Extracting Domain Entities from Scientific Papers Leveraging Author Keywords

Jiabin Peng
School of Economics & Management
Nanjing University of Science and Technology
Nanjing, JiangSu, China
2542505085@qq.com

Jing Chen
School of Economics & Management
Nanjing University of Science and Technology
Nanjing, JiangSu, China
chenjinguuu@126.com

Guo Chen
School of Economics & Management
Nanjing University of Science and Technology
Nanjing, JiangSu, China
delphi1987@qq.com

ABSTRACT

Current methods of domain entity extraction of scientific texts rely heavily on manually annotation corpus and thus have poor generalization ability. In this paper, we proposed a two-stage methodology that can make good use of existed author keywords of the given domain to solve this problem. Firstly, the author keyword set was used to mark the boundary of candidate entities, and then their features are integrated to classify their entity type. In the experiment on artificial intelligence (AI) documents from WOS, our approach obtains an F1 value of 0.753 without manual annotation, which is slightly lower than the BERT-BiLSTM-CRF baseline model (F1=0.772) trained on manual annotation corpus, showing the usability of our approach in practice.

CCS CONCEPTS

• Computing methodologies • Artificial intelligence • Natural language processing • Information extraction

KEYWORDS

Information Extraction, Domain Named Entity Recognition, Analysis of Scientific Papers, Author Keywords

1 Introduction

At present, there have been many studies on knowledge entity extraction in scientific papers, and the biggest problem is the lack of labeled data[1]. As we all know, scientific papers usually belong to a specific domain, so manual annotation needs corresponding domain knowledge, which makes the annotation more expensive, and many popular named entity recognition (NER) models can not play their inherent excellent performance. To ensure the generalization ability of NER models, it is necessary to reduce their dependence on manual annotation. At present, thanks to the rapid development of databases and the Internet, a large number of knowledge resources have been accumulated in many domains, such as knowledge bases, gazetteers, glossaries, dictionaries, etc. These resources are widely used in NER models of distant supervision[2] or semi-supervision learning[3], which reduces the dependence of models on labeled data and improves the generalization ability of models to a certain extent.

Actually, domain entity extraction can be divided into two subtasks: entity boundary recognition and entity type classification. Taking the domain of artificial intelligence (AI) as an example, we firstly used the domain glossary to help to identify the entity boundaries and then constructed a low-cost training data to classify the entities. Problems and solutions were viewed as the *key-insights* of scientific papers[1], so we took them as the main entity types in the experiment. According to related studies, we summarized the research objectives, domains, applications, and tasks in technical papers as *problems*, and the methods, schemes, models, technologies, tools, software, algorithms, and theories used to solve these problems as *solutions*[4][5][6]. The experimental results showed that a good index of our methodology was obtained without manual annotation, and the F1-measure reached 0.753.

2 Related Studies

At present, the mainstream methods of domain NER are divided into two categories: methods based on statistical machine learning (ML) and methods based on deep learning (DL). The NER method based on ML is essentially classification, that is, given multiple types of named entities, and then models are used to classify the entities in the text. And there are two ideas in the implementation. One is to identify the boundaries of all named entities in the text firstly, and then classify them into different types, such as CoBoost[7]. The other is sequence annotation. Each word in the text is given several candidate type labels, which correspond to its position in various entities. The classical NER models based on sequence annotation in ML include HMM[8], CRF[9], etc. The NER models based on DL use pre-trained word vectors to represent words, which can solve the problem of data sparsity in high latitude vector space. Meanwhile, pre-trained word vectors contain more semantic information than manually selected features and can obtain the feature representation in unified vector space from heterogeneous texts, which brings strong development for sequence annotation tasks, especially for NER[10].

