Towards Idea Mining: Problem-Solution Phrase Extraction from Text

Haixia Liu[1], Tim Brailsford[1], James Goulding[2], Tomas Maul[3], Tao Tan[4], and Debanjan Chaudhuri[5]

[1] Computer Science and Creative Technology, University of the West of England, Bristol, UK. haixia.liu@uwe.ac.uk
[2] N/LAB, Nottingham University Business School, Nottingham, UK.
[3] School of Computer Science, University of Nottingham Malaysia, Semenyih, Malaysia.
[4] The Faculty of Applied Science, Macao Polytechnic University, Macao, China.
[5] Computer Science, The University of Bonn, Germany.

Abstract. This paper investigates the feasibility of problem-solution phrases extraction from scientific publications using neural network approaches. Bidirectional Long Short-Term Memory with Conditional Random Fields (Bi-LSTM-CRFs) and Bidirectional Encoder Representations from Transformers (BERT) were evaluated on two datasets, one of which was created by University of Cambridge Computer Laboratory containing 1000 positive examples of problems and solutions (UCCL1000) with the corresponding phrases annotated. The F1-scores computed on the UCCL1000 dataset indicate that BERT is an effective approach to extract solution phrases (with an F1-score of 97%) and problem phrases (with an F1-score of 83%). To test the model's robustness on a different corpus with a different annotation scheme, a dataset consisting of 488 problem-solution samples from the Conference on Neural Information Processing Systems (NIPS488) was collected and annotated by human readers. Both Bi-LSTM-CRFs and BERT performances were dramatically lower for NIPS488 in comparison with UCCL1000.

Keywords: Text mining · Problem-solution extraction · NLP

1 Introduction

The discovery of original and new scientific ideas is a key phase of research innovation. This process usually starts with a literature review. Apart from researchers who are working in academia, scientists from industry and government also need to keep track of new trends. Given increasing publication rates, and the diversification of the literature into ever more specialized fields it is becoming increasingly challenging for both academic and industry researchers to decide how to most productively spend their time on selecting the important parts of a text. It is also difficult for government officers to pick up the most useful pieces of information that are available. The main goal of an abstract includes distilling the

main purpose of the corresponding paper. A paper's novel ideas are embedded in its abstract along with the problems it is solving and these can be extracted using pattern recognition. Mining scientific ideas by manually extracting them from a large body of literature tends to be massively time consuming. People can easily get lost in thousands of abstracts. Scientists in academia are trying to discover state-of-the-art methods for specific problems within their research area and are hoping to invent novel methods that are better than the existing ones; while, researchers in industry are looking for practical solutions that can be implemented and are working effectively in real scenarios. Instructors who are assessing essays online need an assistant that can automatically analyze essays [1].

In order to perform idea mining from text, a functional idea definition is crucial. Liu *et al.* [2] explored idea definition from a technical perspective, where ideas were represented by <problem, solution> pairs. How to extract the important information automatically from the text and make it structured is becoming increasingly important. In this paper, for the first time (to the best of our knowledge), two machine learning methods were compared to extract problem-solution phrases.

2 Related Work

A variety of methods for idea mining from text have been experimented by researchers. Thorleuchter *et al.*[3] introduced an approach for extracting ideas from unstructured text based on the length and the term weights of stop and non-stop words. The extracted ideas are represented by the retrieved words using text patterns, which are built around each targeted term in the new text. The represented words should occur on the left and right side of the non-stop words. The outputs using this method are a list of words and therefore the relations between the extracted words are lost, which makes the pattern less understandable. Some researchers investigated idea mining from the perspective of text classification rather than idea extraction. Christensen *et al.* [4] focused on classifying online community texts into Idea Text and Non-Idea Text using a supervised learning approach. They concluded that it is possible to automatically identify ideas written as text in online communities, however, their study did not provide methods for extracting ideas from text. Liu *et al.* [2] explored idea definition from a technical perspective, where ideas were represented by <problem, solution> pairs. It's stated in the paper [5] that the most important parts of the abstract are the *document problem* and *problem solution*. Liu *et al.* [2] used a part-of-speech tagging technique to extract noun-phrases from scientific publication abstracts. A rule based method was adopted to classify the noun-phrases into problems and solutions. Although <problem, solution> pairs embody an effective definition of ideas, the representation of problems and solutions is not easy to define. While the primary concepts are predominantly carried by the noun-phrases, simply using noun-phrases to represent problems and solutions is not enough. For example, from the sentence researchers have developed a computational method to

