Scientific Keyphrase Extraction: Extracting Candidates with Semi-supervised Data Augmentation

Qianying Liu[1], Daisuke Kawahara[3], and Sujian Li*[2]

[1] School of Mathematical Science, Peking University
[2] Key Laboratory of Computational Linguistics, MOE, Peking University
{yingliu96, lisujian}@pku.edu.cn
[3] Graduate School of Informatics, Kyoto University, Japan
dk@i.kyoto-u.ac.jp

Abstract. Keyphrase extraction can provide effective ways of organizing scientific documents. For this task, neural-based methods usually suffer from performance unstability due to data scarcity. In this paper, we adopt the pipeline two-step method including candidate extraction and keyphrase ranking, where candidate extraction is a key to influence the whole performance. In the candidate extraction step, to overcome the low-recall problem of traditional rule-based method, we propose a novel semi-supervised data augmentation method, where a neural-based tagging model and a discriminative classifier boost each other and get more confident phrases as candidates. With more reasonable candidates, keyphrase are identified with recall promoted. Experiments on SemEval 2017 Task 10 show that our model can achieve competitive results.

Keywords: keyphrase extraction · neural networks · semi-supervised learning.

1 Introduction

With the number of scientific papers increasing dramatically, how to retrieve and manage them is a big problem. Keyphrases are usually used to organize scientific papers, since they carry the core information of an article in a concise way. With keyphrases, readers do not need to read the whole article and can have a general understanding of the content in a short time. Thus, automatic keyphrase extraction has drawn much attention recently and can benefit many downstream applications such as document summarization and question answering.

For keyphrase extraction there have been various approaches [8]. One research line casts it into a sequence tagging problem and builds an end-to-end neural model to extract keyphrases directly [1, 16, 5, 10, 17]. Neural models often suffer from time-consuming parameter fine-tuning and require a large amount of expensive training data. Due to the limited size of the training set, it is hard for

* Correspondence author

neural based models to get reliable results. The out-of-vocabulary (OOV) problem can also deteriorate the performance of neural models. Another research line is to use a multi-step pipeline to extract keyphases. Pipeline models are often composed of two main steps, which are candidate generation and keyphrase ranking. While only using a few hyper-parameters, pipeline models reported competitive performance over neural models with robust performance. However, the performance of candidate extraction is a key that determines the recall of the whole model and limits the overall performance. Wang at el. [27] used N-gram statistics for candidate extraction, although their methods can not extract those keyphrases that appear rarely in the text. Wang and Li [28] relied on rule-based candidate generation which mainly depended on part-of-speech (POS) patterns and performed especially poor on specific categories.

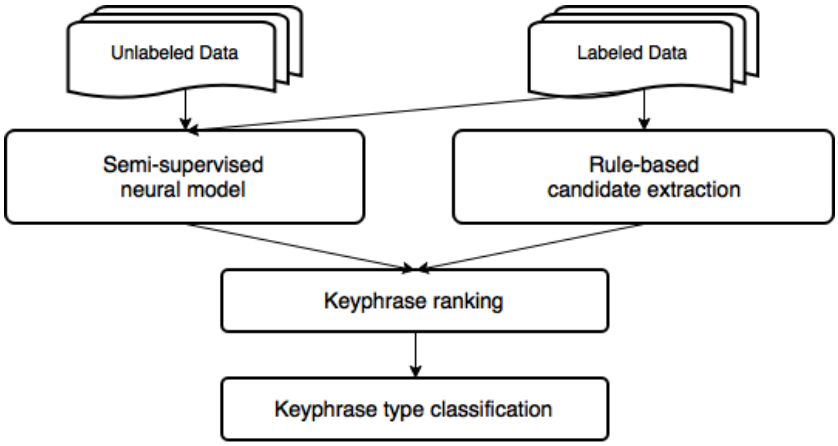

Fig. 1. A brief flow chart of the pipeline of our model.

To overcome the deficiencies of these candidate generation methods, we combine the two research lines by adding a neural model into the pipeline. We design a neural-based model to extract more keyphrase candidates. Following Wang and Li [28], we mix these candidates with the rule-based candidates and feed them into a multi-step pipeline. While extracting candidates with the neural model, we also cast this problem as a sequence tagging problem. As supplement of the rule-based candidates, the neural model can detect candidates of a wide variety of patterns and significantly improve the recall of candidates. Character level features and other handcrafted features are also used to relieve the OOV problem.

