Presenting Problems and Methods under Weak Supervision

Hongyu Sun, Jing Zhang, Cuiping Li, and Hong Chen

Key Laboratory of DEKE (MOE), School of Information,
Renmin University of China, Beijing, 100872, China
{sunhongyu,zhang-jing,licuiping,chong}@ruc.edu.cn

Abstract. The development of computer science is so incredible that a large number of new papers are published every few weeks. However, researchers have limited time and energy, so finding interested papers and filtering out uninterested ones matters. The motivation of our work is that we want to quickly present the problem it studies and the method it uses automatically by a model, if any, from the title of a given paper, without reading full texts. Existing solutions focus on phrase mining, such as SegPhrase, AutoPhrase [30] [1]. They extract important phrases from full texts that describe the problem and the method. However, the quality of extracted phrases is difficulty to guarantee because the process of extraction are unsupervised. We observe that some specific words in the titles can reflect the relation between the problem and the method accurately from massive text corpus. Here we call these specific words pattern words, in the following sections we will explain what pattern word is in detail. However, known pattern words like **for**,**using**,**by** are intuitive and the amount of such words is too small to be generalized. On the other hand, human language is flexible and powerful. So far researchers have delivered countless publications. Another observation is that we believe there exists many pattern words working similarly with like **for**,**using**,**by** have not been thought of directly. Based on these observations, we put forward our framework to present the problem a given paper studies and the method it uses automatically. Moreover, the whole process can be conducted under weak supervision and save more human resources compared with other models. The pipeline of our framework consists of four parts: find new pattern words iteratively, pattern-based phrase extraction, generating large amounts of data with labels and presenting results by a supervised model. Finally extensive experiments on three public datasets show the effectiveness of our framework, around 5% improvements compared with rule-based one. The source code is available at https://github.com/auniquesun/project.

Keywords: weak supervision · bootstrapping iteration · pattern-based phrase extraction · sequence labeling.

1 Introduction

We want to present the problem and the method of a given paper by a model automatically, without reading full texts or abstract. Existing solutions focus on

keywords and phrase mining[30] [1]. Keywords can describe a paper by several important phrases and phrase mining returns a list of phrases. These solutions suffer from lacking labels and don't meet our requirements directly while researchers still need to figure out which phrases describe the problem and which phrases relate to the method. So the key point is how to get labels? Once got labeled data, we can train various kinds of supervised model, let the model learn from labeled data and use the pre-trained model to make predictions. On the one hand, it is expensive for human being to label large amounts of data to achieve good performance. On the other hand, we can't go forward without information from human supervision. We find that some specific words in the titles contain useful semantic information that indicates the relationship between the problem and the method. For example, in Figure 1, **for** is a pattern word since we can infer that the pattern is the method **for** the problem according to its semantic information. Similarly, **using** and **by** are also pattern words as they indicate the problem **using** or **by** the method. But the amount of intuitive pattern words is far from enough. Nevertheless, countless literature has been accumulated by various kinds of academic communities and human language is so powerful and expressive, we believe that many pattern words really exist and have not been thought of directly. In this case, we develop a framework which finds new pattern words by bootstrapping iteration and generates data with labels under weak supervision. Then train a high-performance sequence labeling model to present the problem and the method [28] [23] [21] [26] [27] [25] [24] [22] [19] [20] [18]. The contributions of our work are as following:

1. We put forward a framework that can generate large reliable labeled data under weak supervision and save human labeling efforts by a large margin.
2. Based on the generated labeled data, we train a sequence labeling model. Extensive experiments on public datasets shows the performance of our model improves 5% compared with rule-based one.
3. We present the problem and the method of a given paper effectively and make the project public at https://github.com/auniquesun/project.

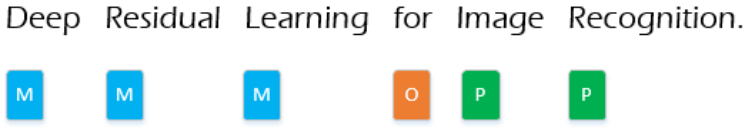

Fig. 1. Pattern word example. In this title, **for** is recognized as a pattern word and the pattern is "a method **for** a problem". Hence, we summarize deep residual learning is the main method the paper puts forward and image recognition is the problem it studies.