predict the function of unknown yeast genes, simply using the noun phrase yeast genes to represent the research problem is not as clear as using a span of consecutive text predict the function of unknown yeast genes. In order to make the expression of the <problem, solution> more comprehensible and understandable, a span of consecutive text to represent problems and solutions are worth study. Heffernan and Teufel [6] created a new corpus containing ground truth for problem-solution strings. They also present an automatic classifier to make a binary decision about problemhood and solutionhood of a given phrase. The classifier was based on supervised machine learning methods that intake a set of 8 features. However, their experiments were focusing on distinguishing problems from non-problems and solutions from non-solutions. Moreover, the 8 features being used were handcrafted, which is time consuming. This paper will utilize the annotated corpus by Heffernan and Teufel [6] for the task of problem-solution phrases extraction from a given sentence using neural networks.

2.1 Problem formation

Considering a single-labeled sentence T represented as an ordered set of N words, where $T = \langle w_1, w_2, \ldots, w_N \rangle$, then the functional definitions used to extract our representation of problem-solution phrases are as follows:

Problem-phrase: is an ordered subset of the text determined to be a *problem* extracted from T:

$$\phi = \langle wp_1, wp_2, \ldots, wp_n \rangle. \tag{1}$$

Solution-phrase: is an ordered subset of the text determined to be a *Solution* extracted from T:

$$\psi = \langle ws_1, ws_2, \ldots, ws_n \rangle. \tag{2}$$

Our goal is to extract ϕ or ψ given T.

As stated in the paper [7], the ground truth for problem-solution strings were defined to be at most one sentence long. The parsed dependencies were examined and some target words such as *problem* and *solution* were used as the seeds to identify subject position. Then, the syntactic arguments were chosen as the candidate Problem-Solution phrase. Semantically similar words of the target words were used to increase the variations. Examples of problem-solution phrases are shown in Figure 1.

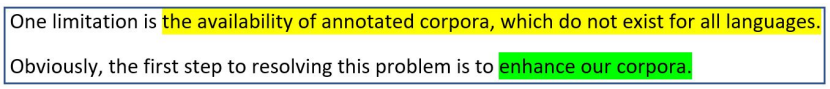

Fig. 1. An example of annotated problem phrase was highlighted in yellow shown on the top. An example of annotated solution phrase was highlighted in green shown on the bottom.

3 Methodology

Detecting Problem-Solution phrases is a form of Named Entity Recognition (NER) [8] since only parts of the sentence are considered as the target to be tagged. In order to detect Problem-Solution phrases, a classification scheme is determined based on the IO [9] format. Here, I is a token inside a chunk and O is a token outside a chunk. Although IO format cannot distinguish between adjacent chunks of the same named entity, it's suitable for our study due to our problem formation: the prediction is based on a single sentence with a single label. The goal of detecting Problem-Solution phrases is to correctly label every word in a sentence as one of the three categories: outside of the chunk (O), inside of the problem (I-P) or inside of the solution (I-S). Therefore, the three classes to be predicted are **I-P, I-S** and **O**.

3.1 Models for Extracting Problem-Solution phrases

Existing models for sequence labeling are linear statistical models, such as Maximum Entropy Markov models (MEMMs) [10] and Conditional Random Fields (CRFs) [11]. Research findings have shown that the model combining bidirectional LSTM (Bi-LSTM) networks and CRF is robust and it can produce accurate tagging performance without resorting to word embedding [12,13]. Considering that the task belongs to the tagging problem category and the words surrounding the problem-solution tags have certain patterns, it is hypothesized that using Bi-LSTM-CRF to detect problem-solution phrases can give better results.

Bidirectional LSTM Long Short-term Memory Networks (LSTM) [14] belong to recurrent neural network (RNN) [15]. LSTM networks are good at learning long-term dependencies. The LSTM had the ability to erase and add information to the cell states and it has regulated gates.

Bidirectional LSTM and CRF tagging (Bi-LSTM-CRFs) Bi-LSTM-CRFs [12] were explored in this study since the advantages of Bi-LSTM-CRFs are: (1) Bi-LSTM takes into account the information from both of the left and right side of the current word; (2) instead of predicting the label of the individual word independently, CRF has the transition matrix connecting the context with the current word. Research findings proved that Bi-LSTM-CRFs have achieved state-of-the-art performance in the task of NER [16]. While most literature focuses on extracting a relatively short span of text such as Location, Person, Organization etc, this study investigates how good Bi-LSTM-CRFs is to extract a longer span of text.