Due to the limited size of the training set, most studies utilize external knowledge or unlabeled data to improve the performance of keyphrase extraction [1, 16]. However, a preliminary experiment shows that simply adding more candidates could make the keyphrase ranking unit suffer from imbalanced data and

drag down its performance. Thus, we need to extract extra data from unlabeled data which has both quality and quantity. Here, we introduce a semi-supervised learning framework to get extra training data of high quality. This approach can both provide data augmentation for the neural model and balance the input data for the ranking unit. Experiments on the ScienceIE task[4] of SemEval 2017 show the effectiveness of our model.

2 Related work

Early approaches of keyphrase extraction often build a pipeline model, which first selects candidate phrases and then casts this problem into a binary classification problem. Handcrafted rules are designed by experts for the candidate selection step. Phrases with certain part-of-speech tags or n-grams that fit in certain syntactic patterns are chosen as candidates. While these methods often reach a high recall, pruning methods are often added to remove the candidates that are unlikely to be keyphrases [14, 13, 27]. Many different algorithms have been used for the second classification stage. Researchers often use various handcrafted features including statistical features, syntactic features and structural features [15, 26].

Wang and Li [28] proposed a traditional pipeline model on this task. A rule-based candidate generator feeds candidates to a linear model based keyphase ranking unit to identify keyphrases. Then random forest stacked upon a linear model is used for keyphrase classification. The performance of the model is limited because the rule-based candidate generator cannot deal with the wide range of keyphrase patterns, and simply adding new rules would make the keyphrase ranking unit suffer from serious imbalanced data.

Various neural models have been proposed in the sequence tagging problem recently. Chiu and Nichols [5] introduced a model for Name Entity Recognition (NER). Using the BIOES(Begin, Inside, Outside, End, Single) tagging scheme, they extracted character level features with a character level convolutional neural network (char-CNN) and used token level bi-directional Long Short-Term Memory Network (BiLSTM) for tagging. While using this tagging scheme, it is obvious that some label sequences (e.g., O I or O E) are unreasonable, so adding a Conditional Random Field (CRF) layer which can label a sequence jointly is a natural choice to improve the performance. Huang et al. [10] compared the performance of LSTM , BiLSTM, LSTM-CRF and BiLSTM-CRF on POS tagging, chunking and NER datasets, and BiLSTM-CRF outperformed all the other models. Ma and Hovy [17] proposed an end-to-end sequence labeling model via BiLSTM-CNN-CRF without task-specific resources, feature engineering or data pre-processing beyond pretrained word embedding. But keyphrase extraction for scientific articles have two main differences from NER that makes a simple BiLSTM-CRF perform poor. Firstly, keyphrase extraction has much smaller training data so that the neural network suffers from overfitting. Sec-

[4] https://scienceie.github.io/

ondly, keyphrases extraction suffers from far more OOV words since most scientific keywords are rare words.

Our approach takes the advantage of these methods, proposing a model that combines neural models and traditional machine learning algorithms to improve the performance.

Current approaches of data augmentation in NLP include Dong et al. [7] which built a neural architecture based on NMT and PPDB for paraphrasing for question answering. Yasunaga et al. [29] generated adversarial examples for POS tagging by giving continuous perturbations directly to the embedding. There are also some approaches of synthesizing adversarial samples to examine the performance of models. Rule based approaches include Hossini et al. [9] and Samanta and Mehta [22]. Jia and Liang [11] created adversarial examples for question answering to evaluate the performance of QA systems under the perturbation of the question sentence. To the best of our knowledge, there is still no previous work which adopts the neural-based data augmentation strategy for keyphrase extraction.

3 Model

Our model is composed of two stages: candidate extraction in section 3.1 and keyphrase ranking in Section 3.2.

3.1 Candidate extraction

Our candidate extracting unit includes two parts: a neural model extractor and a rule based extractor.

Neural model We introduce a neural model of three layers: The embedding layer, the token-level BiLSTM layer and the CRF tagging layer. The embedding combines pretrained word embedding, character level features, POS features and other handcrafted features. The LSTM layer encodes the surrounding information of the tokens. The CRF tagging layer jointly models the tags considering the dependencies across the tags.

Feature Vector For tokens $\{x_1, x_2, ..., x_n\}$ in a sentence, the embedding of each token is a concatenation of these four following parts: Word embedding $w(x_i)$, Character level embedding $c(x_i)$, Part-of-Speech tags $pos(x_i)$ and Handcrafted features $hc(x_i)$.