The biggest problem of domain NER is the lack of labeled corpus nowadays. When a general NER method is applied to a specific domain, corresponding adjustment strategies need to be taken according to the domain corpus. A common idea is to use transfer learning to share data and models among domains. Ni et al. projected labeled data and distributed representation of words without manual annotation in the target domain[11]. Giorgi et al. transferred the source domain model parameters to the target domain for initialization, and then fine-tune the parameters to meet the task[12]. Another idea is to make full use of the existing knowledge resources in domains to automatically build datasets and carry out distant supervision, semi-supervision, weak supervision, etc. Nooralahzadeh et al. adopted a technique of partial annotation and implemented a reinforcement learning strategy with a neural network policy in distant supervision NER[2]. Peters et al. demonstrated a general semi-supervised approach for adding pre-trained context embeddings from bidirectional language models to NLP systems and apply it to NER[3]. Lison et al. relied on a broad spectrum of labeling functions to automatically annotate texts from the target domain[13].

From the above researches, it could be seen that various domain resources were widely used to reduce the manual annotation cost as much as possible, which thus achieved good results. However, the domain NER models based on transfer learning, semi-supervision etc. still could not avoid manual participation in the construction of datasets. Therefore, after analyzing the essence of the NER task, domain NER was divided into two subtasks in this paper, which avoided manual annotation with the help of domain resources. In addition, some new ideas such as zero-shot learning[14] and learning with noisy labels[15] have also been applied to domain NER to help further reduce labor costs.

3 Methodology

3.1 Framework

Traditional NER was regarded as a sequence labeling task, which assigned the corresponding entity type and location label to each token in the text. In fact, NER could be regarded as two subtasks: boundary recognition and entity classification. That was, we can firstly identify the boundaries of named entities in the text, and then classify them into different types. The NER method based on sequence labeling treated two subtasks as a whole, in which the same labeled data was shared by two subtasks so that the requirement for its quality was pretty high. As a result, many classic NER methods could not be applied in some subdivided domains. In addition, the NER method based on sequence labeling could not be effectively integrated into the existing domain resources. At present, the common practice was to use domain terms as auxiliary data to help roughly label data. On the contrary, by dividing NER into boundary recognition and entity classification, we could make full use of the existing domain knowledge resources.

Entity boundary recognition could be regarded as a word segmentation task, which required large-scale resources (i.e. user-defined lexicon). And there usually exist some domain glossaries and a large-scale author keyword set in a given domain, which can help to solve the word segmentation task. Compared with word segmentation, entity classification required smaller-scale resources (i.e. training data). At present, many domains knowledge can be obtained easily through an online database or knowledge graph, which can provide necessary training data for entity type classification without manual annotation. To sum up, the framework of this paper was shown in Figure 1.

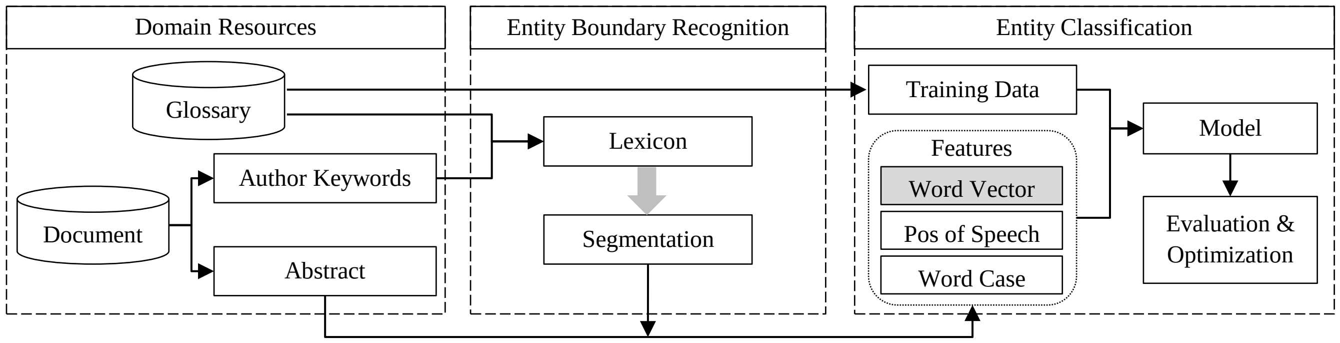

Figure 1 Framework of knowledge entities extraction.