The workflow of utilizing bidirectional LSTM networks (bi-LSTMs) and conditional random fields (CRFs) to extract problem-solution phrases closely follows the steps described in the paper [12].

The widely used transformer based model BERT [17] was also explored on the two datasets.

4 Experiment

4.1 Dataset UCCL1000

UCCL1000 dataset was created by Heffernan and Teufel [7] on a subset of the ACL anthology [6] released in March 2016 containing 22,878 publications. A random subset of 2,500 papers was selected across the entire ACL timeline. Only documents having abstracts were considered. A ground truth for problem-solution strings was defined on the corpus. The annotated samples were independently validated for correctness by two annotators (the two authors of this paper). Correctness was defined by two criteria, which were detailed in the paper Heffernan and Teufel [7].

From the annotated sentences that passed the quality test for both independent assessors, 500 samples of positive problems, 500 samples of negative problems, 500 samples of positive solutions and 500 samples of negative solutions were randomly selected. The resulting 1000 positive samples (500 positive problems and 500 positive solutions) were used in this study.

4.2 Dataset NIPS488

In order to evaluate the neural networks performance on a different corpus with a different annotation scheme, a human annotated dataset is needed. Compared with the contents of a paper, abstracts have fewer licensing issues, resulting in more easily accessible data. Therefore, it's a good decision to obtain problem-solution phrases from the abstracts. A guiding principle underlying the annotation scheme was proposed: keep the sequence as short as possible, while retaining enough information to distinguish the novel contribution of the paper. Four hundred and fifty abstracts were obtained and analysed[7] from the Proceedings of the Neural Information Processing Systems conference (NIPS).

Guidance for annotation The annotation task was conducted using the abstract of the corpus. The annotation rules were as follows:

- A Problem (Solution) sequence might be a word, a list of words or an entire sentence. However, the sequence should not be separated by other words.
- For each abstract, only one problem and one solution were expected to be identified. If there was more than one problem (or solution) in an abstract, the most important one was chosen. When there are multiple Problem-Solution phrases in a paper, then the main problem (or solution) should be the one that is most related to the title.
- The chosen sequence should reflect the novelty of the paper.
- The chosen sequence should be as short as possible.
- The distance between the chosen sequence and the root of the sentence should be as close as possible.[8]

[6] `https://aclanthology.org/`

[7] Publication years: 2008 - 2016.

[8] The distance could be measured by the depth level on the parsed dependency tree.

– If no Problem-Solution phrase were identified, the abstract was excluded from the analysis.

The annotations were collected by seven computer science researchers, who manually highlighted Problem-Solution phrases on printed abstracts with coloured highlighter pens. Four-hundred and fifty abstracts were examined, out of which there were 244 abstracts having the problem-solution phrases clearly stated. An example of a human annotated problem / solution phrases is shown in Examples of problem-solution phrases were shown in Figure 2.

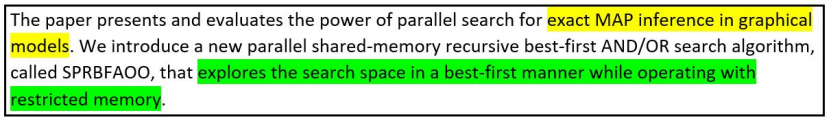

Fig. 2. An example of human annotated abstract. The *problem* (*solution*) phrases were highlighted in yellow (blue).

4.3 Dataset Summary

UCCL1000 dataset contains 7920 O (Outside), 7389 I-S and 6792 I-P entities. NIPS488 contains 6797 O (Outside), 2120 I-S and 1946 I-P entities. In comparison, NIPS488 is imbalanced and smaller.

4.4 Text Preprocessing

After basic text preprocessing such as noise removal, the next step is to make the raw text structured, which includes sentence segmentation, tokenization and token-label assignment.

Take the first sentence shown in Figure 2 for example, let x_1 represent the span of text The paper presents and evaluates the power of parallel search for exact MAP inference in graphical models and y_1 represent their labels. Part of the output of the first step is shown in Table 1.

The second step is to build dictionaries for tokens and tags respectively by converting tokens and tags to numerical values. An uncased tokenizing mechanism was adopted, meaning that all the letters were converted into lower-cased letters. The reason to use an uncased tokenized model is that the problem-solution statements are usually case-insensitive. Each token was assigned with a unique integer, also known as index, such that a sentence was represented by a list of integers. The tokens in the pre-trained embeddings were merged to the token dictionary.