Word embedding $w(x_i)$ We initialized our model with the pretrained GloVe[21] 100d embedding. An unknown word token is added to represent the OOV words and its embedding is initialized as a zero vector.

Character level embedding $c(x_i)$ All tokens are padded to a fix size and we use CNNs of the filter size 2 and 3 to capture character level features. The character lookup table is initialized randomly. We use an full-connected layer to map the embedding into a fixed size.

Part-of-Speech tags $pos(x_i)$ We map the POS tags of the tokens to a vector space and randomly initialize them. The POS tags are labeled by the NLTK toolkit [3].

Handcraft features $hc(x_i)$ We use one hot embedding to map these following features: (1)Rarity, which depends on the TF-IDF of the token, (2)Capitalization, which depends on whether the token is all capitalized and whether the tokens first character is capitalized, (3)Whether the token is an IEEE word or not. We also combine these features to the embedding: (1)The TF-IDF score of the token based on English Wikipedia, (2)The length of the token, (3)The frequency of the token in wikiwords.

The final feature vector of a token x_i is $v(x_i) = (w(x_i); c(x_i); pos(x_i); hc(x_i))$

LSTM We concatenate all features and feed them into a token LSTM layer. The LSTM is bidirectional to capture the contextual information. We use dropout to avoid over-fitting. The LSTM hidden states are passed to a linear layer to map it into the size of the label space.

CRF Because of the strong dependencies across tags, we propose a CRF layer to model the tags jointly. A 'Begin of Sentence(BOS)' tag is added to the start of a sentence and a 'End of Sentence(EOS)' tag is added to the end of a sentence. When testing we use Viterbi algorithm to decode the tags.

Rule-based candidates We follow Wang and Li [28] and add rule-based candidates into the model. We use five rules to generate rule-based candidates. We extract phrases that match the POS pattern $NP = (NN * |JJ*) * (NN*)$ or consist of capital letters. Then we examine their length, whether they contain special digit or consists of only digits. If they pass all the check rules, then we add them as a candidate.

3.2 Keyphrase ranking

For identifying the keywords, we use stacking to ensemble random forest models. Ensemble models allow to combine more features without damping the performance.

Linear models are used to apply the weights of the stacking[23]. We use Linear Regression for this layer. A hyper-parameter α is used here as a score bound to balance the precision and the recall. Candidates whose scores are higher than α would be chosen as keyphrases. The identifier can be represented as follows:

$$C(p) = \begin{cases} 1, & \phi(p) > \alpha \\ 0, & \phi(p) \leq \alpha \end{cases} \tag{1}$$

where p stands for a candidate phrase, $C(p)$ stands for the whole identifier and $\phi(p)$ stands for the ranking model.

Instead of training the random forest and linear model jointly, we train them separately here. The low layer is fitted first and their results are fed into the linear model. The low layer actually works like a feature transformer.

Here we incorporate the linguistic features, context features and external knowledge features. We also use TextRank[18] and SGRank [6] as unsupervised features, examining whether the phrase is in the top n keyphrases according to these two algorithms.

4 Semi-supervised Data Augmentation

Semi-supervised data augmentation is mainly used to balance the data of the keyphrase ranking unit and augment the data for the neural model. In Algorithm 1 and Figure 2, y stands for the labeled data and y^* stands for the unlabeled data. We call the original ranking unit C and the neural candidate extractor G. R stands for the rule-based candidate generation unit. An identifier is built for this process and we call it Q. Q is similar to C except that we choose to use SGD Regression for Q and turn up the hyper-parameter α to ensure its precision remains high.

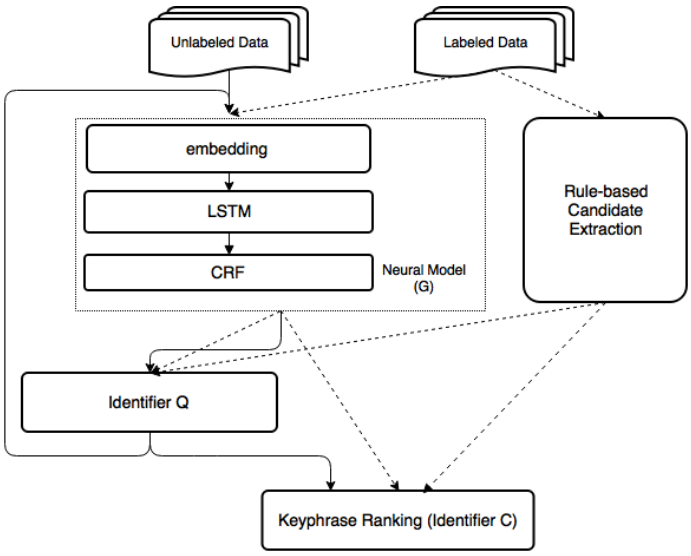

Fig. 2. A brief flow chart of our semi-supervised model. The solid line stands for the semi-supervised training process and the dotted line stands for the supervised training process.