2 Framework

2.1 Problem Definition

The task is defined in the following form: Given a paper with title, the title is denoted as T, which contains a sequence of words $W = \{w_1, w_2, ..., w_n\}$, for each word w_i, we want to know its label $l_i \in \{P, M, O\}$ by a prediction model automatically, finally we get a sequence of labels $L = \{l_1, l_2, ..., l_n\}$. Here tag P stands for the problem, tag M stands fort the method and tag O stands for neither the problem nor the method. So we present readers or researchers the the problem and the the method of a paper in this way.

2.2 Bootstrapping Iteration

It is clear that the supervised prediction model is more powerful. On the opposite side, we lack labeled data to train a supervised model. As mentioned in the introduction, we put forward a weak supervised method called bootstrapping iteration that is one of our contributions. Bootstrapping iteration aims to find more new pattern words according to some known ones and their contexts, especially under weak supervision settings, that means we don't need to spend so many efforts on manual labeling. The detailed implementations of how bootstrapping iteration works are explained in **Algorithm 1**. The original idea is that we believe there exists a number of pattern words while only a small part of them are known to us, and we want to find the remaining. At beginning, some known pattern words are provided, such as **for**, **using**, **by**. Next we can extract their contexts, usually expressed in the form of phrases. . Now that known pattern words are surrounded by these valid phrases, it is probably some new pattern words are also surrounded by these phrases. Then we extract context words of these valid phrases from a new set of titles, treat newly extracted context words as candidate pattern words and count the occurrences of each word. If its occurrences are greater than or equal to our pre-defined threshold, at the same time, it meets the requirements of semantic information, then we append this context word into the set of new pattern words. After each iteration, the set of new found pattern words will be merged with the set of known pattern words, for the purpose of expanding the second set and making it different in the next iteration. Consequently, the process bootstraps the set of known pattern words iteration by iteration, so we call it bootstrapping iteration. However, how to ensure the new found pattern words what we really need? At first, as you can see in the **Algorithm 1**, we have candidates and the threshold settings, and this measure filters out most of words which have less occurrences. Secondly, although the remaining words have frequent occurrences, we are not 100% confident that they are really what we need. It is necessary to require a bit of human efforts of checking. It is a bit of human checking that shows one of manifestations of our weak supervision settings. This operation is feasible because there are only dozens of words remained after each iterations.

Algorithm 1 Bootstrapping Iteration

Input: window size s; number of iterations $iters$; occurrences threshold of pattern words h; groups of paper titles $G = \{G_1, G_2, ..., G_m\}$, each group $G_i = \{t_{i1}, t_{i2}, ..., t_{il}\}$; a set of known pattern words $KPW = \{kpw_1, kpw_2, ..., kpw_n\}$

Output: a set of new pattern words $NPW = \{npw_1, npw_2, ..., npw_k\}$

procedure FindNewPatternWords($s, iters, h, G, KPW$)
  Int: $iter \leftarrow 0$
  Int: $m \leftarrow G.size()$ ▷ get the size of groups G
  Pattern Words Set: $NPW \leftarrow ()$
  while $iter < iters$ **do**
    Phrases Set: $V \leftarrow ()$
    for kpw_i in KPW **do**
      for t_j in G_{iter} **do**
        if kpw_i is in t_j **then**
          $phrases$ = **ExtractContext**($kpw_i, t_j, s, 0$)
          $V \leftarrow$ append $phrases$
    Candidates Dict: $C \leftarrow \{\}$ ▷ initialize pattern words candidates Dict
    for v in V **do**
      for t_k in $G_{iter+\frac{m}{2}}$ **do**
        if v is in t_k **then**
          $words$ = **ExtractContext**($v, t_k, s, 1$)
          for $word$ is in $words$ **do**
            if $word \in C.keys()$ **then** ▷ a counter, $\langle word, cnt \rangle$ pair
              $C[word] \leftarrow C[word] + 1$
            else
              $C[word] \leftarrow 1$
    for $word$ in $C.keys()$ **do**
      if $C[word] \geq h$ **then** ▷ $word$ occurrences need $\geq$ threshold
        $NPW \leftarrow$ append $word$
    $KPW \leftarrow Merge(KPW, NPW)$ ▷ add NPW into KPW
    $iter \leftarrow iter + 1$
  $NPW \leftarrow KPW$
  return NPW
procedure ExtractContext($v, t, s, flag$)
  $idx \leftarrow t.find(v)$
  if $flag = 0$ **then** ▷ extract phrases surrounding v
    Phrases List: $phrases \leftarrow []$
    $phrases \leftarrow$ append $t.substr(idx - s, idx)$
    $phrases \leftarrow$ append $t.substr(idx, idx + s)$
    return $phrases$
  else ▷ extract words surrounding v
    Words List: $words \leftarrow []$
    Int: $i \leftarrow idx$ - s
    while $i \leq idx + s$ **do**
      if $i \neq idx$ **then** ▷ skip the idx(th) word in t
        $words \leftarrow$ append $t[i]$
      $i \leftarrow i + 1$
    return $words$