After the esecond step, the sentences in Figure 2 were converted to a list of lists: [[6965, ..., 139], [15, ..., 1]], where each list represented a sentence. Similarly, each tag was represented by a unique index:

Sentence ID	Word	Tag
Sentence:1	for	O
Sentence:1	exact	I-P
Sentence:1	map	I-P
Sentence:1	inference	I-P
Sentence:1	in	I-P
Sentence:1	graphical	I-P
Sentence:1	models	I-P

Table 1. An example of outputs after the first step of text preprocessing. The words in the second column were excerpted from the first sentence shown in Figure 2.

$$y = [0, 1, 2] \tag{3}$$

4.5 Input Representations

Before training the models, all unique token indexes should be converted to meaningful input features. There are several options to represent the features, such as using one-hot-vector [18] and word embeddings [19]. Muneeb et. al. [20] pointed out two major drawbacks with one-hot-vector representations: first, the length of the vector is huge and second, there is no notion of similarity between words. Word embeddings [19] have proved to be an effective representation in some NLP tasks, such as sentence classification [21] and sentiment detection [22]. In comparison with randomly initialized word embeddings, pre-trained ones carry semantic information. A lot of researchers found that a good initialization of the input layer can improve the performance of models significantly [13]. Chung *et al.* [23] explained that the learned vectors contain semantic information pertaining to the underlying spoken words, and are close to other vectors in the embedding space if their corresponding underlying spoken words are semantically similar. Song *et al.* [24] found out that pre-trained embeddings are more effective than randomized ones. Cases et.al. [25] also demonstrated that pre-trained word2vec embeddings significantly outperformed random one as long as the network is properly configured.

Word embeddings are learned from raw text. A projection matrix is derived using unsupervised learning, which means, the values in the matrix are learned by maximizing the likelihood that words are predicted from their context. Each word can be represented by the corresponding row in the matrix, which is called word vector or word embedding. The dimensionality of the word vectors determines the size of the input layer. Although some researchers claimed [26] that the dimension of the word vectors should be chosen based on corpus statistics as well as NLP tasks, the empirical dimension is usually set between 50 to 300. Chung et.al. [23] found out that increasing the embedding size does not always result in improved performance for their experiment of learning word embeddings from speech and they further emphasised that word embeddings of 50

dimensions are able to capture enough semantic information of the words, as the best result was obtained by them. Bairong et. al. [27] investigated the different embedding vector sizes for the End-to-End Conversation Modeling task. In this experiment, pre-trained word embeddings were used as the input features. The word embedding dimension was set to 300. A vectorized representation of the input data is needed for training the models. Sequences with variable length need transformation to make sure each sequence has the same length. A post-sequence truncation method was adopted in this study, where the values were removed from the end of the sequence if it was larger then maxlen, which was set to 75/128 for Bi-LSTM-CRF/BERT respectively.

4.6 Training and Evaluation

Training Bi-LSTM-CRF and BERT BI-LSTM-CRF models were trained for each dataset separately. Word embedding vectors trained on GoogleNews were used to initialize the embedding layer since it outperforms randomly initialized embedding vectors in the embedding layer. The hidden unit size in the BiLSTM network was set to 50 because researchers found that model performance is not sensitive to hidden layer sizes [12] and 50 units were shown to be a good option [28]. The recurrent dropout rate was set to 0.1. Default parameters for the CRF layer were adopted [9]. Each model was trained for 20 epochs with batch size 32. The Embedding layer and BiLSTM network implementations were based on keras library [10].

BERT models were trained for 20 epochs using Huggingface Bert-base-uncased pretrained model [11]. Comparisons were done between 32 (train), 32 (validation) and 4 (train), 2 (validation) batch sizes, which were named as BS32-32 and BS4-2 respectively.

Evaluation k-fold cross-validation is a popular form of model validation [29]. Typically, researchers perform k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance [30]. Therefore, 5-fold cross-validation was adopted in this study.

F1-scores were reported in this study since the F1-score is a widely used measurement for most NER systems [31]. Because this study focused on extracting problem-solution phrases, the evaluation emphasised F1-scores for problem-solution entity recognition. In addition to F1-scores, precision and recall were reported.

4.7 Result Analysis

The results are shown in Table 2. The F1-scores for problem-solution phrase extraction on UCCL1000 dataset were 0.68/0.91 using Bi-LSTM-CRFs (BLC)

[9] `https://github.com/keras-team/keras-contrib/blob/master/keras_contrib/layers/crf.py`

[10] `https://keras.io/api/`

[11] `https://huggingface.co/bert-base-uncased`

and 0.83/0.97 using BERT. However, the results on NIPS488 dataset are very low. Batch size strategy comparison indicated that B4-2 outperforms B32-32 on NIPS488 dataset.