The training of G and Q includes two steps which are presented by the two kinds of lines in Figure 2. The dotted line in Figure 2 shows the process of pretraining the sequence tagging model G and the identifier Q on labeled

data. The solid line shows the semi-supervised learning process. We use G to tag unlabeled data and feed the candidates into Q. In an ideal situation, the neural model should improve its generalization ability and the independent identifier should only allow high quality data to update the parameters of the neural model. We show the detailed architecture in Section 4.1 and how we use the simulated data to update our models in Section 4.2.

4.1 Architecture

Algorithm 1 Semi(Q, C, G, R, y, y^*)

for y_i in y **do**
    train G with $\{y_i, groundtruth\}$
for p_i in $G(y), R(y)$ **do**
    train Q with $\{p_i, groundtruth\}$
for p_i^* in $G(y^*)$ **do**
    $S = \{p_i^* \mid \text{if } Q(p_i^*) \text{ is true}\}$
    train Q with $\{p_i^*, false\}$
for p_i^S in S **do**
    train G with $\{p_i^S, true\}$
for p_i in $S, G(y), R(y)$ **do**
    train C with $\{p_i, true\}$

We first train G and Q through supervised training with labeled data. Then we let G extract candidates from the unlabeled data and use Q to rank them. For tokens $\{y_1, y_2, ..., y_n\}$ in an unlabeled sentence, the neural model labels them with the BILOS tagging scheme as $\{l(y_1), l(y_2), ..., l(y_n)\}$. We use these labels to extract phrases $P = \{p_1, p_2, ..., p_m\}$. Q identifies them similar to the keyphrase ranking unit and we pick out the positive results $S = \{p_{k_1}, p_{k_2}, ..., p_{k_j}\}$. We call the candidates that Q predict as *simulated data* which is denoted as S. Because of the high precision of Q, we can assume that these simulated data are reliable.

During semi-supervised training, we develop G and Q jointly while C is trained afterwards. To update G, the simulated data is saved and then fed back to G as positive examples for data augmentation. Q also updates its parameters by fitting all the candidates that the neural model extracts as negative data. While updating the two models in the semi-supervised training process, we mix the simulated data and real data together. After all unlabeled data goes through this structure, we stop the process and use G and R to extract candidates on the training data to fit C. When we fit C, S is added to the training data as positive examples to balance the data.

This architecture is only activated when we train the model. While testing, the test data only goes through the neural model G for candidate extraction.

4.2 Model Updating details

Weights While this process can augment reliable new data, the simulated data is still treated differently when we update the parameters. When we update the neural model with the simulated data, we add weights to these data and the weights are determined by the score of the candidates ranked by Q. These weights are also used when we update the ranking unit C. When we update C, we add a hyper-parameter β as a weight for all the simulated data. This hyper-parameter is set to 0.5 in our experiment.

Updating scheme We update the parameters of G by batch, and Q is updated after every epoch. Only the linear stacking layer of Q is updated while the random trees are fixed.

The purpose of updating Q is to force the precision of Q to go higher and the recall to go lower. This will lead to a result that less and less simulated data is used to update G every epoch. In other words, during training G, the potential amount of noise in the mixture of real data and simulated data declines every epoch, which exactly meets the need of G's trimming.

We separate the simulated data into 3 epochs when the neural model is trained. Then we let real data go through the model again until the model reaches convergence.

5 Experiment

5.1 Dataset

We use the SemEval 2017 Task 10 ScienceIE dataset in this approach. In this dataset, 500 paragraphs among the domains Computer Science, Material Sciences and Physics were selected, providing both the full text and additional metadata. Only one paragraph of each article is manually labeled that the size of the training data is very limited. Meanwhile, because of the wide range of terminology vocabulary in those domains, the huge size of OOV words in the pretrained embedding influences the performance of the model. In the process of the unsupervised training, we use the unlabeled context of the articles to extract simulated data.

