09	344.82
Information security question	2981	2.29	772.31

(#complaint orders: the number of complaint orders. Avg. Length: The average number of Chinese characters.)

to that the complaint order's template from information security question is too long, and more detailed information is added by the customer service staff when they record the content of complaint order.

4.2 Agreement Analysis

Each complaint order is annotated the only question label, so the recall is 100%. The accuracy is used as the agreement (the number of complaint orders annotated by two annotators is same/the total number of complaint orders $\times$ 100%.) to measure the annotated effect. Table 2 shows the agreement analysis on COCC. $\text{Result}_{1\&2}$ is the agreement between the first annotation results and the second annotation results. $\text{Result}_{1\&\text{final}}$ is the agreement between the first annotation results and the final annotation results, and $\text{result}_{2\&\text{final}}$ is the consistency between the second annotation results and the final annotation results. It shows that:

(1) The agreements of 8 question labels reach over 91%. Artstein and Poesio [18] show that the agreement of annotated corpus reaches 80% can be considered as trustworthy. So the corpus we construct is reliable.
(2) The first column's agreements are generally lower than that the second column and the third column. This may due to that: the annotated results are influenced by subjective factors, and the final COCC is obtained by unifying the difference between the first annotated results and the second annotated results.
(3) The agreement of cost question and service question are relatively low, and the agreement of marketing activities question, network question and unknowing customization question are relatively high. Because the characteristics and distinctions of each question is different. For example, compared with other classes, cost question and service question annotation are more complicated.

Table 2. Agreement analysis on COCC

Classified label names	$\text{Result}_{1\&2}$ (%)	$\text{Result}_{1\&\text{final}}$ (%)	$\text{Result}_{2\&\text{final}}$ (%)
Network question	96.22	97.89	97.76
Cost question	91.21	93.76	92.83
Marketing activities question	96.06	97.01	96.94
Business use question	94.32	94.67	95.42
Business processing question	92.81	93.11	94.02
Service question	92.23	92.36	92.51
Unknowing customization question	96.11	97.27	97.15
Information security question	93.24	94.69	94.57

The agreement of COCC reaches over 91%. It indicates that COCC is effective and could provide a great support for classifying complaint orders, mining semantic information to carry on the specialized analysis.

5 Conclusion and Future Work

This paper constructs complaint orders classification corpus which consists of 130044 annotated complaint orders. Firstly, we combine K-means and professional knowledge to determine classification labels of complaint orders. Secondly, we craft detailed annotation rules and annotate the complaint orders. Finally, we statistically analyze the constructed corpus, and the agreements of each question class reach over 91%. As mentioned above, the purpose of corpus constructed is to study complaint orders classification, and to enrich the semantic information for thematic analysis. Our future work will focus on studying complaint orders classification on COCC, and to extract and integrate semantic information in mobile customer service field.

References

1. Lowe, R., Pow, N., Serban, I.V., Pineau, J.: The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-turn Dialogue Systems. arXiv preprint arXiv:1506.08909 (2015)
2. Hu, B.T., Chen, Q.C., Zhu, F.Z.: Lcsts: A Large Scale Chinese Short Text Summarization Dataset. arXiv preprint arXiv:1506.05865 (2015)
3. Yang, L.Y.: Research on the establishment and applications of public sentiment corpus based on micro-blog information. Office Inform. **22**, 015 (2016)
4. Xi, X.F., Zhu, X.M., Sun, Q.Y., Zhou, G.D.: Corpus construction for chinese discourse topic via micro-topic scheme. J. Comput. Res. Develop. **54**(8), 1833–1852 (2017)
5. Xue, N.W., Chiou, F.D., Palmer, M.: Building a large-scale annotated Chinese corpus. In: Proceedings of COLING, pp. 1–8. ACL, Taipei (2002)
6. Aksan, Y., Aksan, M., Koltuksuz, A.: Construction of the Turkish National Corpus (TNC). In: Proceedings of LREC, pp. 3223–3227. European Language Resources Association, Istanbul (2012)

7. You, Z.Y., Wang, Y.Q., Shu, H.P.: A corpus-based TCM symptoms of speech tagging. Electron. Technol. Softw. Eng. **21**, 177–178 (2017)
8. Brockett, C., Dolan, W.B.: Support Vector Machines for Paraphrase Identification and Corpus Construction. In: Proceedings of IWP, pp. 1–8. IWP, Iowa (2005)
9. Dolan, W.B., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. In: Proceedings of IWP, pp. 9–16. IWP, Iowa (2005)
10. Vincze, V., Szarvas, G., Farkas, R.: The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. BMC Bioinformatics **9**(Suppl. 11), S9 (2008)
11. Zou, B.W., Zhu, Q.M., Zhou, G.D.: Negation and speculation identification in Chinese language. In: Proceedings of ACL-IJCNLP, pp. 656–665. ACL, Beijing (2015)
12. Zhou, H.W., Yang, H., Xu, J.L., Kang, S.Y.: Construction of Chinese hedge scope corpus. J. Chin. Inf. Process. **31**(3), 77–85 (2017)
13. Yin, P.F., Liu, Z., Xu, A.B.: Tone analyzer for online customer service: an unsupervised model with interfered training. In: Proceedings of CIKM, pp. 1887–1895. ACM, Singapore (2017)
14. Quan, C.Q., Ren, F.J.: Construction of a blog emotion corpus for chinese emotional expression analysis. In: Proceedings of EMNLP, pp. 1446–1454. ACL, Singapore (2009)
15. Chen, J., Nie, J.Y.: Automatic construction of parallel English-Chinese corpus for cross-language information retrieval. In: Proceedings of ANLP, pp. 21–28. ACL, Stroudsburg (2000)
16. Feng, G.J., Yu, L., Tian, S.W.: Auto construction of uyghur emotional words corpus based on CRFs. Data Anal. Knowl. Discov. **27**(3), 17–21 (2011)
17. Yang, J.F., Guan, Y., He, B., Qu, C.Y., Yu, Q.B., Liu, Y.X., Zhao, Y.J.: Corpus construction for named entities and entity relations on chinese electronic medical records. J. Softw. **27** (11), 2725–2746 (2016)
18. Artstein, R., Poesio, M.: Inter-coder agreement for computational linguistics. Comput. Linguist. **34**(4), 555–596 (2008)

The Algorithm of Automatic Text Summarization Based on Network Representation Learning

Xinghao Song[1], Chunming Yang[1,3](✉), Hui Zhang[2], and Xujian Zhao[1]

[1] School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621010, Sichuan, China
yangchunming@swust.edu.cn
[2] School of Science, Southwest University of Science and Technology, Mianyang 621020, Sichuan, China
[3] Sichuan Civil-Military Integration Institute, Mianyang 621010, Sichuan, China

Abstract. The graph models are an important method in automatic text summarization. However, there will be problems of vector sparseness and information redundancy in text map to graph. In this paper, we propose a graph clustering summarization algorithm based on network representation learning. The sentences graph was construed by TF-IDF, and controlled the number of edges by a threshold. The Node2Vec is used to embedding the graph, and the sentences were clustered by k-means. Finally, the Modularity is used to control the number of clusters, and generating a brief summary of the document. The experiments on the MultiLing 2013 show the proposed algorithm improves the F-Score in ROUGE-1 and ROUGE-2.

Keywords: Text summarization · Network representation learning Graph clustering · Modularity

1 Introduction

Automatic text summarization is extracting a piece of concise text that can represent the important content of the original text from a large amount of text. It is an effective method to help people obtain main information quickly and efficiently. It saves users' time and resources, and solves the problem of information overload on the Internet. After the summarization was first proposed by Luhn [1] in 1958, it was used in grammatical, single and descriptive texts such as scientific literature and news. In recent years, Internet texts generated by users have the characteristics of large quantity, wide coverage, subjective and nonstandard language. These problems have challenged the traditional approaches based on statistics, topics. The graph-based methods abstract the semantic relationship between sentences as the vertex relationship of graph.

However, in the case of Internet texts with many fields and redundant contents, it will make the graph sparse and lead to a decline in the quality of summary. Network representation learning is an effective method for solving graph sparseness and reducing computational complexity in recent years. The principle is to use deep learning to map high-dimensional networks to dense low-dimensional vectors, and then use the low-dimensional vectors for further calculation. We propose a graph based summarization algorithm based on the learning algorithm. The vertices of network which represent the text were mapped into low-dimensional dense vectors through representation learning, and then use graph clustering to select appropriate sentences as the summary.

The remained of the paper is organized as follows. In Sect. 2, we summarize the research related to summarization based on the graph model, and analyze the existing problems. In Sect. 3, we introduce the summarization algorithm based on the representation learning. In Sect. 4, we show the experimental data, methods, and results. In Sect. 5, we summarize the method of this article. The contribution of this paper is to propose using network representation to solve the sparseness problem of graph, and use graph clustering to solve the redundancy problem in summary.

2 Related Work

Automatic text summarization can be divided into two types: abstractive and extractive. Abstractive method generates summary by understanding the main content of the text, it uses the NLP technology to understand the text. And it requires that the main content of a text be understood like human, and use a writing technique similar to human to "write" the summary, therefore the abstractive method is very difficult. After several decades of development, no good results have been achieved. With the development of deep learning, some researchers have also begun to use neural networks to achieve certain progress in abstractive summarization. DRGN [2] uses recurrent neural networks (RNN) to make the summary, however, there is still the problem that the generated summary information is inconsistent with the original text, and there are meaningless contents when summarizing multiple sentences.

Extractive method uses sentences as the basic elements; through the various characteristics of sentences, the importance of each sentence is calculated and the appropriate sentence is selected as the summary of the text. The types of extractive method can be divided into methods based on statistics, topics, graph models and machine learning. Graph model algorithm has a wide range of applications in the extractive way, because the topological structure of the graph can reveal important information and relationships between elements in the text; it can reflect more information than other non-graph model methods, so the summarization based on graph model become a prominent current method in recent years.

The earliest graph model work was proposed by Mani and Bloedorn [3], and the general graph model uses text elements such as sentences or words as the vertices of the graph, the relationships and information between the text elements represent the edges. Most of the graph-based methods are scoring graph vertices; they calculate the importance of the sentences in graph model, and get the summary by the sentence score. Such methods are generally called graph sorting algorithms, it has been proved that they are indeed effective in summarization.

LexRank [4] is a typical graph sorting algorithm, it constructs sentences graph by using the sentences in the text as vertices and using PageRank to score the vertices in the graph and obtain the most important sentences in the text. Ferreira [5, 6] incorporates the construction of multiple knowledge extension graph edges and takes into account the semantic relationships to construct sentence graphs, which makes the information richer and better results in more specialized text fields such as thematic information or linguistic knowledge. Giannakopoulos [7] uses words as the vertices of the graph model to extract the important components of the document using N-Gram graphs; it makes the entire process of the proposed summarization system provide rich information about the complex and contextual relationships between characters or words n-grams.

However, there are still the following problems in the above method:

1. Since the algorithms are based on vertex sorting, redundant content may appear in extracted summary sentences. For example, when there are two sentences in the text expressing the same content, the sorting algorithm will give the two sentences an approximate score, it will make two similar sentences in the summary.
2. The sparseness of graphs reduces the quality of summary. If the amount of text information is large, the graph model will be large and sparse, and the quality of the summarization will be affected to a large extent.

Based on the above problems, we propose a graph clustering automatic summarization algorithm based on network representation learning. This method constructs a sentence graph, and maps the graph vertex as a low-dimensional dense vector through the learning algorithm to solve the problems of graph sparseness and algorithm execution efficiency. Then we use the graph clustering to select the appropriate sentences as summaries, while integrating the group relations between sentences, we also solve the redundant information of the text.

3 Our Method

3.1 Graph Construction Through Sentence Similarity

First we need to construct a sentence graph, let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $v_i = \{w_1, w_2, \ldots, w_n\}$ represents a sentence of text T, where w_i represents a word. And E is the set of edges, we need to add undirected edges between sentences by calculating the similarity between sentences. Before calculating the sentence similarity, we need to vectorize the sentences. Here we use the TF and IDF values of the words in the sentence to construct the vector of the sentence.

In information retrieval, TF-IDF [8] or TFIDF, short for *term frequency–inverse document frequency*, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. TF (*Term frequency*) is defined by the formula:

$$tf(w,T) = \frac{t_w}{n_w} \tag{1}$$

Where t_w is the number of occurrences of word w in text T, n_w is the total number of the words. And IDF *(Inverse document frequency)* is defined by the formula:

$$idf(w,s) = \log\frac{n_s}{1 + df(w,s)} \tag{2}$$

Where n_s is the total number of text sentences, $df(w,s)$ is the number of sentences s that contains the word w.

Then we use the bag-of-words to represent each sentence s as an n-dimensional vector $\boldsymbol{a} = (a_1, a_2, .., a_n)$, where n is the number of all words in the text T and $a_i = tf(w_i, T) \times idf(w_i, s)$. The similarity between two sentences is defined as the angle cosine of two sentence vectors:

$$tf - idf - cosine(x,y) = \frac{\sum_{w \in x,y} tf^2(w,T)idf(w,x)idf(w,y)}{\sqrt{\sum_{x_i \in x}(tf(x_i,T)idf(x_i,y))^2}\sqrt{\sum_{y_i \in y}(tf(y_i,T)idf(y_i,y))^2}} \tag{3}$$

Given the ε, for any two sentences s_i and s_j in the sentences set S; if $tf - idf - cosine(s_i, s_j) \geq \varepsilon$, we add an undirected edge e_{ij} between v_i and v_j to the graph G. The method we use to build sentence graph is presented in Algorithm 1.

```
Algorithm 1 Build Sentence Graph
Input: Text $T$, Threshold $\varepsilon$
Output: Sentence Graph $G = (V,E)$
1:  Build $G$ with $n$ nodes, $n$ is the sentences number of $T$
2:  for $i = 0$ to $n$ do
3:      for $j = i + 1$ to $n$ do
4:          if $tf - idf - cosine(s_i, s_j) \geq \varepsilon$
5:              add $e_{ij}$ to $E$
6:      end for
7: end for
```

In the construction of graph G, the choice of the threshold ε will affect the sparseness of the graph. If ε is too large, it will cause the graph G to be too sparse and there are too many isolated points. If ε is too small, the graph G will be too dense to affect the result of the last cluster. Figure 1 shows the results of a graph constructed using 10 sentences as examples.

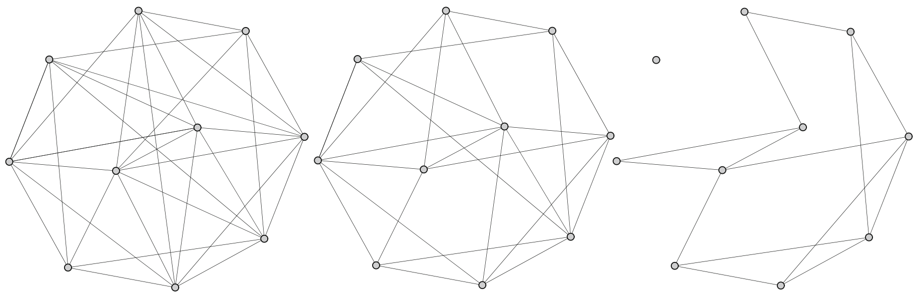


Fig. 1. Different ε sentence diagrams (from left to right ε values is 0.05, 0.1, 0.15)

3.2 Automatic Text Summarization via NRL

Network representation learning (NRL) algorithm refers to the algorithm for learning vector representation of each vertex from the network data, and the "network" refers to information networks such as social networks, web linked networks, and logistics networks. These networks are usually represented by data structure of "graph". Although graphs can be represented by adjacency matrices, adjacency matrices have great problems in computational efficiency, and the vast majority of adjacency matrices are 0, and the data is very sparse. This kind of data sparsity makes the application of fast and effective statistical learning methods difficult. Therefore, researchers turn to learning low dimensional dense vector representations for vertices in the network. Formally, the goal of NRL is to learn a real vector $a_v \in R^d$ for each vertex $v \in V$, in which the dimension d of the vector is far smaller than the total number of vertices $|V|$.

In 2013, the famous "word2vec" [9] used probabilistic neural networks to map words in natural language into low dimensional vector space, and then Perozzi *et al.* proposed *DeepWalk* [10] on the basis of the Skip-Gram model of "word2vec", for the first time the technology of deep learning was introduced into the NRL field. *DeepWalk* generalizes the idea of the Skip-Gram model that utilizes word context in sentences to learn latent representations of words, to the learning of latent vertex representations in networks, by making an analogy between natural language sentence and short random walk sequence. Given a random walk sequence with length L, $\{v_1, v_2, \ldots, v_L\}$, following Skip-Gram, *DeepWalk* learns the representation of vertex v_i by using it to predict its context vertices, which is achieved by minimize the optimization problem $-logPr(\{v_{i-w}, \ldots, v_{i+w}\} \backslash v_i | \Phi(v_i))$, where $\{v_{i-w}, \ldots, v_{i+w}\} \backslash v_i$ are the context vertices of vertex v_i within w window size. Subsequent work *node2vec* [11] and *LINE* [12] have further improved *DeepWalk*.

In order to remove redundant information in the document, we cluster the sentence vectors, clustering can group together similar sentences. When we select a summary sentence, we only need to select representative sentences from each cluster to compose the summary of the text; the clustering algorithm here we choose the k-means. One problem of the k-means algorithm is the choice of the number of clusters. In order to get better clustering results and solve the problem, we paper introduces "Modularity" to obtain the optimal number of clusters.

"Modularity" [13] is one measure of the structure of networks or graphs. It was designed to measure the strength of division of a network into modules (also called groups, clusters or communities). Networks with high modularity have dense connections between the nodes within modules but sparse connections between nodes in different modules. Given a community division C, the modularity is defined as:

$$Q = \frac{1}{2m}\sum\nolimits_{ij}\left(\boldsymbol{A}_{ij} - \frac{k_i k_j}{2m}\right)\delta\left(C_i, C_j\right) \tag{4}$$

Where $\boldsymbol{A}$ is the adjacency matrix, k_i is the degree of vertex v_i, $\delta\left(C_i, C_j\right)$ is defined as:

$$\delta\left(C_i, C_j\right) = \begin{cases} 1 & C_i = C_j \\ 0 & C_i \neq C_j \end{cases} \tag{5}$$

For different k of k-means, we select the clustering result with the largest modularity as the final sentence clustering result, and then select the vertex sentence with the largest degree in each cluster as the candidate summary sentence. The details of the method we used to generate automatic summarization via NRL is presented in Algorithm 2, here the NRL algorithm we used is the *node2vec*.

```
Algorithm 2  Auto Summarization via NRL
Input: Sentence graph $G$,Text $D$,dimension $d$
Output: Sentence of text summarization $s_1,\dots,s_n$
1:    $X \leftarrow$ node2vec$(G, d)$
2:    $C_{best} \leftarrow k - means(X, 2)$
3:    $Q_{max} \leftarrow Q(C)$
4:    for $i = 3$ to $n$ do
5:        $C \leftarrow kmeans(G, i)$
6:        if $Q(C) > Q_{max}$
7:            $C_{best} \leftarrow C$
8:            $Q_{max} \leftarrow Q(C)$
9:    end for
10:  $s_i \leftarrow maxdegree(C_i)$
```

4 Experiment

4.1 Datasets and Evaluation Method

We used the MultiLing 2013 dataset in our experiment; it contains a collection of texts in 40 languages, each text collection contains 30 single text files, and there are about 100–300 sentences in each single text. In the experiment, we select the data set of the English language as the experimental data set to generate a summary for each single text.

For evaluation, we used the automatic summary evaluation metric, ROUGE [14]. ROUGE is a recall-based metric for fixed-length summaries which is based on n-gram co-occurrence. It reports separate scores for 1, 2, 3, and 4-gram matching between the model summaries and the summary to be evaluated. In the experiment, we mainly used ROUGE-1 and ROUGE-2 evaluation standards. And the ROUGE-N is defined by the formula:

$$ROUGE - N = \frac{\sum_{S \in ref} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in ref} \sum_{gram_n \in S} count(gram_n)} \tag{6}$$

Where n stands for the length of the n-gram, $gram_n$ and $count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

4.2 Results

We evaluate the performance of our method against the following automatic text summarization algorithms:

- Text-Rank [15]: A graph-based ranking model using PageRank for text processing and text summarization.
- LSA [16]: A summarization methods based on LSA, which measure a content similarity between an original document and its summary.

The number of summary sentences used for Text-Rank and LSA is 10; for our method we set $\varepsilon = 0.14$ for Algorithm 1 and $d = 20$ for Algorithm 2. Table 1 lists the average results of 30 documents ROUGE-1, and Table 2 lists the average results of 30 documents ROUGE-2.

Table 1. Comparison of the results of the three algorithms ROUGE-1 ($\varepsilon = 0.14, d = 20$)

Method	Avg_Recall	Avg_Precision	Avg_F-Score
Text-Ran	0.5676	0.3268	0.4019
LSA	0.4386	0.3515	0.3876
Our	0.4446	0.4445	0.4332

Table 2. Comparison of the results of the three algorithms ROUGE-2 ($\varepsilon = 0.14, d = 20$)

Method	Avg_Recall	Avg_Precision	Avg_F-Score
Text-Ran	0.1214	0.1305	0.1256
LSA	0.1026	0.0839	0.0916
Our	0.1355	0.1372	0.1326

From the evaluation results on the MultiLing 2013 single text dataset; we can see that our method is better than the text-rank and LSA on the F-score of ROUGE-1 and

ROUGE-2. And the results of text-rank and our algorithm are all better than the LSA algorithm which based on semantic analysis. This shows that the graph model is indeed better than the traditional statistical method in summarization the algorithm. It can also be seen that the graph model represented by the learning algorithm has improved on the summary result.

4.3 Parameter Sensitivity

There are two important parameters ε and *d* in our method, we analyzed the F-score of ROUGE-1 and ROUGE-2 under different ε-values and dimensions *d* on the data set, the result is shown in Figs. 2, 3 and 4:

Experiments show that when ε is less than 0.14, the F value of ROUGE-1 and ROUGE-2 increases with the increase of ε. When ε is greater than 0.14, the summary results begin to decline. This is consistent with our previous analysis that sentences are too dense or sparse will affect the effectiveness of the summary, and map the sentence to low dimensional vector will result in better quality of text summary.

From Figs. 3 and Fig. 4, the effect of the dimension on the Rouge value can be seen that when the value of the dimension d is between 20 and 30, the result of the summary is the best, and the overall trend of the summary results is decreasing with the increase of the dimension. This shows that embedding a sentence into a low-dimensional vector space not only improves the efficiency of the algorithm but also improves the quality of the summarization.

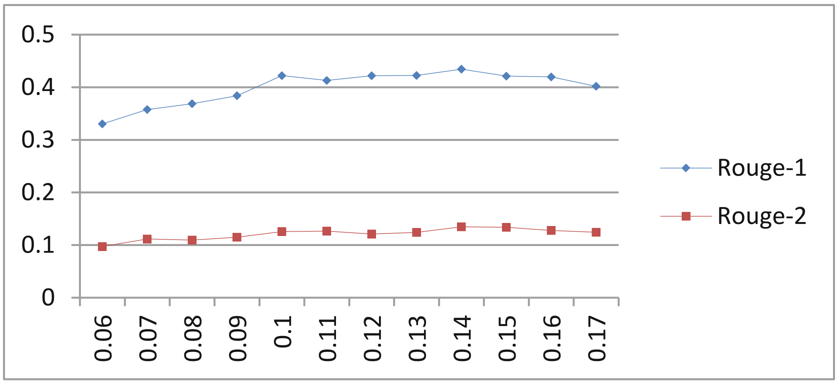


Fig. 2. Rouge with different ε values (d = 20)

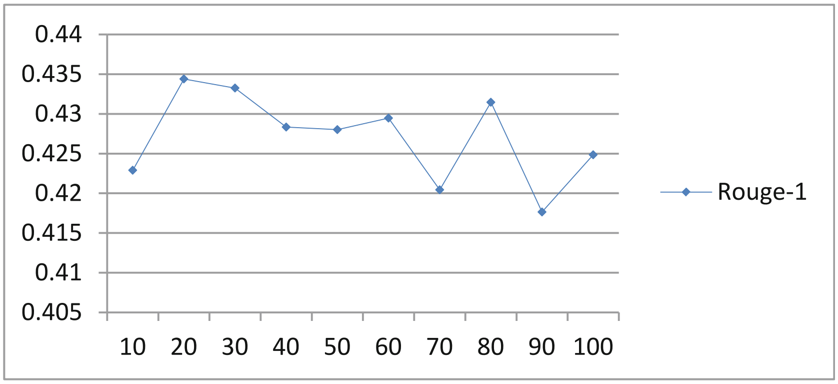


Fig. 3. Rouge-1 values for different dimensions d ($\varepsilon = 0.14$)

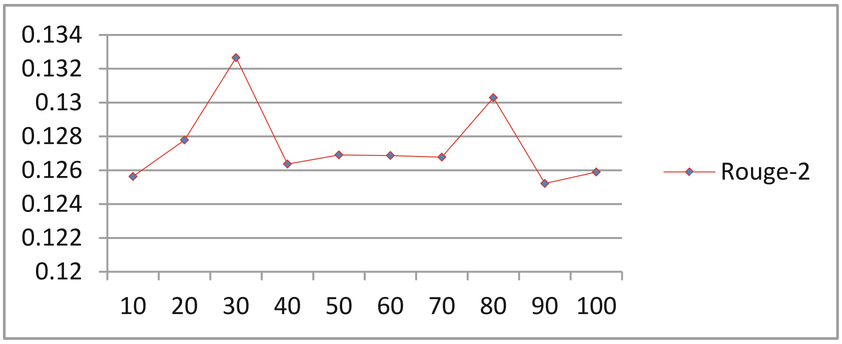


Fig. 4. Rouge-2 values for different dimensions d ($\varepsilon = 0.14$)

5 Discussion and Conclusion

In this paper, we use network representation learning to propose a new summarization algorithm based on graph model. This method is slightly better than the other two methods on the MultiLing 2013 document set. The NRL has attracted much attention since 2014. The current research is still at the initial stage and there is no representation learning method for the sentence graphs of the text. This paper only studies the clustering-based summarization methods after expressing the vector representation of the sentences through learning. In the future work, there are many works worthy of further research and exploration in the study of sentence graph and sentence vector based summarization algorithms.

Acknowledgement. This work is supported by the Ministry of education of Humanities and Social Science project (17YJCZH260), the Next Generation Internet Technology Innovation Project (NGII20170901), the Fund of Fundamental Sichuan Civil-military Integration (JMRHH01, 18sxb017, 18sxb028).

References

1. Luhn, H.P.: The automatic creation of literature abstracts. IBM Corp. (1958)
2. Li, P., Lam, W., Bing, L., et al.: Deep Recurrent Generative Decoder for Abstractive Text Summarization. arXiv preprint arXiv:1708.00625 (2017)
3. Mani, I., Bloedorn, E.: Multi-document summarization by graph search and matching. In: Proceedings of AAAI 1997, pp. 622–628 (1997)
4. Erkan, G., Radev, D.R.: LexRank: graph-based lexical centrality as salience in text summarization. J. Qiqihar Jr. Teach. Coll. **2011**, 22 (2004)
5. Ferreira, R., Freitas, F., Cabral, L.D.S., et al.: A four dimension graph model for automatic text summarization. In: IEEE/WIC/ACM International Joint Conferences on Web Intelligence, pp. 389–396. IEEE (2013)
6. Ferreira, R., Lins, R.D., Freitas, F., et al.: A new sentence similarity method based on a three-layer sentence representation. In: ACM Symposium on Document Engineering, pp. 25–34. ACM (2014)
7. Giannakopoulos, G., Karkaletsis, V., Vouros, G.: Testing the use of N-gram graphs in summarization sub-tasks. In: Proceedings of Text Analysis Conference, TAC 2008, pp. 158–167 (2008)
8. Salton, G., Fox, E.A., Wu, H.: Extended boolean information retrieval. Cornell University (1982)
9. Mikolov, T., Chen, K., Corrado, G., et al.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
10. Perozzi, B., Alrfou, R., Skiena, S.: DeepWalk: online learning of social representations, pp. 701–710 (2014)
11. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD, p. 855 (2016)
12. Tang, J., Qu, M., Wang, M., et al.: LINE: large-scale information network embedding, vol. 2, pp. 1067–1077 (2015)
13. Newman, M.E.: Fast algorithm for detecting community structure in networks. Phys. Rev. E Stat. Nonlinear Soft Matter Phys. **69**(6 Pt 2), 066133 (2003)
14. Lin, C.: ROUGE: a package for automatic evaluation of summaries. In: ACL, pp. 74–81 (2004)
15. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. UNT Scholarly Works, pp. 404–411 (2004)
16. Steinberger, J., Jezek, K.: Using latent semantic analysis in text summarization and summary evaluation. In: International Conference ISIM, pp. 93–100 (2004)

Semi-supervised Sentiment Classification Based on Auxiliary Task Learning

Huan Liu, Jingjing Wang, Shoushan Li(✉), Junhui Li, and Guodong Zhou

Natural Language Processing Lab, School of Computer Science and Technology, Soochow University, Suzhou, China
hliu0909@stu.suda.edu.cn, djingwang@gmail.com, {lishoushan,lijunhui,gdzhou}@suda.edu.cn

Abstract. Sentiment classification is an important task in the community of Nature Language Processing. This task aims to determine the sentiment category towards a piece of text. One challenging problem of this task is that it is difficult to obtain a large number of labeled samples. Therefore, a large number of studies are focused on semi-supervised learning, i.e., learning information from unlabeled samples. However, one disadvantage of the previous methods is that the unlabeled samples and the labeled samples are studied in different models, and there is no interaction between them. Based on this, this paper tackles the problem by proposing a semi-supervised sentiment classification based on auxiliary task learning, namely Aux-LSTM, which is used to assist learning the sentiment classification task with a small amount of human-annotated samples by training auto-annotated samples. Specifically, the two tasks are allowed to share the auxiliary LSTM layer, and the auxiliary expression obtained by the auxiliary LSTM layer is used to assist the main task. Empirical studies demonstrate that the proposed method can effectively improve the experimental performance.

Keywords: Sentiment classification · Auxiliary task · Auto-annotation samples

1 Introduction

With the development of the social media, people are accustomed to commenting on characters, events and products on the internet to express their opinion and sentiment. Sentiment analysis is a hot research direction that is produced under such a background, and text sentiment classification is the basic task in sentiment analysis. Specifically, the task of text sentiment classification is to determine the sentiment orientation of a text, i.e., *positive* and *negative*. For example, the text "*This book is simply boring!*" is considered as a *negative* sentiment. There are a large number of product comments in the electronic commerce platform. Correctly identifying the sentiment of these comments helps to understand the evaluation of the products, thereby improving the product quality and providing better service to customers. From the perspective of customers, they can judge the quality of products by analyzing the sentiment of comments, so as to make correct choices.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 372–382, 2018.
https://doi.org/10.1007/978-3-319-99501-4_33

Early research on sentiment classification mainly focus on supervised learning using only labeled samples [1, 2]. However, supervised learning requires a large number of labeled samples, studies in recent years have used semi-supervised learning method to reduce the dependence on labeled samples. For example, collaborative training (Co-training) [3], label propagation (LP) [4] and deep learning [5] are widely used in semi-supervised sentiment classification task. This paper mainly focuses on the method of semi-supervised sentiment classification.

At present, several studies have verified the effectiveness of the semi-supervised sentiment classification method, i.e., the use of unlabeled samples can improve the performance of sentiment classification. However, these existing methods have their own advantages and disadvantages, so it is difficult to determine which algorithm is suitable for which domain of sentiment classification. For example, the co-training algorithm can achieve good performance in the domain of Book and Kitchen, while the LP algorithm has better performance in the domain of DVD and Electronic [6]. Therefore, the semi-supervised sentiment classification method of integrated learning is also produced, which can improve the performance of sentiment classification through multiple semi-supervised learning methods. The above methods are aimed at reducing the error of annotating unlabeled samples in semi-supervised learning.

However, the unlabeled samples and labeled samples in the above semi-supervised learning algorithm are usually studied in two different models, ignoring the loss correlation information between models. In order to further study the link between the labeled samples and unlabeled samples information, and reduce the error of annotating unlabeled samples, we propose a semi-supervised sentiment classification method based on auxiliary task learning. The method firstly annotates unlabeled samples automatically, so as to obtain auto-annotated samples. Then, two sentiment classification tasks, the main task and the auxiliary task, are designed respectively according to the human-annotated samples and the auto-annotated samples. The main task obtains the auxiliary representation through the auxiliary LSTM layer, which is shared with the auxiliary task, and adds this auxiliary representation to the main task to assist the main task to complete the sentiment classification. The experimental results show that the proposed method in this paper can effectively improve the semi-supervised sentiment classification performance by utilizing the information of unlabeled samples.