2.3 Pattern-based Phrase Extraction

Phrase extraction or phrase mining is a active research topic in data mining and natural language processing [30] [29] [31] [32]. It aims at extracting phrases from massive text corpus for both specific domains and general ones. In most cases, the procedure of phrase mining is conducted under unsupervised or semi-supervised settings. Then what is pattern-based phrase extraction? The difference between pattern-based phrase extraction and other phrase extraction methods is that the former has clear rules, it only extract phrases related to some specific patterns and skip all the others. This process is specially designed for the set of new pattern words found by bootstrapping iteration. In this section, I will describe this process in detail and give the **Algorithm 2**. For each title in the corpus, if it contains a pattern word, then we split the title into two parts based on the pattern word. After that, the chunk operation is applied to each part. The results from chunk operations are usually phrases [8]. So the quality of new pattern words is significant to our pattern-based phrase extraction.

Algorithm 2 Pattern-based Phrase Extraction

Input: a set of New Pattern Words NPW returned by **Algorithm 1**; a corpus of titles $T = \{t_1, t_2, ..., t_n\}$

Output: a list of phrases P, a 2-dimensional list ID (the first dimension is the size of and the second dimension is the number of words in each title)

procedure PHRASEEXTRACTION(NPW, T)
    empty list: $P \leftarrow []$
    empty list: $ID \leftarrow []$
    for t_i in T **do**
        list: $li \leftarrow []$
        bool: $flag \leftarrow$ true ▷ by default, t_i doesn't contain any pattern word
        for kpw_i in NPW **do**
            if kpw_i in t_i **then**
                $parts = t_i.split(kpw_i)$
                for $part$ in $parts$ **do**
                    $phrase \leftarrow$ Chunk($part$) ▷ chunk operation to get phrase
                    $P \leftarrow$ append $phrase$
                $flag \leftarrow$ false
                break
        if $flag =$ true **then**
            for w_j in t_i **do**
                $li[j] \leftarrow 0$
        else ▷ in fact, find pattern word in t_i
            for w_j in t_i **do**
                $id =$ **Decide**(w_j)
                $li[j] \leftarrow id$
    return P, ID

2.4 Generating Data with Labels

In this paper we expect to present the problem and the method by the means of giving a tag to each word in the title. It is a problem of sequence tagging. Words in the sequence are related rather than independent. And supervised models, such as CRF, LSTM, Bert are good at capturing the dependencies within contexts [11] [12] [19] [21] [16]. Where does the labeled data come from? Based on the bootstrapping iteration and pattern-based phrase extraction, we can answer this question directly. As **Algorithm 2** shows, the 2-dimensional list ID contains information of labels, the first dimension is equal to the number of titles, and the second dimension is the number of words in the corresponding title. For an element $id[i][j]$ in the ID, it has three possible values 0,1,2. Concretely, value 2 corresponds to label M(ethod), value 1 corresponds to label P(roblem) and value 0 corresponds to label O(ther, neither problem nor method). There is no doubt that these labeled data contains a certain amount of noisy labels, after all, it was generated under weak supervision. In the section of experiments, we will show the confidence score of the labels prediction on Groundtruth dataset.