5.2 Tagging scheme

In the process of extracting candidates with a neural model, we follow Luan et al. [16] and Ammer et al.[1] and cast the problem into a sequence tagging problem. We use the BIOES(Begin, Inside, Outside, End, Single) system to tag the context and assign three labels to each token. For example, the PROCESS keyword okta Markov Chain is labeled as B-PROCESS I-PROCESS E-PROCESS.

5.3 Hyperparameters and details

All parameter are tuned on the development set. We select the hyperparameters for the neural model and the identifiers when we warm them up separately. We use two filter sizes for CNN which are 2 and 3. The character embedding dimension is 50. The word embedding dimension is 100. The handcraft features dimension is 8. The Part of Speech feature dimension is 50. The token-level hidden dimension is 200. The token level LSTM layer is made up of two layers of BiLSTM. We use 0.18 and 0.5 for the hyperparameter α in the two identifiers.

While training, the batch size is 32 and dropout is set to 0.5 for the LSTM model and CNNs. We use the Adam[12] optimizer with a learning rate of 0.001 and a weight decay of 1e-8. We monitor the performance on the development set and use early stopping. The best parameters are fixed on the development set while we experiment supervised learning.

We use scikit-learn[20] to implement the linear models and random forest models. The neural model is implemented by Pytorch[19].

5.4 Performance comparisons

Span Level	**Identification**
Wang and Li [28]	0.510
Ammer et al. [1] (SM)	0.499
Ammer et al. [1] (Semi SM)	0.541
LuanOH17 [16]	0.521
LuanOH17 [16](Semi)	0.576
Ours	0.543

Table 1. Overall F1 scores for identification and classification(SemEval Subtask A and Subtask B). SM stand for single model.

Table 1 reports the results of our model and compares them with other SemEval Systems. We can see that the performance of our model is competitive against other models and out performs all the supervised single models, which proves the effectiveness of our model.

Table 2 compares our model with Wang and Li [28] whose candidates are rule-based. It shows the significant boost of candidate recall of our system and the improvement in the **TASK** category. This shows that our model can capture candidates of a wider variety while balancing between recall and precision.

We also examine how the semi-supervised architecture affects overall performance in Table 2. We report results of simply adding more candidates to show the importance of our semi-supervised architecture. It shows that simply adding more candidates does not work because the imbalanced data harms the performance of the ranking unit. Adding candidates without consideration of data balance seriously drags down the performance.

Span Level	Candidate Recall(train)				Results				
	M	P	T	Overall	K	M	P	T	Overall
Wang	0.715	0.608	0.334	0.683	0.510	**0.460**	0.399	0.072	0.409
Wang + IEEE	0.715	0.608	0.334	0.683	0.509	0.421	0.372	0.066	0.377
Wang + Unsupervised	0.715	0.608	0.334	0.683	0.501	0.418	0.340	0.059	0.361
Wang + Ngram	0.776	0.646	0.340	0.649	0.499	0.398	0.361	0.056	0.360
Ours without semi	0.814	0.707	0.511	0.717	0.494	0.413	0.357	0.084	0.366
Ours	**0.820**	**0.721**	**0.531**	**0.728**	**0.543**	0.450	**0.417**	**0.104**	**0.414**

Table 2. Comparisons of Wang and Li [28] and our model.

Span Level		Results				
		K	M	P	T	Overall
Ours without semi		0.494	0.413	0.357	0.084	0.366
+ Undersampling	AllKNN[24]	0.507	0.422	0.357	0.067	0.371
	TomekLinks[25]	0.485	0.408	0.348	0.101	0.359
+ Oversampling	SMOTE[4]	0.520	0.420	0.364	**0.115**	0.373
+ Combined Resampling	SMOTEENN[2]	0.510	0.414	0.366	0.065	0.371
Ours		**0.543**	**0.450**	**0.417**	0.104	**0.414**

Table 3. Comparisons of our model and other data balancing methods

Table 3 shows the results of applying other data balancing methods before the ranking unit. Here we remove the semi-supervised architecture and use other methods to balance data before the random forest model for keyphase ranking. We can see that our semi-supervised architecture works better than other data balancing algorithms that are based on resampling in the feature space.

6 Conclusion

In this paper we propose a system for the task of scientific keyphrase extraction which has a wide range of OOV words and keyphrase patterns. To improve the pipeline of candidate extraction and keyphrase ranking, we design a semi-supervised data augmentation method to identify more reliable keyphrase candidates to ensure the final extraction performance. We also use abundant features including character level features, linguistic features, context features and external knowledge features to relieve the OOV problem.