The rest of our paper is structured as follows. Section 2 briefly discusses the related work. Section 3 gives the overall framework. Section 4 describes the algorithm of obtaining auto-annotated samples. Section 5 describes the semi-supervised sentiment classification based on auxiliary task learning in detail. Section 6 gives experimental settings and experimental results. Finally, the last section is the conclusion of this paper.

2 Related Work

Early sentiment classification research mainly focus on supervised learning. Ye et al. [1] compare three supervised machine learning algorithms of Naïve Bayes, SVM and the character based *n*-gram model for sentiment classification of the reviews on travel blogs for seven popular travel destinations in the US and Europe. Pang et al. [2]

introduce a variety of classification methods into the task of sentiment classification and achieves good classification results.

Since supervised learning requires a large number of labeled samples, the semi-supervised learning method has gradually attracted the attention of researchers. Wan [3] takes two different languages (English and Chinese) as two different views and adopts co-training method to semi-supervised sentiment classification. Zhu and Ghagramni [4] propose a graph-based semi-supervised learning method, namely label propagation (LP). The basic idea is to use the relationship between the samples to establish a relational complete graph model. In the complete graph, the nodes include labeled and unlabeled samples, and the edges represent the similarity of the two nodes. The labels of the nodes are transmitted to other nodes according to similarity. Xia et al. [7] make use of the original and antonymous views in pairs, in the training, bootstrapping and testing process, all based on a joint observation of two views. Sharma et al. [8] propose a semi-supervised sentiment classification method that uses sentiment bearing word embedding form to generate continuous ranking of adjectives with common semantic meaning. Yu and Jiang [9] borrow the idea from Structural Correspondence Learning and use two auxiliary tasks to help induce a sentence embedding that supposedly words well across domains for sentiment classification.

Different from traditional semi-supervised learning method, this paper proposes a semi-supervised sentiment classification method based on auxiliary task learning, which is constructing the joint loss function of the main task and the auxiliary task. It eliminates the need to add auto-annotated samples to human-annotated samples for modeling, thereby reducing the error of annotating unlabeled samples.

3 Overall Framework

Figure 1 shows the overall framework of semi-supervised sentiment classification based on auxiliary task learning. The basic idea is making human-annotated samples, i.e., labeled samples, and auto-annotated samples to learn from each other to assist completing the sentiment classification. Specifically, two sentiment classification tasks are designed, i.e., a main task and an auxiliary task. The main task implements the sentiment classification of human-annotated samples. The auxiliary task implements the sentiment classification of auto-annotated samples. The two tasks share the auxiliary LSTM layer, i.e., the auxiliary task completes the auxiliary sentiment classification through the auxiliary LSTM layer, and the main task completes the main sentiment classification with the auxiliary expression obtained by the auxiliary LSTM layer. Finally, joint learning the loss function of the two task to improve the performance of the main task.

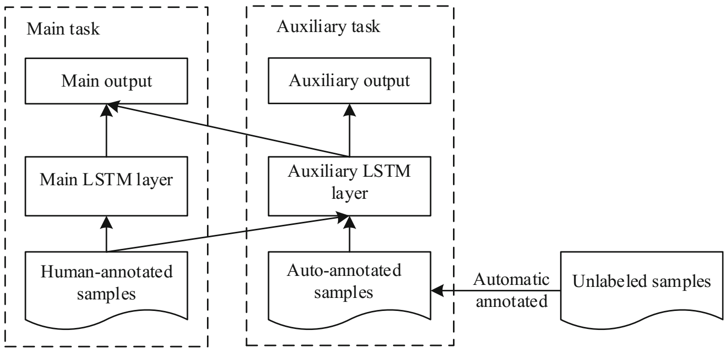


Fig. 1. The overall framework

4 Automatic Labeling Method for Unlabeled Samples

In order to complete the experiment of the semi-supervised sentiment classification method proposed in this paper, we need to obtain the label of the unlabeled samples. Firstly, we use the information gain (IG) algorithm [10–12] to extract 1000 *positive* feature words from *positive* labeled samples and 1000 *negative* feature words from *negative* labeled samples. The extraction method of *positive* feature words is to calculate the IG value of each word appearing in the *positive* samples, sort the IG values in descending order, and take the words corresponding to the first 1000 IG values as the *positive* feature words. The extraction method of *negative* feature words is the same as the *positive* feature words. Then, the unlabeled samples are divided into *positive* and *negative* categories according to the number of *positive* and *negative* feature words included in each sample in the unlabeled sample, wherein only the occurrence or non-occurrence of the feature words is considered, and the frequency is not considered. The specific algorithm is shown in Fig. 2:

```
Input:
    Positive feature set $P$;
    Negative feature set $N$;
    Unlabeled samples $U$;
Output:
    Auto-annotated samples $T$
Procedure:
    Loop until $U = \varnothing$
    1)  Calculate the number of positive words $C_P$ contained in the sample according to the set $P$
    2)  Calculate the number of negative words $C_N$ contained in the sample according to the set $N$
    3)  If $C_P > C_N$, label the sample as positive
        If $C_P < C_N$, label the sample as negative
        If $C_P = C_N$, label the sample randomly
    4)  Add the new auto-annotated sample into $T$
```

Fig. 2. The algorithm of obtaining auto-annotated samples

5 Semi-supervised Sentiment Classification Based on Auxiliary Task Learning

This section mainly introduces semi-supervised sentiment classification method based on auxiliary task learning. First, we introduce the basic LSTM neural network. Second, we propose a method of the sentiment classification of human-annotated samples sharing the auxiliary LSTM layer with the sentiment classification of auto-annotated samples, so as to make full use of the information of the unlabeled samples.

5.1 LSTM Model for Sentiment Classification

Long Short-Term Memory Network (LSTM) is a special kind of Recurrent Neural Network (RNN) and it aims to learn long-dependency correlations in a sequence [13]. We adopt the standard LSTM layer used by Graves [14].

First, the one-hot feature representation of the text T is used as an input to the LSTM layer and a new representation h is obtained. The formula is as follows:

$$h = LSTM(T) \tag{1}$$

Then, the output of the LSTM layer is propagated to the fully connected layer, and the output of the fully connected layer h^* is obtained by weighting the activation function, i.e.,

$$h^* = dense(h) = \phi(\theta^T h + b) \tag{2}$$

where ϕ is a non-linear activation function, and “Relu” is used as an activation function [15]. “Relu” will cause the output of some neurons in the network to be 0, which reduces the dependence between parameters and is closer to the biological activation model, alleviating the occurrence of overfitting. θ^T is the weight matrix and b is the bias.

In order to reduce the model complexity and prevent the network from overfitting the training samples, we add the dropout layer after the fully connected layer. The dropout layer randomly ignores some hidden units in the network during training. This paper uses the dropout layer as a hidden layer in the network:

$$h^d = h^* \cdot D(p^*) \tag{3}$$

where D denotes the dropout operation and p^* denotes the dropout probability. h^d is the output of h^* after the dropout layer operation.

Finally, using the softmax layer to complete the classification task, the predicted probability is obtained by the following formula:

$$p = softmax(W^d h^d + b^d) \tag{4}$$

where W^d and b^d are the parameters for the softmax layer. p is the conditional probability distribution over the two categories of sentiment, i.e., *positive* and *negative*.

5.2 Aux-LSTM Model for Semi-supervised Sentiment Classification

Figure 3 shows the overall architecture of the semi-supervised sentiment classification method based on auxiliary LSTM (Aux-LSTM), which mainly includes one main task and one auxiliary task.

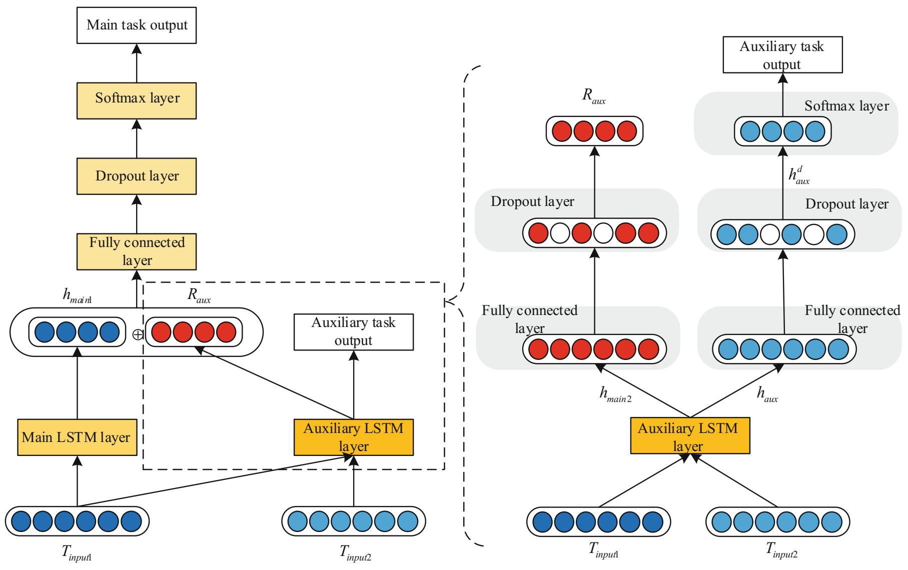


Fig. 3. Overall architecture of Aux-LSTM for semi-supervised sentiment classification

- **The main task-using human-annotated samples**

This part describes the main task of the semi-supervised sentiment classification method, consisting of the main LSTM layer and the auxiliary LSTM layer:

$$h_{main1} = LSTM_{main}(T_{input1}) \tag{5}$$

$$h_{main2} = LSTM_{aux}(T_{input1}) \tag{6}$$

where T_{input1} represents the one-hot feature representation of the human-annotated samples. h_{main1} and h_{main2} represent the output through the main LSTM layer ($LSTM_{mian}$) and the auxiliary LSTM layer ($LSTM_{aux}$) respectively.

Then, we feed h_{main2} to the fully connected layer and get an auxiliary representation R_{aux} through a dropout layer:

$$R_{aux} = dense(h_{main2}) \cdot D(p^*) \tag{7}$$

We can obtain a novel representation after concatenating above two representation h_{main1} and R_{aux} and use them as the input of a fully connected layer followed by a dropout layer in the main task:

$$h^{d}_{main} = dense(h_{main1} \oplus R_{aux}) \cdot D(p^{*}) \tag{8}$$

where $\oplus$ denotes the concatenate operator.

Finally, softmax layer is used to complete the classification, refer to 5.1 for details.

- **The auxiliary task-using auto-annotated samples**

This part describes the auxiliary task of the semi-supervised sentiment classification method. The auxiliary LSTM layer, which is the LSTM layer shared by the main task and the auxiliary task, has the same input sequence and weight as the auxiliary LSTM layer in the main task:

$$h_{aux} = LSTM_{aux}(T_{input2}) \tag{9}$$

where T_{input2} denotes the one-hot feature representation of the auto-annotated samples.

Then, we use the output of the auxiliary LSTM layer as the input to the hidden layer:

$$h^{d}_{aux} = dense(h_{aux}) \cdot D(p^{*}) \tag{10}$$

Finally, softmax layer is used to complete the classification, refer to 5.1 for details.

- **Joint learning-using both human-annotated and auto-annotated samples**

In order to better learn the parameters of the auxiliary LSTM layer in the model, we weighted the loss function of the main task and the auxiliary task to obtain the joint learning loss function, i.e.,

$$loss = \lambda(loss_{main}) + (1 - \lambda)(loss_{aux}) \tag{11}$$

where λ denotes the weight parameter, here we set 0.75 to reduce the noise of the auxiliary task. $loss_{main}$ is the loss function of the main task, while $loss_{aux}$ is the loss function of the auxiliary task. We take Adam [16] as our optimizing algorithm.

6 Experiments

6.1 Experimental Settings

In this paper, we use the corpus of Amazon product reviews, which is annotated by Blitzer et al. [17]. The corpus consist of four domains: Book, DVD, Electronics, and Kitchen. In the experiment of each domain, we select 100 instances as labeled data for training and 400 instance are used as test. The task of the experiment is to determine whether the sentiment of a text is *positive* or *negative*. According to the number of unlabeled samples, we make four sets of experiments. The number of training samples for each set of auxiliary tasks, i.e., the number of auto-annotated samples, is 750, 950, 1200, and 1450 respectively. The test samples for each set of experiments in auxiliary task is the same as the main task.

The classification feature used in the experiment is the one-hot representation of text. Specifically, we first construct a dictionary in descending order of occurrence frequency of word features in all corpus, and then use the subscripts of the word in the dictionary, thereby constructing the feature vectors of the samples. We use the LSTM neural network as the basic classification algorithm. The specific parameter settings of the LSTM neural network model are shown in Table 1.

Table 1. Parameters setting in LSTM

Parameter description	Value
Dimension of the LSTM layer output	128
Dimension of the full-connected layer output	64
Dropout probability	0.5
Epochs of iteration	30

We employ accuracy to measure the performance of the classification. It gives an average degree of the similarity between the predicted and ground truth label sets of all test samples, i.e.,

$$Accuracy = \frac{1}{m}\sum_{i=1}^{m} 1_{y_i = y_i^{'}} \tag{12}$$

where m is the number of all test samples, y_i is the true label and $y_i^{'}$ is the estimated label.

6.2 Experiments

For thorough comparison, we implement the following approaches to semi-supervised sentiment classification:

- **ME:** We employ the maximum entropy classifier in the MALLET Machine Learning Toolkit[1]. All parameters of the algorithm are set to default values. Here we only use the human-annotated samples to train the classification model.
- **Co-training:** The idea of the Co-training algorithm is to train multiple classifiers with multiple independent views, and then iteratively expands labeled samples and retrains them using that new labeled samples. In the implementation, we use each feature subspace as a representation view of text, and multiple feature subspaces correspond to different views.

[1] http://mallet.cs.umass.edu/.

- **LP:** LP algorithm uses the relationship between the samples to establish a relational complete graph model. In the complete graph, the nodes include labeled and unlabeled samples, and the edges represent the similarity of the two nodes. The labels of the nodes are transmitted to other nodes according to similarity.
- **Semi-stacking:** Li et al. [18] integrate two or more semi-supervised learning algorithms from an ensemble learning perspective. Specifically, they apply *meta*-learning to predict the unlabeled samples and proposed *N*-fold cross validation to guarantee a suitable size of the data for training the *meta*-classifiers.
  LSTM: We use the standard LSTM model, which includes a LSTM layer, a fully connected layer, and a dropout layer. The method used here for unlabeled samples is to train the one-hot feature with the labeled and unlabeled samples.
- **Aux-LSTM:** The method of the auxiliary LSTM described in Sect. 5.

In this paper, four sets of experiments are conducted on sentiment classification based on different numbers of auto-annotated samples. Figure 4 shows the experimental results which the number of the auto-annotated samples is 2900.

From Fig. 4, it can be seen that the results of sentiment classification using auto-annotated samples are significantly better than those using only human-annotated samples, i.e., ME. The method of Co-training has significantly improved in the domains of DVD, Book and Kitchen, but has not improved in the Electronic. However, the method of LP has slightly improved in the domains of DVD, Electronic and Kitchen, but there is almost no improvement in the Book. Semi-stacking combines the advantages of the Co-training and LP algorithms, and the accuracy in the four domains is significantly improved. The results of our method (Aux-LSTM) in four domains are clearly superior to those using only human-annotated samples for sentiment classification. Although performance similar to the Semi-stacking method was achieved in the domain of Kitchen (accuracy is 0.2% lower), performance in the DVD, Book, and Electronic was significantly higher than other semi-supervised methods. For example, in the domains of DVD, Book, and Electronic, our method has improved 4%, 2.3% and 2.8%

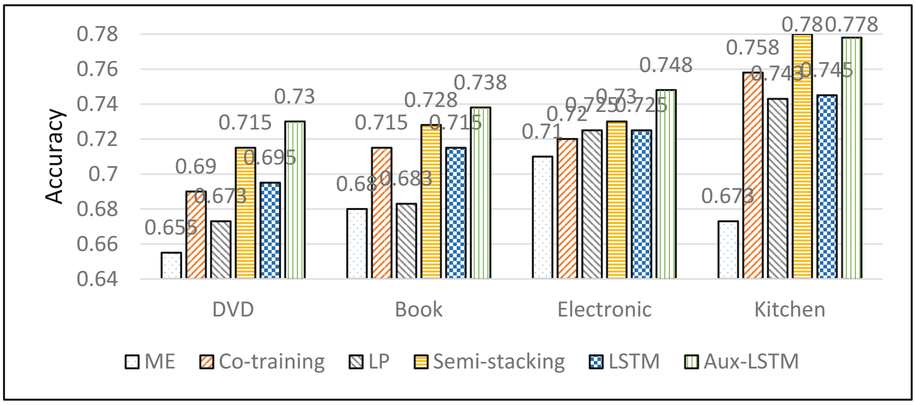


Fig. 4. Performances of different approaches to semi-supervised sentiment classification in four domains (The number of the auto-annotated samples is 2900)

respectively compared with Co-training method. This result fully shows that the Aux-LSTM model can effectively reduce the impact of incorrect auto-annotated samples on the semi-supervised sentiment classification task, and can better improve classification performance than other traditional semi-supervised sentiment classification methods.

7 Conclusion

This paper proposes a semi-supervised sentiment classification method based on auxiliary task learning. The method first annotates the unlabeled samples automatically with IG algorithm to obtain the auto-annotated samples. Then, it assists in sentiment classification of the human-annotated samples (main task) through the sentiment classification of the auto-annotated samples (auxiliary task). Finally, joint learning the loss function of the two task to improve the performance of the main task. The experimental results show that the semi-supervised sentiment classification method proposed in this paper can make full use of unlabeled samples to improve the performance of sentiment classification, and is superior to the current mainstream semi-supervised sentiment classification methods.

Acknowledgments. This research work has been partially supported by two NSFC grants, No. 61672366 and No. 61331011.

References

1. Ye, Q., Zhang, Z., Law, R.: Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. Expert Syst. Appl. **36**(3), 6527–6535 (2009)
2. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: Proceedings of EMNLP, Philadelphia, pp. 79–86 (2002)
3. Wan, X.: Co-training for cross-lingual sentiment classification. In: Proceedings of ACL-IJCNLP, Singapore, pp. 235–243 (2009)
4. Zhu, X., Ghahramani, Z.: Learning from Labeled and Unlabeled Data with Label Propagation. CMU CALD Technical report, CMU-CALD-02-107
5. Zhou, S., Chen, Q., Wang, X.: Active deep networks for semi-supervised sentiment classification. In: Proceedings of COLING, Beijing, pp. 1515–1523 (2010)
6. Li, S., Xue, Y., Wang, Z., et al.: Active learning for cross-domain sentiment classification. In: Proceedings of IJCAI, Beijing, pp. 2127–2133 (2013)
7. Xia, R., Wang, C., Dai, X., et al.: Co-training for semi-supervised sentiment classification based on dual-view bags-of-words representation. In: Proceedings of ACL-IJCNLP, Beijing, pp. 1054–1063 (2015)
8. Sharma, R., Somani, A., Kumar, L., et al.: Sentiment intensity ranking among adjectives using sentiment bearing word embeddings. In: Proceddings of EMNLP, pp. 547–552 (2017)
9. Yu, J., Jiang, J.: Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In: Proceddings of EMNLP, pp. 236–246 (2016)
10. Shi, H., Jia, D., Miao, P.: Improved information gain text feature selection algorithm based on word frequency information. J. Comput. Appl. **34**, 3279–3282 (2014)
11. Xu, J., Jiang, H.: An improved information gain feature selection algorithm for SVM text classifier. In: Proceddings of IEEE, pp. 273–276 (2015)

12. Zhang, H., Ren, Y.G., Yang, X.: Research on text feature selection algorithm based on information gain and feature relation tree. In: Proceddings of IEEE, pp. 446–449 (2014)
13. Hochreiter, K., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Graves, A.: Generating sequences with recurrent neural networks. Computer Science (2013)
15. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: JMLR W&CP, vol. 15, pp. 315–323 (2012)
16. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. Computer Science (2014)
17. Blitzer, J., Dredze, M., Pereira, F.: Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In: Proceedings of ACL, Prague, pp. 440–447 (2007)
18. Li, S., Huang, L., Wang, J., et al.: Semi-stacking for semi-supervised sentiment classification. In: Proceedings of ACL-IJCNLP, Beijing, pp. 27–31 (2015)

A Normalized Encoder-Decoder Model for Abstractive Summarization Using Focal Loss

Yunsheng Shi, Jun Meng, Jian Wang(✉), Hongfei Lin, and Yumeng Li

Dalian University of Technology, Dalian 116023, Liaoning, China
wangjian@dlut.edu.cn

Abstract. Abstractive summarization based on seq2seq model is a popular research topic today. And pre-trained word embedding is a common unsupervised method to improve deep learning model's performance in NLP. However, during applying this method directly to the seq2seq model, we find it does not achieve the same good result as other fields because of an over training problem. In this paper, we propose a normalized encoder-decoder structure to address it, which can prevent the semantic structure of pre-trained word embedding from being destroyed during training. Moreover, we use a novel focal loss function to help our model focus on those examples with low score for getting better performance. We conduct the experiments on NLPCC2018 share task 3: single document summary. Result showed that these two mechanisms are extremely useful, helping our model achieve state-of-the-art ROUGE scores and get the first place in this task from the current rankings.

Keywords: Summarization · Seq2Seq · Pre-trained word embedding Normalized encoder-decoder structure · Focal loss

1 Introduction

Summarization is the task to compress a piece of text to a shorter version that contains the main ideal of the original. There are two main approaches to summarization: extractive and abstractive. Extractive method is taking some sentences directly from the source text. While Abstractive method is generating novel words and sentences not featured in source text. Abstractive is more difficult than extractive because it transforms the source text to summary in human-like style, which require its incorporation of real-world knowledge. Recently, due to the success of seq2seq model with attention mechanism [1, 2], abstractive approaches get greatly development and most of studies are based on this model [3–5].

However, although these systems are promising, they are based on large-scale corpus, which may not perform well in small dataset. NLPCC2018 share task 3 is a single-document-summary task with small training dataset, which only contains 50,000 articles with summary. To get the better performance on this small corpus, we try to use the pre-trained word embedding like [4] but do not get the corresponding improvement because of that the semantic structure of pre-trained word embedding is very easy to be destroyed during training, causing the model to diverge (see Fig. 2 from Sect. 2).

After many attempts, we present a novel normalized encoder-decoder structure to solve this problem, in which we add a normalization layer in both encoder and decoder to prevent the semantic structure of pre-trained word embedding from being damaged by over training, which has an incredible effect in our experiments.

What's more, inspired by the reference [6], we also identify the same sample imbalance problem in text summarization that there is a small part of examples from the whole dataset the model difficultly fits well and gets low score on it. we propose a novel focal loss function to force our model pay more attention on those hard examples during training for getting better performance. This loss function is from above paper and has never been used in text summarization.

We apply our models in NLPCC2018 share task 3, and the experimental results show that these two mechanisms are extremely useful and help us achieve state-of-the-art ROUGE scores, getting the first place in this task finally.

2 Our Models

In this section, we describe (1) baseline sequence-to-sequence model (2) copy and coverage mechanisms (3) our normalized encoder-decoder structure and focal loss.

2.1 Sequence-to-Sequence Attentional Model

Our baseline model is similar to that of [7]. After being segmented, Chinese sentence was transformed to tokens w_s_i, which will be fed into the encoder (a single-layer bid-reaction LSTM), producing a sequence of encoder hidden state h_s_i, abstractly computed as:

$$h\_s_i = f(h\_s_{i-1},\ w\_s_i) \tag{1}$$

Where f computes the current hidden state given the previous hidden state h_s_{i-1} and source tokens w_s_i and can be either an Elman RNN unit, a GRU, or a LSTM unit. In this paper, we use LSTM unit as encoder and decoder.

During decoding, the structure is the same as encoder in addition to an input of last hidden state h_t_{t-1}, abstractly computed as:

$$h\_t_{\mathrm{t}} = f(h\_t_{\mathrm{t}-1},\ [w\_t_{t-1}; h\_t_{t-1}]) \tag{2}$$

Them, an attentional hidden state is produced as follows:

$$\tilde{h}_t = \tanh(W_c[c_t; h\_t_t]) \tag{3}$$

$$e_i^t = v^T \tanh(W_a[h\_t_t; h\_s_i] + b_a) \tag{4}$$

$$a^t = \mathrm{softmax}(e^t) \tag{5}$$

$$c_t = \sum\nolimits_i a_i^t h\_s_i \tag{6}$$

At first, we calculate the attention distribution a_i as the concat attention in [7] (4) (5). Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the context vector c_t (6). At last, an attentional hidden will be calculate by (3), where the W_c is learnable parameter.

Finally, we produce the vocabulary distribution P_{vocab} by $\tilde{h}$:

$$p_{vocab} = \mathrm{softmax}(W_{vocab}\tilde{h} + b_{vocab}) \tag{7}$$

During training, the loss for time step t is the negative log likelihood of the target word's vocabulary probability $P_{vocba}(w\_t_t)$ for that time step:

$$loss_{\mathrm{t}} = -\log p_{vocab}(w\_t_t) \tag{8}$$

2.2 Copy and Coverage Mechanisms

Pointer-Generator Network

To deal with the out-of-vocabulary (OOV) words problem, we take the pointer-generator network, which was proposed from [5] as copy mechanism (see Fig. 1). A copy probability $P_{copy} \in [0,1]$ will control the model whether generate the target word from P_{vocab} (7) or from a^t.

$$p_{\mathrm{copy}} = \sigma(W_{\tilde{h}}^T \tilde{h} + b_{copy}) \tag{9}$$

Where $\tilde{h}$ is an attentional hidden state calculated by Eq. (3).

The final word distribution is presented as follow:

$$P(w) = p_{copy}P_{vocab} + (1 - p_{copy})\sum\nolimits_{w_i = w} a_i^t \tag{10}$$

Note that if the p_{copy} is zero, the model will copy the word from the ith word of source sequence text whose a_i^t is the Maximum value in a^t. Otherwise, if p_{copy} is one, The model will generate target word from vocabulary distribution.

The loss function is the same as Eq. (8) by replace p_{vocab} with $P(w)$.

Coverage Mechanism

We also apply coverage mechanism [5] to prevent the model generate the same word repeatedly. In each decoder step t, we will calculate a coverage vector

$$co^t = \sum\nolimits_{k=1}^{t-1} a^k \tag{11}$$

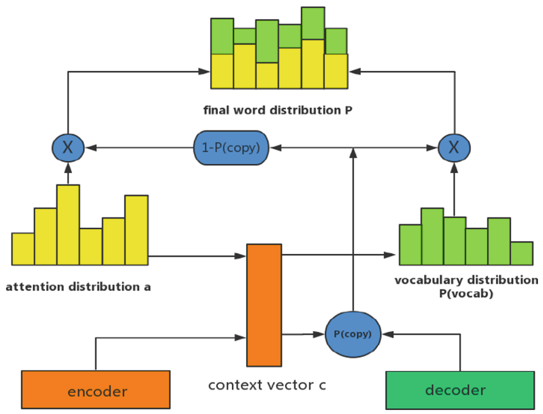


Fig. 1. Copy mechanism in the model. In the decoder step t, the final $P(w)$ is calculated by the sum of attention distribute a^t and vocabulary distribution P_{vocab}

The co^t will be used to prevent each word from being generated more than once by the penalty loss as follow:

$$loss_{\text{cov}}^t = \sum_{k=1}^{L} \min(co_k^t, a_k^t) \tag{12}$$

Where the L is the length of source text.

Finally, the total loss function for training the model is the weighted sum of negative log-likelihood $P(w)$ (10) and coverage loss (12), and the λ will be set to 1 as hyperparameter:

$$Loss = \frac{1}{T}\sum_{t=1}^{T} (-\log P(w_t) + \lambda loss_{\text{cov}}^t) \tag{13}$$

2.3 Normalized Encoder-Decoder Structure and Focal Loss

Normalized Encoder-Decoder Structure

Conventional seq2seq abstractive models are built for training on the large-scale corpus such as CNN/Daily Mail dataset (with 287,113 training examples) [8] and Gigaword corpus (with 5 million examples) [9], which may cause their hardly performing well on small datasets. We try to use pre-training word embedding such as word2vec [10] to deal with this problem.

However, during our experiments, we find that directly loading the pre-trained word embedding to model cannot make a good result, even make the model eventually diverge because of over training problem (see Fig. 2).

The semantic structure of pre-trained word embedding is very easy to be destroyed during training the whole seq2seq model even if fixed it before training.

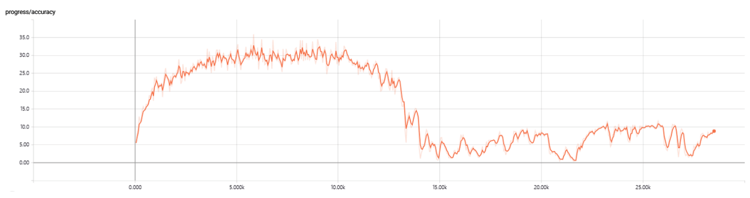


Fig. 2. The accuracy curve of baseline model with pre-trained word embedding.

After several trials, we find that using normalization layers can protect the semantic structure of word embedding from being destroyed during training, which is a new feature and never have been proposed in NLP.

Normalization layer was proposed for accelerating model's training. In this paper, we find it have ability of keeping the semantic structure of word embedding. we propose a normalized encoder-decoder structure (see Fig. 3). We add a normalization layer [11] between RNN and Embedding layer, apply this structure in both encoder and decoder. The result (see Fig. 5 from the Sect. 4) shows that our model is effectively prevent the model from diverging and maximizing the performance of pre-trained word embedding.

$$x_{norml} = \gamma \frac{x - \bar{x}}{\sqrt{\sigma_x^2 + \varepsilon}} + \beta \tag{14}$$

In Eq. (14), $\bar{x}$ is the mean value of x, σ_x^2 is the variance, γ and β are learnable parameters.

Focal Loss

Inspired by the approach in reference [6], there may have the same sample imbalance problem in text summarization. The training difficulty of each example in the training data is not the same. Moreover, the training difficulty of each word in a sentence (example) is also not the same, which can be represented by the $P(w)$ from Eq. (10). The $P(w)$ means that a word is an easy training example where its $P(w)$ closed to 1 because the model can predict it with confidence of one hundred percent, while it will be a difficultly training word if its $P(w)$ is small to zero.

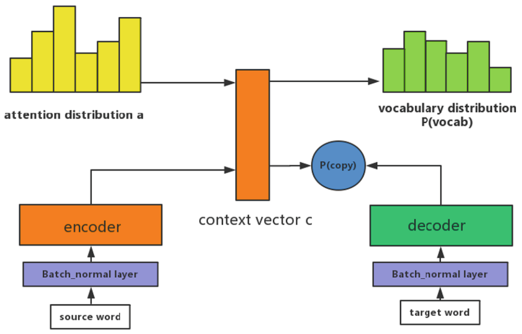


Fig. 3. Normalization layer will normalize the word vector before being put in RNN network, which will be applied in both encoder and decoder.

Consequently, we need to assign a soft weight to each word's loss for the model can pay more attention on those whose $P(w)$ is small for get better performance on it. The focal loss for it will be presented like follow:

$$loss_{focal\_loss}(w_t) = \alpha(1 - P(w_t))^{\gamma}(-\log(P(w_t)) \tag{15}$$

Where the $\alpha(1 - P(w_t))^{\gamma}$ is the weight of $loss_w$. When the $P(w_t)$ is small, the $loss_{focal\_loss}$ of this word will be enlarged by its weight, which will lead the model optimizing on this word's loss more. While if the $P(w_t)$ is large to one, the $loss_{focal\_loss}$ will be close to zero, and it does not provide any help for the final optimization.

The α and the γ are hyperparameters, we will set them for 0.25 and 1 for respectfully in this experiment.

The final loss will calculate by $\mathrm{loss}_{\mathrm{focal\_loss}}$ and coverage loss (12) like this:

$$Loss = \frac{1}{T}\sum_{t=1}^{T}(loss_{focal\_loss}(w_t) + \lambda loss^{t}_{\mathrm{cov}}) \tag{16}$$

3 Dataset and Experiments

3.1 TTNews Corpus

We training our model on the TTNews corpus provided by NLPCC2018 share task 3. We use all the news to pre-trained the word embedding. We take 5,000 news with summary as training set. We take 1,239 examples from the NLPCC2017 test set (because there are some examples in training set redundantly) as validation set. Finally, we will predict the summary for the NLPCC2018 test set (with 2,000 examples).