The framework was divided into three parts. The first was the acquisition of domain resources. The domain resources used in this paper included domain glossary and domain documents. Domain glossary could be obtained directly through browser (such as Google, Firefox, etc.) or relevant domain knowledge websites (such as Wikipedia, Baidu baike, etc.). In addition, we also got the types of terms when constructing a domain glossary. Domain documents could be obtained through databases (such as WOS, CNKI, etc.), then author keywords and abstract were extracted. Author keywords were indispensable large-scale resources for entity boundary recognition, and abstracts could be used in constructing features for the training data. The second was entity boundary recognition, which was regarded as a word segmentation task. The user-defined lexicon of the word segmentation task was constructed by combining domain glossary and author keywords set and helped to realize entity boundary recognition at a low cost. The third was entity classification. The training data required for classification were extracted from the domain glossary, and the text features were obtained from abstract training or counting.

3.2 Implementation

3.2.1 Entity Boundary Recognition. As mentioned above, entity boundary recognition was transformed into word segmentation. Inspired by the Chinese automatic word segmentation method, the forward maximum matching algorithm based on string matching was used in this paper. Slightly different from Chinese word segmentation, it was necessary to extract the stem of English words before English word segmentation to avoid the influence of word form on word segmentation. In addition, there would be some noise when using word segmentation lexicon to label the candidate entities, so the following entity classification task was actually multi-class.

3.2.2 Entity Classification. Entity classification was essentially word classification, which was a typical supervised task, and the training data was indispensable. At present, it is difficult to construct a large number of high-quality classification data in a given domain. However, a small number of high-quality data could usually be obtained at a low cost with the help of domain knowledge bases or domain experts.

1) **Construct training data**. Training data consisted of positive samples and negative samples. Positive samples consisted of entities and corresponding types. And the pre-constructed glossary contained the types of terms, so we directly extracted some high-quality terms and their types as the positive samples. Negative samples, i.e. non-entities, were randomly extracted from keyword sets and texts.
2) **Construct text features**. According to the task, word vector, part of speech (POS) feature, and word case feature were constructed. Word vectors could be obtained by training large-scale domain unlabeled corpus, and semantic information was given to discrete words according to the context. POS feature was obtained by counting the corpus without word segmentation. The acquisition of the case feature was basically consistent with the POS feature, but the corpus with word segmentation was used and the cases needed to be self-defined.
3) **Model selection, training, evaluation, and optimization**. According to the task, the models we used included four classical machine learning models: Random Forest (RF), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and TextCNN, which performed well in sentence classification[16]. The detailed steps of our experiment were as follows: ① *feeding the training data to models to obtain the basic results*; ② *optimizing the word vector according to the model effect and adding features in the training process*; ③ *evaluating the effect of models to decide whether to continue optimizing*.

3.3 Feature Processing

3.3.1 Word Vector. At present, common models for training word vectors included Word2Vec, GloVe, ELMO, GPT, BERT, etc. The word vectors trained by the first two models were context-independent. Because our core task was phrase classification, which not needed context information, we chose Word2Vec to train word vectors. Before training, we used underline to concatenate the words in a phrase in the corpus after word segmentation, to ensure that phrases were regarded as a whole when training word vectors.

Word2Vec included two algorithms: Skip-gram and CBOW. Research showed that Skip-gram contained more semantic information, while CBOW contains more grammatical information[17]. The window size was also very important for training word vectors, and the commonly used window sizes were 5 and 10[1]. Therefore, we would first explore the above two factors affecting word vector results in the classification experiment. According to related studies[18], other parameters were shown in Table 1. In addition, to make word vectors more robust, we used stemmed corpus to train Word2Vec.