Tag-Model	Precision/Recall/F1(UCCL1000)	Precision/Recall/F1(NIPS488)
P-BLC32-32	0.64/0.72/0.68	0.09/0.12/0.10
P-BERT32-32	0.79/0.85/0.82	0.15/0.26/0.18
P-BERT4-2	0.81/0.85/0.83	0.17/0.29/0.22
S-BLC32-32	0.89/0.94/0.91	0.06/0.08/0.07
S-BERT32-32	0.97/0.98/0.97	0.10/0.18/0.13
S-BERT4-2	0.95/0.98/0.97	0.14/0.23/0.17

Table 2. Results (Precision/Recall/F1-score) generated by the model Bi-LSTM-CRFs (BLC) and BERT. UCCL1000 and NIPS488 indicated the corresponding dataset that the experiments were carried out on. 33-33 and 4-2 indicated the batch-sizes for train and validation (train-validation) datasets.

Examples of error analysis on UCCL1000 and NIPS488 datasets are shown in Figure 3 and 4 respectively.

```
  Word            ||True ||Pred
==============================
Another         : O     O
limitation      : O     O
that            : O     O
was             : O     O
discovered      : O     O
after           : O     O
the             : O     I-P
data            : O     I-P
analysis        : O     I-P
was             : O     I-P
that            : O     I-P
a               : I-P   I-P
data            : I-P   I-P
input           : I-P   I-P
error           : I-P   I-P
caused          : I-P   I-P
All             : I-P   I-P
Negative        : I-P   I-P
Chinese         : I-P   I-P
n-gram          : I-P   I-P
category        : I-P   I-P
```

Fig. 3. An example of problem phrase extraction error analysis on UCCL1000 dataset. The predicted problem phrases indicated that extra words were recognized as part of the problem entities but the ground truth showed that only the words in the clause are considered to be correct in this particular case.

```
Word           ||True ||Pred
==============================
We             : O    O
propose        : O    O
multiplicative : I-S  I-S
approximation  : I-S  I-S
scheme         : I-S  I-S
MAS            : I-S  I-S
for            : O    I-S
inference      : O    O
problems       : O    O
```

Fig. 4. An example of solution phrase extraction error analysis on NIPS488 dataset. The word *for* was wrongly detected as part of the solution phrases.

5 Discussion

A problem for training and evaluation in experiments of this nature is that it is difficult to enforce consistent annotation rules due to the differing subjective perceptions of the annotators. For the NIPS488 dataset, some of the annotators highlighted the problem (solution) explanations stated in a clause, rather than selecting the actual problem (solution) names. One of the biggest challenges when extracting only one main problem-solution phrase from an abstract is dealing with multiple problem-solution phrases that exist in the same abstract. This challenge might be the reason that the model could not achieve good result on the NIPS488 dataset.

6 Future Work

In the future, several aspects could be improved:

Adding a sentence classification stage To overcome the challenge caused by the second rule in the *Guidance for annotation*, labeling each sentence with one of the labels: main-problem, main-solution, main-ps (the examined sentence contains both main problem and main solution), non-main-ps (the examined sentence is neither main problem nor main solution sentence) before extracting problem-solution phases could be useful. Seventy nine abstracts were obtained from the Journal of Machine Learning Research (JMLR79). Each sentence was annotated with one of the labels described above. The annotation was done by one computer science researcher.

Annotation tool for collecting more data In the future, it is possible to use crowd sourcing techniques to get the same abstracts annotated by many different people. Many more annotations from authors should be collected.

Novel idea computation Potential useful ideas can be discovered by analyzing the problem-solution phrases that are not seen together in one abstract. Using a similar method in the paper [2], it is possible to accelerate the ideation process using a collaborative filtering algorithm, where problem phrases are considered as users and solution phrases as the items to be recommended.

7 Conclusion

The idea to extract problem-solution phrases from a given sentence using neural network techniques is new, to the best of our knowledge. With high quality dataset, the model Bi-LSTM-CRFs can spot meaningful patterns in text, which is intriguing and potentially valuable. It is hoped in the future, the work may contribute to novel idea computation and information retrieval (IR) in such a way that based on users' problems, the IR system can retrieve the papers that contain the solutions that can potentially solve these problems. However, although this work is promising, it needs to be repeated with larger high quality datasets.