3.2 Experiments

We do with two main experiments. Firstly, we train the following five models (1) baseline model, (2) baseline model + word2vec, (3) baseline model + fixed word2vec, (4) normalized encoder-decoder model, (5) normalized encoder-decoder model + word2vec, comparing their accuracy to prove the validity of our model.

Secondly, we will take the best model from the first result, changing its NLL (negative log likelihood) into Focal Loss to do contrast experiment.

For all experiments, our basic seq2seq models have 600-dimensional hidden states bid-LSTM encoder and 1200-dimensional hidden states LSTM decoder. And each encoder and decoder only have 1 layer. We use a vocabulary of 60k words for both source and target and set embedding size as 400, and all the models are using copy and coverage mechanisms.

What's more, We use Adam as our optimizing algorithm by default hyperparameters (learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\varepsilon = 10^{-8}$).

During training and at test time we do not truncate the article or the summary. We train on a single Titan XP GPU with a batch size of 2. At test time our summaries are produced using beam search with beam size 5.

We trained all models for about 100,000 iterations, because we will get the best model in the first 4 epochs for preventing over fitting. Each epoch take about an hour.

We tried different truncated text to train the model, but we found it will reduce the model performance. We try different α and γ of FL and found that the model with FL will outperform than with NLL in the same learning rate when $\alpha = 0.25$ and $\gamma = 1$.

4 Validation and Result

4.1 Accuracy Evaluation

For quickly evaluating training model's performance on validation set, we count a accuracy, which is obtained by dividing the number of words appearing both in prediction summary and in target summary by total number of target summary words. The equation is presented as follow, in which the n is the number of examples.

$$\text{acc} = \frac{1}{n}\sum_{i=1}^{n}\frac{number(w_{predict} \cap w_{target})}{number(w_{target})} \tag{17}$$

We draw the training accuracy curve of baseline model+word2vec and of normalized encoder-decoder model+word2vec with the iteration numbers as X-axis and the accuracy as Y-axis (see Figs. 4 and 5).

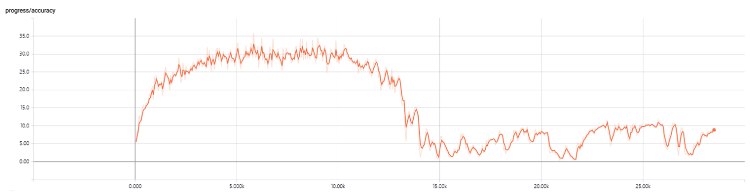


Fig. 4. The training accuracy curve of baseline model with pre-trained word embedding.

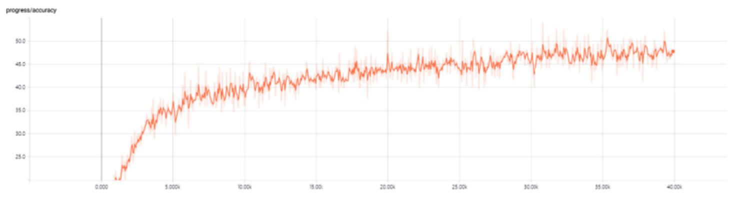


Fig. 5. The training accuracy curve of normalized encoder-decoder model with pre-trained word embedding.

Figures 4 and 5 show that the accuracy curve of baseline model with pre-trained word embedding will be shock to zero suddenly after 15,000 iterations. While for normalized encoder-decoder model, it is improving stably with the increase of iteration. Consequently, it is no doubt that our normalized encoder-decoder model is truly useful for keeping the semantic structure of pre-trained word embedding from being destroyed during training.

Moreover, to prove the superiority of our model, we train the five models and gets their accuracy on validation set after each epoch. The result is showed as Fig. 6.

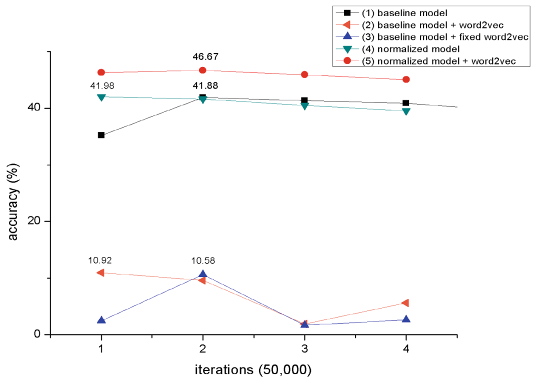


Fig. 6. The accuracy curves of the following five models on validation set: (1) baseline model, (2) baseline model+word2vec, (3) baseline model+fixed word2vec, (4) normalized encoder-decoder model, (5) normalized encoder-decoder model+word2vec.

There are three conclusions we can get from above Fig. 6. Firstly, directly loading the pre-trained embedding to baseline model will lead a terrible result that model will diverge during training even if fixed it (see curve (2) and curve (3)). Secondly, simply using normalized encoder-decoder model without pre-trained word embedding will not bring a significant improvement to result (see curve (1) and curve (4)). Finally, as we see, the normalized encoder-decoder model with pre-trained word embedding get a very high accuracy on validation set, which is at least 5% higher than others.

Consequently, there is a reason to believe that our model can maximum the performance of pre-trained word embedding.

We also evaluate normalized encoder-decoder model + word2vec with NLL (negative log likelihood) and with FL (focal loss) on validation (see Fig. 7). The model with FL is about 2% accuracy higher than that with NLL, in which we can learn that FL can really improve the model's performance compared with NLL.

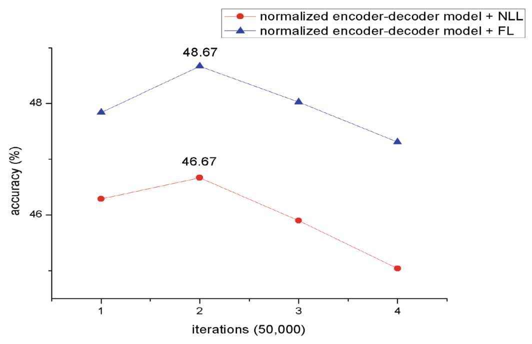


Fig. 7. The accuracy curves for normalized encoder-decoder model with NLL and with FL

4.2 ROUGE

We evaluate our models with the standard ROUGE metric [12]. We report ROUGE-F2, ROUGE-F4 and ROUGE-SU* for our validation set (with 1,239 examples) by the official ROUGE script (version 1.5.5) as NLPCC2018 share task 3 require.

We also evaluate our models through online evaluation. NLPCC2017 share task 3 testset: https://www.biendata.com/competition/nlptask03/ and NLPCC2018 share task 3 testset: https://biendata.com/competition/nlpcc2018/.

Showed as the follow Table 1, the performance of each model evaluated by ROUGE is the same as evaluated by Accuracy.

Table 1. ROUGE-F from validation set and average ROUGE from NLPCC2017 and NLPCC2018 online evaluation

Models	ROUGE-F2	ROUGE-F4	ROUGE-SU*	Testset 2017	Testset 2018
(1) baseline model	0.48764	0.18675	0.64069	0.30524	/
(2) baseline model+word2vec	0.14633	0.01136	0.14545	0.08666	/
(3) baseline model+fixed wor2vec	0.14425	0.01129	0.14396	0.08592	/
(4) normalization encoder-decoder model + NLL	0.49177	0.18714	0.64607	0.30544	/
(5) normalization encoder-decoder model + word2vec + NLL	0.50220	0.19615	0.65490	0.31650	/
(6) normalization encoder-decoder model + word2vec + FL	**0.51779**	**0.20895**	**0.66309**	**0.32450**	**0.31292**

Our model with focal loss achieve much better scores than others in each ROUGE points (Despite there is some discrepancy between the data we evaluate and the online assessment because of the lack of the specific value of rouge script's parameter using in online evaluation).

What's more, our best model (NLPCC2018_DLUT_815) is also the best model in NLPCC2018 share task 3 competition, getting the **31.292** average ROUGE scores, which is **1.3** points higher than the second place.

5 Conclusion

In this work, we presented a normalized encoder-decoder model to maximize the effect of pre-trained word embedding. What's more, a focal loss we take to help model fit those difficultly training examples better. The experiments proved these two mechanisms had obviously improved the performance of seq2seq model applying in text summarization.

Finally, we applied our model on NLPCC2018 share task 3: single document summary, achieving state-of-the-art ROUGE scores and getting the first place from the current rankings.

Acknowledgments. This research is supported by the National Key Research Development Program of China (No. 2016YFB1001103).

References

1. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Neural Information Processing Systems (2014)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representation (2014)
3. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 379–389 (2015)
4. Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, pp. 280–290 (2016)
5. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1073–1083 (2017)
6. Lin, T.-Y., Goyal, P., Girshick, R., He, K.: Focal loss for dense object detection (2018). arXiv preprint: arXiv:1708.02002
7. Luong, M.-T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Empirical Methods on Natural Language Processing (2015)
8. Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Advances in Neural Information Processing Systems, pp. 1693–1701 (2015)
9. Graff, D., Kong, J., Chen, K., Maeda, K.: English gigaword. Linguistic Data Consortium, Philadelphia (2003)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR, abs/1310.4546 (2013)
11. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Learning Representation (2015)
12. Lin, C.-Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches Out: ACL Workshop (2004b)

A Relateness-Based Ranking Method for Knowledge-Based Question Answering

Han Ni[1(✉)], Liansheng Lin[2], and Ge Xu[3]

[1] NetDragon Websoft Inc., Fuzhou, China
nihan@nd.com.cn
[2] NetDragon Websoft Inc., Fuzhou, China
linliansheng@nd.com.cn
[3] Minjiang University, Fuzhou, China
xuge@pku.edu.cn

Abstract. In this paper, we report technique details of our approach for the NLPCC 2018 shared task knowledge-based question answering. Our system uses a word-based maximum matching method to find entity candidates. Then, we combine editor distance, character overlap and word2vec cosine similarity to rank SRO triples of each entity candidate. Finally, the object of the top 1 score SRO is selected as the answer of the question. The result of our system achieves 62.94% of answer exact matching on the test set.

Keywords: Question answer · Knowledge base · Entity linking Relation ranking

1 Introduction

Automatic open-domain question answering has attracted great attention with the development of Natural Language Processing (NLP) and Information Retrieval (IR) techniques. One of the typical tasks named Knowledge-Based Question Answering (KBQA) is defined to retrieve a specific entity from knowledge base as the answer to a given question.

The challenge of retrieval-based KBQA is how to match unstructured natural language questions with structured data in knowledge base. To understand a question, it is necessary to figure out the topic entity and relation chain inside the question. Thus, topic entity linking and relation ranking are the most important modules in our system.

2 Related Work

Knowledge-based question answering is a challenging task in the field of NLP. The mainstream approaches can be divided into three categories: semantic parsing based [1–5], information extraction based [6–8] and retrieval based [9–11].

The semantic parsing based approaches translate natural language questions into a series of semantic representations in logic forms. They query the answer in knowledge base through the corresponding query statement. Yih et al. [12] present a semantic parsing method via staged query graph generation. Convolution neural network is used to calculate the similarities between question and relation chains.

The information extraction based approaches extract topic entities from questions and generate a knowledge base subgraph with the topic entity node as the center. Each node in the subgraph can be used as a candidate answer. By examining the questions and extracted information according to some rules or templates, they obtain the feature vectors of the questions. A classifier is then constructed to filter candidate answers based on input feature vectors. Yao and Van Durme [13] associate question features with answer patterns described by Freebase. They also exploit ClueWeb, mined mappings between knowledge base relations and natural language text, and show that it helps both relation prediction and answer extraction.

The idea of retrieval-based method is similar to that of information extraction based methods. The question and candidate answers are mapped to distributed representation. The distributed representations are trained on labeled data, aiming to optimize the matching function between the question and the correct answer. Zhang et al. [14] combine bi-directional LSTM with an attention mechanism to represent the questions dynamically according to diverse focuses of various candidate answers.

These approaches work well on the English open dataset WebQuestion. However, their performances on a Chinese KBQA dataset have not been presented before.

3 Our Approach

Figure 1 shows the system architecture of our approach. For each question, the system finds the entity candidates firstly. And then entity ranking and relation ranking are conducted seperately to assign each entity candidate and relation a rank score. Finally, in the answer ranking stage, the system finds the top 1 triple according to the entity score and relation score. The object entity of the top 1 triple is the answer of the question.

3.1 Entity Linking

Since the entity in the knowledge base has various name, such as Chinese name, English name, nick name, alias and so on, we build a Entity Map which maps these names to the original entity. In order to detect the topic entity in the question, we use a word-based maximum matching method to find entity candidates. First of all, the question is segmented by ltp[1] [15] segmenter. Then, we join the

[1] http://ltp.ai/.

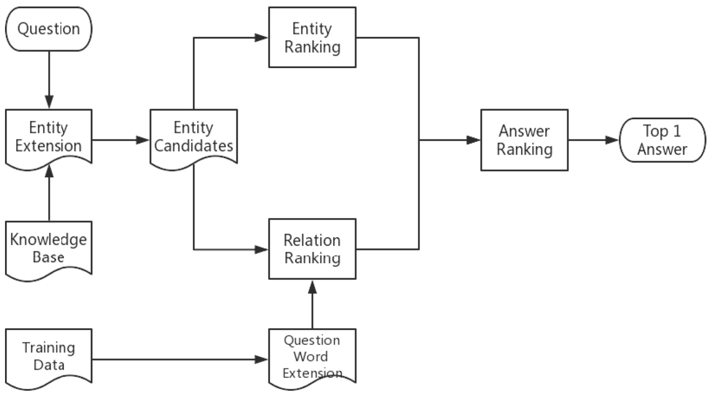


Fig. 1. System architecture

words in the question one by one and search it in the Entity Map. If it exists in the keys of Entity Map, the corresponding value to the key will be added to a entity candidates list.

Here is an example: consider the question, 马丁·泰勒青少年时在哪支球队踢球. After segmentation, we get a list of words 马丁·泰勒, 青少年, 时, 在, 哪, 支, 球队, 踢球. Then we filter stop words 时, 在, 支 and question words 哪, because they are impossible to be part of entity. After that, two sub-list of words are left, which are 马丁·泰勒, 青少年 and 球队, 踢球 . Then, we do word-based maximum matching for each sub-list. For sub-list, 马丁·泰勒, 青少年, we first join all the words (马丁·泰勒青少年) and search it in the Entity Map. Apparently, it is not a entity. Then, we shorten the string length by one. Now, we search 马丁·泰勒 and 青少年 in the Entity Map separately. They are both existing entity, so they are added to the entity candidates list. For sub-list 球队, 踢球, we conduct the same operation. In the end, we get entity candidates:
马丁·泰勒
马丁·泰勒(英国足球运动员)
马丁·泰勒(英国足球评论员)
青少年
《青少年》(2007年英国电影)
青少年(按年龄划分的社会群体)
青少年(2011年贾森·雷特曼导演美国电影)
踢球

3.2 Ranking

The knowledge base consists of millions of Subject-Relation-Object (SRO) triples. Each subject entity has dozens of Relation-Object pairs, each relation corresponding to only one object entity. Therefore, finding the answer to the question is equal to rank the relations of the subject entity and the object entity corresponding to the top one relation is supposed to be the answer.

After entity candidates have been found in the entity linking section, we can now collect all SRO triples of theses entities from the knowledge base. In this section, we combine editor distance score, character overlap score and word2vec cosine similarity score to rank each entity candidate and their relations.

Edit Distance. The edit distance is a way of quantifying how dissimilar two strings are to one another by counting the minimum number of operations required to transform one string into the other. The edit distance score we used is a variant of the original edit distance. Suppose that the original edit distance of two strings s_1 and s_2 is notated as $ed(s_1, s_2)$, the edit distance score we use is

$$score_{ed} = 1 - \frac{ed(s_1, s_2)}{max(len(s_1), len(s_2))} \tag{1}$$

Character Overlap. The character overlap is the number of overlapped characters in two strings. Greater character overlap suggests that the two strings are more topic related. We notate the character overlap score as $score_{co}$.

$$score_{co} = \frac{|set(s_1)| \cap |set(s_2)|}{|set(s_1)| \cup |set(s_2)|} \tag{2}$$

Word2vec Cosine Similarity. We train a word2vec model with a 20G chinese news corpus so that we can obtain a vector for each chinese word in the vocabulary. Then the string vector is computed as

$$v(s) = \sum_{w_i \in s} v(w_i) \tag{3}$$

so the word2vec cosine similarity of two strings is computed as the cosine similarity of two string vectors.

$$score_{w2v} = \frac{v(s_1) \cdot v(s_2)}{||v(s_1)|| \cdot ||v(s_2)||} \tag{4}$$

Related Score. The related score of two string is defined as:

$$score_{related} = score_{ed} + score_{co} + score_{w2v} \tag{5}$$

Entity Ranking. We rank entity candidate by how many object entity of the candidate are related with question.

Consider the question 巴西球员托罗是哪天出生的 . Suppose that there are more than one entities 托罗 in the knowledge base and they have different nationalities such as 波多黎各, 墨西哥, 巴西 etc. and different occupations such as

球员, 导演, 演员 etc. When we calculate the related score of each entity candidate and the question, apparently the related score of the entity of which the nationality is 巴西 and the occupation is 球员 will be higher than that of others. In experience, if the related score is greater than a threshold λ, then we think that the object entity is related with the question. So entity score is computed as

$$score_{award} = (1 + score_{ed}) \times (1 + score_{co}) \times (1 + score_{w2v}) \tag{6}$$

$$score_S = 1 \times \prod_{o \in S} score_{award}(O, Q) \tag{7}$$

$$score_{award}(O, Q) = \begin{cases} score_{award}(O, Q) & score_{award}(O, Q) \geq \lambda \\ 1 & score_{award}(O, Q) < \lambda \end{cases} \tag{8}$$

We tune the value of λ from 1.0 to 2.0, gap 0.1, and find that when $\lambda = 1.5$ it achieves the best result on the training data.

Relation Ranking. Before calculating the score, we remove the string corresponding to the entity candidate and related object entity for simplifying the computation. For example, after removing the string, the question 巴西球员托 罗是哪天出生的 becomes 是哪天出生的.

In addition, we also do question word extension. In some cases, the relation of entity does not appear in the question. For example, the relation to the question is supposed to be 出生日期 or 生日 (both refers to "birthday" in English), however, neither of them exists in the question. So, we map 哪天出生 to 出生日期 and 生日 , the latter is called the extension of question word.

Finally, we rank relations of each entity candidate by calculating the related score of each relation with the quesiton and the extension of question word.

$$score_R = \alpha \times score_{related}(R, Ext) + \beta \times score_{related}(R, Q \cup Ext) \tag{9}$$

where, R refers to relation, Ext refers to the extension of question word, and $Q \cup Ext$ refers to the union of question and ext. The α and β are weight factors. We set α to be 0.47 and β to be 0.53 according to the experiment.

Answer Ranking. For a SRO triple, we calculate the score as below:

$$score_{SRO} = score_R \times score_S \tag{10}$$

$$score_S = \begin{cases} score_S & score_{award}(O, Q) < \lambda \\ 1 & score_{award}(O, Q) \geq \lambda \end{cases} \tag{11}$$

where O refers to Object of SRO triple, Q refers to the question and λ is set to be 1.5 according to the experiment.

If the $score_{award}(O, Q) \geq \lambda$, it suggests that the object is a known fact and can not be the answer of the question shown as Eq. (11).

We rank SRO by multiplying the score of each relation and the score of corresponding entity candidate and get the Object from the top 1 SRO as the answer.

4 Experiments

4.1 Dataset

In this paper, we use the dataset provided by the NLPCC 2018 open domain KBQA shared task. The dataset includes 24,479 single-relation question-answer pairs for training, a Chinese knowledge base with 43M SRO triples, and 7M mapping data from mentions to entities. The test set contains 618 questions.

Since the mapping data is not what our system desires, we rebuild a Entity Map from mentions to entities with no word segmentation.

4.2 Setup

The word embeddings used in our system is pre-trained by gensim[2]. We use the skip-gram model [16] and the dimension is set to be 300.

4.3 Results

The results of our system achieves 62.94% of answer exact matching on the test set, which ranks 3rd place in the final leaderboard.

4.4 Error Analysis

We analyze the causes of the error cases (229 in total). 37.6% of errors are caused by entity linking and 27.9% are caused by relation ranking. 16.6% of errors are attributed to that the desire answer of the quesiton is the subject entity of the SRO triple and we can not use object entity to infer subject entity.

In addition, 5.2% of errors are caused by the confliction of knowledge base. For example, to the question 国脚黄博文在场上司职什么角色?, our answer is 中场, while the official answer is 前卫. However, in the knowledge base, the entity 黄博文 contains both triples 黄博文 ||| 场上位置 ||| 中场 and 黄博文 |||位置 ||| 前卫 .

For the last 16.6% of errors, in fact, we find the correct answers, but the official system judge them as incorrect ones. For example, to the question 泰国武里南联队是哪年成立的? , our answer is 1970年 , while the official answer is 1970. And in the preprocessing stage, we convert all characters in the knowledge base from full-width to half-width and convert all upper case letter to lower case, which also cause the official system to judge our correct answer as wrong one. For example, to the question 牛津联足球俱乐部的主席是谁? , our answer is 达利尔·伊尔斯 , while the official answer is 达利尔·伊尔斯 . If these cases caused by wrong judgement and knowledge base confliction are revised, the answer exact matching of our results will be 69.42%.

[2] https://radimrehurek.com/gensim/.

5 Conclusion

In this paper, we report technique details of our approach for the NLPCC 2018 shared task knowledge-based question answering. Our system uses a word-based maximum matching method to find entity candidates. Then, we combine editor distance, character overlap and word2vec cosine similarity to rank SRO triples of each entity candidate and get the object of the top 1 score SRO as the answer of the question.

We also try to use deep learning in entity linking and question-relation match. However, for entity linking, since the questions of test set are greatly different from that of training set, the model can not generalize from training data to test data. For question-relation match problem, it seems to be quite difficult to match thousands of questions to millions of relations, even by deep learning. And the number of relations in the training set is 4,385, however the number of that in the knoweldge base is up to 587,576. It is impractical to train a relation match model from such a small dataset. Even though, after replacing the provided mention2id with the entity extension built by us and revising some errors in the knowledge base, we also achieve good results with statistic and rule-based methods.

References

1. Zettlemoyer, L.S., Collins, M.: Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In: UAI (2005)
2. Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., Steedman, M.: Inducing probabilistic CCG grammars from logical form with higher-order unification. In: EMNLP (2010)
3. Liang, P., Jordan, M.I., Klein, D.: Learning dependency-based compositional semantics. In: ACL (2011)
4. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on freebase from question-answer pairs. In: EMNLP (2013)
5. Berant, J., Liang, P.: Semantic parsing via paraphrasing. In: ACL (2014)
6. Bast, H., Haussmann, E.: More accurate question answering on freebase. In: Information and Knowledge Management (2015)
7. Fader, A., Zettlemoyer, L., Etzioni, O.: Open question answering over curated and extracted knowledge bases. In: Knowledge Discovery and Data Mining (2014)
8. Yao, X.: Lean question answering over freebase from scratch. In: ACL (2015)
9. Bordes, A., Chopra, S., Weston, J.: Question answering with subgraph embeddings. In: EMNLP (2014)
10. Bordes, A., Weston, J., Usunier, N.: Open question answering with weakly supervised embedding models. In: ECML (2014)
11. Dong, L., Wei, F., Zhou, M., Xu, K.: Question answering over freebase with multicolumn convolutional neural networks. In: ACL (2015)
12. Yih, W., Chang, M.-W., He, X., Gao, J.: Semantic parsing via staged query graph generation: question answering with knowledge base. In: ACL (2015)
13. Yao, X., Van Durme, B.: Information extraction over structured data: question answering with freebase. In: ACL (2014)

14. Zhang, Y., Liu, K., He, S., Ji, G., Liu, Z., Wu, H., Zhao, J.: Question answering over knowledge base with neural attention combining global knowledge information. arXiv:1606, p. 00979 (2016)
15. Che, W., Li, Z., Liu, T.: LTP: A Chinese language technology platform. In: Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations, pp. 13–16. Association for Computational Linguistics, August 2010
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space, 7 September 2013. arXiv:1301.3781v3 [cs.CL]

A Sequence to Sequence Learning for Chinese Grammatical Error Correction

Hongkai Ren[1,2], Liner Yang[1,2](✉), and Endong Xun[1,2]

[1] Beijing Advanced Innovation Center for Language Resources, Beijing, China
[2] School of Information Science, Beijing Language and Culture University, Beijing, China
renhongkai27@gmail.com, yangtianlin08@gmail.com, edxun@126.com

Abstract. Grammatical Error Correction (GEC) is an important task in natural language processing. In this paper, we introduce our system on NLPCC 2018 Shared Task 2 Grammatical Error Correction. The task is to detect and correct grammatical errors that occurred in Chinese essays written by non-native speakers of Mandarin Chinese. Our system is mainly based on the convolutional sequence-to-sequence model. We regard GEC as a translation task from the language of "bad" Chinese to the language of "good" Chinese. We describe the building process of the model in details. On the test data of NLPCC 2018 Shared Task 2, our system achieves the best precision score, and the $F_{0.5}$ score is 29.02. Our final results ranked third among the participants.

Keywords: Grammatical Error Correction
Convolutional Sequence to Sequence Model
Neural machine translation

1 Introduction

The rapid development in China attracts people from all over the world to learn Chinese. Chinese is a historically influential and versatile language. Chinese is unique in many aspects as opposed to English and other languages. One of the distinction worth mentioning is its lack of verb conjugations and plural suffixes. Besides, the sentence expression is very flexible, which means that the rearrangement of word order in various ways may not impact on the sentence meaning. While handling grammatical complexity comes very naturally to native Chinese speaker, to be proficient and competent is very challenging to CSL (Chinese as Second Language) learners. Therefore, it is practical to develop a system automatically correcting grammatical errors, which is the goal of the NLPCC 2018 Shared Task 2.

English grammar correction has been studied for many years with great progress. In particular, after Ng et al. [13] organize the CONLL-2013 shared task, a large number of methods based on statistics and neural networks emerge, which greatly promote the study of English grammar error correction. Chollampatt et al. [1] propose the phrase-based statistical machine translation (SMT) approach, in which GEC is firstly treated as a translation task. By training the model to "translate" the "bad" English into the "good" English, they carry out very promising results. Following the previous work, several neural encoder-decoder approaches have been put forward for this task. Chollampatt et al. [2] firstly employ a convolutional encoder-decoder model that achieved good performance for GEC. Among those, Junczys-Dowmunt et al. [9] demonstrate parallels between neural GEC and low-resource neural MT. They successfully adopt several methods from low-resource MT to neural GEC, and achieve the state-of-the-art results on this task.

While English Grammatical Error Correction is being intensively studied for years, the same task on Chinese is poorly focused until very recently. In 2014, Yu et al. [10] organize a Shared Task on Grammatical Error Diagnosis (GED) for Learning Chinese as a Foreign Language (CFL). The goal of this shared task is to develop computer-assisting tools for GED of several kinds (i.e., redundant word, missing word, word disorder, and word selection). The task had led researchers to focus on Chinese grammar errors correction in computational linguistics. Until 2017, Rao et al. [14] organize the IJCNLP 2017 Shared Tasks on CGED, where the task still solely concentrates on the detection of the grammatical errors rather than the automatic generation of corrections. The NLPCC 2018 Task 2 gives NLP researchers an opportunity to develop the Chinese grammatical error correction system.

This paper is organized as follows: Sect. 2 describes the GEC shared task. Section 3 illustrates the details of our structure. In Sect. 4, we present our experiment in details, including the data preprocessing and results. In Sect. 5, we introduce some related work both in English and in Chinese. Last but not least, the conclusion and a prospect of future work are given in Sect. 6.

2 Grammatical Error Correction

With the expanding influence of China, learning Mandarin Chinese has grown in popularity around the world. Even though the study of second language learning has started many years ago, the research of CSL still has a long way to go.

The goal of the NLPCC 2018 Shared Task 2 is to evaluate algorithms and systems for the automatic detection and correction of grammatical errors from second language learners of Chinese. Given a Chinese sentence, a GEC system is expected to correct four types of grammatical errors, including redundant words (R), missing words (M), bad word selection (S) and disorder words (W).

A grammatical error correction system is evaluated by how well its proposed corrections or edits match the gold-standard edits. A sentence is first segmented before evaluation is carried out on a set of sentences. The metrics measured at the testing stage are: Precision, Recall and $F_{0.5}$. Let g_i is the set of gold-standard edits for sentence, and e_i is the set of system edits for sentence. The measurements are defined as follows:

$$P = \frac{\sum_{i=1}^{n} |e_i \cap g_i|}{\sum_{i=1}^{n} |e_i|}, \tag{1}$$

$$R = \frac{\sum_{i=1}^{n} |e_i \cup g_i|}{\sum_{i=1}^{n} |g_i|}, \tag{2}$$

$$F_{0.5} = \frac{(1+0.5^2) \times R \times P}{R + 0.5^2 \times P}, \tag{3}$$

where the intersection between e_i and g_i is defined as:

$$e_i \cup g_i = \{e \in e_i | \exists g \in g_i \ , match(e, g)\}. \tag{4}$$

We choose $F_{0.5}$ which emphasizes precision twice as much as recall as our F-measure for that when a grammar checker is put into actual use, the accuracy of its corrections is profoundly valued in order to gain users' acceptance. Negligence in offering a correction is not as bad as giving a wrong one. The NLPCC 2018 Shared Task 2 use the MaxMatch (M^2) scorer[1] [4] as the official scorer. The M^2 scorer efficiently searches for a set of system edits that maximally matches the set of gold-standard edits specified by an annotator.

3 Methodology

Sequence-to-sequence model has been proven to be powerful in many tasks such as machine translation [12], speech recognition [3] and text summarization [15]. Our model for the task of GEC, inspired by the work of Gehring et al. [6], is based on a fully convolutional encoder-decoder architecture with multiple layers of convolutions and attention mechanisms. Most grammatical errors are often localized and dependent more heavily on the nearby words. Therefore, we take advantage of the convolutional neural networks (CNNs), as it can capture local context more effectively than RNNs by performing on smaller windows over the word sequences. Wider contexts and interactions between distant words can also be captured by a multilayer hierarchical structure of convolutions. Moreover, an attention mechanism that assigns weights over the source words based on their relevance is used when predicting the target word. One benefit of our model is that only a fixed number of nonlinear operations are implemented on the input disregarding its length, whereas when using RNNs, the number of nonlinear operations is proportional to the length of the input, diminishing the effects of distant words. In the following section, we will describe our model in details.

3.1 Convolutional Sequence to Sequence Model

We embed input source sentence S given as a sequence of m source words $s_1, \cdots, s_m$ then lookup embedding vector from embedding matrix for each word

[1] http://www.comp.nus.edu.sg/~nlp/software.html.

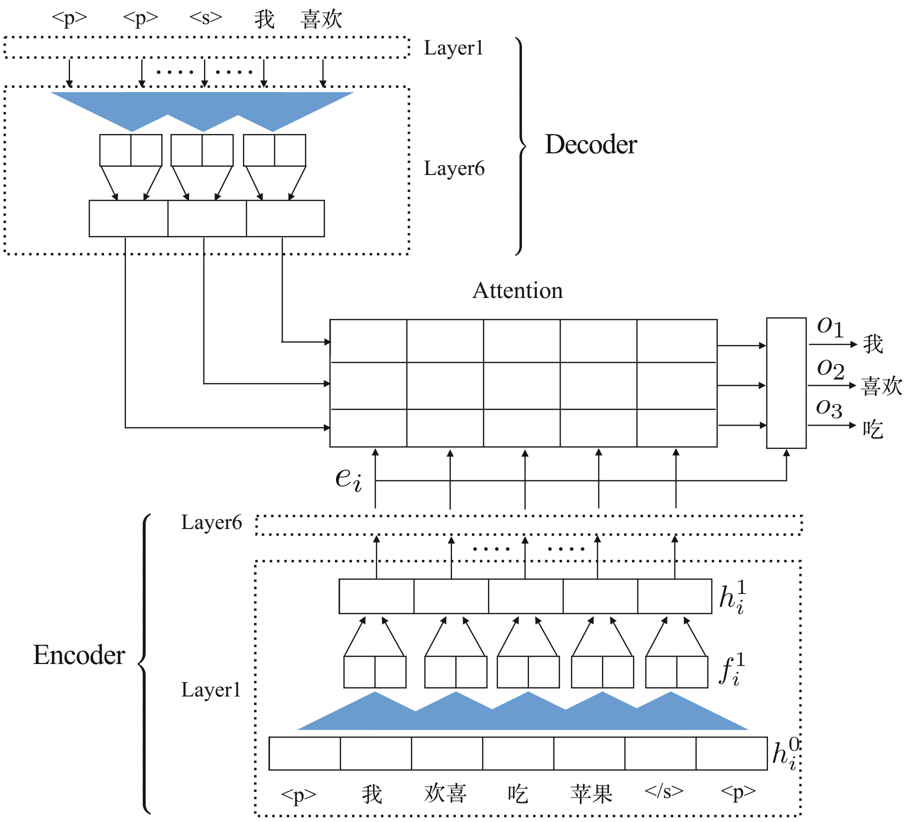


Fig. 1. The architectures of convolutional sequence-to-sequence model.

s_i as $w_{s_i} \in \mathbb{R}^d$. We also equip our model with a sense of order by embedding the absolute position of input elements $p = (p_1, \cdots, p_m)$ where $p_j \in \mathbb{R}^d$. Both are combined to obtain input sentence representations $s = (w_1 + p_1, \cdots, w_m + p_m)$. We proceed similarly for output elements that were already generated by the decoder network to yield output element representations that are being fed back into the decoder network $g = (g_1, \cdots, g_n)$. Position embeddings are useful in our architecture since they give our model a sense of which portion of the sequence in the input or output it is currently dealing with.