Algorithm 3 Generating Data with Labels

Input: a corpus of titles $T = \{t_1, t_2, ..., t_n\}$; their ID returned by Algorithm 2
Output: a set of lists of labels $L = \{l_1, l_2, ..., l_m\}$

procedure GetLabels(T, ID)
  $L \leftarrow []$
  for id, t_i in zip(ID, T) **do** ▷ ID and T have same dimension
    $l_i \leftarrow []$
    for w_j in t_i **do**
      if $id[i][j] = 1$ **then** ▷ convert id to *char*, *samebelow*
        $li \leftarrow$ append $'P'$
      else if $id[i][j] = 2$ **then**
        $li \leftarrow$ append $'M'$
      else
        $li \leftarrow$ append $'O'$
    $L \leftarrow$ append l_i ▷ finishing tagging a whole title
  return L

2.5 Sequence Labeling

Finally, we want to present the problem and the method by means of sequence labeling. For an input sentence, the sequence labeling model predicts a tag for each word in the sentence. Now we have got training data by running **Algorithm 3**. So the performance of sequence labeling model depends on the architecture itself. Classic machine methods like hidden markov model [9], conditional random fields are designed for sequence tagging [10]. But they were surpassed by later deep learning models, such as LSTM, GRU, BERT [16] etc. Another

common way is to combine machine learning methods and deep learning ones, such as LSTM+CRF, BiLSTM+CRF, usually their architecture consists of an encoder and a decoder. LSTM plays the role of encoder which can extract features from context and CRF plays the role of decoder responsible for prediction. Recently, Google released Bert which is a more powerful pre-trained model for extracting long distance dependencies relationship within contexts [16]. Naturally, Bert+CRF can be used for sequence labeling and other models based on Bert also emerged. In the following section of we will give comparisons of all sequence labeling models here. Regardless of what models we use, the optimization object is identical, we aims to minimize the loss function below:

$$Loss = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} I_i \left(y_j \neq \hat{y_j}\right)}{\sum_{i=1}^{n} \sum_{j=1}^{m_i} (I = 1)} \tag{1}$$

Equally maximizing the accuracy of prediction,

$$Acc = 1 - Loss \tag{2}$$

Here **n** is the number of training examples, each example stands for a paper title, m_i is the number of words in i(th) example, y_j is the label of j(th) word in i(th) example and $\hat{y_j}$ is the corresponding prediction given by our model, I_i equals 0 when $y_j = \hat{y_j}$, otherwise 1.

In this section, the details of our framework are separated into four parts, let we put them together. Figure 2 presents the flowchart of the framework we put forward. At first, we develop an algorithm to find more new pattern words in step 1 from massive text corpus in the settings of weak supervision. Step 2 involves in pattern-based phrase extraction. Then we generate data with labels in step 3 according to the semantic information indicated by pattern words and phrases extracted from previous step. At last, it turns to step 4, sequence labeling models are trained on labeled data to present the problem and the method by a prediction sequence of tags.

3 Experiments

3.1 Datasets

We choose our experiments datasets from two sources. The first one is a widely used open academic graph (OAG) [14], which unifies two billion-scale academic graphs: Microsoft Academic Graph (MAG) and AMiner [35] [36]. Both of the two graphs collects thousands of millions of academic papers and the datesets are public [15] [13]. The second source is a popular GitHub repository whose full name is Paper with Code (PWC) [1] [34]. Different from OAG, dataset from PWC is not well organized. We collect thousands of titles and format them later on. The subset of Aminer used contains more than 3 million papers, and the subset

[1] https://github.com/zziz/pwc

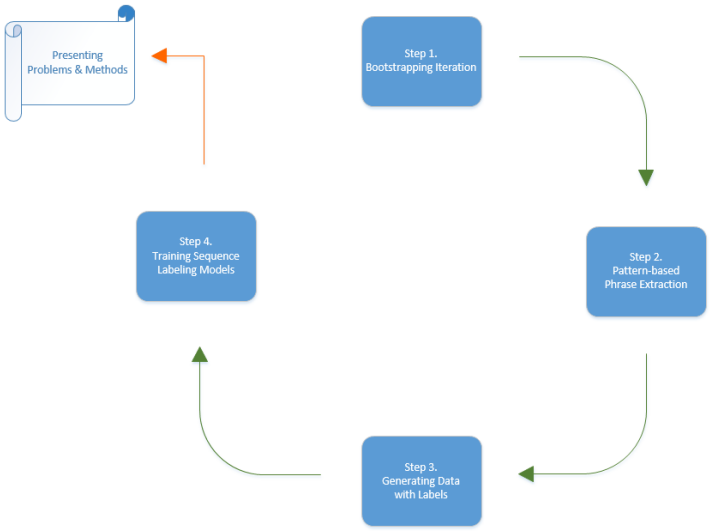

Fig. 2. Flowchart of Our Framework.

from MAG includes more than 2 million papers. For each paper, it was stored as a JSON object which consists of several key-value pairs such as title, authors, venue information, abstract, references, number of citations and so on. Before training, we process these three datasets into the format that is consistent with [33].