Parameters	Values
sg	1 / 0
window size(w)	5 / 10
min count	5
iteration number	20
embedding size	200

Table 1 Parameters of Word2Vec. sg=1 meant algorithm was Skip-gram, sg=0 was CBOW. w was used to represent the window size later.

3.3.2 POS Feature. POS of words in sentences could be obtained through the Python third-party package nltk. There were 36 kinds of POS in nltk, which meant the length of the POS vector of a single word was 36. In the classification experiment, POS vectors of training data were obtained by concatenating the POS vectors of component words. To avoid the inconsistent length of POS vectors, we counted the lengths of phrases (the number of words in phrases) in the segmentation lexicon to obtain the maximum phrase length. When the lengths of training data were less than the maximum phrase length, POS vectors of training data would be padded with 0. Finally, the length of POS vectors of training data was $36 * \max(\mathrm{len}(\mathrm{phrase}_{\{\mathrm{lexicon}\}}))$ (Maximum length of phrases in lexicon). POS vectors were used by concatenating word vectors in the experiment.

3.3.3 Case Feature. Three types of phrase cases were defined in this paper: initial uppercase, all uppercase and all lowercase. The lengths of case vectors of training data were 3. Similarly, case vectors were used by concatenating word vectors.

3.4 Classification Models

Classification models used in this paper included RF, KNN, SVM, MLP, and TextCNN. The first four models were implemented by sklearn in Python. In the RF, the number of decision trees was set to 100. All parameters of the KNN took the default values. In the SVM, the probability was set to True, that was, probability estimation was enabled. In the MLP, the number of neurons in the

[1] https://www.bbsmax.com/A/A2dm2D7zen/

hidden layer was (100, 50). TextCNN was originally used to classify sentences, and phrases could be regarded as shorter sentences. The input of TextCNN[2] were vectors generated by Word2Vec. The embedding size, sequence length, batch size, and training epoch in TextCNN were set as 200, 10, 32, and 20 respectively. Parameters not mentioned above took the default values.

4 Experiment

To verify the effectiveness of our methodology, we took the domain of AI as an example in the experiment.

4.1 Data Acquisition and Preprocessing

Firstly, we obtained the bibliography data of the AI domain. The data was from the category of AI in the core collection of WOS (Web of Science). Documents were retrieved with *WC = computer science and WC = artificial intelligence*, and the time range was set as from 1996 to 2020. Then, abstracts and keywords were extracted from the bibliography data, including 927675 abstracts and 161169 keywords.

Secondly, we constructed a glossary of AI domain. The data came from a knowledge website[3] in AI domain, from which we obtained all problem and solution entities. The problem entities were from the tasks of the *Browse State-of-the-Art* page in the website, and the solution entities were from the machine learning components of the *Methods* page. After removing duplications, 1887 problem entities and 1209 solution entities were remained.

Finally, we processed the above data to get the final experimental data. The user-defined lexicon of English word segmentation was constructed by merging keywords set and domain glossary. The training data of classifiers consisted of entities and non-entities, in which 360 entities of each type were manually extracted from the glossary, and 360 non-entities were manually constructed. Non-entities included phrases and words, in which phrases were extracted from high-frequency keywords and words were constructed randomly. The ratio of phrases to words in non-entities was about 2:1. This was because almost all entities were phrases, so more phrase-level non-entities were needed to help train models. Finally, 1080 pieces of classification data were obtained. The training set and validation set were randomly divided according to the ratio of 5:1. In addition, to evaluate the performance of our methodology, we set our baseline as the traditional BERT-BiLSTM-CRF NER model. A previous annotated corpus containing 3000 sentences was used for the baseline model, in which 2000 sentence were randomly selected as the training data and the rest 1000 sentences were used as the common test set.