In this section, we will describe our model in details. The encoder and decoder are made up of L layers each, share a simple block structure that computes intermediate states based on a fixed number of input elements. Each block contains a one dimensional convolution and a non-linearity. The goal of this framework is to estimate the conditional probability $p(y_{i+1}|y_1, \cdots, y_i, S)$, where S is an input sentence and $\{y_1, y_2, \cdots, y_m\}$ is the corresponding output sequence. The architecture of the network is indicated in Fig. 1.

Encoder. Pass the source token embeddings, $s_1, \cdots, s_m$, over the linear layer to get the input vectors of the first encoding layer, $h_1^0, \cdots, h_m^0$, where $h_i^0 \in \mathbb{R}^d$ and h is the input and output dimension of all encoder and decoder layers. In the first encoder layer, convolution kernel is parameterized as $W \in \mathbb{R}^{(2d \times kd)}$, $b_w \in \mathbb{R}^{2d}$ and takes as input $X \in \mathbb{R}^{k \times d}$ which is a concatenation of k input elements embedded in d dimensions and maps them to a single output element $Y \in \mathbb{R}^{2d}$ that has twice the dimensionality of the input elements. Paddings (denoted by in Fig. 1) are added at the beginning and end of the source sentence to retain the same number of output vectors as the source tokens after the convolution operations.

$$Y = [A \quad B] \in \mathbb{R}^{2d} \tag{5}$$

This is followed by a non-linearity using gated linear units (GLU) [5]:

$$GLU(Y) = A \otimes \sigma(B) \tag{6}$$

where $A, B \in \mathbb{R}^d$ are the inputs to the non-linearity, $\otimes$ and σ represent element-wise multiplication and sigmoid activation functions, respectively. To enable deep convolutional networks, we add residual connections from the input vectors of each encoder layer to the output of the layer. The output vectors of the 1^{th} encoder layer are given by,

$$h_i^l = GLU(Y) + h_i^{l-1} \qquad i = 1, \cdots, m \tag{7}$$

Each output vector of the final encoder layer, $h_i^L \in \mathbb{R}^h$, is linearly mapped to get the encoder output vector, $e_i \in \mathbb{R}^d$, using weights $W_e \in \mathbb{R}^{d \times h}$ and biases $b_e \in \mathbb{R}^d$:

$$e_i = W_e h_i^L + b_e \qquad i = 1, \cdots, m \tag{8}$$

Decoder. Each decoder layer has its own multi-step attention. To compute the attention, we combine the current decoder state $y_n^l \in \mathbb{R}^h$ with an embedding of the previous target element t_{n-1}:

$$z_n^l = W_z y_n^l + b_z + t_{n-1} \qquad W_z \in \mathbb{R}^{d \times h} \quad b_z \in \mathbb{R}^d \tag{9}$$

The attention weights $\alpha_{n,i}^l$ are computed by a dot product of the encoder output vectors $e_1, \cdots, e_m$ with z_n^l and normalized by a softmax:

$$\alpha_{n,i}^l = \frac{\exp(e_i^T z_n^l)}{\sum_{k=1}^m \exp(e_k^T z_n^l)} \qquad i = 1, \cdots, m \tag{10}$$

The addition of the source embeddings helps to better retain information about the source tokens. The conditional source context vector x_n^l is a weighted sum of the encoder outputs as well as the source embeddings:

$$x_n^l = \sum_{i=1}^m \alpha_{n,i}^l (e_i + s_i) \tag{11}$$

The context vector x_n^l is then linearly mapped to $c_n^l \in \mathbb{R}^h$. The output vector of the l^{th} decoder layer, g_n^l, is the summation of c_n^l, y_n^l, and the previous layer's output vector g_n^{l-1}.

$$g_n^l = y_n^l + c_n^l + g_n^{l-1} \tag{12}$$

The final decoder layer output vector g_n^L is linearly mapped to $d_n \in \mathbb{R}^d$. Dropout [18] is applied at the decoder outputs, embeddings, and before every encoder and decoder layer. Finally, we compute a distribution over the T possible next target elements y_{i+1} by transforming the top decoder output d_n via a linear layer with weights W_o and bias b_o:

$$p(y_{i+1}|y_1, \cdots, y_i, S) = softmax(W_o d_n + b_o) \in \mathbb{R}^T \tag{13}$$

4 Experimental Setup

4.1 Data

We conduct our experiment on the dataset of NLPCC 2018 Evaluation Task 2, which is collected from Lang-8 website[2], a multilingual language learning platform providing language exchange Social Networking Service, with native speakers from more than 190 countries and 90 languages.

The full dataset, containing 1,220,069 sentence pairs, has no validation set. We randomly split the whole dataset into two parts: a validation set with 5k sentence pairs that have inconsistency between the source sentence and the target sentence and a training set with all the remaining 1,215,876 sentence pairs. In our experiments, we found that adding the sentence pairs that are identical on both sides could improve the result. Therefore we add all sentences with no grammatical error into the training set. The test data contains 2k sentence pairs. The statistic of the dataset shows in Table 1.

Table 1. Statistics of training, validation and test data. Unchanged refers to unchanged sentence pair. Changed refers to changed sentence pair. Src refers to source wrong sentences. Trg refers to target correct sentences.

	Unchanged	Changed	Words (Jieba)	Characters
Training Src	123,500	1,091,569	15,532,349	25,102,706
Training Trg	123,500	1,091,569	16,261,275	26,318,823
Validation Src	–	5,000	63,974	103,543
Validation Trg	–	5,000	67,342	108,965
Test Src	–	2,000	37,420	61,314

[2] http://lang-8.com/.

Table 2. Examples of two word segmentation methods.

The result of jieba tool	The result of subword method
这个/游戏/叫龙/和/地下城/。	这个/游戏/叫@@/龙/和/地下@@/城/。

4.2 Data Preparation

Word Segmentation. Since the evaluation criteria is based on the word-level, we firstly segment the large corpus using jieba[3] toolkit, which is a Python module for Chinese word segmentation. As is well known, Chinese word segmentation constantly faces the difficulty of multi-granularities. Remarkably, jieba deals with this kind of problem with flying colors. By comparing the experimental results of word segmentation with other word segmentation tools, we found that using jieba can achieve superior performance.

Subword. The subword method is initially proposed by Byte Pair Encoding (BPE) which is an effective data compression technique. Sennrich et al. [17] adopted BPE for word segmentation in neural machine translation (NMT) task which helps to solve the problem of rare and unknown words. The task of grammatical error correction has the similar problem like out-of-vocabulary (OOV) words in the summary generation. Hence, we apply this BPE algorithm to our task, which splits rare words into multiple frequent subwords. The results of the two segmentation methods are showed in Table 2. In our experiments, the use of BPE algorithm can greatly enhance the performance of the model. For detailed comparison results, see Sect. 4.3.

Word Embeddings. Word representations learned from large corpus have shown to be beneficial in many NLP tasks, such as part-of-speech tagging, dependency parsing and machine translation. We initialize the word embeddings for the source and target words with pre-trained word embeddings learned from a unlabeled large corpors. Word segmented by jieba tool, and rare words in this Chinese corpus are split into subword units by Byte Pair Encoding algorithm as we use similar preprocessing for the training dataset that is used to train the network. We use the structured-skipngram model in Wang2Vec tool [11] to train word vectors, which can solve syntax problems well and have information about the words order. In our experiments, the use of pre-trained word embedding can greatly enhance the performance of the model than initializing the network randomly. For detailed comparison results, see Sect. 4.3.

4.3 Experiment Results

We adopt the widely used MaxMatch Scorer toolkit for evaluation. Table 3 shows the results. Our basic model (CS2S) with no use of any additional knowledge

[3] https://github.com/fxsjy/jieba.

or strategy achieves 18.11 in $F_{0.5}$. The $F_{0.5}$ score increase to 20.11 with the utilization of pre-trained word embedding. By adapting the BPE algorithm to the preparation of the dataset, the performance boosts by 9.69 in $F_{0.5}$ (from 18.11 to 27.80). The results are consistent with our intuition that the BPE is supportive to the seq2seq model by upgrading its ability to generate unknown words.

Led by the previous experiments, we equip our model with both pre-trained embedding and BPE algorithm (CS2S+BPE+Emb). This model achieves 29.02 in $F_{0.5}$. Last but not least, we initialize the parameters of the fully equipped model with 4 different seeds. By ensembling the 4 models saved with different initializations, our approach achieves an $F_{0.5}$ score of 30.57, surpassing the best published result of 29.91 in $F_{0.5}$ (TeamID is Fighter_Plane) previously. Due to time suppress, we couldn't submit the results of the ensembled model.

Compared to English GEC, the best $F_{0.5}$ score we gain is an unsatisfying 30.57. To some extent, it lies on the scale of task and the deficiency of training data. So there is much to be explored for the task of Chinese GEC.

Table 3. Results on test dataset. +BPE refers to using byte pair encoding algorithm to preprocess data. +Emb indicates of using pre-trained embedding. Ensemble refers to merging results of 4 models with different initialization.

System	P	R	$F_{0.5}$
CS2S	21.28	11.36	18.11
CS2S+Emb	23.22	13.10	20.11
CS2S+BPE	40.27	12.90	27.80
CS2S+BPE+Emb	41.73	13.08	**29.02**
CS2S+BPE+Emb (ensemble)	47.63	12.56	**30.57**

5 Related Works

Grammatical Error Detection and Correction in CONLL-2013 and CONLL-2014 shared Task attracted a lot of English NLP researchers. Many different approaches were proposed by those participants, e.g. hand-crafted rules, statistical model, translation model and language model.

Statistical machine translation [8] has achieved good results, which can correct various types of errors and complex error patterns. However, SMT-based systems suffer from limited generalization capabilities compared to neural approaches and are unable to access longer source and target contexts effectively. To address these issues, several seq2seq approaches relying on RNNs were proposed for GEC.

At present, encoder-decoder frameworks are widely used for tasks like machine translation. Yuan et al. [19] first applied a popular neural machine translation model, *RNNSearch*. Ji et al. [7] proposed a hybrid word-character model based on the hybrid machine translation model, by adding nested levels of attention at the word and character level. More recently, Schmaltz et al. [16] used a word-level bidirectional LSTM network trained on Lang-8 and NUCLE with edit operations (insertions, deletions, and substitutions) marked with special tags in the target sentences.

More recently, Gehring et al. [6] propose an architecture for sequence to sequence modeling that is entirely convolutional. The model is equipped with gated linear units and residual connections, and also use attention in every decoder layer and demonstrate that each attention layer only adds a negligible amount of overhead. It performs well on some published dataset, because convolutional networks do not depend on the computations of the previous time step and therefore allow parallelization over every element in a sequence, so training and decoding speed is faster.

Due to the similarity between MT and GEC illustrated above, an encoder-decoder model can also be employed for the latter, where the encoder network is used to encode the potentially erroneous source sentence in vector space and a decoder network generates the corrected output sentence by attending to the output of the encoder stack.

6 Conclusion and Future Work

This paper describes our system in the NLPCC 2018 Shared Task 2 for GEC. We explored a seq2seq model based entirely on convolutional neural network. The application of BPE-based algorithm to split rare words into multiple frequent subwords makes the GEC model more capable of handling OOV problem. We achieved highest precision scores and $F_{0.5}$ score is 30.57.

At this stage, we believe the task is far from solved. Lots of improvements can be made to our current model. In the future, we will continue to work on this problem. Possible future directions include combining grammatical error correction with other related multi-task models, adding more features to the model and adapting pre-trained language model. Aside from the model architecture, due to the flexibility and intricacy of Chinese grammar, how to evaluate the automatic grammatical correction also remains a big challenge. In our future work, we will investigate better measurements and criteria for evaluation. Our code is released at https://github.com/blcu-nlp/NLPCC_2018_TASK2_GEC.

Acknowledgment. This research project is supported by the funds of Beijing Advanced Innovation Center for Language Resources (No. TYR17001), the Key Project of National Social Science Foundation of China (No. 16AYY007), and the Fundamental Research Funds for the Central Universities in BLCU (No. 18YCX001). We also thank Huimeng Zhang, JingJing Miao and Jin Zhao for helping to modify the paper.

References

1. Chollampatt, S., Ng, H.T.: Connecting the dots: towards human-level grammatical error correction. In: BEA@EMNLP (2017)
2. Chollampatt, S., Ng, H.T.: A multilayer convolutional encoder-decoder neural network for grammatical error correction. CoRR abs/1801.08831 (2018)
3. Chorowski, J., Bahdanau, D., Cho, K., Bengio, Y.: End-to-end continuous speech recognition using attention-based recurrent NN: first results. CoRR abs/1412.1602 (2014)
4. Dahlmeier, D., Ng, H.T.: Better evaluation for grammatical error correction. In: NAACL (2012)
5. Dauphin, Y., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: ICML (2017)
6. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.: Convolutional sequence to sequence learning. In: ICML (2017)
7. Ji, J., Wang, Q., Toutanova, K., Gong, Y., Truong, S., Gao, J.: A nested attention neural hybrid model for grammatical error correction. In: ACL (2017)
8. Junczys-Dowmunt, M., Grundkiewicz, R.: Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In: EMNLP (2016)
9. Junczys-Dowmunt, M., Grundkiewicz, R., Guha, S., Heafield, K.: Approaching neural grammatical error correction as a low-resource machine translation task. CoRR abs/1804.05940 (2018)
10. Yu, L.-C., Lee, L.H., Chang, L.P.: Overview of grammatical error diagnosis for learning Chinese as foreign language. In: NLP-TEA (2014)
11. Ling, W., Dyer, C., Black, A.W., Trancoso, I.: Two/too simple adaptations of word2vec for syntax problems. In: NAACL (2015)
12. Neubig, G.: Neural machine translation and sequence-to-sequence models: a tutorial. CoRR abs/1703.01619 (2017)
13. Ng, H.T., Wu, S.M., Wu, Y., Hadiwinoto, C., Tetreault, J.R.: The CoNLL-2013 shared task on grammatical error correction. In: CoNLL Shared Task (2013)
14. Rao, G., Zhang, B., Xun, E., Lee, L.H.: IJCNLP-2017 task 1: Chinese grammatical error diagnosis. In: IJCNLP (2017)
15. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: EMNLP (2015)
16. Schmaltz, A., Kim, Y., Rush, A.M., Shieber, S.M.: Adapting sequence models for sentence correction. In: EMNLP (2017)
17. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. CoRR abs/1508.07909 (2016)
18. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. JMLR **15**, 1929–1958 (2014)
19. Yuan, Z., Briscoe, T.: Grammatical error correction using neural machine translation. In: NAACL (2016)

Ensemble of Neural Networks with Sentiment Words Translation for Code-Switching Emotion Detection

Tianchi Yue, Chen Chen, Shaowu Zhang(✉), Hongfei Lin, and Liang Yang

Dalian University of Technology, Dalian 116023, Liaoning, China
zhangsw@dlut.edu.cn

Abstract. Emotion detection in code-switching texts aims to identify the emotion labels of text which contains more than one language. The difficulties of this task include problems in bridging the gap between languages and capturing crucial semantic information for classification. To address these issues, we propose an ensemble model with sentiment words translation to build a powerful system. Our system first constructs an English-Chinese sentiment dictionary to make a connection between two languages. Afterwards, we separately train several models include CNN, RCNN and Attention based LSTM model. Then combine their classification results to improve the performance. The experiment result shows that our method has a good effect and achieves the second place among nineteen systems.

Keywords: Emotion detection · Code-switching · Neural networks
Sentiment words translation

1 Introduction

With the rapid development of the Internet, more and more people tend to express their emotions through text in the online community. Emotion detection has become a hot research topic. Previous work on emotion detection mostly focused on analyzing emotions from monolingual text [1]. However, many users often post code-switching texts in social media, and some researchers start to study how emotions are expressed with code-switching texts.

Code-switched emotion detection is a challenging task in emotion analysis which aims to assign emotion labels to code-switching texts. Code-switching texts contain more than one language [2]. For example, the code-switching instances below show that happiness emotion in [a] is expressed by monolingual form and sadness emotion in [b] is expressed by bilingual form.

[a] 美好假期已经开始, have a nice time。 (happiness)
(The happy vocation has begun, have a nice time.)
[b] 上了一天的课，嗓子hold不住了啊。 (sadness)
(I have been teaching the whole day, my throat can’t take it any more)

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 411–419, 2018.
https://doi.org/10.1007/978-3-319-99501-4_37

Most existing methods respectively focus on emotion detection [3–6] and code-switching text analysis [7, 8]. Relatively few researches [9, 10] consider detecting emotion in code-switching text. The problems of this task are how to bridge the gap to between two languages and how to model the code-switching text to detect emotion.

Motivated by the significant improvements of deep neural networks in many NLP tasks [11], we propose an effective ensemble model with sentiment words translation to tackle these problems. Our system mainly consists of three parts. Firstly, we utilize both Chinese and English sentiment lexicons and an English-Chinese dictionary to translate the English sentiment words in a sentence to Chinese. Afterwards, we separately train several models include CNN, RCNN and Attention based LSTM model. Finally, we ensemble their classification results to improve the performance.

The main contributions of our work can be summarized as follows:

- We construct an English-Chinese sentiment words dictionary to build a connection between bilingual forms.
- We propose an ensemble of neural networks to detect emotion which can extract local features, focus on silent parts and capture contextual information of a whole sentence.
- Experimental result indicates that our approach outperforms several base methods and has a good effect.

2 Related Works

Most existing emotion detection methods focus on the monolingual text. Yang et al. propose an emotion-aware topic model to build a fine-grained domain-specific emotion lexicon [4]. Li et al. build a factor graph to incorporate both the label and context dependency for emotion classification [5]. Neural networks are also introduced because of the good performance in many NLP tasks. Abdul-Mageed et al. use gated recurrent neural networks (GRNNs) for fine-grained emotion detection [6]. Emotion detection can be formalized as the task of classifying whether the post belongs to the given emotion or not. There are some neural networks proposed for text classification, including convolutional neural network (CNN) [12], recurrent convolutional neural network (RCNN) [13], and hierarchical attention network (HAN) [14]. Inspired by these neural networks, we propose an ensemble model to combine the virtues of them.

Recently with the trends of code-switching text on social media, several methods have been proposed to detect emotions of bilingual texts. Lee et al. construct a Chinese-English code-switching corpus for emotion detection and combine maximum entropy based Chinese text classifier and English text classifier [1]. Wang et al. use both the machine translation based bilingual information and sentimental information to build a relation between two languages [9]. Wang et al. apply a bilingual attention model to focus on important words on both monolingual and bilingual contexts and combine the attention vectors to predict the emotion [10].

3 System Preprocessing

3.1 Text Preprocessing

Many texts contain hyperlinks to other web pages which are senseless for emotion detection. We use regular expression to replace all the links with the token URL.

We convert all the English characters into lower case and remove some special tokens such as ‘\xa0’ and ‘\u3000’.

We further segment the tokens which contain bilingual words. For example, split ‘high起’ into ‘high’ and ‘起’.

3.2 Word Embeddings

Word embeddings are continuous low-dimensional vector space representations of words which can better capture semantic and syntactic information. Word embeddings play an important role in sentiment analysis with neural networks [11]. The pre-trained word embeddings from a large task-related unlabelled data can enhance the performance of classifiers. In our experiments, all word embeddings are initialized by word2vec [12], the word vectors are pre-trained on a large unlabelled corpora which is collected from Sina Weibo. We measure the performance of some bilingual pairs from our pre-trained word embeddings, we find some English words and their Chinese translation have been mapped to close vector space. Thus, in this paper, we only translate the English sentiment words.

3.3 Sentiment Words Translation

Sentiment words are vital for emotion detection. If the English sentiment words in text have been mapped to a wrong space, the neural model can’t predict the emotion correctly. To bridge the gap between two languages, we build an English-Chinese sentiment words dictionary. Firstly we use iciba dictionary[1] to translate all of English sentiment words [16] to Chinese. Then choose the word from candidate translation which is also included in a Chinese sentiment lexicon[2] as the only translation. Finally we get an English-Chinese sentiment dictionary with the length of 4040 and we utilize it to map the English sentiment words in text to the Chinese sentiment words.

4 Ensemble Model

In this paper, we respectively implement three models include CNN, RCNN and Attention based LSTM. For each model we first embed every word in the sentence into a word vector space so that a sentence can be represented as a matrix $X = [x_1, x_2, \ldots, x_t]$

[1] http://dict-co.iciba.com/search.php?word=good.

[2] DUTIR Chinese Sentiment Lexicon: http://ir.dlut.edu.cn/EmotionOntologyDownload.aspx.

where $x_t \in R^d$ and t is the length of sequence. After that, we use different networks to obtain the sentence representation individually. Finally, we apply the same classification layer to get prediction probabilities.

4.1 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) uses the convolution layers to learn local features of text. Our model of CNN, illustrated on Fig. 1 is similar to Kim [12].

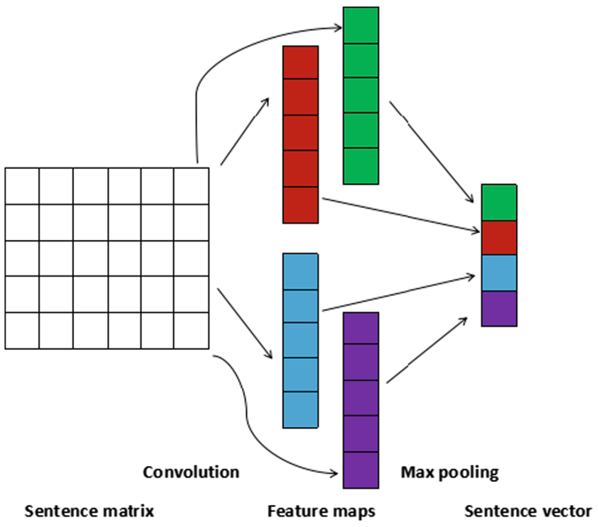


Fig. 1. CNN

We apply several filters of different sizes to the words representation matrix X to extract n-gram features, with general formulations:

$$c_i = f(WX_{i:i+h-1} + b) \tag{1}$$

We then feed the extracted feature maps to a max-pooling layer to get the most important features $c_{max} = \max_{i=1} c_i$. We combine all the c_{max} of each filter into one vector s to represent the whole sentence.

4.2 Long Short-Term Memory (LSTM)

For Attention and RCNN, we use Bi-LSTM to get contextual information of sentence. Let's first briefly introduce LSTM. Long Short-Term Memory network (LSTM) can avoid gradient vanishing and expansion, it is good at learning long-term dependencies and modeling sequences [17]. There are three gates and a memory cell in the LSTM architecture. Formally, LSTM cell can be computed as follows:

$$i_k = \sigma\left(W^i x_k + V^i h_{k-1} + b^i\right) \tag{2}$$

$$f_k = \sigma\left(W^f x_k + V^f h_{k-1} + b^f\right) \tag{3}$$

$$o_k = \sigma(W^o x_k + V^o h_{k-1} + b^o) \tag{4}$$

$$c_k = f_k \odot c_{k-1} + i_k \odot \tan h(W^c x_k + V^c h_{k-1} + b^c) \tag{5}$$

$$h_k = o_k \odot \tan h(c_k) \tag{6}$$

Where W and V are the weighted matrix and b are biases. σ is the sigmoid function and $\odot$ is element-wise multiplication. h_k is the vector of hidden layer which is the final word representations for text.

Since words in a sentence have strong dependence on each other and bi-directional LSTM can better capture the contextual information in text, we employ the bi-directional LSTM network which consists of a forward and a backward LSTM to learn the representation of each word in a sentence.

$$\begin{aligned} \overrightarrow{h}_k &= \overrightarrow{\text{LSTM}}(x_k) \\ \overleftarrow{h}_k &= \overleftarrow{\text{LSTM}}(x_k) \\ h_i &= [\overrightarrow{h}_k, \overleftarrow{h}_k] \end{aligned} \tag{7}$$

4.3 Attention Based LSTM

Each word has different levels of importance on the representation of the sentence semantic. For example, ‘nice’ plays a more critical role than ‘have’ in summarizing the example sentence [a] from Sect. 1. Thus, we introduce an attention mechanism to capture the important information in a sentence. Our model of attention based LSTM is illustrated on Fig. 2.

After obtaining the hidden state h_t by Bi-LSTM, we use a non linear transformation layer to get u_t as a representation of h_t. We use dot product function between ut and a word level vector v to get the relative importance and a softmax transformation to get the final attention signal α_t. Then we can get the weighted representation of sentence s with the attention signal and it can be used as features for text classification.

$$u_t = \tanh(W_a h_t + b_a) \tag{8}$$

$$\alpha_t = \frac{exp\left(u_t^T v\right)}{\sum_t exp\left(u_t^T v\right)} \tag{9}$$

$$s = \sum_t \alpha_t h_t \tag{10}$$

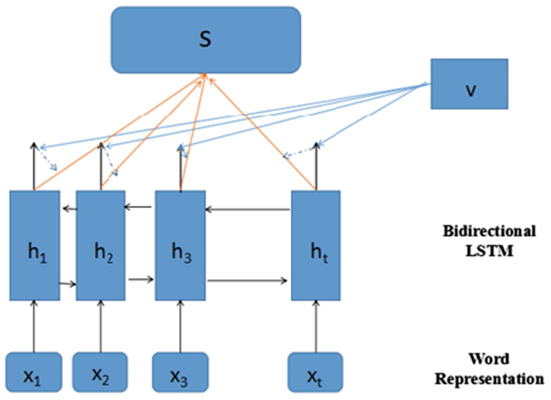


Fig. 2. Attention based LSTM model

4.4 Recurrent Convolutional Neural Network (RCNN)

LSTM is good at capturing the long contextual information. However, it can't sufficiently model the whole sentence since later words are more dominant than earlier words in LSTM. CNN is good at capturing local features while can't capture long gram features. Thus, we use RCNN to tackle these problems. In RCNN network we first apply a Bi-LSTM layer to capture the information h in text which is the same as the Bi-LSTM layer of attention model. Then we further apply a CNN which is identical to Sect. 4.1 on the hidden representation h_t to get the sentence representation s.

4.5 Final Predictions

After getting the sentence vector s, we apply a MLP (Multi-Layer Perception) layer and softmax function to produce the prediction probability.

$$p = \mathrm{softmax}(W_2\sigma(W_1 s + b_1) + b_2) \tag{11}$$

We have five emotions, so we separately train our model for each emotion. Each model is an individual binary classifier. Cross-entropy loss is used as the objective function for training. We also weight the loss function by inverse the frequency of each class for handling the imbalanced data problem.

We have three models for each emotion. Soft voting strategy is used to ensemble these models to reduce variance and improve performance. That means, given a sentence and an emotion, we can get three 2-dimensional output probabilities and average them to identify whether the text expresses the emotion or not.

4.6 Parameters

In our experiments, all word vectors are initialized by word2vec, and out of vocabulary words are initialized by sampling from the uniform distribution $U(-0.01, 0.01)$. The dimension of word vectors is 640.

We employ Adam [18] with initial learning rate of 0.0005 for the optimization of models. After the first 3 epochs, we reduce learning rate decay by a factor of 10. We use early stopping which means stop training when the performance of development set doesn't improve in 5 epochs and dropout with rate of 0.2 to avoid overfitting. We set the batch size to 50. For LSTM based models, hidden output size is 300. For CNN based models, filter windows of 2, 3, 4 with 250 feature maps each.

We use the development sets to tune the hyper parameters and select the best model based on performance on the development set, and evaluate on the test set.

4.7 Results Analysis

We conduct the experiment on the dataset of NLPCC 2018 Evaluation Task 1 which contains 6000 instances for training, 728 for development set and 1200 for test set. Each post contains five emotion labels, i.e. happiness, sadness, fear, anger and surprise. The results of system are evaluated by macro-averaged F1 Score.

We compare the performance of different configurations on Table 1. It shows that the basic neural networks significantly outperforms baseline (SVM model). The ensemble model which incorporates three models achieves the best performance. The reason is that our ensemble model is capable of extracting local features, focusing on silent parts and capturing contextual information which will contribute to sentence representation and text classification.

Table 1. Performance results of different models on the test data.

Models	Happiness	Sadness	Anger	Fear	Surprise	MacF1
Baseline	0.587	0.500	0.390	0.108	0.128	0.342
Ensemble NN	**0.715**	0.521	0.541	0.166	0.396	**0.468**
CNN	0.671	0.507	0.493	0.177	**0.404**	0.450
RCNN	0.709	0.541	0.532	0.171	0.344	0.459
ATTENTION	0.685	**0.543**	0.495	0.191	0.336	0.450
CNN + RCNN	0.694	0.513	**0.543**	0.193	0.380	0.464
CNN + ATT	0.679	0.530	0.534	**0.203**	0.387	0.466
RCNN + ATT	0.686	0.525	0.530	0.166	0.360	0.453

We have also found that single model CNN and Attention based LSTM get similar effect. While RCNN achieves the best performance among the single models, because the RCNN model fully considers local features and contextual information. Moreover, systems which ensemble two single models also outperform the other individual models, which demonstrate the effectiveness of ensemble model. Due to the small number of instances and some posts are also hard to classify by manual annotation, we can see the F-score of some emotions is below 0.4.

Table 2 shows the performance of our system (DUTIR_938) compared with the top, the third and the median ranked team. Our system ranks second among the nineteen teams. However, for the comparison with the top system, our system is over 9%

lower in Sadness and Fear emotion detection. We think the word embedding layer and quality of sentiment words translation are the most important factors. Some sentiment related words can't be mapped to correct vector space even with the fine tuning of word embeddings and will result in misclassification.

Table 2. Performace of the top-3 and median-ranked system offered by the organizer

Team	Happiness	Sadness	Anger	Fear	Surprise	MacF1
DeepIntell	0.734	0.616	0.543	0.264	0.418	0.515
DUTIR_938	0.715	0.521	0.541	0.166	0.396	0.468
Shining	0.710	0.652	0.540	0.292	0.139	0.467
Yang_NEU	0.568	0.432	0.351	0.207	0.255	0.363
Baseline	0.587	0.500	0.390	0.108	0.128	0.342

5 Conclusion and Future Work

In this paper we use bilingual sentiment lexicons and an English-Chinese dictionary to build a connection between two languages and then we explore a method which ensembles several neural networks to detect emotion. Our method can also reduce the impact of data imbalance and boost the performance. Our submission result ranks the second place in all of teams and shows the effectiveness of our method.

We find that the translation of bilingual sentiment related words plays an important role in code-switching text emotion detection, for the future work, we will try to utilize more external knowledge to better bridge the gap to between two languages.

Acknowledgements. This work is supported by National Natural Science Foundation of China (61562080, 61632011, 61572102, 61702080).