References

1. A. Azman, M. Alksher, S. Doraisamy, R. Yaakob, E. Alshari, A framework for automatic analysis of essays based on idea mining, in: Computational Science and Technology, Springer, 2020, pp. 639–648.
2. H. Liu, J. Goulding, T. Brailsford, Towards computation of novel ideas from corpora of scientific text, in: Machine Learning and Knowledge Discovery in Databases, Springer, 2015, pp. 541–556.
3. D. Thorleuchter, D. Van den Poel, A. Prinzie, Mining ideas from textual information, Expert Systems with Applications 37 (10) (2010) 7182–7188.
4. K. Christensen, S. Nørskov, L. Frederiksen, J. Scholderer, In search of new product ideas: Identifying ideas in online communities by machine learning and text mining, Creativity & Innovation Management (2016).
5. B. Trawiński, A methodology for writing problem structured abstracts, Information processing and management 25 (6) (1989) 693–702.
6. K. HEFFERNAN, S. TEUFEL, Identifying problem statements in scientific text, Patrick Saint-Dizier 18.
7. K. Heffernan, S. Teufel, Identifying problems and solutions in scientific text, Scientometrics 116 (2) (2018) 1367–1382.
8. E. F. T. K. Sang, F. D. Meulder, Introduction to the conll-2003 shared task: Language-independent named entity recognition 21 (08) (2003) 142–147.
9. V. Krishnan, V. Ganapathy, Named entity recognition, Stanford Lecture CS229 (2005).
10. A. Mccallum, D. Freitag, F. C. N. Pereira, Maximum entropy markov models for information extraction and segmentation, Proc of Icml (2000) 591–598.
11. J. Lafferty, A. McCallum, F. Pereira, et al., Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the eighteenth international conference on machine learning, ICML, Vol. 1, 2001, pp. 282–289.
12. Z. Huang, W. Xu, K. Yu, Bidirectional lstm-crf models for sequence tagging, Computer Science (2015).
13. G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360 (2016).
14. S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.
15. T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, S. Khudanpur, Recurrent neural network based language model, in: Eleventh annual conference of the international speech communication association, 2010.

16. K. Mai, T.-H. Pham, M. T. Nguyen, T. D. Nguyen, D. Bollegala, R. Sasano, S. Sekine, An empirical study on fine-grained named entity recognition, in: Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 711–722.
17. J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
18. A. V. Uriartearcia, One-hot vector hybrid associative classifier for medical data classification, Plos One 9 (4) (2014) e95715.
19. T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781 (2013).
20. T. Muneeb, S. Sahu, A. Anand, Evaluating distributed word representations for capturing semantics of biomedical concepts, in: Proceedings of BioNLP 15, 2015, pp. 158–163.
21. A. Komninos, S. Manandhar, Dependency based embeddings for sentence classification tasks, in: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1490–1500.
22. B. Xue, C. Fu, S. Zhan, A study on sentiment computing and classification of sina weibo with word2vec, in: IEEE International Congress on Big Data, 2014, pp. 358–363.
23. Y.-A. Chung, J. Glass, Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech, Proc. Interspeech 2018 (2018) 811–815.
24. Y. Song, S. Shi, Complementary learning of word embeddings., in: IJCAI, 2018, pp. 4368–4374.
25. I. Cases, M.-T. Luong, C. Potts, On the effective use of pretraining for natural language inference, arXiv preprint arXiv:1710.02076 (2017).
26. K. Patel, P. Bhattacharyya, Towards lower bounds on number of dimensions for word embeddings, in: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2017, pp. 31–36.
27. Z. Bairong, W. Wenbo, L. Zhiyu, Z. Chonghui, T. Shinozaki, Comparative analysis of word embedding methods for dstc6 end-to-end conversation modeling track, in: Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop, 2017.
28. S. Trivedi, H. Rangwani, A. K. Singh, Iit (bhu) submission for the acl shared task on named entity recognition on code-switched data, in: Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching, 2018, pp. 148–153.
29. S. Geisser, The predictive sample reuse method with applications, Journal of the American statistical Association 70 (350) (1975) 320–328.
30. G. James, D. Witten, T. Hastie, R. Tibshirani, An introduction to statistical learning, Vol. 112, Springer, 2013.
31. B. Pal, A. K. Tarafder, M. S. Rahman, Synthetic samples generation for imbalance class distribution with lstm recurrent neural networks, in: Proceedings of the International Conference on Computing Advancements, 2020, pp. 1–5.