3.2 Models Comparison

In this section, there are various ways to conduct sequence labeling to present the problem and the method, here we select five typical models to compare. The evaluation metric is accuracy that equals to $1 - loss$ according to Eq. 2

Rule-based Labeling The rule-based method find the pattern word in a title at first, then split the title into two parts based on the pattern word, and chunk every part to get phrases, finally tag each word one of labels P, M, O. Checking the results in Table 3.2, we conclude that for one thing this method has higher precision, recall and f1 except accuracy. For another, rule-based method has poorer generalization because of strong rules, resulting in relatively low accuracy.

BiLSTM+CRF Once BiLSTM+CRF was the standard architecture for sequence labeling and achieved state-of -the-art in many cases [22] [18]. Felix et al. put forward LSTM [37] for solving the problem of gradient vanishing in the process of learning long distance dependencies from contexts, and bidirectional LSTM combined contextual information for better performance. The inputs of LSTM cells are concatenations of word embedding and its chars embedding, word embedding uses pre-trained glove model, the Conditional Random Field (CRF)

layer can be treated as a decoder which outputs predicting label of each word in the titles. The BiLSTM+CRF not only learn rules from training data but also generalizes rules, it improves the accuracy clearly compared with rule-based one. We also conducted experiments using BiLSTM model without decoder layer CRF. In this case, the BiLSTM model without decoder layer CRF has comparable accuracy to BiLSTM+CRF, although it is far from BiLSTM+CRF in terms of precision, recall and f1 score.

BERT+CRF Recently the BERT won new state-of-the-art results on eleven natural language processing tasks [16], including sequence labeling. Based on the architecture of transformer, BERT takes the advantage of multi-head attention and adopts the masked language model, which captures the information more accurately and efficiently. Table 3.2 proves BERT outperforms previous models by large margin. The origin implementations of NER model in BERT feed final hidden representations into a classifier layer over NER label set and did not use CRF. In our experiments, we implement the BERT model with CRF and compare them to figure out their differences. They have comparable performances.

Table 3.2 shows results of different models on the groundtruth, in this case BERT-related models still get good performances and go far beyond others while this advantage is relative small on AMiner and MAG. At the same time, a lesson learned from Table 3.2 and 3.2 is that if we present the problem and the method totally depending on some specific pattern words(or rules), the results lack the ability of generalization and are unsatisfactory. It is the necessity that we generate data with labels to train powerful supervised model.

Table 1. Experiments Results on Dataset AMiner.

Models	**AMiner**							
	Dev				Test			
	precision	recall	f1	acc	precision	recall	f1	acc
Rule-based	0.8882	0.8910	0.8896	0.9141	0.8942	0.8905	0.8923	0.9167
BiLSTM	0.7027	0.7651	0.7326	0.9394	0.6981	0.7571	0.7264	0.9384
BiLSTM+CRF	0.8255	0.8017	0.8134	0.9408	0.8224	0.7972	0.8096	0.9412
Bert	0.7907	0.7926	0.7911	**0.9670**	0.7920	0.7925	0.7917	**0.9676**
Bert+CRF	0.7849	0.7979	0.7911	0.9663	0.7859	0.7984	0.7919	0.9670

3.3 Parameter Setting

The rule-based method predicts labels guided by specific patterns, so it has nothing to do with parameters and we do not include it in this section. Instead, BiLSTM and BERT models really take the advantage of powerful neural network architecture and training data. The input vectors of BiLSTM model consist of two parts: word embedding and chars embedding. Word embedding use pre-trained glove model, whose dimension is 300. Chars embedding take the last output of LSTM cells of all chars in a word, then concatenate it with word

Table 2. Experiments Results on Dataset MAG.