References

1. Lee, S., Wang, Z.: Emotion in code-switching texts: corpus construction and analysis. In: Eighth Sighan Workshop on Chinese Language Processing, pp. 91–99 (2015)
2. Wang, Z., Lee, S.Y.M., Li, S., et al.: Emotion analysis in code-switching text with joint factor graph model. IEEE/ACM Trans. Audio Speech Lang. Process. **25**(3), 469–480 (2017)
3. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: Isa-belle, P. (ed.) Proceedings of the EMNLP 2002, pp. 79–86. ACL, Morristown (2002)
4. Yang, M., Zhu, D., Chow, K-P.: A topic model for building fine-grained domain-specific emotion lexicon. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, 22–27 June 2014, Baltimore, MD, USA, vol. 2: Short Papers, pp. 421–426 (2014)
5. Li, S., Huang, L., Wang, R., Zhou, G.: Sentence-level emotion classification with label and context dependence. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, 26–31 July 2015, Beijing, China, vol. 1: Long Papers, pp. 1045–1053 (2015)

6. Abdul-Mageed, M., Ungar, L.: EmoNet: fine-grained emotion detection with gated recurrent neural networks. In: Meeting of the Association for Computational Linguistics, pp. 718–728 (2017)
7. Zhou, H., Chen, L., Shi, F., Huang, D.: Learning bilingual sentiment word embeddings for cross-language sentiment classification. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2015) (2015)
8. Ling, W., Xiang, G., Dyer, C., Black, A.W., Trancoso, I.: Microblogs as parallel corpora. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4–9 August 2013, Sofia, Bulgaria, vol. 1: Long Papers, pp. 176–186 (2013)
9. Wang, Z., Lee, S., Li, S., et al.: Emotion detection in code-switching texts via bilingual and sentimental information. In: Meeting of the Association for Computational Linguistics and the, International Joint Conference on Natural Language Processing, pp. 763–768 (2015)
10. Wang, Z., Yue, Z., et al.: A bilingual attention network for code-switched emotion prediction. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics, pp. 1624–1634 (2016)
11. Zhang, L., Wang, S., Liu, B.: Deep learning for sentiment analysis: a survey. Wiley Interdiscip. Rev. Data Min. Knowl. Disc. (2018)
12. Kim, Y.: Convolutional neural networks for sentence classification. In: Empirical Methods in Natural Language Processing, pp. 1746–1751 (2014)
13. Lai, S., Xu, L., Liu, K., et al.: Recurrent convolutional neural networks for text classification. In: National Conference on Artificial Intelligence, pp. 2267–2273 (2015)
14. Yang, Z., Yang, D., Dyer, C., et al.: Hierarchical attention networks for document classification. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489 (2016)
15. Mikolov, T., Chen, K., Corrado, G., et al.: Efficient estimation of word representations in vector space. Comput. Sci. (2013)
16. Liu, B., Hu, M., Cheng, J., Opinion observer: analyzing and comparing opinions on the web. In: Proceedings of the 14th International World Wide Web conference (WWW-2005), Chiba, Japan, pp. 10–14 (2005)
17. Graves, A.: Long short-term memory. In: Graves, A. (ed.) Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, vol. 385. Springer, Heidelberg (1997)
18. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. Comput. Sci. (2014)

NLPCC 2018 Shared Task User Profiling and Recommendation Method Summary by DUTIR_9148

Xiaoyu Chen, Jian Wang(✉), Yuqi Ren, Tong Liu, and Hongfei Lin

Dalian University of Technology, Dalian Liaoning 116023, China
wangjian@dlut.edu.cn

Abstract. User profiling and personalized recommendation plays an important role in many business applications such as precision marketing and targeting advertisement. Since user data is heterogeneous, leveraging the heterogeneous information for user profiling and personalized recommendation is still a challenge. In this paper, we propose effective methods to solve two subtasks working in user profiling and recommendation. Subtask one is to predict users' tags, we treat this subtask as a binary classification task, we combine users' profile vector and social Large-scale Information Network Embedding (LINE) vector as user features, and use tag information as tag features, then apply a deep learning approach to predict which tags are related to a user. Subtask two is to predict the users a user would like to follow in the future. We adopt social-based collaborative filtering (CF) to solve this task. Our results achieve second place in both subtasks.

Keywords: User tags prediction · User following recommendation User modeling · Collaborative filtering · Deep learning

1 Introduction

With the rapid development of the Internet, a large amount of user information has been generated on the Internet. User profiling [1] can effectively use this information to analyze user's attributes, and it is widely used both in recommendation system and precise advertisement [2]. Social media is one of the most important components of Internet, and it has become essential for people's life. People like to share all kinds of information through social applications, which makes the user research based on social media become more and more important. Users usually use a set of phrases called tag to indicate their personal characteristics such as hobbies and interests and characters on social network. Many social networking services provide tag recommendation function for users to help them set the tags. We can use users' tags to solve problems such as community discovery [3]. Meanwhile, users like to make friends who have a lot in common through social networks. And friend recommendation is a very important function of social networking services, therefore, it has been the focus of social network research.

This shared task contains two subtasks. Subtask one gives users' other information except tags, we need predict which tags are related to a user. Subtask two gives users' following relationship and other provided information, we need predict the users a user would like to follow in the future. We regard subtask one as a classification problem and subtask two as a recommendation problem. We will introduce the two subtasks respectively in the following sections.

1.1 Subtask One

As an important function of social networking services, user tag recommendation has become the focus of research. The classic methods include content-based user tag recommendation method [4, 5] and graph-based user tag recommendation method [6–8]. Yamaguchi et al. [9] proposed a method to predict the tags which are related to a user by using Twitter list. Twitter list is an official functionality to list users may share information about the topic represented in list name. So they extracted the phrases from list names as user's tags, and exploited the relationship between tags and users, then recommended the most relevant tags to the users. Wu et al. [10] considered all of user's tweets as a document, then adopted TextRank [11] to extract keyphrases from it and treated the keyphrases as user's tags. In addition to tweets, there are complex relationships and links among users in social networks, which can be used in user tag recommendation. Lappas et al. [12] believed that users had many different interests. The reason why a user follow others is that they have the same interests. Therefore, they used the underlying social endorsement network to extract useful tags for users. And Chen et al. [13] proposed a method that recommended tags of the users who a user followed to a user. Because they considered that a user tended to follow the users who had the same interests with him. Since deep learning has been successfully used in many research fields, it is gradually applied to user tag recommendation task.

For subtask one, we propose a deep learning approach to predict which tags are related to a user. There are three parts in our approach, we will elaborate in the next section.

1.2 Subtask Two

Friend recommendation is a very popular personalized service in social networking services. Collaborative filtering(CF) [14] is the classic approach to the recommendation problem, which is comprised of two main methods, model-based CF and memory-based CF. The latter includes user-based CF [15] and item-based CF [16]. And friend recommendation is also a recommendation problem, there is some research on it. Bian et al. [17] proposed a method called MatchMaker, which was based on personality matching and collaborative filtering. It leveraged the mutual understanding and social information among users in social networks. Agarwal et al. [18] proposed an implicit rating model, for estimating a user's affinity toward his friends, which uncovered the strength of relationship, utilizing both attribute similarity and user interaction intensity. It was also a CF-based framework. Users usually share their check-in location in social networks, and the check-in data can be used for friend recommendation. Lin et al. [19] considered that customary location a user checked in can represent his social circle, so they achieved friend recommendation based on social circle similarity which was computed by the distance between the two geographical locations.

For subtask two, we propose a social-based collaborative filtering method and rules to realize users' friend recommendation. We will elaborate in the next section.

2 Method

2.1 Subtask One

We regard this subtask as a binary classification task and propose a deep learning framework to solve it. We encode user's personal information such as age, province and city into a vector, use the Large-scale Information Network Embedding(LINE) [20] to encode the social network information into a vector, and join these two vectors into a vector as the user features. Although a user's tags are desensitized, they still contain a lot of semantic information. By encoding them into a vector as the tag features, and then we join these two features as the sample feature, finally we apply our model to achieve the subtask.

As is shown in Fig. 1, the architecture of our model consists of three components: (1) Embedding layer to encode user and tag information; (2) Fully connected layer to obtain user and tag features from the embedding layer; (3) Output layer to predict a user's tags.

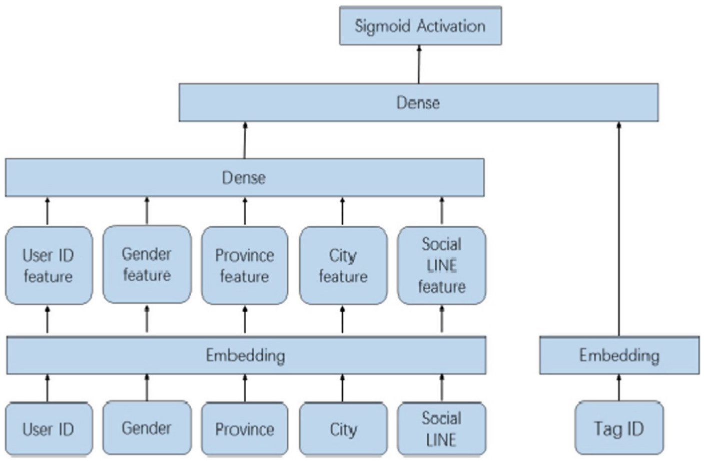


Fig. 1. Architecture of our proposed method

Embedding Layer

Embedding approach is convenient for us to capture the relationship between key information from data. Therefore, each information about user and tag is encoded into a real-valued vector by build embedding matrix $I^{emb} \in \sum_{\mathrm{i}}^{\mathrm{n}} V^i \times N$ in the embedding layer, where V^i denotes the dimension of the each information vector and N is the number of users.

We mainly adopt profiles data as the user information to generate user embedding vector, including user ID, gender, province and city. According to the data of the social ties, one user has followed many users. We connect users who have social relationships with each other and eventually form a social network. We use LINE approach to map all the users nodes in the network to a d-dimensional vector, and try to keep the original network structure. Finally, we merge the user IDs, genders, provinces, cities and social information embedding vectors together as user matrix.

For tag information, tag ID can reflect the relevant characteristics of tag. So the tag matrix consists of the embedding vector of tag IDs.

Fully Connected Layer
We extract user features from the merged user matrix by fully connected layer. Next, we merge the user features with the tag features. We use the fully connected layer again to get connection between user and tag.

Output Layer
Finally we use sigmoid function to predict a user's tags. The sigmoid function is a probabilistic classification algorithm that classifies by probability distribution. The calculation result is from 0 to 1. We consider that if the output probability is greater than 0.5, the tag is related to the user, otherwise the tag is not related to the user.

2.2 Subtask Two

Friend recommendation based on the proportion of common friends
We usually know that, if A and B are friends, and B and C are friends, then A is also likely to be willing to make friends with C. Therefore, we can achieve friend recommendation by calculating the proportion of common friends of two users. But there is also a problem here, the ratio of the total number of friends of a user to the number of common friends is usually very likely to have a certain influence on the friends to be recommended. So first, we use the ratio of the number of common friends to the total number of friends to weight them, and then recommend top-10 of results to a user.

Friend recommendation based on social-based collaborative filtering
As the saying goes, birds of a feather flock together, and people always like to make friends with like-minded people. The like-minded people mean that the people who have interests in the same. Therefore, we can also recommend friends by computing the similarity between users.
We use collaborative filtering (CF) to achieve the similarity computation between uses. One of the main functions of collaborative filtering is to make recommendations. Users can be analyzed based on their historical data. And recommend similar users based on their different preferences. Collaborative filtering is divided into user-based collaborative filtering algorithms and article-based collaborative filtering algorithms.

The key to achieve the friend recommendation through a social-based CF method is to find users similar to the target user. In this subtask, we use the data which describes user's following relationship to compute similarity between users. We denote the user set as $U = \{u_1, u_2, u_3, \ldots, u_m\}$, and denote every user's following user set as $F = \{f_1, f_2, f_3, \ldots, f_n\}$, at the same time, we also create a matrix $S \in R^{M \times N}$ indicates users and the users they follow, and if user i follows user j, $S_{ij} = 1$, otherwise $S_{ij} = 0$. To simplify the calculations, we normalize each line of the matrix S to get a new similarity matrix M_{uf}.

$$M_{uf} = Norm(S) \cdot Norm(S)^T \tag{1}$$

Where,$M_{uf} \in R^{M \times M}$ and each element m_{ij} represents the similarity between user i and user j. With the user similarity, we can use another matrix multiplication to obtain the score of new friends for each user as follow:

$$P = M_{uf} \cdot S - S \tag{2}$$

Where, $P \in R^{M \times N}$ and each element p represents the score for user i and user j.

Friend recommendation based on the most-popular rule
On social networking platforms, such as Weibo, people tend to follow well-known users whose common feature is that they have a high number of followers, so we use this rule to make friend recommendation and recommend the most popular users to a user.

3 Experiments and Results Analysis

3.1 Subtask One

Dataset and Evaluation
We use profiles data, tags data and socials data for subtask one. The number of unique users in the tags file is 11995. We split tags data to form an offline training set and test set by users group. For all users, split out 75 percent of the users with tags from tags data to get training set, the rest as the offline test set. Since the provided data contains only positive samples, we choose the top 20 tags of the frequency of occurrence to build negative samples. If user has no relationship with the top 20 tags, the combination of the user and tag serves as a negative sample. Our offline training set contains 9,796 users, each user has multiple tags, the total number of combination of users and tags is 176,619, the number of negative samples is 117,746, the number of positive samples is 58,873. The rest 2,399 users with tags for the test set, the total number of data is 47,980. The number of training set tag space is 18,496, and the tag space of test set is top 20.

The dimension of user ID's embedding vector is 128. We map the gender f and m to 1 and 0 respectively. The dimensions of embedding vector of province and city are both 32. Each node in the user social network represents a user. We use the LINE method to train the node vector and set the vector dimension to 100.

The quality of subtask one is evaluated by $F1@K$ (K = 3), and we also list $P@K$, $R@K$ for analysis. The calculation formula is as follows:

$$P_i@K = \frac{|H_i|}{K} \tag{3}$$

$$R_i@K = \frac{|H_i|}{|V_i|} \tag{4}$$

$$F1_i@K = \frac{P_i@K \times R_i@K}{P_i@K + R_i@K} \tag{5}$$

$$\mathrm{P}@K = \frac{1}{N}\sum\nolimits_{i=1}^{N} P_i@K \tag{6}$$

$$\mathrm{R}@K = \frac{1}{N}\sum\nolimits_{i=1}^{N} R_i@K \tag{7}$$

$$\mathrm{F1}@K = \frac{1}{N}\sum\nolimits_{i=1}^{N} F1_i@K \tag{8}$$

Where $|H_i|$ is the correctly predicted tags for user i's top K prediction, $|V_i|$ is the correct tags for user *i*. $P_i@K$, $R_i@K$, $F1_i@K$ is the precision, recall and F1 for a user *i*. N is the user count.

Experimental Results and Analysis

We evaluate our method in test set. First we evaluate the deep recommendation model with user features from profiles data. Then we merge the features of the social network with the trained user features through fully connected layer. Finally, we merge user's social feature and profile information as user features together. The results are shown in Table 1.

Table 1. The result of each method (K = 3).

Method	P@K	R@K	F1@K
Profile	4.175%	3.349%	3.652%
Fully connected layer + social	5.669%	3.418%	3.876%
profile + social	5.992%	3.505%	4.037%

Based on the results in Table 1, we find that the method combine profile data with user social information as embedding layer has the best performance. This demonstrates that social network information is effective in tag recommendation. The social information can better represent the commonalities between users.

3.2 Subtask Two

Dataset and Evaluation

We use social network data for subtask two. We split social data to form an offline training set and test set by friend(user2) groups. For each friend, split out 75 percent of

the social data to get the training set, so that all the friend already appear in the training set. Then we filter out the users who don't exist in the training set from the rest of social data to become test data set.
The quality of subtask two is evaluated by $F1@K$ (K = 10), and we also list $P@K$ and $R@K$ for analysis. The calculation formula is the same as subtask one.

Experimental Results and Analysis
We evaluate our method on the offline test set. The results are shown in Table 2.

Based on the results in Table 2, we find that the social-based CF has the best performance. This demonstrates that social network information is effective in User Following Recommendation. Because the users that have common friends are similar. It is effective to recommend the friends that the similar users have to a user.

Table 2. The result of each method (K = 10).

Method	P@K	R@K	F1@K
The proportion of common friends	0.202%	0.210%	0.213%
Social-based CF	0.679%	0.762%	0.679%
Most-popular rule	0.642%	0.747%	0.640%

4 Conclusion and Future Work

In this paper, we elaborate our methods and ideas on User Profiling and Recommendation shared tasks. For subtask one, we take it as a classification problem and adopted a deep learning method to predict user's tags. We focus more on data analysis and the combination of features. Our online submission results are based on fully connected layer with social information result. However, we find that profile with social features is much more effective than the method. And there are too few tag features in the method, and we don't use the tweet data and check-in data. In the future, we are going to use tag co-occurrence relationship to train a LINE vector to rich tag features in this subtask and split the users into two groups, the users that have tweets is divided into one group, the rest users is divided into another group, then construct a dual channel model to predict user tags.

For subtask two, we treat the task as a recommendation problem. We adopt a social-based collaborative filtering method. Since the number of users' common friends is small, the results of this method are not ideal. The collaborative filtering method can gain the similarity between users, so as to realize the friend recommendation. In this subtask, we only use social data. In the future, we can try to calculate similarity by using check-in data and profile data, and then combine multiple approach results to achieve the recommendation.

Acknowledgments. This research is supported by the National Key Research Development Program of China (No. 2016YFB1001103).

References

1. Farnadi, G., Tang, J., Cock, M.D., et al.: User profiling through deep multimodal fusion. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, pp. 171–179. ACM (2018)
2. Farseev, A., Nie, L., Akbari, M., et al.: Harvesting multiple sources for user profile learning: a big data study. In: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, pp. 235–242. ACM (2015)
3. Yan, C.L., Zhang, Y.Y.: Research of community discovery algorithm based on user tags. Sci. Technol. Eng. **11**(6), 1237–1240 (2011)
4. Harvey, M., Baillie, M., Ruthven, I., Carman, Mark J.: Tripartite hidden topic models for personalised tag suggestion. In: Gurrin, C., et al. (eds.) ECIR 2010. LNCS, vol. 5993, pp. 432–443. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12275-0_38
5. Zhang, Y., Zhang, B., Gao, K., et al.: Combining content and relation analysis for recommendation in social tagging systems. Phys. A Stat. Mech. Appl. **391**(22), 5759–5768 (2012)
6. Sigurbjörnsson, B., Zwol, V.R.: Flickr tag recommendation based on collective knowledge. In: Proceedings of the 17th International Conference on World Wide Web, pp. 327–336 (2008)
7. Liu, Z., Shi, C., Sun, M.: FolkDiffusion: a graph-based tag suggestion method for folksonomies. In: Cheng, P.-J., Kan, M.-Y., Lam, W., Nakov, P. (eds.) AIRS 2010. LNCS, vol. 6458, pp. 231–240. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17187-1_22
8. Hu, J., Wang, B., Liu, Y., et al.: Personalized tag recommendation using social influence. J. Comput. Sci. Technol. **27**(3), 527–540 (2012)
9. Yamaguchi, Y., Amagasa, T., Kitagawa, H.: Tag-based user topic discovery using twitter lists. In: Proceedings of International Conference on Advances in Social Networks Analysis and Mining, pp. 13–20. IEEE (2011)
10. Wu, W., Zhang, B., Ostendorf, M.: Automatic generation of personalized annotation tags for twitter users. In: Proceedings of Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, pp. 689–692 (2010)
11. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: Proceedings of EMNLP, pp. 404–411 (2004)
12. Lappas, T., Punera, K., Sarlos, T.: Mining tags using social endorsement networks. In: Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 195–204. ACM (2011)
13. Chen, Y., Lin, L., Sun, C., et al.: A tag recommendation method for microblog users. Intell. Comput. Appl. **1**(3), 21–26 (2011)
14. Rangel, F., Rosso, P.: On the impact of emotions on author profiling. Inf. Process. Manage. **52**(1), 73–92 (2015)
15. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52 (1998)
16. Sarwar, B., Karypis, G., Konstan, J., et al.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International Conference on World Wide Web, pp. 285–295. ACM (2001)
17. Li, B., Holtzman, H.: Online friend recommendation through personality matching and collaborative filtering. In: Proceedings of the 5th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 230–235 (2011)

18. Agarwal, V., Bharadwaj, K.K.: A collaborative filtering framework for friends recommendation in social networks based on interaction intensity and adaptive user similarity. Soc. Netw. Anal. Min. **3**(3), 359–379 (2013)
19. Lin, K., Chen, Y., Li, X., et al.: Friend recommendation algorithm based on location-based social networks. In: Proceedings of IEEE International Conference on Software Engineering and Service Science, pp. 233–236. IEEE (2017)
20. Tang, J., Qu, M., Wang, M., et al.: LINE: large-scale information network embedding. In: Proceedings of the 24th International World Wide Web Conference, pp. 1067–1077 (2015)

Overview of NLPCC 2018 Shared Task 1: Emotion Detection in Code-Switching Text

Zhongqing Wang(✉), Shoushan Li, Fan Wu, Qingying Sun, and Guodong Zhou

Natural Language Processing Lab, Soochow University, Suzhou, China
{wangzq, lishoushan, gdzhou}@suda.edu.cn,
20174227039@stu.suda.edu.cn, qingying.sun@foxmail.com

Abstract. This paper presents the overview of the shared task, emotion detection in code-switching text, in NLPCC 2018. The submitted systems are expected to automatically determine the emotions in the Chinese-English code-switching text. Different from monolingual text, code-switching text contains more than one language, and the emotion can be expressed by either mono-lingual or bilingual form. Hence, the challenges are: how to integrate both monolingual and bilingual forms to detect emotion, and how to bridge the gap to between two languages. Our shared task has 19 team participants. The highest F-score was 0.515. In this paper, we introduce the task, the corpus, the participating teams, and the evaluation results.

Keywords: Emotion detection · Code-switching text
Annotation and evaluation

1 Introduction

With the rapid development of Web 2.0, emotion analysis in social media has become of great value to market predictions and analysis [1–3]. Previous researches on emotion analysis have mainly focused on emotion expressions in monolingual texts [5–7]. However, in informal settings such as micro-blogs, emotions are often expressed by a mixture of different natural languages. Such a mixture of language is called *code-switching*. Specifically, code-switching text is defined as text that contains more than one language (‘*code*’). It is a common phenomenon in multilingual communities [4].

In this shared task, the submitted systems are expected to automatically determine the emotions in the Chinese-English code-switching text. It is more difficult to detect emotions in code-switching texts than in monolingual ones since emotions in code-switching posts can be expressed through one or two languages [9–12]. Hence, traditional automatic emotion detection methods which simply consider monolingual texts would not be readily applicable.

2 Data

In this task, we provide training, development, and testing data. Each post in dataset contains two language text (Chinese and English), we call such post as code-switched text. Following are some examples of code-switched posts with different emotions.

[E1] 婚礼上新娘大秀鼓技, high 翻全场！(**happiness**)
(The bride played the drums at the wedding, everyone was very high!)

[E2] 美好假期已经开始, have a nice time. (**happiness**)
(The happy vacation has begun, have a nice time.)

[E3] 上了一天的课, 嗓子 hold 不住了啊。(**sadness**)
(I have been teaching the whole day, my throat can't take it anymore.)

For each post, five emotions were annotated, namely *happiness*, *sadness*, *fear*, *anger* and *surprise*. The number of samples of different emotions in training, development, and testing data is in Table 1, respectively.

Table 1. Samples of different emotions

	Train	Dev	Test
Happiness	1824	220	490
Sadness	1086	120	296
Anger	570	84	111
Fear	648	85	37
Surprise	651	92	68

3 Participants

There are 46 teams registered the share task, and 19 teams submitted their final results. The teams that have submitted the results are shown in Table 2.

Table 2. Introduction of participating teams

Team name	Organization name
DeepIntell	Research team of DeepIntell co., Ltd.
DUTIR_938	Dalian University of Technology
Shining	University of South China
lxzlx624	University of South China
xiamx-rali	Université de Montréal – Laboratoire RALI
zzuhhjx	University of Zhengzhou
Team_1	Beijing Guangnian Infinity Technology Co., Ltd.
CASIA-ED	Research Center for Brain-inspired Intelligence
cmos_nlp	Move online text algorithm group
Yang_NEU	Northeastern University
rax	Peking University
NLP@WUST	Wuhan University of Science and Technology
scau_geek	South China Agricultural University
Lab1010	Southwest University
The Dream Team of NLP	Nanjing University Science and Technology
GDUFSLEC	Guangdong University of Foreign Studies
CSUNLP	Guangdong University of Foreign Studies
CQUT_301_1	Chongqing University of Technology
BISTU_IIPI	Beijing Information Science and Technology University

4 Evaluation

4.1 Evaluation Metric and Baseline

We compute precision (P), recall (R), and F1-Score for each emotion separately, and compute the macro averaged P, R and F1 with all emotions. Our official scoring metric is macro-averaged F1-Score.

We implement a simple baseline on training data, and test on the development data. The performance is shown in Table 3. We use SVM[1] as the classification method, and unigram as features.

Table 3. Performance of baseline

	P	R	F1
Happiness	0.610	0.691	0.648
Sadness	0.307	0.650	0.417
Anger	0.299	0.667	0.413
Fear	0.256	0.788	0.386
Surprise	0.142	0.391	0.208

4.2 Submission Results and Discussions

There are 19 teams submitted their valid results, the results are shown in Table 4. As Table 4 given, DeepIntell, DUTIR_938 and Shining have better results than others. In particular, **DeepIntell** converts multi-label classification into binary classification task and employ ensemble learning for code-switching text with sampling and emotion lexicon. **DUTIR_938** also use ensemble separately trains several models include CNN, RCNN and Attention based LSTM model. Then combine their classification results to improve the performance. **Shining** considers the relationship between different emotions in a single post, at the same time, the attention mechanism is employed to find the importance of different features and predict all emotions expressed by each post.

[1] http://svmlight.joachims.org/.

Table 4. Evaluation results

Team	Happiness	Sadness	Anger	Fear	Surprise	Marco-F1
DeepIntell	**0.734**	0.616	**0.543**	0.264	**0.418**	**0.515**
DUTIR_938	0.715	0.521	0.541	0.166	0.396	0.468
Shining	0.710	**0.652**	0.540	**0.292**	0.139	0.467
lxzlx624	0.734	0.637	0.570	0.204	0.164	0.462
xiamx-rali	0.624	0.494	0.457	0.200	0.366	0.428
zzuhhjx	0.692	0.428	0.406	0.186	0.376	0.418
Team_1	0.594	0.510	0.440	0.143	0.310	0.399
CASIA-ED	0.596	0.417	0.510	0.130	0.337	0.398
cmos_nlp	0.632	0.414	0.352	0.161	0.265	0.365
Yang_NEU	0.568	0.432	0.351	0.207	0.255	0.363
rax	0.576	0.421	0.392	0.103	0.306	0.360
NLP@WUST	0.630	0.374	0.287	0.146	0.354	0.358
scau_geek	0.615	0.480	0.398	0.000	0.230	0.345
Baseline	0.587	0.500	0.390	0.108	0.128	0.342
Lab1010	0.385	0.458	0.271	0.061	0.063	0.248
The Dream Team of NLP	0.480	0.350	0.160	0.062	0.137	0.238
GDUFSLEC	0.491	0.131	0.190	0.045	0.299	0.231
CSUNLP	0.441	0.354	0.033	0.093	0.149	0.214
CQUT_301_1	0.246	0.161	0.093	0.053	0.049	0.121
BISTU_IIPI	0.456	0.007	0.018	0.000	0.000	0.096

5 Conclusion

This paper briefly introduces the overview of emotion detection in code-switching text shared task at NLPCC 2018. There are 19 participants having submitted final results. And some participants get exciting results in this corpus. Meanwhile, we release a large code-switching text emotion corpus for more large-scale research in emotion and bilingual analysis.

Acknowledgment. We would like to thank the participants for their valuable feedback and results. We should thank Dr. Sophia Yat Mei Lee and Helena Yan Ping Lau for their excellent works on corpus annotation and analysis. The work is supported by the National Natural Science Foundation of China (61331011, 61751206), and the Early Career Scheme (ECS) sponsored by the Research Grants Council of Hong Kong (No. PolyU 5593/13H).

References

1. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. In: Empirical Methods in Natural Language Processing, pp. 79–86 (2002)
2. Yat, S., Lee, M., Li, S., Huang, C.: Annotating events in an emotion corpus. In: Proceedings of the Ninth International Conference on Language Resources and Evaluation, pp. 3511–3516 (2014)
3. Solorio, T., Liu, Y.: Learning to predict code-switching points. In: Conference on Empirical Methods in Natural Language Processing, pp. 973–981(2008)
4. Adel, H., Vu, N., Schultz, T.: Combination of recurrent neural networks and factored language models for code-switching language modeling. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pp. 206–211 (2013)
5. Mohammad, S.M., Kiritchenko, S.: Using hashtags to capture fine emotion categories from tweets. Comput. Intell. **31**(2), 301–326 (2015)
6. Crossley, S.A., Kyle, K., McNamara, D.S.: Sentiment analysis and social cognition engine (SEANCE): an automatic tool for sentiment, social cognition, and social-order analysis. Behav. Res. Methods **49**, 1–19 (2016)
7. Gupta U., Chatterjee A., Srikanth R., Agrawal P.: A sentiment-and-semantics-based approach for emotion detection in textual conversations (2017)
8. Sun, R., Wilson, N., Lynch, M.: Emotion: a unified mechanistic interpretation from a cognitive architecture. Cogn. Comput. **8**(1), 1–14 (2016)
9. Jamatia, A., Das, A.: Part-of-speech tagging for code-mixed English-Hindi Twitter and Facebook chat messages. In: Recent Advances in Natural Language Processing, pp. 139–248 (2015)
10. Lee, S., Wang, Z.: Emotion in code-switching texts: corpus construction and analysis. In: Eighth Sighan Workshop on Chinese Language Processing, pp. 91–99 (2015)
11. Wang, Z., Zhang, Y., Lee, S., Li, S., Zhou, G.: A bilingual attention network for code-switched emotion prediction. In: Proceeding of COLING (2016)
12. Wang, Z., Lee, S., Li, S., Zhou, G.: Emotion analysis in code-switching text with joint factor graph model. In: Transactions on Audio Speech and Language Processing, pp. 469–480 (2017)

Overview of the NLPCC 2018 Shared Task: Automatic Tagging of Zhihu Questions

Bo Huang(✉) and Zhenyu Zhao(✉)

Zhihu Institute, No. 5 Xueyuan Road, Beijing, China
{huangbo,zhaozhenyu}@zhihu.com

Abstract. In this paper, we give an overview for the shared task at the CCF Conference on Natural Language Processing & Chinese Computing (NLPCC 2018): Automatic Tagging of *Zhihu* Questions. The dataset is collected from the Chinese question-answering web site *Zhihu*, which consists 25551 tags and 721608 training samples in this shared task. This is a multi-label text classification task, and each question can have as much as five relevant tags. The dataset can be assessed at http://tcci.ccf.org.cn/conference/2018/taskdata.php.

Keywords: Automatic tagging · Multi-label classification
Text classification

1 Introduction

The task aims to tag questions in *Zhihu* with relevant tags from a collection of predefined ones. This is a multi-label classification problem, several tags can be relevant to a given question. With the rise of social media, the text data on the web is growing exponentially. Furthermore, the label space is relatively huge compared to traditional text classification tasks. Make it is impractical for a human being to accurately assign tags to all those data. Machine learning methods are quite suitable for this task, and accurate tags can benefit several downstream applications such as recommendation and search.

Formally, the task is defined as follows: given a question with its title $x_t = (x_{t_1}, x_{t_2}, \cdots, x_{t_n})$ and description $x_d = (x_{d_1}, x_{d_2}, \cdots, x_{d_m})$, where x_{t_j} denotes the jth word in the title. The objective is to find its possible relevant tags in the predefined tag set. More specifically, given a specific tag tag_i, we need to find a function to predict whether tag_i is relevant to the current question with title x_t and description x_d.

$$p(tag_i|x_t, x_d) = f(x_t, x_d, tag_i, \theta) \tag{1}$$

where θ is the parameter of the function.

2 Data

In this task, we provide training, development, and test data. Each question in the dataset contains a title, an unique id and an additional description. The labels are tagged collaboratively by users from the community question answering web site *Zhihu*. To improve the quality of the data, we removed infrequency tags, and relabeled manually to build development and test dataset.

There are 25551 tags and 721608 training samples in training data, 8947 samples in development data and 20597 samples in test data. Some samples from training dataset are shown in Table 1.

The dataset is different from widely used text classification datasets. Firstly, the label space is relatively huge and there is a data imbalance problem. Table 2 shows the statistics of the numbers of training samples for each label, we can see that almost 30% labels only have 5 to 10 training samples, while there still are some labels may have more than 5000 training samples. Secondly, the task is a multi-label problem and the number of labeled tags is not fixed for each question with a range from 1 to 5, Table 3 shows the statistics of the numbers of labeled tags for each question. Thirdly, since the dataset is collected from *Zhihu* whose contents are all generated by users, the text styles vary from user to user, Fig. 1 shows the length distributions of titles and descriptions respectively.

Table 1. Training samples from the dataset.