4.2 Result Analysis

The macro average of precision, recall, and F1-measure were used to evaluate the models.

Word vectors were the basic input of models, so we firstly explored the influences of word vectors trained by the two algorithms in different window sizes, and the results were shown in Table 2.

	sg=1		sg=0	
	w=5	w=10	w=5	w=10
RF	0.672	**0.681**	0.666	0.655
KNN	0.55	**0.672**	0.151	0.146
SVM	**0.736**	0.701	0.709	0.679
MLP	0.672	**0.685**	0.428	0.478
TextCNN	**0.695**	0.685	0.67	0.67

Table 2 Macro F1-measure of models using different word vectors on the test set.

Firstly, we compared the results in Table 2 vertically. When sg=1, five models in two window sizes achieved good results on the whole. Only when w=5, KNN performed poorly. However, when sg=0, the performances of all models decreased, especially KNN and MLP. The possible reason was that Skip-gram focused on semantics, which was more conducive to the NER task than CBOW. In addition, KNN and MLP had higher requirements for data quality, but word vectors trained by CBOW could not meet the requirement. Secondly, we compared the results in Table 2 horizontally, and the best F1-measures of each model were about 0.7, which were highlighted in bold in Table 2. In the following experiments, the word vector that made each model achieved the best performance was used, and POS features and case features were added to models. The results were shown in Table 3.

It could be found in Table 3 that the addition of two features effectively improved the F1-measures. When all features were fused, optimal results were obtained in all models. Among the five models, SVM had the best performance, with an F1-measure of 0.753. This might be because the underlying training mechanism of SVM made SVM more suitable for small sample classification. When the word vector parameters were sg=1 and w=10, the voting model had the best performance, and the F1-measure was 0.752. SVM or voting model could be selectively used in practical application. The best F1-measure of TextCNN was 0.715, which was far from its performance in sentence classification. One possible reason was that the length of phrases was much smaller than sentences.

The baseline BERT-BiLSTM-CRF[4] performed well on the domain NER task. Its F1-measure was 0.772, which was far less than its performance in the general NER task, but it had been a very good result in the subdivided domain. And the result was 0.019 higher than our optimal model. From the experimental result, there was still a gap in our methodology, but from the cost of experimental data, the gap was acceptable. In the following work, we can further optimize the word vectors and add more features to improve the performance.

[2] https://github.com/cjymz886/text-cnn
[3] https://paperswithcode.com/
[4] https://github.com/macanv/BERT-BiLSTM-CRF-NER

5 Conclusion

Aiming at the problem that the current domain NER models heavily rely on manually annotation data and thus has poor domain generalization ability, we propose a two-stage knowledge entity extraction methodology, which can get rid of the dependence on manually annotation data. Experiments in WOS documents in the domain of AI showed that good results can be achieved in the extraction of problem and solution entities without manual annotation using our approach.

In general, our approach has good domain generalization because it does not need manual annotation, and can be applied to many subdivided domains at a low cost. However, the performance of our scheme still has some room for improvement. In the follow-up work, we can try to use better word vectors and more features to improve the accuracy of entity extraction, and gradually extend the model to the extraction of more knowledge types.

	f1			f1+f2			f1+f3			f1+f2+f3		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
RF	0.588	0.812	0.681	0.629	0.824	0.713	0.618	0.812	0.702	0.661	0.833	**0.736**
KNN	0.593	0.78	0.672	0.603	0.773	0.676	0.604	0.784	0.681	0.614	0.783	**0.687**
SVM	0.677	0.81	0.736	0.701	0.809	0.749	0.69	0.812	0.744	0.706	0.812	**0.753**
MLP	0.593	0.813	0.685	0.604	0.815	0.694	0.605	0.814	0.694	0.621	0.826	**0.709**
TextCNN	0.63	0.785	0.695	0.631	0.815	0.701	0.64	0.765	0.697	0.65	0.81	**0.715**
Voting	-	-	-	-	-	-	-	-	-	0.689	0.831	0.752
BERT-BiLSTM-CRF(baseline)					P: 0.756			R: 0.789			F1:0.772	

Table 3 Macro P, R, F1-measure of models using different features on the test set. f1 was word vector, f2 was POS feature, f3 was case feature.