7 Acknowledgement

We thank the anonymous reviewers for their insightful comments on this paper. This work was partially supported by National Natural Science Foundation of China (61572049 and 61273278).

References

1. Ammar, W., Peters, M.E., Bhagavatula, C., Power, R.: The AI2 system at semeval-2017 task 10 (scienceie): semi-supervised end-to-end entity and relation extraction. In: Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017. pp. 592–596 (2017)
2. Batista, G.E., Bazzan, A.L., Monard, M.C.: Balancing training data for automated annotation of keywords: a case study. In: WOB. pp. 10–18 (2003)
3. Bird, S., Loper, E.: Nltk: the natural language toolkit. In: Proceedings of the ACL 2004 on Interactive poster and demonstration sessions. p. 31. Association for Computational Linguistics (2004)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. Journal of artificial intelligence research **16**, 321–357 (2002)
5. Chiu, J.P., Nichols, E.: Named entity recognition with bidirectional lstm-cnns. Transactions of the Association for Computational Linguistics **4**, 357–370 (2016)
6. Danesh, S., Sumner, T., Martin, J.H.: Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. In: Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics, *SEM 2015, June 4-5, 2015, Denver, Colorado, USA. pp. 117–126 (2015)
7. Dong, L., Mallinson, J., Reddy, S., Lapata, M.: Learning to paraphrase for question answering. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017. pp. 875–886 (2017)
8. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: A survey of the state of the art. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1262–1273 (2014)
9. Hosseini, H., Kannan, S., Zhang, B., Poovendran, R.: Deceiving google's perspective api built for detecting toxic comments. arXiv preprint arXiv:1702.08138 (2017)
10. Huang, Z., Xu, W., Yu, K.: Bidirectional lstm-crf models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
11. Jia, R., Liang, P.: Adversarial examples for evaluating reading comprehension systems. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017. pp. 2021–2031 (2017)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
13. Liu, Z., Huang, W., Zheng, Y., Sun, M.: Automatic keyphrase extraction via topic decomposition. In: Conference on Empirical Methods in Natural Language Processing. pp. 366–376 (2010)
14. Liu, Z., Li, P., Zheng, Y., Sun, M.: Clustering to find exemplar terms for keyphrase extraction. In: Conference on Empirical Methods in Natural Language Processing. pp. 257–266 (2009)
15. Lopez, P., Romary, L.: Humb: Automatic key term extraction from scientific articles in grobid. In: Proceedings of the 5th International Workshop on Semantic Evaluation. pp. 248–251. SemEval '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
16. Luan, Y., Ostendorf, M., Hajishirzi, H.: Scientific information extraction with semi-supervised neural tagging. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017. pp. 2641–2651 (2017)

17. Ma, X., Hovy, E.H.: End-to-end sequence labeling via bi-directional lstm-cnns-crf. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers (2016)
18. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain. pp. 404–411 (2004)
19. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: NIPS-W (2017)
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
21. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. pp. 1532–1543 (2014)
22. Samanta, S., Mehta, S.: Towards crafting text adversarial samples. arXiv preprint arXiv:1707.02812 (2017)
23. Schwenker, F.: Ensemble methods: Foundations and algorithms [book review]. IEEE Comp. Int. Mag. **8**(1), 77–79 (2013)
24. Tomek, I.: An experiment with the edited nearest-neighbor rule. IEEE Transactions on systems, Man, and Cybernetics **SMC-6**(6), 448–452 (1976)
25. Tomek, I.: Two modifications of cnn. IEEE Trans. Systems, Man and Cybernetics **SMC-6**(11), 769–772 (1976)
26. Wang, C., Li, S.: Corankbayes: Bayesian learning to rank under the co-training framework and its application in keyphrase extraction. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management. pp. 2241–2244. CIKM '11, ACM, New York, NY, USA (2011)
27. Wang, C., Li, S., Wang, W.: Experiment research on feature selection and learning method in keyphrase extraction. In: International Conference on Computer Processing of Oriental Languages. pp. 305–312. Springer (2009)
28. Wang, L., Li, S.: Pku_icl at semeval-2017 task 10: Keyphrase extraction with model ensemble and external knowledge. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017). pp. 934–937 (2017)
29. Yasunaga, M., Kasai, J., Radev, D.: Robust multilingual part-of-speech tagging via adversarial training. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). pp. 976–986. Association for Computational Linguistics (2018)