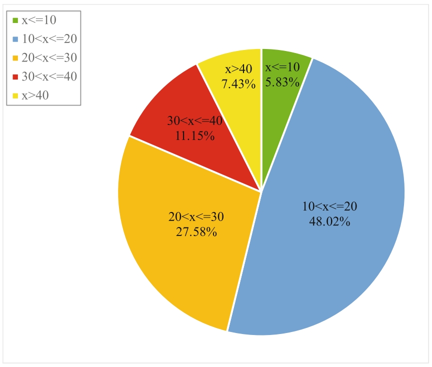
question_title	question_description	tags
如何做到不让别人影响自己的心情? How to keep others from affecting your mood?		情绪, 情绪管理 mood, mood control
怎样写商业计划书? How does one create a business plan?	我有些想法，怎么获得投资 I have some ideas, how can I get investment?	商业计划书，风投 business plan, VC

Table 2. Statistics of the numbers of training samples for each label.

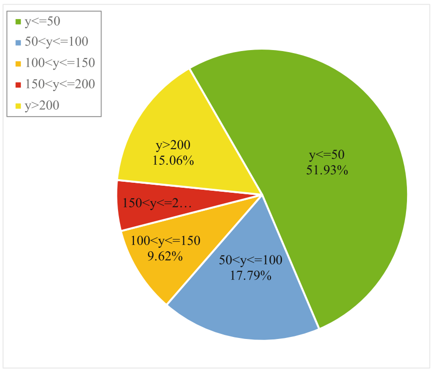
Number of training samples	5 to 10	10 to 50	50 to 500	500 to 1000	1000 to 5000	5000+
Count of labels	7651	11988	5158	422	296	36
Percentage of labels(%)	29.94	46.92	20.19	1.65	1.16	0.14

Table 3. Statistics of the numbers of labeled tags for each sample in training data.

Number of labeled tags	1	2	3	4	5
Count of samples	134190	123397	151553	143388	169080
Percentage of samples(%)	18.60	17.10	21.00	19.87	23.43



(a) Length distribution of titles.



(b) Length distribution of descriptions.

Fig. 1. Length distributions of training data.

3 Evaluation

For each question in the test set, the model is required to predict as much as five relevant tags, and the tags are sorted by their predicted probabilities. Specifically, the number of predicted tags for a given question can be less than 5 or even be 0 if the model can't find enough relevant tags to the question.

The results are evaluated on the F_1 measure. We compute the positional weighted precision. Let $correct\_num_{p_i}$ denotes the correct count of predicted tags at position i, and $predict\_num_{p_i}$ denotes the count of predicted tags at position i.
The precision, recall and F_1 measure are computed as following formulas:

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{2}$$

$$P = \frac{\sum_{i=1}^{5} correct\_num_{p_i} / \log(i+2)}{\sum_{i=1}^{5} predict\_num_{p_i} / \log(i+2)} \tag{3}$$

$$R = \frac{\sum_{i=1}^{5} correct\_num_{p_i}}{ground\_truth\_num} \tag{4}$$

4 Baseline Implementations

Text classification is an important task in Natural Language processing with many applications, including search query classification, sentiment analysis, news categorization, which have been studied for years. In recent years, Deep

Learning on text classification has gained much attention due to its prominent achievement.

We have implemented several deep learning baseline models on text classification, which are effective and widely used in recent years, including LSTM [1], FastText [2] and CNN [3]. The results of the baselines are listed in Table 4.

Table 4. Results of baselines.

Method	Weighted precision	Recall	F1
LSTM	33.47	46.15	38.80
CNN	34.62	46.87	39.82
FastText	32.79	48.52	39.13

5 Participants Submitted Results

There are total 15 participants actively participate and submit their predictions on the test set. The number of submissions is limit to 5 in total, and we report the best result for each participant. The predictions are evaluated and the results are listed in Table 5.

Table 5. Participants Submitted Results.

Participant	Weighted precision	Recall	F1	Number of submissions
P1	0.5251	0.7783	0.6271	1
P2	0.5384	0.6380	0.5840	3
P3	0.5267	0.5123	0.5194	5
P4	0.5048	0.4692	0.4863	4
P5	0.5031	0.4664	0.4841	1
P6	0.3859	0.5502	0.4536	2
P7	0.3423	0.4759	0.3982	2
P8	0.3743	0.3770	0.3756	4
P9	0.2882	0.4157	0.3404	5
P10	0.2880	0.4100	0.3383	5
P11	0.2894	0.3427	0.3138	1
P12	0.2996	0.3221	0.3104	5
P13	0.2431	0.3477	0.2861	5
P14	0.4333	0.1546	0.2279	2
P15	0.1614	0.1242	0.1404	1

6 Conclusion

Text classification has been studied for years, several text classification datasets have been studied extensively in recent years. In this task we collected a new text classification dataset, which addresses two problems: (1) the label space is relatively huge, (2) the training samples are very imbalance. We contributed the dataset to the research community for further study.

References

1. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
2. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T.: FastText.zip: compressing text classification models. arXiv preprint arXiv:1612.03651 (2016)
3. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)

Overview of the NLPCC 2018 Shared Task: Grammatical Error Correction

Yuanyuan Zhao[1,2], Nan Jiang[1,2], Weiwei Sun[1,2,3](✉), and Xiaojun Wan[1,2]

[1] Institute of Computer Science and Technology, Peking University, Beijing, China
zhaoyy1461@gmail.com,jnhsyxxy@126.com,{ws,wanxiaojun}@pku.edu.cn

[2] The MOE Key Laboratory of Computational Linguistics, Peking University, Beijing, China

[3] Center for Chinese Linguistics, Peking University, Beijing, China

Abstract. In this paper, we present an overview of the Grammatical Error Correction task in the NLPCC 2018 shared tasks. We give detailed descriptions of the task definition and the data for training as well as evaluation. We also summarize the approaches investigated by the participants of this task. Such approaches demonstrate the state-of-the-art of Grammatical Error Correction for Mandarin Chinese. The data set and evaluation tool used by this task is available at https://github.com/zhaoyyoo/NLPCC2018_GEC.

1 Introduction

Grammatical Error Correction (GEC) is a challenging task in natural language processing and it has attracted more and more concerns recently. This year, we organize the first shared task of GEC for Mandarin Chinese, with a focus on speech errors produced by Chinese learners. In particular, our task is defined as to detect the grammatical errors in the essays from non-native speakers and return the corrected texts [1]. The previous research on grammatical errors in Chinese is mainly devoted to error detection [2], while our shared task also include automatic correction of such grammatical errors. To the best of our knowledge, this task provides the first benchmark data set for GEC for Chinese.

The goal of the task is to develop techniques to automatically detect and correct errors made by writers of CSL (Chinese as a Second Language). We provide large-scale Chinese texts written by non-native speakers in which grammatical errors have been annotated and corrected by native speakers. Blind test data is used to evaluate the outputs of the participating teams using a common scoring software and evaluation metric.

A total of 23 teams signed up for the shared task and six of them submitted final results. This overview paper provides detailed descriptions of the shared task and it is organized as follows. Section 2 gives the task definition. Section 3 presents a detailed introduction of the data sets and annotation guidelines. Section 4 provides the evaluation metric and Sect. 5 introduces different approaches from participants. Section 6 shows the final results and Sect. 7 gives the conclusion of the paper.

M. Zhang et al. (Eds.): NLPCC 2018, LNAI 11109, pp. 439–445, 2018.
https://doi.org/10.1007/978-3-319-99501-4_41

2 Task Definition

Automatically correcting grammatical errors is a challenging task which has attracted an increasing attention recently. The goal of this shared task is to detect and correct grammatical errors present in Chinese essays written by non-native speakers of Mandarin Chinese. Given annotated training data with corrections of grammatical errors and blind test data, the participating teams are expected to submit automatically corrected version of texts in test data. An example of mistaken quantifiers under the task definition is shown in Table 1.

Table 1. An example of the input and the output under the task definition.

Source Input	那是一个牛。
Segmented Input	那 是 一 个 牛 。
Outputs	那是一头牛。
	那 是 一 头 牛 。

3 Data

This section presents the released training and test data in the shared task.

3.1 Training Data

The training data provided in the shared task is collected from http://lang-8.com/, a language-learning website where native speakers freely choose learners' essays to correct. Following [3], we collect a large-scale Chinese Mandarin learners' corpus by exploring "language exchange" social networking services (SNS). There are about 68,500 Chinese Mandarin learners on this SNS website. By collecting their essays written in Chinese and the revised version by Chinese natives, we set up an initial corpus of 1,108,907 sentences from 135,754 essays.

As correcting specifications are not unified and there is lots of noise in raw sentences, we take a series of measures to clean up the data. First, we drop words surrounded by <spanclass = "sline"> since this indicates redundant contents. As for other kinds of tags, correctors use them in different ways. We just remove the tag and remain inner words for consistency and clarity. Learners often ask questions in their native languages, bringing about extra noise into the corpus. We need to get rid of sentences with too many foreign words by checking their Unicode values. There is one more situation where writers use Chinese phonetic alphabet to represent the word that they want to express but do not know how to write it in Chinese characters. Such nonstandard sentences are excluded from final dataset. To improve compactness, we also drop rather simple sentences such as 大家好 *(Hello everyone)*, 晚上好 *(Good night)*. According to our observation, writers sometimes provide optional corrections using "/", "or", " "或 (or)" (or)" or " "或者 (or)" (or)". In such situations, the first correction is reserved. Moreover, to explain the reason why the original sentence is ungrammatical, correctors may write comments in the position of revised sentences. We utilize a rule-based classifier to determine whether to include the sentence into the corpus.

Through above cleaning operations, we finally sort out a Chinese Mandarin learners' corpus of 717,241 sentences from writers of 61 different native languages. Among these sentences, there are 123,501 sentences considered to be correct, 300,004 sentences with one correction, 170,407 sentences with two corrections and the maximum number of corrections about one sentence is twenty-one. Sample sentences are shown in Table 2. Besides, we use PKUNLP tool (http://www.icst.pku.edu.cn/lcwm/pkunlp/downloads/libgrass-ui.tar.gz) for word segmentation.

Table 2. Sample sentences from the training data.

Source Sentence	**Corrected Sentences**
长成大人，我盒饭做的很开心。	长大成人后，我做盒饭做得很开心。
城市里的人能度过多方面的生活。	城市里的人能过丰富多彩的生活。
	城市里的人能过多种多样的生活。
	城市里的人能过多方面的生活。

3.2 Test Data

The test data is extracted from *PKU Chinese Learner Corpus*. *PKU Chinese Learner Corpus* is constructed by Department of Chinese Language and Literature, Peking University. The goal is to promote research on international education and Chinese interlanguage. And it is composed of essays written by foreign college students. We collected 2,000 sentences from the corpus and release the source sentences and the segmented version.

To obtain gold edits of grammatical errors, two annotators annotated these sentences. The annotation guidelines follow the general principle of *Minimum Edit Distance*. This principle regulates how to reconstruct a correct form of a given sentence containing errors and it selects the one that minimizes the edit distance from the original sentence [4]. This means that we choose to follow the original intention of the writer as much as possible. Following [2], errors are divided into four types: redundant words (denoted as a capital "R"), missing words ("M"), word selection errors ("S"), and word ordering errors ("W"). The first annotator marked the edit alone, and the second annotator was asked to check the annotation and make a revision if he thought the current edit was not appropriate. We release evaluation results on both the two kinds of gold annotations and their integration.

4 Evaluation Metric

We use the *MaxMatch* (M_2) Scorer for evaluation [5]. M_2 Algorithm is a widely used method for evaluating grammatical error correction. The general idea is computing the phrase-level edits between the source sentence and the system output. Specifically, it will choose the system hypothesis that holds the highest overlap with the gold edits from annotators. And [1] extends the M_2 Scorer to deal with multiple alternative sets of gold-standard annotations, in which case there are more than one corrections that are reasonable for the current sentence.

Suppose the gold edit set is $\{g_1, g_2, ..., g_n\}$, and the system edit set is $\{e_1, e_2, ..., e_n\}$. The precision, recall and $F_{0.5}$ are defined as follows:

$$P = \frac{\sum_{i=1}^{n} |e_i \cap g_i|}{\sum_{i=1}^{n} |e_i|} \tag{1}$$

$$R = \frac{\sum_{i=1}^{n} |e_i \cap g_i|}{\sum_{i=1}^{n} |g_i|} \tag{2}$$

$$F_{0.5} = 5 \times \frac{P \times R}{P + 4 \times R} \tag{3}$$

where the intersection between e_i and g_i is defined as

$$e_i \cap g_i = \{e \in e_i | \exists g \in g_i(match(e, g))\}. \tag{4}$$

Take the sentence in Fig. 1 as an example, suppose the source sentence is "随着通迅技术的发达我们的生活也是越来越放便。(With the development of communication technology, our life is becoming more and more convenient.)", the set of gold edits **g** and the set of system edits **e** are shown in this figure. Then there will be $P = 1$, $R = 2/3$, $F_{0.5} = 10/11$.

Source Sentence: 随着通迅技术的发达我们的生活也是越来越放便。
Gold Edits (**g**): {通迅 → 通讯, 也 → ϵ, 放便 → 方便}
System Edits (**e**): {通迅 → 通讯, 放便 → 方便}

Fig. 1. An example of the evaluation metric.

5 Approaches

There are altogether 18 submissions from six teams, at most three submissions per team. The detailed information of participants is shown in Table 3.

Most of the systems treat the GEC problem as a machine translation (MT) task. Rule-based models and language models are also explored. *AliGM* [6] proposes two modules for this problem: the correction module and the combination module. In the former module, correction candidates are generated for each input sentence with two statistical models and one neural model. The statistical models include a rule-based model and a statistical machine translation (SMT) -based

Table 3. The detailed information of participants.

System	Organization
AliGM	Alibaba Group
CU-Boulder	Department of Linguistics, University of Colorado Boulder
YouDao	Department of ML & NLP, Youdao
BUPT	Beijing University of Posts and Telecommunications
PKU	Institute of Computational Linguistics, Peking University
BLCU	School of Information Science, Beijing Language and Culture University

model. The neural model refers to a neural machine translation (NMT) -based model. In the latter module, they combine these models in a hierarchical manner. *CU-Boulder* uses a Bi-LSTM model with attention to make corrections. And they use the character-level minimum edit distance (MED) to select the correction version among multiple candidates. Joint voting of five models is implemented to advance the performance. *YouDao* [7] also casts the problem as a machine translation task. It is worth noting that they use a staged approach and design specific modules targeting at particular errors, including spelling, grammatical, etc. *BUPT* uses a two-stage procedure method. In the first stage, they adopt neural models for error detection. In the second stage, they use a statistical method following [8]. *PKU* uses a character-based MT model to deal with this problem. Besides, they propose a preprocessing module for the correction of spelling errors. First, the error detection is based on the binary features including co-occurrence probability, mutual information and chi-square test. Then confusion sets are introduced to generate candidates at the detected point. The final correction is the candidate with the highest language model probability. To improve the precision score, they set a high threshold. In addition, they check each correction with confidence levels in a post-processing stage. *BLCU* [9] proposes a system mainly based on the convolutional sequence-to-sequence model.

6 Results

We perform evaluations on all the eighteen submissions regarding to both of the two kinds of gold annotations and their integration. The best performance of each system referring to the integrated gold standard edits is shown in Table 4.

From Table 4, we can see that grammatical error correction for Chinese language is a challenging task. There still remains large gaps between automatic GEC systems and native speakers. In detail, *YouDao* gets the highest recall and $F_{0.5}$ score while *BLCU* wins the highest precision score. Both of the two systems treat the GEC problem as a MT task. By contrast, the rule-based models and language models perform unsatisfactorily.

Table 4. Evaluation Results

System name	Precision	Recall	$F_{0.5}$
YouDao	35.24	18.64	29.91
AliGM	41.00	13.75	29.36
BLCU	41.73	13.08	29.02
PKU	41.22	7.18	21.16
CU-Boulder	30.07	6.54	17.49
BUPT	4.22	1.49	3.09

7 Conclusion

This paper provides the overview of the Grammatical Error Correction (GEC) shared task in NLPCC 2018. We release a large Chinese learner corpus and briefly introduce participants' methods. The final results show that it is still a challenging task which deserves more concern.

Acknowledgement. This work was supported by National Natural Science Foundation of China (61772036, 61331011) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We thank the Department of Chinese Language and Literature, Peking University for providing the original inputs of the test data. Weiwei Sun is the corresponding author.

References

1. Ng, H.T., Wu, S.M., Wu, Y., Hadiwinoto, C., Tetreault, J.: The CoNLL-2013 shared task on grammatical error correction. In: Proceedings of the 17th Conference on Computational Natural Language Learning, Association for Computational Linguistics, Sofia, pp. 1–12 (2013)
2. Rao, G., Zhang, B., Xun, E., Lee, L.: IJCNLP-2017 Task 1: Chinese grammatical error diagnosis. In: Proceedings of the IJCNLP 2017, Shared Tasks, pp. 1–8. Asian Federation of Natural Language Processing, Taipei (2017)
3. Mizumoto, T., Komachi, M., Nagata, M., Matsumoto, Y.: Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In: Proceedings of 5th International Joint Conference on Natural Language Processing, pp. 147–155. Asian Federation of Natural Language Processing, Chiang Mai (2011)
4. Nagata, R., Sakaguchi, K.: Phrase structure annotation and parsing for learner English. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1837–1847. Association for Computational Linguistics, Berlin (2016)
5. Dahlmeier D, Ng H T.: Better evaluation for grammatical error correction. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 568–572. Association for Computational Linguistics (2012)

6. Zhou, J., Li, C., Liu, H., Bao, Z., Xu, G., Li, L.: Chinese grammatical error correction using statistical and neural models. In: Proceedings of NLPCC-2018 (2018)
7. Fu, K., Huang, J., Duan Y.: Youdao's Winning solution to the NLPCC-2018 Task 2 challenge: a neural machine translation approach to Chinese grammatical error correction. In: Proceedings of NLPCC-2018 (2018)
8. Chen, S., Tsai, Y., Lin, C.: Generating and scoring correction candidates in Chinese grammatical error diagnosis. In: Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications, Osaka, pp. 131–139 (2016)
9. Ren, H., Yang, L. Xun, E.: A sequence to sequence learning for Chinese grammatical error correction. In: Proceedings of NLPCC-2018 (2018)

Overview of the NLPCC 2018 Shared Task: Multi-turn Human-Computer Conversations

Juntao Li[2] and Rui Yan[1,2](✉)

[1] Institute of Computer Science and Technology (ICST), Peking University, Beijing 100871, China
ruiyan@pku.edu.cn

[2] Institute of Big Data Research, Peking University, Beijing 100871, China
lijuntao@pku.edu.cn

Abstract. In this paper, we give an overview of multi-turn human-computer conversations at NLPCC 2018 shared task. This task consists of two sub-tasks: conversation generation and retrieval with given context. Data-sets for both training and testing are collected from Weibo, where there are 5 million conversation sessions for training and 40,000 non-overlapping conversation sessions for evaluating. Details of the shared task, evaluation metric, and submitted models will be given successively.

Keywords: Multi-turn conversation · Conversation generation
Conversation retrieval · Sequence matching

1 Task Background

Building multi-turn conversation system in open-domain is a research hotpot in academia and draws mounting attention in the industry. In earlier years, researchers mainly design rule-based or template-based systems for task-oriented conversation. With the access to myriad conversation data, it is prompting to develop conversation systems for open-domain. Existing open-domain conversation systems in recent few years can be roughly summarized as two categories: retrieval-based models and generation-based approaches.

Most conversation generation approaches are based upon the sequence to sequence framework [1]. For example, Vinyals and Le propose to utilize such a sequence to sequence model for addressing the issue of conversation generation [2]. However, the simple sequence to sequence model can not adequately model context information in conversations. With limited context information, improper or even unrelated output will be yielded by conversation systems. To address the obstacle of "out of context", context-sensitive conversation generation approach was proposed [3,4]. Serban et al. [5,6] further propose to model conversation context as a hierarchical structure to avoid data sparsity.

Besides, specific information in context are extracted such as topic [7], diversity [8,9], and persona [10], for generating conversations.

For retrieval-based conversation systems, it can be constructed as a sequence matching problem by computing the matching degree of candidate responses and user-issued query. Yan et al. [11] treat responses retrieval as a ranking problem through incorporating multi-dimension ranking evidences, where conversation context in a continuous session in multi-turns is also captured. To better extract matching information from conversation context, the sequential matching network is designed and achieves the state-of-art performance [12].

Although substantial progress of multi-turn human-computer conversation has been achieved by either retrieval-based or generation-based systems, there are still room for improvement. Herein the NLPCC 2018 shared task 5 is designed for putting Chinese multi-turn conversation forward. This task consists of two sub-tasks: conversation generation and response retrieval. There are 10 teams targeting at conversation generation and 5 teams for response retrieval.

2 Task Description

2.1 Task Formulation

The multi-turn conversation generation task is formulated as follows: given the previous $n-2$ sentences in an continuous conversation session $X = (x_1, ..., x_{n-2})$ as *context* and the $(n-1)$-th sentence x_{n-1} as *query*, the goal is to generate the *response* x_n. Table 1 gives two conversation sessions used for generation.

Table 1. Two conversation sessions used for generation.

Conversation Sessions	
Context	谢谢你所做的一切
Context	你开心就好
Context	开心
Context	嗯因为你的心里只有学习
Query	某某某，还有你
Response	这个某某某用的好
Context	你们宿舍都是这么厉害的人吗
Query	眼睛特别搞笑，这土也不好捏但是就是觉得挺可爱
Response	特别可爱啊

For the sub-task of response retrieval in multi-turn conversations, it can be defined as follows: given the *context* $X = (x_1, ..., x_{n-2})$ and *query* x_{n-1}, the proper *response* x_n will be retrieved from 10 response candidates $\{c_1, c_2, ..., c_{10}\}$. As illustrated in Table 2, c_4 is the right answer for the given query and will be selected.

Table 2. An example of response retrieval.

Conversation Session	
Context	你们宿舍都是这么厉害的人吗
Query	眼睛特别搞笑，这土也不好捏但是就是觉得挺可爱
Candidate1	佛山有这个地方吗
Candidate2	佛山都是以吃的为准，怎么会有海呢，珠海，潮汕惠州就有
Candidate3	男朋友不是赵阳玩你手机的话那你简直可怕
Candidate4	特别可爱啊
Candidate5	那么什么时候一起游西湖
Candidate6	自己宿舍不给进那是大四认识的学妹宿舍
Candidate7	我今年参加高考，想报你们学校，云南的学生希望大吗?
Candidate8	王小懒小瘦子你票都没买号反正要聚餐啊
Candidate9	我只能说这厮能长到是要积了多少德
Candidate10	你是怎么赔偿的阿

2.2 Datasets

The datasets employed in this task are collected from Sina Weibo, which contain training set and testing set. We clean the datasets by removing emojis and repeated utterances. For conversation generation, there are 5,000,000 conversation sessions in the training set and extra 40,000 conversation sessions in the testing set. Each session contains at least 3 sentences. As for response retrieval, there are 5,000,000 conversation sessions for training and non-overlap 10,000 sessions for testing. Participants are asked to submit at least one system for either one of the sub-tasks or both.

2.3 Evaluation Metric

It is still challenging for evaluating the results of conversation generation. Although automatic evaluation is not aligned with user experience, we still use BLEU score [13] as the evaluation metric for conversation generation insomuch as human evaluation is not affordable for this task. For the sub-task of response retrieval, we use precision of selected candidates as the evaluation metric.

3 Results Statistics

Ultimately, 10 teams submitted their results for the sub-task of conversation generation and 5 teams participated in the response retrieval. As shown in Table 3, the best result for conversation generation is yielded by the system Yiwise-DS, i.e. 16.58. It can be seen that other five systems also achieve presentable results, e.g. G930, Lmll, BD-chatbot. Table 4 shows the evaluation results of response retrieval. The best performance is generated by ECNU and Wyl-buaa achieves a comparable results while other systems have a large space for improvement.

Table 3. Results for conversation generation.

Submitted systems	Score
DialogMind	5.24
BLCU-NLP	0.19
Yiwise-DS	16.58
Jiaoyanqiaowuwang	1.54
Laiye-rocket	12.05
G930	12.85
BD-chatbot	15.51
Phantomgrapes	11.51
Lmll	12.98
ECNU	0.91

Table 4. Results for response retrieval.

Submitted systems	Precision
BLCU-NLP	10.54
Yiwise-DS	26.68
Laiye-rocket	18.13
Wyl-buaa	59.03
ECNU	62.61

4 Representative Systems

In this part, we give a brief analysis of 4 representative systems in NLPCC 2018 shared task 5, where there are 2 systems for conversation generation and response retrieval respectively.

For conversation generation, G930 re-implements the VHRED model [6] or part of KgCVAE [9] on the Weibo Dataset released by NLPCC 2018. The VHRED model utilizes the latent variable for addressing the wording novelty issue of RNNs. The hierarchical encoder of VHRED can effectively take conversation context into account and thus yields relatively good responses. The Lmll model augments HERD [5] with keywords and an attention mechanism for addressing the issue of topic irrelevance in generated responses.

For response retrieval, Wyl-buaa presents a novel RCMN model for addressing the relationship between utterances in context through utilizing the self-matching information in context. To obtain the relevance between response and each utterances in both word-level and sentence-level, the Sequential Matching Network (SMN) [12] is used. The SMN and the RCMN are further combined as an ensemble model and achieve rank second in the shared task of NLPCC 2018. ECNU presents a framework that combines NLP features, SMN, and memory-based matching network (MBMN) for addressing the issue of response selection.

Specifically, the MBMN is designed for learning global context information and important long-distance dependence on the query, while SMN is utilized for modeling sequential relationships of contexts. Inspired by the performance improvement yielded by NLP features, three features are designed in this system. The overall combination of the three parts achieves rank 1st in all teams.

5 Conclusion

In this paper, we present the details of NLPCC 2018 shared task 5, including task formulation, datasets, evaluation metrics, results of submitted systems, and representative systems. This task investigates the performance of various systems on Chinese multi-turn conversations. We release a large corpus which contains more than 5 million multi-turn conversation sessions. There are 10 teams targeting at conversation generation and 5 teams at retrieval. As presented, there is still a long way for both multi-turn conversation generation and response retrieval models. It is expected that there are more and more researchers focusing on multi-turn conversation and moving the state-of-art forward.

References

1. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
2. Vinyals, O., Le, Q.V.: A neural conversational model. arXiv preprint arXiv:1506.05869 (2015)
3. Sordoni, A., et al.: A neural network approach to context-sensitive generation of conversational responses. In: HLT-NAACL, pp. 196–205 (2015)
4. Tian, Z., Yan, R., Mou, L., Song, Y., Feng, Y., Zhao, D.: How to make context more useful? an empirical study on context-aware neural conversational models. In: ACL, vol. 2, pp. 231–236 (2017)
5. Serban, I.V., Sordoni, A., Bengio, Y., Courville, A.C., Pineau, J.: Building end-to-end dialogue systems using generative hierarchical neural network models. In: AAAI, pp. 3776–3784 (2016)
6. Serban, I.V. et al.: A hierarchical latent variable encoder-decoder model for generating dialogues. In: AAAI, pp. 3295–3301 (2017)
7. Xing, C., et al.: Topic aware neural response generation. In: AAAI, pp. 3351–3357 (2017)
8. Li, J., Galley, M., Brockett, C., Gao, J., Dolan, B.: A diversity-promoting objective function for neural conversation models. In: HLT-NAACL, pp. 110–119 (2016)
9. Zhao, T., Zhao, R., Eskenazi, M.: Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In: ACL, vol. 1, pp. 654–664 (2017)
10. Li, J., Galley, M., Brockett, C., Spithourakis, G.P., Gao, J., Dolan, W.B.: A persona-based neural conversation model. In: ACL, vol. 1, pp. 994–1023 (2016)
11. Yan, R., Song, Y., Wu, H.: Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In: SIGIR, pp. 55–64 (2016)

12. Wu, Y., et al.: Sequential matching network: a new architecture for multi-turn response selection in retrieval-based chatbots. In: ACL, vol. 1, pp. 496–505 (2017)
13. Papineni, K., Roukos, S., Ward, T., et al.: IBM research report bleu: a method for automatic evaluation of machine translation. In: ACL, vol. 2, pp. 311–318 (2002)

Overview of the NLPCC 2018 Shared Task: Open Domain QA

Nan Duan(✉)

Microsoft Research Asia, Beijing, China
nanduan@microsoft.com

Abstract. We give the overview of the open domain QA shared task in the NLPCC 2018. In this year, we release three sub-tasks including Chinese knowledge-based question answering (KBQA) task, Chinese knowledge-based question generation (KBQG) task, and English knowledge-based question understanding (KBQU) task. The evaluation results of final submissions from participating teams will be presented in the experimental part.

Keywords: Question answering · Question generation
Question understanding

1 Background

Question Answering (or QA) is a fundamental task in Artificial Intelligence, whose goal is to build a system that can automatically answer natural language questions. In the last decade, the development of QA techniques has been greatly promoted by both academic and industry fields, and many QA-related topics have been well studied by researchers from all over world.

In order to further advance QA-related research in China, we organize this open domain QA shared task series in the past several years via NLPCC, and in this year, we release following 3 sub-tasks: (1) Chinese Knowledge-based Question Answering (KBQA); (2) Chinese Knowledge-based Question Generation (KBQG); and (3) English Knowledge-based Question Understanding (KBQU). You can see that comparing to previous two shared tasks, we retain the KBQA task and add KBQG and KBQU as two new tasks. The reason of adding these two new tasks is that we think the capabilities of asking questions in a proactive way and understanding user utterances in a deep way are very important to building human-computer interaction engines, such as search engine, chitchat bot, and task bot.

2 Task Description

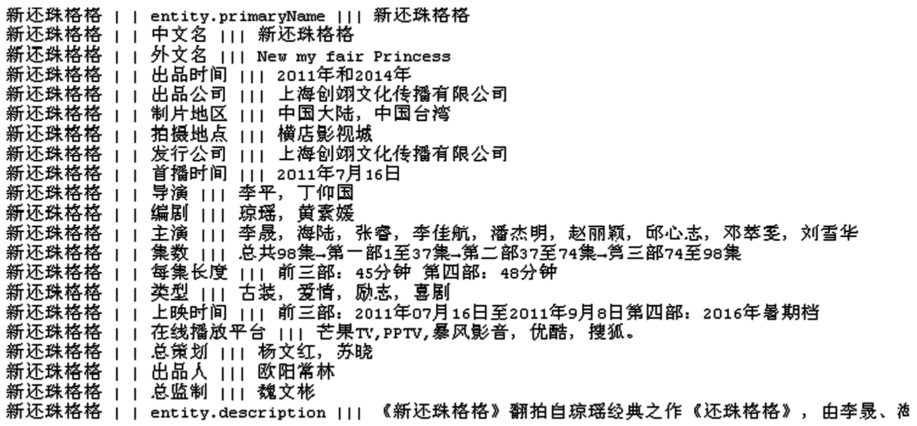
The NLPCC 2018 open domain QA shared task includes 2 sub-tasks for Chinese language: KBQA and KBQG, and 1 sub-task for English language: KBQU.

2.1 KBQA Task

For KBQA task, we provide a train set and a test set. In train set, both questions and their golden answers are provided. In test set, only questions are provided. The participating teams should predict an answer for each question in test set, based on a given large-scale Chinese KB. If no answer can be predicted for a given question, just set the value of <answer id="X"> to an empty string. The quality of a KBQA system will be evaluated by answer exact match. An example in train set is given below:

<question id="X">	你知道迪克牛仔是什么属相吗？
<answer id="X">	鼠

We provide a large-scale Chinese KB to participating teams, and it includes knowledge triples crawled from web. Each knowledge triple has the form: <Subject, Predicate, Object> , where 'Subject' denotes a subject entity, 'Predicate' denotes a relation, and 'Object' denotes an object entity. A sample of knowledge triples is given in Fig. 1, and the statistics of the Chinese KB is given in Table 1.



```
新还珠格格 | | entity.primaryName ||| 新还珠格格
新还珠格格 | | 中文名 ||| 新还珠格格
新还珠格格 | | 外文名 ||| New my fair Princess
新还珠格格 | | 出品时间 ||| 2011年和2014年
新还珠格格 | | 出品公司 ||| 上海创翊文化传播有限公司
新还珠格格 | | 制片地区 ||| 中国大陆，中国台湾
新还珠格格 | | 拍摄地点 ||| 横店影视城
新还珠格格 | | 发行公司 ||| 上海创翊文化传播有限公司
新还珠格格 | | 首播时间 ||| 2011年7月16日
新还珠格格 | | 导演 ||| 李平，丁仰国
新还珠格格 | | 编剧 ||| 琼瑶，黄素媛
新还珠格格 | | 主演 ||| 李晟，海陆，张睿，李佳航，潘杰明，赵丽颖，邱心志，邓萃雯，刘雪华
新还珠格格 | | 集数 ||| 总共98集→第一部1至37集→第二部37至74集→第三部74至98集
新还珠格格 | | 每集长度 ||| 前三部：45分钟 第四部：48分钟
新还珠格格 | | 类型 ||| 古装，爱情，励志，喜剧
新还珠格格 | | 上映时间 ||| 前三部：2011年07月16日至2011年9月8日第四部：2016年暑期档
新还珠格格 | | 在线播放平台 ||| 芒果TV,PPTV,暴风影音，优酷，搜狐。
新还珠格格 | | 总策划 ||| 杨文红，苏晓
新还珠格格 | | 出品人 ||| 欧阳常林
新还珠格格 | | 总监制 ||| 魏文彬
新还珠格格 | | entity.description ||| 《新还珠格格》翻拍自琼瑶经典之作《还珠格格》，由李晟、
```

Fig. 1. An example of the Chinese KB.