ACKNOWLEDGMENTS

This study is supported by the MOE (Ministry of Education in China) Project of Humanities and Social Sciences.

REFERENCES

[1] Zara Nasar, Syed Waqar Jaffry and Muhammad Kamran Malik, 2018. Information extraction from scientific articles: a survey. Scientometrics 117 3(2018), 1931-1990. DOI: https://doi.org/10.1007/s11192-018-2921-5.

[2] Nooralahzadeh Farhad, Lønning Tore Jan and Øvrelid Lilja, 2019. Reinforcement-based denoising of distantly supervised NER with partial annotation. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP*. 225-233.

[3] Peters E. Matthew, Ammar Waleed and Bhagavatula Chandra, et al., 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1756–1765.

[4] Gupta Sonal and Manning D, 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*. 1-9.

[5] Singh Mayank, Dan Soham, Agarwal Sanyam, Goyal Pawan and Mukherjee Animesh, 2017. AppTechMiner: Mining Applications and Techniques from Scientific Articles. In *Proceedings of the Joint Conference on Digital Libraries Joint Conference on Digital Libraries*. 1-8.

[6] Heffernan Kevin and Teufel Simone, 2018. Identifying Problems and Solutions in Scientific Text. Scientometrics 116 2 (2018), 1367–1382. DOI: https://doi.org/10.1007/s11192-018-2718-6.

[7] Michael Collins and Yoram Singer, 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. 100-110.

[8] Zhou Guodong and Su Jian, 2002. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Stroudsburg: Association for Computational Linguistics, 473-480.

[9] McCallum Andrew and Li Wei, 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*. Stroudsburg: Association for Computational Linguistics, 188-191.

[10] Cherry Colin and Guo Hongyu, 2015. The unreasonable effectiveness of word representations for Twitter named entity recognition. In *The 2015 Annual Conference of the North American Chapter of the ACL*. Stroudsburg: Association for Computational Linguistics, 735-745.

[11] Ni Jian, Dinu Georgiana and Florian Radu, 2017. Weakly Supervised Cross-Lingual Named Entity Recognition via Effective Annotation and Representation Projection. In *Proceedings of the 54th Annual Meeting on Association for Computational Linguistics*. 1470-1480.

[12] Giorgi M John and Bader D Gary, 2018. Transfer learning for biomedical named entity recognition with neural networks. Bioinformatics 34 23 (2018), 4087-4094.

[13] Lison Pierre, Barnes Jeremy, Hubin Aliaksandr and Touileb Samia. 2020. Named Entity Recognition without Labelled Data: A Weak Supervision Approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 1518-1533.

[14] Dai, Damai, et al. "Inductively Representing Out-of-Knowledge-Graph Entities by Optimal Estimation Under Translational Assumptions." (2020).

[15] Ifeoluwa David Adelani, A. Michael Hedderich, Dawei Zhu, den Esther van Berg and Dietrich Klakow, 2020. Distant Supervision and Noisy Label Learning for Low Resource Named Entity Recognition: A Study on Hausa and Yor\`ub\'a. *arXiv preprint arXiv: 2003.08370* (2020).

[16] Kim Y . Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.

[17] Mikolov Tomas, Chen Kai, Corrado Greg and Dean Jeffrey, 2013. Efficient Estimation of Word Representations in Vector Space. Computer Science. *arXiv preprint arXiv:1301.3781v3* (2013).

[18] Siwei Lai, Kang Liu, Liheng Xu and Jun Zhao, 2016. How to Generate a Good Word Embedding. IEEE Intelligent Systems 31 6 (2016), 5–14.