Models	MAG							
	Dev				Test			
	precision	recall	f1	acc	precision	recall	f1	acc
Rule-based	0.8813	0.8646	0.8729	0.9301	0.8792	0.8617	0.8703	0.9289
BiLSTM	0.8608	0.8481	0.8544	0.9746	0.8611	0.8493	0.8552	0.9745
BiLSTM+CRF	0.8947	0.8545	0.8742	0.9753	0.8967	0.8567	0.8763	0.9755
Bert	0.8012	0.7976	0.7994	**0.9756**	0.8012	0.7978	0.7995	**0.9756**
Bert+CRF	0.7952	0.7994	0.7973	0.9737	0.7953	0.7993	0.7973	0.9737

Models	PWC			
	precision	recall	f1	acc
Rule-based	0.9307	0.6405	0.7586	0.6764
BiLSTM	0.9419	0.7684	0.8463	0.7782
BiLSTM+CRF	0.9443	0.7626	0.8453	0.7745
BERT	0.7296	0.7318	0.7222	**0.8817**
BERT+CRF	0.7288	0.7308	0.7211	0.8802

Table 3. Experiments Results on PWC. Obviously, even the highest accuracy of these models is lower than the same metric on AMiner and MAG. Since Algorithm 1, 2 and 3 generate data with labels under weak supervision resulting in noisy labels. Further, noisy labels have not so much influence on accuracy because the highest score achieves around 0.88 which is relative satisfactory in the circumstance of little supervision information provided by human.

embedding again. The dimension of char embedding is 100. We use adam as optimizer [38], the learning rate is 0.001 and the batch size is 64.
For models related to BERT, we almost keep all parameters default [16] except for max size of a sentence, batch size and learning rate. We set max length of a sequence to 22 while the origin value is 512. As we know, it is impossible for a paper title to have more than hundreds of words. The batch size is 64 and learning rate is 2e-5. All models in our experiments are trained on two Nvidia GeForce RTX 2080 Ti graphics cards. Further details are available at our project https://github.com/auniquesun/project.

4 Related Work

Bootstrapping iteration is a very important part of our work. It is bootstrapping iteration that finds new pattern words. Actually, many efforts have been taken into bootstrapped entity extraction system [4–6] by previous work in the setting of lacking training data. Bootstrapping iteration depends on two score functions that are pattern score function and entity score function. Riloff et al. put forward that pattern score can be measured using the ratio of the positive entities among all entities extracted by this pattern and entity score can be measured by the number and quality of its matched patterns [7]. Gupta et al. have similar ideas[5].

Recently, Shen et al. developed the framework SetExpan which expand entities set and contexts set via bootstrapping each other [3]. Qu et al. extract relation between entity pairs in the weakly supervised setting by optimizing a co-training model [2].

Our work is also related to sequence labeling. When it comes to sequence labeling, named entity recognition, chunking and part of speech tagging are typical cases. Statistical models are used to solve sequence labeling problems in earlier days, including Hidden Markov Models, Maximum entropy Markov Models [9] and Conditional Random Fields [10]. With the development of deep learning, convolutional neural network [11] based models are applied to tackle sequence labeling problem. Huang et al. firstly apply a bidirectional LSTM CRF model to the same problem. Lample et al. got better performance in NER task using similar neural architectures without the assistance of hand-crafted features and domain-specific features [12]. Recently, transformer based models achieve he state of the art in task of sequence labeling, such BERT and XLNet [16] [17].

5 Conclusion

Nowadays thousands of new papers are delivered everyday in scientific communities. But researchers have limited time and energy, an efficient and effective toolkit is required to figure out the problem it studies and the method it uses according to a given paper to save their time. We put forward a framework to present the problem and method of a specific paper. Under the framework, one of most challenging tasks solved by us is that we want to engage in the powerful deep learning models but shortage of labeled data. The Algorithm 1, 2, 3 bridge this gap under weak supervision, it means we get reliable labels by means of the algorithms rather than expensive human labeling. Similar results of experiments on three public datasets show that our framework is adaptable and deep learning models go beyond the rule-based method by a large margin. In turn, it confirms the value of labeled data provided by our algorithms. We make this project open for reproducing the experiments and pushing the relative research forward.