Table 1. Statistics of the Chinese KB.

# of Subject Entities	8,721,640
# of Triples	47,943,429
# of Averaged Triples per Subject Entity	5.5

2.2 KBQG Task

For KBQG task, we provide a train set and a test set. In train set, both triples and their golden questions are provided. In test set, only triples are provided. The participating teams should generate a natural language question for each triple in test set, and this generated question can be answered by the object entity of the given triple. The quality of a KBQG system will be evaluated by BLEU-4. An example in train set is given below:

<triple id="X">	微软 ||| 创始人 ||| 比尔盖茨
<question id="X">	是谁创办了微软公司？

2.3 KBQU Task

For KBQU task, we provide a train set and a test set. In train set, both questions and their golden logical forms are provided. In test set, only questions are provided. The participating teams should predicate a logical form for each question in test set. The quality of a KBQU system will be evaluated by logical form exact match. An example in train set is given below:

<question id="X">	what is fight songs of Maryland
<logical form id="X">	(lambda ?x (sports.team.fight_song Maryland ?x))

3 Evaluation Results

There are 19 submissions to the KBQA task, and Table 2 lists the evaluation results.

Table 2. Evaluation results of the KBQA task.

Organization	System name	Answer extract match
Central China Normal University	CCNU-319	0.3900
天津深思维科技有限公司	DeeperThought	0.4498
Chinese Academy of Sciences, Institute of Automation	Dream_on_Road	0.4676
Dalian University of Technology	DUTIR_9147	0.2832
East China Normal University	ECNU	0.5275
Guangdong University of Foreign Studies	GDUFSLEC	0.0971
China Academy of Engineering Physics	Lawe	0.3366
AI Lab, Lenovo Research	LEQAU 主系统	0.5647
AI Lab, Lenovo Research	LEQAU副系统a	0.5696

(*continued*)

Table 2. (*continued*)

Organization	System name	Answer extract match
AI Lab, Lenovo Research	LEQAU副系统b	0.5502
NetDragon Websoft Inc	NDers	0.6294
Northeastern University	NEUQA	0.5825
Peking University	Pkult	0.4984
Southeast University	SEU- WDS-KBQA	0.6926
Soochow University	SUDA-HLT	0.4337
University of Science and Technology of China	USTC-NELSLIP	0.5647
Xi'an Jiao Tong University	XJBot	0.6294
Zhejiang University	Yiwise-KBQA	0.6359
Central China Normal University	zilean	0.5023

There are 9 submissions to the KBQG task, and Table 3 lists the evaluation results.

Table 3. Evaluation results of the KBQG task.

Organization	System name	Character BLEU-4
Southwest University	AQG	0.4723
Southwest University	AQG-PAC_greedy_relation_predict	0.4360
Southwest University	AQG-PAC_soft_relation_predict	0.4188
Southwest University	AQG-question_sentence&relation_predict	0.4141
Central China Normal University	CCNU-319	0.4131
Peking University	ICL-1	0.4781
Peking University	ICL-2	0.3820
Southeast University	LPAI	0.3647
Central China Normal University	unique AI group	0.3652

There are 3 submissions to the KBQU task, and Table 4 lists the evaluation results.

Table 4. Evaluation results of the KBQU task.

Organization	System name	Logical form exact match
AI Lab, Lenovo Research	LEQAU 主系统	0.3020
AI Lab, Lenovo Research	LEQAU 副系统a	0.3060
AI Lab, Lenovo Research	LEQAU 副系统b	0.1900

4 Conclusion

This paper briefly introduces the overview of this year's 3 open domain QA shared tasks. In the future, we plan to provide more QA datasets for Chinese QA field. In the future, we will build more datasets for QA research, such as multi-turn QA dataset and cross-lingual QA dataset.

Overview of the NLPCC 2018 Shared Task: Single Document Summarization

Lei Li[1(✉)] and Xiaojun Wan[2]

[1] Bytedance AI Lab, Beijing, China
lileilab@bytedance.com
[2] Institute of Computer Science and Technology, Peking University, Beijing, China
wanxiaojun@pku.edu.cn

Abstract. In this report, we give an overview of the shared task about single document summarization at the seventh CCF Conference on Natural Language Processing and Chinese Computing (NLPCC 2018). Short summaries for articles are consumed frequently on mobile news apps. Because of the limited display space on the mobile phone screen, it is required to create concise text for the main idea of an article. This task aims at promoting technology development for single document summarization. We describe the task, the corpus, the participating teams and their results.

Keywords: Text summarization · TTNews corpus · NLPCC 2018

1 Introduction

Summarizing documents is an important task in today's fast pace daily life. Self publishing media producers are creating millions of articles every day. Therefore it is impossible to digest every piece fully. Automatic summarization technologies provides concise text snippets which can be consumed in short and fragmented time. Some mobile news reading apps such as Toutiao app provide a mode for summary view – summary text are displayed along with their titles, therefore they do not require users to click into the article page to read the full content. Some media publishers compile a summary article with daily highlights about certain topics of interest. Both case require techniques to automatically generate concise summaries for long text.

Document summarization methods can be categorized into two classes: extractive summarization and abstractive summarization [3]. The extractive summarization attempts to extract key sentences or key phrases from the original document, and then reorders these fragments into a whole piece. While the abstractive summarization focuses on generating new text and expressions which are based on the understanding of this document. Additionally, the document summary can be produced from a single document or multiple documents [3]. This shared task focuses on summarizing from a single document.

2 The Task

Traditional news article summarization techniques have been widely explored on the DUC and TAC conferences, and existing corpora for document summarization are mainly focused on western languages, while Chinese news summarization has seldom been explored. In the shared task of previous year's conference (NLPCC 2017), we prepared a corpus of news articles in Chinese, along with ground truth summaries [1]. This year, we continue to offer the single document summarization task. The goal is to generate a concise summary text for a give long article in Chinese. We provide a large corpus with both original document and ground truth summary for training. In addition, we have a separate corpus for evaluation. Only the documents nor the summaries are provided for the evaluation set. To further promote research in semi-supervised summarization techniques, we also prepare an additional set of documents, without reference summaries.

3 The Dataset

The provided dataset is referred as TTNews corpus in the following. It contains a training set and a test set. For the training set, it contains a large set of news articles browsed on Toutiao app and corresponding human-written summary which was used on news pushing and viewing on Toutiao app. The summaries are written by experts from Bytedance Inc [1], the parent company running the Toutiao app. Furthermore it contains another large set of news articles without summary. For the test set, it just contains the news articles. The news articles are from a variety of different sources and meanwhile contain of different topics, such as sports, foods, entertainments, politics, technology, finance and so on. As far as we know, TTNews corpus is the largest corpus for single document summarization in Chinese. There are 50,000 news articles with summary and 50,000 news articles without summary in the training set, and 2000 news articles in the test set. Note the training set remains the same as previous year's shared task [1], while the testing set are freshly baked. As shown in Table 1, the mean length of the short summary is 45 Chinese characters. The example of a news article and its reference summary is shown in Table 2.

3.1 Data Format

The training data contains two files, one for the articles with summaries and the other without summaries. Each line of the file contains a string of record in json format. Each record contains two fields: "article" and "summarization".

For evaluation, every line contains a record, in json format, with "index", "article", and "summarization" fields. The 'summarization' field is empty in the distributed data. Each submission must contain a single file with the name

[1] http://www.bytedance.com.

`submission.txt`. Each line of the submission file must contain one line of json string, with corresponding fields 'index' and "summarization".

All files are encoded in UTF-8.

Obtaining the dataset: You may download the training data from http://lab.toutiao.com/wp-content/uploads/2017/nlpcc2017textsummarization.zip. The testing data is available at http://lab.toutiao.com/wp-content/uploads/2018/05/nlpcc2018textsummarization_eval.zip

Use of The Data: You are free to use the data for research purpose. If you only use the training corpus, please cite the overview paper [1] with the following bib entry.

```
@InProceedings{hua2018overview,
    author="Hua, Lifeng and Wan, Xiaojun and Li, Lei",
    editor="Huang, Xuanjing and Jiang, Jing and Zhao, Dongyan
                 and Feng, Yansong and Hong, Yu",
    title="Overview of the NLPCC 2017 Shared Task:
             Single Document Summarization",
    booktitle="Natural Language Processing and Chinese Computing",
    year="2018",
    pages="942--947",
    isbn="978-3-319-73618-1"
}
```

Table 1. Statistics of the TTNews corpus

	# articles avg.	Article length avg.	Summary length
Training (w/ summary)	50000	994	45
Training (w/o summary)	50000	1526	-
Testing	2000	733	36

4 Evaluation Metric

We used ROUGE for automatic evaluation metric. ROUGE is the acronym name of Recall-Oriented Understudy for Gisting Evaluation, and contains a set of metrics used for automatic document summarization, machine translation evaluation and other tasks in NLP [2]. We defined the mean value of ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4, ROUGE-L, ROUGE-SU4, ROUGE-W-1.2 scores as the overall evaluation score. And we used ROUGE-1.5.5 toolkit to compute the overall score. Note that the length of each summary was limited to 60 Chinese characters at our shared task, so we used "`-l 60`" as the command line parameter for truncating longer news summary.

Table 2. Sample article and summary

Summary
距离地球8000光年的天鹅座星群黑洞再度活跃，在周围还有其他物质的情况下，黑洞进入休眠很罕见。
Article
6月28日消息，据国外媒体报道，在距离地球8000光年的天鹅座星群中，一个黑洞与一颗恒星正宛如天鹅般翩翩起舞。通常，物质在被黑洞巨大的引力吞没之前，会被加热并释放能量。这个名为V404Cygni的黑洞自1989年起就变得相对沉寂，然而，最近它又开始变得活跃起来。欧洲航天局于6月26日报道称，该机构的Integra（国际伽玛射线天体物理实验室）卫星观测到来自黑洞的"异常爆发的高能射线"。V404Cygni的活动亦被NASA的雨燕卫星观测到。该处产生的X射线耀斑同时被MAXI单元捕获到，后者是国际空间站日本模块的一部分。6月17日，欧洲航天局将Integral卫星用于观测V404Cygni，该机构随后证实，位于此处的黑洞确实再度活跃，而且活跃程度相当高。欧洲航天局Integral项目科学家埃里克·库克斯（ErikKuulkers）表示，耀斑产生的间隔非常短，不到一个小时，这种现象在其他黑洞系统中很罕见。其曾经一度成为天空中最明亮的物体，较蟹状星云还要亮50倍。后者通常是通过X光观测的天空中最明亮的地方。2013年6月，曾经有报道称，NASA观测到位于玉夫座中心的一处巨大黑洞逐渐进入休眠。通常，在吸收消耗完周围所有物质之后，黑洞会归于沉寂。在周围还有其他物质存在的情况下，黑洞进入休眠状态的情形相当罕见。例如，此次的V404Cygni附近就有恒星存在。这些质量巨大的黑洞在休眠与活跃之间变化的机理还尚不完全清楚，但是我们现在有更多的工具来观测这些过程。上述现象让天文学界感到相当兴奋。

5 Participating Teams and Submissions

Each team was allowed to submit at most 10 runs of results in the period of this shared task. The best score of the up to 10 submissions will be selected as the final score of each team. The participants were allowed to use any NLP resources and toolkits, but not allowed to use any other news articles with reference summaries. There were 18 teams submitting their final results in this shared task. The participating teams are shown in Table 3. Both extractive summarization and abstractive summarization techniques were developed by the participating teams.

6 Evaluation Results

There are 18 submitted teams in this shared task, and the results are shown in Table 4. As the table shows, WILWAL, Summary++, and CCNU_NLP are among the top three winners with the highest scores.

6.1 Representative System

We analyze the submission from the team Summary++, who ranked the second place in the final standing. Their solution is a modified version of pointer

Table 3. Participating teams

Team name	Organization
WILWAL	Wisers Information Limited, Wisers AI Lab
Summary++	Computational Linguistics Lab at Tsinghua University School of Humanities
CCNU_NLP	Huazhong Normal University School of Computer Science
freefolk	Peking University, National Engineering Research Center For Software Engineering
NLPCC2018_kakami	NLPCC2018_kakami
Casia-S	Research Center for Brain-inspired Intelligence,Institute of Automation, Chinese Academy of Science
Felicity_Dream_Team	Natural Language Processing Lab, Soochow University
dont lie	dont lie
CQUT_301_1	Chongqing University of Technology
lll_go	lll_go
NLPCC2018_LHZ_FD	NLPCC2018_LHZ_FD
DLUT_815	Dalian University of Technology
The Dream Team of NLP	Nanjing University Science and Technology
CMOS Doc Summarizer	China Mobile
CQUT_301_2	Chongqing University of Technology
NLP@WUST	College of Computer Science and Technology, Wuhan University of Science and Technology
基于神经网络与特征工程相结合的单文本摘要抽取	Networking Technology Lab at Beihang University School of Computer Science
Coordinate System	University of Electronic Science and Technology of China, School of Computer Science and Engineering, Computational Intelligence Laboratory

generator network [4]. The pointer generator uses a decoder with a mixture of generation from sequence-to-sequence model and a pointer to words in the source sentence. Two modifications are made based on this. Firstly, they proposed to use character embedding instead of word embedding, therefore there is no need to segment the words in Chinese. Secondly, they proposed to add one additional coverage vector in the decoding, which is useful in machine translation task.

Table 4. Evaluation results

Team name	Score
WILWAL	0.2938
Summary++	0.2853
CCNU_NLP	0.2827
freefolk	0.2814
NLPCC2018_kakami	0.2783
Casia-S	0.2739
Felicity_Dream_Team	0.2721
dont lie	0.2707
CQUT_301_1	0.2599
lll_go	0.2561
NLPCC2018_LHZ_FD	0.2293
DLUT_815	0.2170
The Dream Team of NLP	0.2133
CMOS Doc Summarizer	0.1638
CQUT_301_2	0.1629
NLP@WUST	0.1628
基于神经网络与特征工程相结合的单文本摘要抽取	0.1024
Coordinate System	0.0452

7 Conclusion

This paper briefly introduces the overview of single document summarization shared task at NLPCC 2018. There are 18 participants with successful submissions, which is a significant growth from the last year's shared task. Quite a few participants get exciting results in this corpus. Meanwhile, we release a large Chinese news articles and reference summaries corpus (TTNews corpus) for more large-scale research in Chinese document summarization.

Acknowledgement. We thank the colleagues from Bytedance to write summaries for the articles. We thank Huiru Zhang for processing and cleaning the testing data. We thank Lifeng Hua who prepared previous's summarization task. We also thank Jie Tang and his team Knowledge Engineering Group of Tsinghua University for verifying the evaluation process. We thank biendata.com for providing the online submission system which kept a standing board for every submission. Finally we also would like to thank the participants for their valuable feedback and outstanding results.

References

1. Hua, L., Wan, X., Li, L.: Overview of the NLPCC 2017 shared task: single document summarization. In: Huang, X., Jiang, J., Zhao, D., Feng, Y., Hong, Y. (eds.) NLPCC 2017. LNCS (LNAI), vol. 10619, pp. 942–947. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73618-1_84
2. Lin, C.y.: Rouge: a package for automatic evaluation of summaries. In: Proceedings of the ACL-04 Workshop: Text Summarization Branches Out, pp. 25–26 (2004)
3. Nenkova, A., McKeown, K.: A survey of text summarization techniques. In: Aggarwal, C., Zhai, C. (eds.) Mining Text Data, pp. 43–76. Springer, Boston (2012). https://doi.org/10.1007/978-1-4614-3223-4_3
4. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1073–1083. Association for Computational Linguistics (2017).https://doi.org/10.18653/v1/P17-1099

Overview of the NLPCC 2018 Shared Task: Social Media User Modeling

Fuzheng Zhang[1](✉) and Xing Xie[2]

[1] Meituan AI Lab, Beijing, China
zhangfuzheng@meituan.com
[2] Microsoft Research Asia, Beijing, China
xingx@microsoft.com

Abstract. In this paper, we give the overview of the social media user modeling shared task in the NLPCC 2018. We first review the background of social media user modeling, and then describe two social media user modeling tasks in this year's NLPCC, including the construction of the benchmark datasets and the evaluation metrics. The evaluation results of submissions from participating teams are presented in the experimental part.

Keywords: User modeling · Social media · Recommendation

1 Background

With the widespread of social media websites in the internet, and the huge number of users participating and generating infinite number of contents in these websites, the need for personalization increases dramatically to become a necessity. One of the major issues in personalization is building users' profiles, which depend on many elements; such as the used data [1, 2], the application domain they aim to serve [3, 4], the representation method and the construction methodology [5, 6]. Another major issue in personalization is personalized recommendation, which can be divided into different methods including contented based methods [7, 8], collaborative filtering based methods [9, 10], and hybrid methods [11, 12].

In the industry field, many influential user modeling products have been built, such as Netflix movie recommendation system, Amazon item recommendation system, etc. These kinds of systems are immerging into every user's life.

Under such circumstance, in this year's NLPCC shared task, we call the social media user modeling task that cover both personalized recommendation and user profiling tasks.

The remainder of this paper is organized as follows. Section 2 describes the provided dataset. In Sect. 3, we describe the detail of these two shared tasks. Section 4 describes evaluation metrics, and Sect. 5 presents the evaluation results of different submissions. We conclude the paper in Sect. 6, and point out our plan on future user modeling activities.

2 Data Description

The data, collected from a social media platform, contains the following five aspects:

(1) profile.txt describes users' profiles. Currently gender, province and city are provided.

user	gender	province	city	tags

(2) tags.txt describes users' tags. Each line contains a user and related tag.

user	tag

(3) social.txt describes users' following relationship, where user1 follows user2 on this social media platform.

user 1	user2

(4) tweets.txt describes what user posted. Each line contains a user and the posted tweet.

user	tweet

(5) checkins.txt describes users' location visits. The format is as follows, where POI is the location user visits, cate1, cate2, cate3 is the category of the POI in a hierarchical level. lat and lng is the latitude and longitude information and Name is the location name.

user	POI	cate1	cate2	cate3	lat	lng	name

All the files are UTF-8 encodes and tab separated.

3 Task Description

Given the social media dataset including the following heterogenous information: users' profiles (gender, province, city, tags), social ties (following relationship), users' published tweets, and users' location visits, the NLPCC 2018 social media user modeling shared task includes two shared tasks for social media dataset: User Tags Prediction task and User Following Recommendation task.

3.1 User Tags Prediction Task

Given users' other information except tags, predict which tags are related to a user.

3.2 User Following Recommendation Task

Given users' following relationship and other provided information, predict the users a user would like to follow in the future.

4 Evaluation Metrics

The quality of **User Tags Prediction (UTP)** and **User Following Recommendation (UFR)** subtasks will both be evaluated by F1@K,

$$\boldsymbol{P_i@K = \frac{|H_i|}{K}}, \quad \boldsymbol{R_i@K = \frac{|H_i|}{|V_i|}}, \quad \mathbf{F1}_i@\boldsymbol{K} = \frac{\boldsymbol{P_i@K * R_i@K}}{\boldsymbol{P_i@K + R_i@K}}$$

$$\mathbf{F1}@\boldsymbol{K} = \frac{\mathbf{1}}{\boldsymbol{N}}\sum_{i=1}^{N} \mathbf{F1}_i@\boldsymbol{K}$$

where $|H_i|$ is the correctly predicted item set (item refers to tag in UTP and user in UFR) for user i 's top K prediction, $|V_i|$ is the ground truth item set for user i. $P_i@K$, $R_i@K$ and $\mathrm{F1}_i@K$ is the precision, recall and F1 for a user i.

In **UTP**, we set $K = 3$.

In **UFR**, we set $K = 10$.

5 Evaluation Results

There are totally 39 teams registered for the above two shared tasks, and 4 teams submitted their results for UTP subtask and 3 teams submitted their results for UFR subtask. Table 1 and Table 2 lists the evaluation results respectively.

Table 1. Evaluation results of the User Tags Prediction (UTP) subtask.

	F1@3
Team 1	0.046268361
Team 2	0.033795288
Team 3	0.000124505
Team 4	0

Table 2. Evaluation results of the User Following Recommendation (UFR) subtask.

	Accuracy
Team 1	0.011645037
Team 2	0.004177592
Team 3	0.00000101

6 Conclusion

This paper briefly introduces the overview of this year's two social media user modeling shared tasks. We have 39 teams registered and 4 teams submitted final submissions. In the future, we plan to provide more social media datasets and call for new user modeling tasks.

References

1. Zhong, Y., Yuan, N.J., Zhong, W., Zhang, F., Xie, X.: You are where you go: inferring demographic attributes from location check-ins. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, pp. 295–304 (2015)
2. Abel, F., Gao, Q., Houben, G.-J., Tao, K.: Semantic enrichment of twitter posts for user profile construction on the social web. In: Extended Semantic Web Conference, pp. 375–389 (2011)
3. Zhang, F., Zheng, K., Yuan, N.J., Xie, X., Chen, E., Zhou, X.: A novelty-seeking based dining recommender system. In: The International Conference, pp. 1362–1372 (2015)
4. Morita, M., Shinoda, Y.: Information filtering based on user behavior analysis and best match text retrieval. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 272–281 (1994)
5. Adomavicius, G., Tuzhilin, A.: User profiling in personalization applications through rule discovery and validation. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–381 (1999)
6. Zhang, F., Yuan, N.J., Lian, D., Xie, X.: Mining novelty-seeking trait across heterogeneous domains. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 373–384 (2014)
7. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10
8. Popescul, A., Pennock, D.M., Lawrence, S.: Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. In: Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence, pp. 437–444 (2001)
9. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Comput. (Long. Beach. Calif.) **42**(8), 30–37 (2009)
10. 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, pp. 452–461 (2009)
11. Zhang, F., Yuan, N.J., Lian, D., Xie, X., Ma, W.-Y.: Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 353–362 (2016)
12. Zhang, F., Yuan, N.J., Zheng, K., Lian, D., Xie, X., Rui, Y.: Exploiting dining preference for restaurant recommendation. In: Proceedings of the 25th International Conference on World Wide Web, pp. 725–735 (2016)

Overview of the NLPCC 2018 Shared Task: Spoken Language Understanding in Task-Oriented Dialog Systems

Xuemin Zhao[1](✉) and Yunbo Cao[2](✉)

[1] Tencent Technology (Chengdu) Company Limited, Chengdu, China
xueminzhao@tencent.com

[2] Tencent Technology (Beijing) Company Limited, Beijing, China
yunbocao@tencent.com

Abstract. This paper presents the overview for the shared task at the 7th CCF Conference on Natural Language Processing & Chinese Computing (NLPCC 2018): Spoken Language Understanding (SLU) in Task-oriented Dialog Systems. SLU usually consists of two parts, namely intent identification and slot filling. The shared task made publicly available a Chinese dataset of over 5.8 K sessions, which is a sample of the real query log from a commercial task-oriented dialog system and includes 26 K utterances. The contexts within a session are taken into consideration when a query within the session was annotated. To help participating systems correct ASR errors of slot values, this task also provides a dictionary of values for each enumerable type of slot. 16 teams entered the task and submitted a total of 40 SLU results. In this paper, we will review the task, the corpus, and the evaluation results.

Keywords: SLU · Intent identification · Slot filling

1 Introduction

In task-oriented dialog systems, understanding of users' queries (expressed in natural language) is a process of parsing users' queries and converting them into some structure that machine can handle. The understanding usually consists of two parts, namely intent identification and slot filling. For example, given the utterance "给我来一首谭咏麟的朋友", the user's intent is to play a song, and "谭咏麟" fills one slots (singer) and "朋友" fills another (song).

Intents are global properties of utterances, which signify the goal of a user. Slots, on the other hand, are local properties in the sense that they span individual words rather than whole utterances. And the words that fill slots tend to be the only semantically loaded words in the utterance (i.e., the other words are function words). In the dialog systems, each type of intent corresponds to a particular service API, and the slots correspond to the parameters required by the API. SLU helps the dialog system to call the right back-end service using the right parameters to satisfy users' goals.

Traditionally, both of intent identification and slot filling are considered one utterance at a time by the SLU process, and the context information (including both the preceding queries in the same session and the user's situation information) is ignored by SLU and then handled by the dialog manager. The high cost to construct and maintain corpus is the main reason why the context information is not used in the SLU process. Usually, each utterance occurs within the context of a larger discourse between a person and a dialog system. Table 1 shows some example sessions, where without the context information from previous intra-session utterances we can't correctly do intent identification and slot filling for the utterance "取消" (utterance u_2 in session s_1, utterance u_2 in session s_2, utterance u_2 in session s_3) and "蒙曼" (utterance u_3 in session s_4). As the SLU process occurs in the early stage of a dialog system, well utilizing the context information can help avoid cascaded errors throughout the rest of the system.

Table 1. Example sessions, including session id **SID** and utterance ids **UID** in each session. Each utterance has an associated intent, while the corresponding slots are shown within each utterance using XML style tags.

SID	UID	Intent	Utterance
s_1	u_1	music.play	来一首<singer>冷漠</singer>的歌
s_1	u_2	music.pause	取消
s_2	u_1	navigation. navigation	导航去<destination>锡山紫金城</destination>
s_2	u_2	navigation. cancel_navigation	取消
s_3	u_1	phone_call. make_a_phone_call	呼叫<phone_num>4000008</phone_ num>
s_3	u_2	phone_call.cancel	取消
s_4	u_1	Others	说话
s_4	u_2	phone_call. make_a_phone_call	打电话给
s_4	u_3	phone_call. make_a_phone_call	<contact_name>蒙曼</contact_name>
s_4	u_4	navigation. navigation	<destination>增城宾馆</destination>
s_4	u_5	music.play	放音乐

Numerous techniques for SLU have been proposed, including traditional machine learning methods and hand-crafted features [1, 2, 4], deep learning methods [3, 5, 6], incorporating context information [1, 3], jointly optimizing intent detection and slot filling [5]. Despite this progress, direct comparisons between methods have not been possible because different datasets and domains are used in past studies.

The NLPCC 2018 Shared Task 4 (Spoken Language Understanding in Task-oriented Dialog Systems) provides a common testbed and evaluation suite for the SLU process. The shared task made publicly available a corpus of over 5.8 K sessions including 26 K utterances, which is a sample of the real query log from a commercial task-oriented dialog system. 16 teams entered the task, submitting a total of 40 SLU results.

This paper is organized as follows. First, Sect. 2 provides an overview of the task, the data and the evaluation metrics, all of which will remain publicly available to the community (NLPCC Shared Task 4, 2018). Then, Sect. 3 summarizes the results of the task. Finally, Sect. 4 briefly concludes.

2 Task Overview

2.1 Problem Statement

Spoken language understanding (SLU) comprises two tasks, intent identification and slot filling. That is, given the current query along with the previous queries in the same session, an SLU system predicts the intent of the current query and also all the slots associated with the predicted intent.

Included with the data is an ontology, which gives details of all the intents and the corresponding slots. To simplify the task, the dictionaries (e.g., singer, song, etc.) are

Table 2. Ontology and requirement in the task.

Intent	Slot	Provide-Slot-Dictionary	Do-Error-Correction
music.play	Song	YES	YES
	Singer	YES	YES
	Theme	YES	YES
	Style	YES	YES
	Age	YES	YES
	Toplist	YES	YES
	Emotion	YES	YES
	Language	YES	YES
	Instrument	YES	YES
	Scene	YES	YES
music.pause	–	–	–
music.prev	–	–	–
music.next	–	–	–
navigation. navigation	Destination	NO	NO
	custom_destination	YES	YES
	Origin	NO	NO
navigation.open	–	–	–
navigation. start_navigation	–	–	–
navigation. cancel_navigation	–	–	–
phone_call. make_a_phone_call	phone_num	NO	NO
	contact_name	NO	NO
phone_call.cancel	–	–	–
Others	–	–	–

provided for the slots with enumerable values while the slots with the non-enumerable values (e.g., phone_num, destination, contact_name, etc.) should be handled by rules or machine learning models. The textual strings, fed into a dialog system as input utterances, are mostly the transcripts translated from spoken language by ASR (Automatic Speech Recognition) and thus subject to recognition errors. If the enumerable slot values contain ASR errors, the SLU system should do slot value correction against the provided slot dictionaries. The non-enumerable slots don't need to do this for simplification. Table 2 gives details on the ontology used in this task.

The task studies the problem of SLU as a corpus-based task - i.e., the SLU systems are trained and tested on a static corpus of dialogs. The task is to re-run the SLU process on these dialogs - i.e., to take as input the dialogs translated from spoken language by ASR, and to output the SLU results. This corpus-based design was chosen because it allows different SLU systems to be evaluated on the same data.

2.2 Data

The dataset adopted by this task is a sample of the real query log from a commercial task-oriented dialog system, which is an in-car voice interface product. The data is all in Chinese. The evaluation includes three domains, namely music, navigation and phone call. Within the dataset, an additional domain label 'OTHERS' is used to annotate the data not covered by the three domains (as shown in Table 2). To simplify the task, we keep only the intents and the slots of high-frequency while ignoring others although they appear in the original data.

The entire data can be seen as a stream of user queries ordered by time stamp. The stream is further split into a series of segments according to the gaps of time stamps between queries and each segment is denoted as a "session". The annotation was achieved by first running an existing SLU system over the transcriptions, and then crowdsourcing to check the labels. Finally, the authors re-checked the labels by hand. The contexts within a session are taken into consideration when a query within the session was annotated. Table 1 gives some example sessions with annotations.

The entire dataset was randomly split into training and test dataset with a ratio of 4:1 at the session dimension. The statistics of the datasets are shown in Table 3. To help participating systems correct ASR errors, this task also provides a dictionary of values for each enumerable type of slot. Note that dictionaries are pruned such that they include all the values occurring in the dataset, but do not necessarily include all the values in real world. The statistics of the dictionaries are show in Table 4.

2.3 Evaluation

Depending on whether or not external resources can be used, the task can be divided into two types:

- Close evaluation – use only the training dataset provided by the task for model training and tuning, and output the results (in the evaluation stage) based only on the provided test set, not on any other dataset or resources.

- Open evaluation – can use any datasets and resources (in addition to the provided training dataset) for model training and tuning; and output the results (in the evaluation stage) based only on the provided test set, not no any other dataset or resources.

Table 3. The statistics of the datasets, where "# of" stands for "number of".

Item		Train dataset	Test dataset
# of sessions		4,705	1,177
# of utterances		21,352	5,350
Average session length		4.54	4.55
Average utterance length		5.93	6.08
# of error slot values		306	83
Intent	music.play	6,425	1,631
	music.pause	300	73
	music.prev	5	4
	music.next	132	34
	navigation.navigation	3,961	1,038
	navigation.open	245	55
	navigation.start_navigation	33	4
	navigation.cancel_navigation	835	207
	phone_call.make_a_phone_cal	2,796	670
	phone_call.cancel	22	18
	Others	6,598	1,616

Table 4. The statistics of the dictionaries.

Slot dictionary	Size
Song	6,870
Singer	2,667
Theme	140
Style	102
Age	139
Toplist	69
Emotion	135
Language	41
Instrument	30
Scene	145
custom_destination	3

Besides, we divided the task into another two sub-tasks: intent identification, and intent identification plus slot filling. In addition to the close and open evaluation, we got the following four sub-tasks:

- Sub-task 1: Intent Identification – Close;
- Sub-task 2: Intent Identification – Open;
- Sub-task 3: Intent Identification and Slot Filling – Close;
- Sub-task 4: Intent Identification and Slot Filling – Open.

However, it's very hard to do a close evaluation as the participating systems may use different Chinese word segmentor, word embedding, Name Entity Recognizer and dictionary resources. After the discussion with the participating teams, finally only Sub-task 2 and Sub-task 4 were retained in the final report, and Sub-task 1 and Sub-task 3 not.