References

1. Jingbo S., Jialu L., Meng J., Xiang R., Clare R V., Jiawei H.: Automated Phrase Mining from Massive Text Corpora, accepted by IEEE Transactions on Knowledge and Data Engineering, Feb. 2018.
2. Meng, Qu.: Overcoming Limited Supervision in Relation Extraction: A Pattern-enhanced Distributional Representation Approach. CoRR (2017)
3. Jiaming, S., Zeqiu, W., Dongming, L.: SetExpan: Corpus-Based Set Expansion via Context Feature Selection and Rank Ensemble. 288-304 (2017)
4. Etzioni, O.: Unsupervised named-entity extraction from the Web: An experimental study. Artificial Intelligence. **165**(1), 91-134 (2005)
5. Gupta, S.: Induced lexico-syntactic patterns improve information extraction from online medical forums. JAMIA **21** (2014)

6. Gupta, S., Manning, C.: Improved Pattern Learning for Bootstrapped Entity Extraction. In: Eighteenth Conference on Computational Natural Language Learning, pp. 98–108. Association for Computational Linguistics (2014)
7. Riloff, E.: Automatically generating extraction patterns from untagged text. AAAI **2** (1996)
8. Akbik, A., Blythe, D., Vollgraf, R.:Contextual String Embeddings for Sequence Labeling. COLING (2018)
9. McCallum, A., Freitag, D.: Maximum entropy Markov models for information extraction and segmentation. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 591-598. Morgan Kaufmann Publishers Inc (2000)
10. Lafferty, J., McCallum A.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 282-289. Morgan Kaufmann Publishers Inc (2001)
11. Collobert, R.: Natural Language Processing (Almost) from Scratch. JMLR, 2493–2537 (2011)
12. Lample G.: Neural Architectures for Named Entity Recognition. CoRR, 260–270 (2016)
13. AMiner Homepage. https://www.aminer.cn. Last accessed 5 Apr 2019
14. Open Academic Grpah. https://www.openacademic.ai/oag/. Last accessed 8 Oct 2019
15. Microsoft Academic. https://academic.microsoft.com/home. Last accessed 8 Oct 2019
16. Devlin, J., Chang, M., Lee, K., Toutanova, K.: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR (2018)
17. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.: XLNet: Generalized Autoregressive Pretraining for Language Understanding. CoRR (2019)
18. Manning, C.: Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics? CICLing, 171–189 (2011)
19. Ratnaparkhi, A.: A Maximum Entropy Model for Part-Of-Speech Tagging. EMNLP (1996)
20. Jurafsky, D., Martin, J.H.: WORD CLASSES AND PART-OF-SPEECH TAGGING (2005)
21. Nadeau, D.: A survey of named entity recognition and classification. Linguisticae Investigationes, 3–26 (2007)
22. Yadav, V., Bethard, S.: A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 2145-2158. Association for Computational Linguistics, USA (2018)
23. Nadeau, D., Turney, P.D., Matwin S.: Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In: Advances in Artificial Intelligence, pp. 266–277. Springer, Berlin Heidelberg (2006)
24. Eftimov, T.: A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. Plos One **12**(6), (2017)
25. Marrero, M.: Named Entity Recognition: Fallacies, challenges and opportunities. Computer Standards & Interfaces **35**, 482–489 (2013)
26. Ratinov, L., Roth, D.: Design Challenges and Misconceptions in Named Entity Recognition. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning, pp. 147–155. Association for Computational Linguistics, Boulder, Colorado (2009)

27. Chiticariu, L., Krishnamurthy, R., Li Y.: Domain Adaptation of Rule-based Annotators for Named-entity Recognition Tasks. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 1002–1012. ACM, ambridge, Massachusetts (2010)
28. Evans, R.: A framework for named entity recognition in the open domain. In: In Proceedings of the Recent Advances in Natural Language Processing, pp. 137–144. (2003)
29. Shang J.: Automated Phrase Mining from Massive Text Corpora. CoRR (2017)
30. Liu, J., Shang, J., Wang, C.: Mining Quality Phrases from Massive Text Corpora. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1729–1744. ACM, Melbourne, Victoria, Australia (2015)
31. Bedathur, S.: Interesting-phrase Mining for Ad-hoc Text Analytics. PVLDB **3**(1-2), 1348–1357 (2010)
32. Krishnan, A., Sankar, A., Zhi, S.: Unsupervised Concept Categorization and Extraction from Scientific Document Titles, 1339–1348. ACM, Singapore, Singapore (2017)
33. Tjong, K.S., Erik, F., De Meulder, F.: Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. In: Proceedings of the Seventh Conference on Natural Language Learning, pp. 142–147. Association for Computational Linguistics, Edmonton, Canada (2003)
34. public GitHub repository. Paper with Code, https://github.com/zziz/pwc. Last accessed 12 Oct. 2019
35. Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'2008). pp.990-998.
36. Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion). ACM, New York, NY, USA, 243-246.
37. Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM. (1999).
38. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. CoRR (2014)