For Sub-task 2, in order to balance the importance of each intent, we use $F1_{macro}$ of all the intents (not including the intent OTHERS) as the evaluation metric, calculated as the following equations,

$$P_{macro} = \frac{1}{N}\sum\nolimits_{i=1}^{N} \frac{\#\text{ of queries correctly predicted as intent } c_i}{\#\text{ of queries predicted as intent } c_i},$$

$$R_{macro} = \frac{1}{N}\sum\nolimits_{i=1}^{N} \frac{\#\text{ of queries correctly predicted as intent } c_i}{\#\text{ of queries labelled as intent } c_i},$$

$$F1_{macro} = \frac{2}{1/P_{macro} + 1/R_{macro}}.$$

For Sub-task 4, the evaluation metric is as given by the following equation,

$$P = \frac{\#\text{ of queries correctly parsed}}{\#\text{ of queries}},$$

where "# of queries" is the number of queries in the test set (including the queries with intent annotated as 'OTHERS'). "# of queries correctly parsed" denotes the number of queries for which the predicted intent and the predicted slot values (including the corrected values if correction is needed) are both exactly same as the annotations.

3 Results and Discussion

Altogether 16 teams participated in both of sub-tasks. Each team could submit a maximum of 3 results for each sub-task (Sub-task 2 and Sub-task 4), and both sub-tasks had 40 submitted entries in total. Table 5 gives the results on the metrics for each sub-task entry. As can be seen, the best result of Sub-task 2 is achieved by **AlphaGOU.entry3**, $F1_{macro} = 0.96157$; and the best result of Sub-task 4 is also achieved by **AlphaGOU.entry3**, P = 94.916% .

Table 5. Results of the evaluation.

Team ID	Sub-task 2			Sub-task 4	
	Entry	$F1_{micro}$	$F1_{macro}$	Entry	P
AlphaGOU	1	0.97090	0.96039	1	94.486%
	2	0.97234	0.96109	2	94.785%
	3	0.97365	**0.96157**	3	**94.916%**
CVTE_SLU	1	0.93390	0.92951	1	87.383%
	2	0.93454	0.93163	2	87.533%
	3	0.92675	0.91964	3	86.318%
DeepIntell	1	0.91659	0.60858	1	84.804%
	2	0.91942	0.67917	2	83.607%
	–	–	–	3	83.907%
DLUFL_SLU	1	0.93881	0.91612	1	88.112%
	2	0.94039	0.89501	2	88.710%
	3	0.93863	0.88936	3	88.243%
FAQRobot-wds	1	0.90584	0.86891	1	83.084%
	2	0.92594	0.91236	2	82.075%
	3	0.91481	0.88031	3	83.364%
HappyRogue	1	0.92785	0.76696	1	87.570%
	2	0.94211	0.89966	2	89.869%
	3	0.94105	0.89249	3	89.794%
HCCL	1	0.94873	0.92637	1	89.121%
	2	0.93211	0.91558	2	87.458%
	3	0.93339	0.91148	3	90.729%
ISCLAB	1	0.94474	0.85473	1	90.710%
laiye_rocket	1	0.93212	0.90285	1	79.813%
Learner	1	0.94886	0.91546	1	90.841%
	2	0.95197	0.93271	2	90.804%
	3	0.95223	0.94193	3	90.523%
orion_nlp	1	0.93065	0.88945	1	84.636%
	2	0.93038	0.90068	2	84.336%
	3	0.93035	0.88690	–	–
rax	1	0.93811	0.90409	1	88.168%
	2	0.93619	0.86398	2	87.028%
	3	0.93091	0.81014	3	86.430%
scau_SLU	1	0.94913	0.92989	1	78.374%
	2	0.94881	0.92962	2	78.486%
	3	0.94906	0.92972	3	79.720%
SLU-encoder	1	0.91981	0.87178	1	84.467%
	2	0.91978	0.87167	2	84.766%
	3	0.92023	0.87177	3	84.822%
SMIPG	1	0.91650	0.85222	1	82.972%

(*continued*)

Table 5. (*continued*)

Team ID	Sub-task 2			Sub-task 4	
	Entry	$F1_{micro}$	$F1_{macro}$	Entry	P
	2	0.91616	0.84256	2	82.916%
Team_4	1	0.85826	0.68953	1	74.785%

Table 5 also lists the metrics $F1_{micro}$ and $F1_{macro}$ for Sub-task 2. We could see that the metric $F1_{macro}$ is always less than the metric $F1_{micro}$ for all the entries. **CVTE_SLU.entry2** gets the least gap between $F1_{micro}$ and $F1_{macro}$, which is 0.00291. **DeepIntell.entry1** gets the greatest gap, which is 0.30801. The F1 metrics of all the intents for the two entries are shown in Fig. 1. In our released dataset, the example size of different intents is very different, and the maximum size is 895 times of the minimum. Because macro-averaging weights the metric toward the smaller classes, Teams should optimize the model performance for smaller classes (e.g. intents music.prev, navigation.start_navigation, and phone_call.cancel in Sub-task 2).

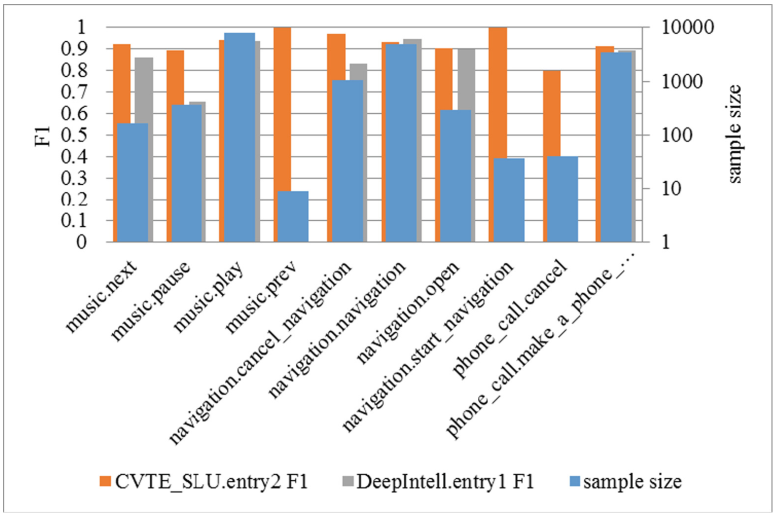


Fig. 1. Intent identification results of Sub-task 2 from **CVTE_SLU.entry2** and **DeepIntell.entry1**, and the sample size (including both of train and test datasets) for each intent. The F1 metrics for intents music.prev, navigation.start_navigation, and phone_call.cancel are all 0, and the sample size for these intents is 9, 37, and 40, respectively.

Figure 2 shows the results on slot filling (not combining the step of intent identification) of Sub-task 4 from **AlphaGOU.entry3**, which achieved the 1st place of Sub-task 4. Only the slots, whose sample size is larger than 100, are shown. One reason for the high performance of 'singer' and 'song' slots is that we released the slot dictionaries including all the values occurring in the dataset. The rich training data and obvious features is the main reason for the high performance of the destination slot. The main reason for the relative low performance of the contact_name slot is that, firstly we didn't release the users' contact name lists because of the privacy protection, secondly the ASR performance of contact names is very poor.

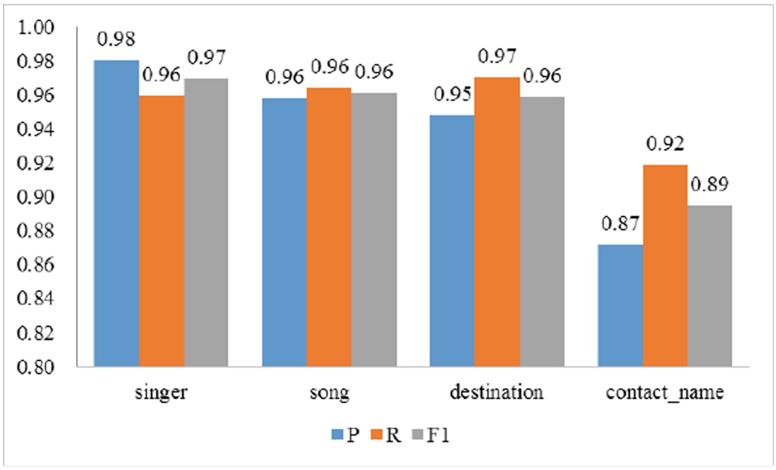


Fig. 2. Slot filling results (not combining the step of intent identification) of Sub-task 4 from **AlphaGOU.entry3**, where P stands for Precision, R for Recall, and $F1 = \frac{2}{1/P + 1/R}$. The results are computed from the utterances whose intent identification is correct.

Figure 3 shows the results on slot value correction (not combining the steps of intent identification and slot filling) of Sub-task 4. We can see a big difference for the performance. The top right 3 points are given by the 3 entries of Team 1, who has achieved a precision of around 0.75 and a recall of around 0.76. 18 points lie in the bottom left corner (0, 0), which means that 18 entries from 8 teams didn't correct slot value errors.

3.1 Some Representative Systems

In this section, some representative systems will be briefly introduced. While most of the systems use the neural networks, the 1st places of the two sub-tasks are achieved by the **AlphaGOU** system using the traditional techniques.

AlphaGOU system is a hybrid of context-independent model and context-dependent rules; the former is a pipelined framework which includes slot boundary detection, slot type classification, slot correction and intent classifier. Although all the used techniques are very traditional, the system achieved promising results.

Learner system uses a hierarchical LSTM based model. The dialog history is memorized by a turn-level LSTM, which is used to assist the intent identification and slot filling.

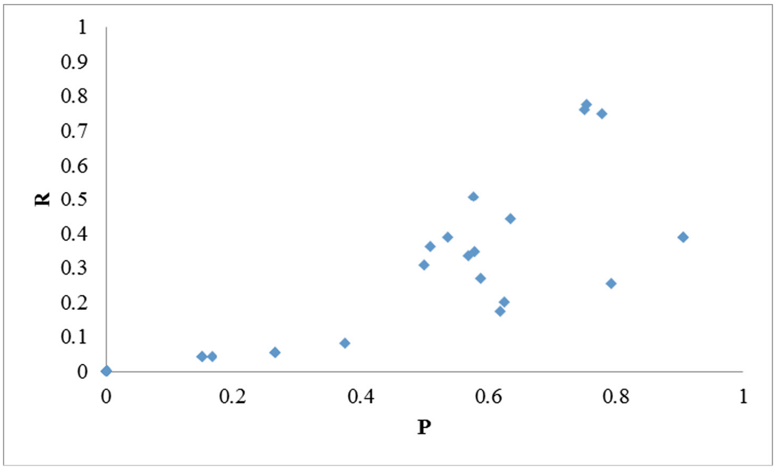


Fig. 3. Slot value correction results (not combining the steps of intent identification and slot filling) of Sub-task 4, where one point represents one entry result. P stands for Precision, and R stands for Recall.

ISCLAB system proposes a neural framework, named SI-LSTM model, which combines intent identification and slot filling together, and the slot information is used for determining the intent while the intent type is used to rectify the slot filling deviation.

4 Conclusion

In this paper, we present the overview of the NLPCC 2018 Shared Task: Spoken Language Understanding in Task-oriented Dialog Systems. The dataset adopted by this task is a sample of the real query log from a commercial task-oriented dialog system, which is an in-car voice interface product. The data is all in Chinese. The contexts within a session are taken into consideration when a query within the session was annotated. The entire dataset was randomly split into train and test dataset with a ratio of 4:1 at the session dimension. In the evaluation, two sub-tasks are designed. Sub-task 2 is intent identification, and Sub-task 4 is intent identification and slot filling. Both sub-tasks had 40 submitted entries in total. The best result of Sub-task 2 is achieved by **AlphaGOU.entry3**, $F1_{macro} = 0.96157$, and the best result of Sub-task 4 is also achieved by **AlphaGOU.entry3**, P = 94.916% .

Acknowledgement. We are very grateful to the colleagues from our company for their efforts to annotate the data. And we also would like to thank the participants for their valuable feedback.

References

1. Bhargava, A., Celikyilmaz, A., Hakkani-Tür, D., Sarikaya, R.: Easy contextual intent prediction and slot detection. In: ICASSP 2013 (2013)
2. Gong, N., Shen, T., Wang, T., Qi, D., Li, C.H.: The Sogou spoken language understanding system for the NLPCC 2018 evaluation. In: NLPCC 2018 (2018)
3. Xu, P., Sarikaya, R.: Contextual domain classification in spoken language understanding systems using recurrent neural network. In: ICASSP 2014 (2014)
4. Mairesse, F., et al.: Spoken language understanding from unaligned data using discriminative classification models. In: ICASSP 2009 (2009)
5. Xu, P., Sarikaya, R.: Convolutional neural network based triangular CRF for joint intent detection and slot filling. In: ASRU 2013 (2013)
6. Luo, B., Feng, Y., Wang, Z., Huang, S., Yan, R., Zhao, D.: Marrying up regular expressions with neural networks: a case study for spoken language understanding. In: ACL 2018 (2018)

WiseTag: An Ensemble Method for Multi-label Topic Classification

Guanqing Liang(✉), Hsiaohsien Kao, Cane Wing-Ki Leung, and Chao He

Wisers AI Lab, Wisers Information Limited, Wan Chai, Hong Kong
{quincyliang,shanekao,caneleung,chaohe}@wisers.com

Abstract. Multi-label topic classification aims to assign one or more relevant topic labels to a text. This paper presents the WiseTag system, which performs multi-label topic classification based on an ensemble of four single models, namely a KNN-based model, an Information Gain-based model, a Keyword Matching-based model and a Deep Learning-based model. These single models are carefully designed so that they are diverse enough to improve the performance of the ensemble model. In the NLPCC 2018 shared task 6 "Automatic Tagging of Zhihu Questions", the proposed WiseTag system achieves an F1 score of 0.4863 on the test set, and ranks no. 4 among all the teams.

Keywords: Topic classification · Tagging · Multi-label

1 Introduction

Multi-label topic classification aims to assign one or more relevant topic labels to a text. It can contribute to many downstream natural language processing applications including recommendation, user profiling and information retrieval. In the NLPCC 2018 shared task 6 "Automatic Tagging of Zhihu Questions task", participants are required to build a multi-label model that assigns relevant tags to a question from a set of predefined topic tags. Specifically, participants are given a training dataset of questions collected from Zhihu, a Chinese community question answering website, where each question in the dataset contains a title, an unique id and an additional description. The predefined tag set contains over 25,000 topic tags, and the task is to assign at most 5 topic tags to each question.

There are two major challenges in this task. First, the high dimensionality and sparsity of the output space increases the difficulty of model training. Second, the quality of the training data is inconsistent, since the questions are tagged collaboratively by users from the Zhihu community. Only the development and testing datasets are relabeled manually for the shared task.

To address the above challenges, we propose an ensemble model combining four single models that are trained based on different features and algorithms. Experimental results on the test set show that the ensemble model is more accurate and robust than the individual models, and ranks no. 4 among all participating teams with a F1 score of 0.4863.

The remainder of the paper is as follows. Section 2 reviews related work on multi-label topic classification. Section 3 details our WiseTag system including its overall architecture, data pre-processing steps, design of each single model and the ensemble model. Section 4 describes the evaluation setup and experimental results, and finally Sect. 5 concludes the paper.

2 Related Work

Generally, multi-label classification algorithms [12] can be classified into two categories, namely problem transformation methods and algorithm adaptation methods.

Problem Transformation Methods. These methods focus on transforming the multi-label classification problem into other existing well-studied problems. Widely used algorithms include Binary Relevance [13] which transforms the multi-label classification problem into a set of binary classification problems; Calibrated Label Ranking [14] which transforms multi-label classification into label ranking, and Random k-labelsets [15] which transforms the multi-label classification problem into the multi-class classification problem. However, the computation costs of these methods will be very high since there are over 25,000 labels to predict in the task at hand. We therefore do not consider these methods.

Algorithm Adaptation Methods. These methods aim to adapt existing single-label learning algorithms to the multi-label setting. For example, K-Nearest-Neighbors (kNN) has been extended to ML-kNN [16] for multi-label classification, Decision Tree has been extended to ML-DT [17], Support Vector Machine (SVM) has been extended to Rank-SVM [18], etc. After the consideration of the computational cost of the model, we only choose kNN model for further experiments.

More recently, researchers turn to use deep learning-based models for multi-label classification. In the recent Zhihu Machine Learning Challenge [19], all the wining solutions adopt Convolutional Neural Network (CNN) or Recurrent Neural Networks (RNN) models. The evaluation results show that deep learning models achieve the state-of-the-art solution for multi-label classification problem. Therefore, we will consider CNN and RNN model in this paper.

3 System Description

3.1 Overview

Figure 1 depicts the overall architecture of the WiseTag system. The system takes both title and description of a question as input, and generates the top-5 topic tags along with their predicted scores as output. First, the system performs data pre-processing including data cleaning and word segmentation. Second, the pre-processed data are fed into four different topic tagging models, namely the KNN model, the Information Gain (IG) model, the Keyword Matching (KM) model, and the Deep Learning (DL) model respectively. Finally, an ensemble model combines the four prediction results to output the top-5 predicted topic tags and their scores.

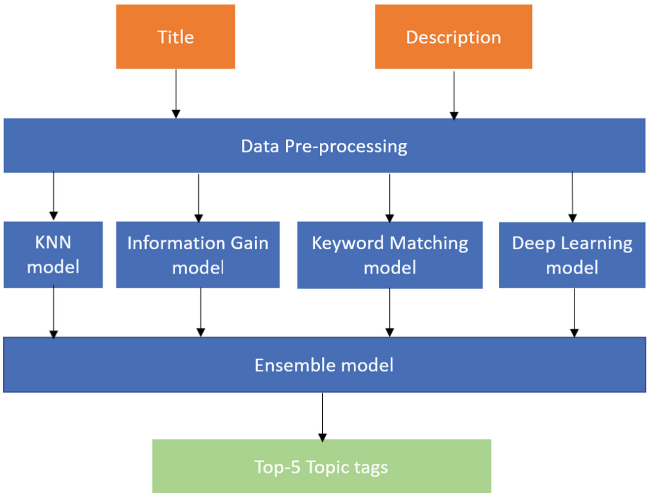


Fig. 1. Overview of the WiseTag system

Table 1. Samples of duplicated data

question_id	question_title	question_detail	tag_names	tag_ids
3926783272	有哪些极短的日剧推荐?		日剧	357
3509771290	有哪些极短的日剧推荐?		日剧	357

Table 2. Samples of question_detail irrelevant to question titles and topic tags

question_detail	Count
如题	3990
本问题已被收录至撰写你的知乎「城市手册」活动，更多活动详情请点击查看。	1031
RT	609

3.2 Data Pre-processing

The following pre-processing steps are performed on each input question:

1. Deduplicate the training data to remove instances having the same question title, descriptions, tag names and tag ids. Table 1 shows a duplicated data example.
2. Remove question descriptions (question_detail) that are redundant or irrelevant to the question titles or topic tags. Table 2 shows some samples of such question_detail which we identified based on data analysis.
3. Convert all characters to half-width and convert all characters to lowercase.
4. Perform word segmentation on question title and description using the Language Technology Platform (LTP) [5] engine.

5. Revise LTP's word segmentation results based on common phrases in the training set identified by the Pointwise Mutual Information (PMI) [8] algorithm.

3.3 KNN Model

KNN [10] is a simple and widely used model for classification, and often proven to be effective for text classification problems. For a given question, we first identify its top 10 ($k = 10$) similar questions and their assigned tags from the training set based on cosine similarity of TF-IDF [11] features. Then, we take the normalized frequencies of these tags as their predicted scores for the given question.

We observe that the KNN model performs well on short questions, but more frequent tags tend to dominate the predictions for new questions.

3.4 Information Gain (IG) Model

Information Gain (IG) is used to measure how much information a word contained in a question provides about the tag of the question. In the training phase, the IG of each word v is first computed for each tag t, denoted as $IG(v, t)$. It is then normalized such that $\sum_{t \in T} IG(v, t) = 1$, where T denotes the set of all predefined tags. In the prediction phase, the normalized IGs of all words in a given question are summed with respect to each tag separately. The summations are then normalized to obtain the predicted scores of the tags. The IG model has built-in feature selection property, therefore, we consider it to be one of our single models.

This model might suffer from over-fitting but has high interpretability because the tags are inferred based on the occurrence of the high-IG words. We observe that it outperforms the other single models except for the Deep Learning model.

3.5 Keyword Matching (KM) Model

We implement a rule-based classifier based on keyword matching. This model is motivated by our observation that some questions are simply labeled using keywords they contained. The KM model counts the predefined tags a given question contains, and takes the normalized tag frequencies as their predicted scores for the question. Table 3 shows some sample questions with their matched keywords (highlighted in red) and predicted tags.

In general, the KM model is able to identify some very specific topics, such as human and school names, thus serves as a good single model candidate for our ensemble model.

Table 3. Samples of keyword matching model prediction

question_title	**question_detail**	**prediction**
鹿晗到底做了什么_这么多喷他求科普		p(鹿晗)=0.50, p(科普)=0.50
张飞大战张郃的宕渠之战，张是否迁民成功?	网上有种，张飞得虚名，张得实惠的说法，是否正确?	p(张飞)=0.67, p(成功)=0.33

3.6 Deep Learning (DL) Model

Deep learning models often report the state-of-the-art performance in text classification tasks, so it is necessary to include one in our ensemble model. During the initial investigation, several DL models including CNN [3], RNN [23] and fastText [2] have been attempted on the given dataset. However, preliminary experimental results showed that RNN and fastText model are difficult to converge, which we attribute to the high dimensionality of the output layer. Thus, we choose CNN as our deep learning model for further experiments.

Figure 2 depicts the architecture of our CNN model, inspired by the 1st Place Solution for Zhihu Machine Learning Challenge [9]. The first layer is an embedding layer with dimension $(150 * 50)$, which allows a maximum of 150 words as input and the embedding size is set to 50. Note that 150 words are able to capture enough information, since the average lengths of question title and description are 13.17 and 70.84 words respectively. On top of the embedding layer, there are five parallel convolution layers with kernel sizes ranging from 1 to 5. The output of the convolution layers are concatenated and fed into a dense layer. A dropout layer with rate 0.5 is added after the dense layer to avoid overfitting. The dense layer is fully connected with the output layer with sigmoid as the activation function. The embedding layer adopts a word2vec embedding pre-trained using the given training set with Gensim [6]. The CNN model is implemented in the Keras framework [7].

3.7 Ensemble Model

To improve classification accuracy, WiseTag uses an ensemble model to combine the four aforementioned models $M = \{KNN, IG, KM, DL\}$ as follows:

$$ensemble\_model = \sum_{m \in M} w_m * m \tag{1}$$

where w_m is the weight assigned to model m.

The final evaluation only accepts up to 5 predicted labels per question. Hence, we output a tag only if it is ranked among the top-5 by our system with a predicted score above a decision threshold. In order to obtain the optimized weights of the different models and the decision threshold, we use the tool Hyperopt [1] for parameter tuning. The Hyperopt supports Tree-structured Parzen Estimators (TPE) algorithm, which is better than random search and grid search [20].

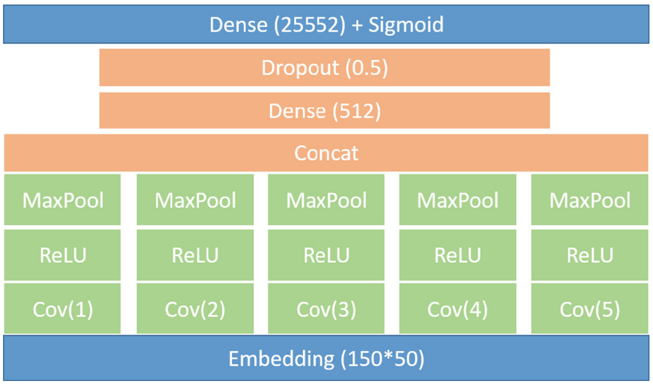


Fig. 2. CNN architecture

4 Experiments

4.1 Dataset

The original training set contains 721,608 instances, which are questions collected from Zhihu. Each question contains a title, an unique id and an additional description; and is tagged collaboratively by users from the Zhihu community. The average lengths of question title and description are 22.23 and 116.29 characters respectively, or 13.17 and 70.84 words respectively after performing word segmentation. There are 3.13 tags per question on average. After deduplication, only 721,531 instances are left for further training. The development and test sets contain 8,946 and 20,596 questions respectively, with their labels manually relabeled for the evaluation task at hand.

4.2 Evaluation Metrics

Each question in the test set will be assigned at most 5 predicted topic tags, sorted by their predicted relevant scores (or probabilities). Performance is evaluated based on the F1 measure with positional weighted precision. Let $correct\_num_{Pi}$ denote the number of correctly predicted tags at position i, and $predict\_num_{Pi}$ denote the number of predicted tags at position i. The precision, recall and F1 measure are computed as follows:

$$F_1 = \frac{2 * P * R}{P + R} \tag{2}$$

$$P = \frac{\sum_{i=1}^{5} correct\_num_{Pi}/log(i+2)}{\sum_{i=1}^{5} predict\_num_{Pi}/log(i+2)} \tag{3}$$

$$R = \frac{\sum_{i=1}^{5} correct\_num_{Pi}}{ground\_truth\_num} \tag{4}$$

4.3 Experimental Results

In order to understand the effectiveness of different modules and parameters, we carry out an extensive series of experiments.

We first study the impact of different pre-processing tasks on classification performance based on the validation set, where the training data is split into two parts, with 90% for training, and the remaining 10% for validation. Specifically, three pre-processing settings are evaluated:

- *raw ltp*: which uses the LTP engine for word segmentation.
- *character conversion + raw ltp*: which performs character conversion (half-width and lowercase normalization) and uses the LTP engine for word segmentation.
- *character conversion + raw ltp + PMI*: which performs character conversion, uses the LTP engine for word segmentation, and then performs the proposed word segmentation revision based on common words identified by the PMI algorithm.

Table 4 presents the results, which show that the proposed revised word segmentation with *character conversion + raw ltp + PMI* settings achieves consistent improvement over the *raw ltp* method under different neural network architectures (Table 5).

Table 4. Impact of pre-processing tasks on classification performance

Pre-processing tasks	Dense layer dimension	Validation score
raw ltp	1024	0.4015
character conversion + raw ltp	1024	0.4031
character conversion + raw ltp + PMI (proposed)	1024	**0.4044**
raw ltp	2048	0.3751
character conversion + raw ltp	2048	0.3758
character conversion + raw ltp + PMI (proposed)	2048	**0.3779**

Next, we empirically evaluate the impact of different parameters on the CNN-based DL model. The results with IDs (2, 3, 4) in Table 4 show the effects of varying the filter number, which produces the best performance when set to 512. The results with IDs (1, 4) show that with the same filter number, a larger kernel size increases classification performance. According to [22], Adam optimizer performs better than Stochastic Gradient Descent (SGD), so we only evaluate Adam with different numbers of the restarts. The results with IDs (4, 5) show that Adam [21] with 2 restarts achieves better validation score than Adam optimizer under the same parameter settings.

Table 5. Impact of different parameters on the DL model

ID	Filter kernel size	Filter number	Optimizer	Validation score
1	2, 3, 4	512	Adam	0.4067
2	1, 2, 3, 4, 5	1024	Adam	0.4066
3	1, 2, 3, 4, 5	256	Adam	0.4045
4	1, 2, 3, 4, 5	512	Adam	0.4083
5	1, 2, 3, 4, 5	512	Adam (with 2 restarts)	**0.4157**

Table 6. Comparison of different single models

Data set	Model	Precision	Recall	F1
Validation	KM	0.1846	0.2005	0.1922
	KNN	0.3949	0.3141	0.3499
	IG	0.3931	0.3784	0.3856
	DL	**0.4373**	**0.3985**	**0.4170**
Dev	KM	0.2395	0.2618	0.2501
	KNN	0.4327	0.3317	0.3755
	IG	0.4141	0.3999	0.4069
	DL	**0.4681**	**0.4218**	**0.4437**

This finding is consistent with that in [4], which reveals that Adam with 2 restarts and learning rate annealing is faster and performs better than SGD with annealing. In particular, we set the learning rate to 0.001 and train the model until convergence. We then halve the learning rate and restart by loading the previous best model.

The optimal parameters of the CNN model we adopted are {'batch size': 128, 'Filter number': 512, 'Kernel size': (1, 2, 3, 4, 5),'Dense layer size': 512, 'Threshold': 0.15 }.

Table 7. Evaluation on dev and test set

Data set	Model	Precision	Recall	F1
Test	Ensemble Model (1)	**0.5048**	0.4692	**0.4863**
	Ensemble Model (2)	0.4878	**0.4839**	0.4858
	Best single model (DL)	0.4715	0.4258	0.4475
Dev	Ensemble Model (1)	**0.5041**	0.4646	**0.4835**
	Ensemble Model (2)	0.4870	**0.4800**	**0.4835**
	Best single model (DL)	0.4681	0.4218	0.4437

Table 8. Model weights in ensemble model

Model/weight	KNN	IG	KM	DL	Threshold
Ensemble (1)	0.28	0.29	0.15	0.28	0.0759
Ensemble (2)	0.3445	0.2803	0.1432	0.2318	0.07812

Table 9. Evaluation results of the task 6 (Top-10 teams)

Rank	Team	Score
1	Tomwindows	0.6271
2	YiWise-QT	0.5840
3	NEUTag	0.5194
4	**WILWAL**	0.4863
5	Team_Wang	0.4840
6	Dream_driver	0.4536
7	iipnku	0.3981
8	HCY_FANS	0.3756
9	scau_AT	0.3404
10	CQUT_301_1	0.3383
...	...	...

In Table 6, we give the comparisons of different single models. The CNN-based DL model outperforms other models on both validation and dev datasets, while KNN and IG achieve comparable results.

Finally, we compare the performance of the ensemble models and the single models as shown in Table 7. Clearly, the ensemble models achieve significantly better results than the best single model on both dev and test datasets. Note that the difference between Ensemble Model (1) and Ensemble Model (2) is that the former use the weights optimized by the Hyperopt tool, while the later uses the weights designed based on rule of thumb that the weight is in proportion to the model's validation score. The weights of ensemble model are shown in Table 8. The final evaluation results of task 6 are presented in Table 9. Our team WILWAL ranks no. 4 with an F1 score of 0.4863 (evaluated on Ensemble Model (1)) on the test set.

5 Conclusion

This paper describes the proposed WiseTag system which performs multi-label topic classification based on an ensemble model. This ensemble model is built upon four diversified models, including: a KNN-based model, an Information Gain-based model, a Keyword Matching-based model and a Deep Learning-based model. Experimental results on the NLPCC-2018 shared task 6 show that the proposed model is effective, and ranks no. 4 with an F1 score of 0.4863. In our future work, we plan to investigate into semi-supervised approaches to multi-label topic classification.

References

1. https://github.com/hyperopt/hyperopt
2. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
3. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
4. Denkowski, M., Neubig G.: Stronger baselines for trustable results in neural machine translation. In: Proceedings of the First Workshop on Neural Machine Translation, Association for Computational Linguistics, Vancouver, pp. 18–27 (2017)
5. https://github.com/HIT-SCIR/ltp-cws
6. https://radimrehurek.com/gensim/models/word2vec.html
7. Chollet, F.: Keras: deep learning library for theano and tensorflow (2016). https://keras.io
8. https://en.wikipedia.org/wiki/Pointwise_mutual_information
9. https://github.com/chenyuntc/PyTorchText
10. https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
11. https://en.wikipedia.org/wiki/Tf-idf
12. Zhang, M., Zhou, Z.: A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. **26**(8), 1819–1837 (2014)
13. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern Recognit. **37**(9), 1757–1771 (2004)
14. Fürnkranz, J., Hüllermeier, E., Mencía, E.L., Brinker, K.: Multilabel classification via calibrated label ranking. Mach. Learn. **73**(2), 133–153 (2008)
15. Tsoumakas, G., Vlahavas, I.: Random *k*-labelsets: an ensemble method for multilabel classification. In: Kok, J.N., Koronacki, J., Mantaras, R.L., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 406–417. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74958-5_38
16. Zhang, M.L., Zhou, Z.H.: ML-kNN: a lazy learning approach to multi-label learning. Pattern Recognit. **40**(7), 2038–2048 (2007)
17. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: De Raedt, L., Siebes, A. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 42–53. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44794-6_4
18. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (eds.) Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS 2001), pp. 681–687. MIT Press, Cambridge (2001)
19. https://zhuanlan.zhihu.com/p/28912353
20. Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS 2011), pp. 2546–2554. Curran Associates Inc., USA (2011)

21. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
22. http://ruder.io/deep-learning-nlp-best-practices/index.html#fn:21
23. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735

Author Index