A Domain Independent Approach for Extracting Terms from Research Papers

Birong Jiang[1]**, Endong Xun[1], and Jianzhong Qi[2]

[1] Beijing Language and Culture University, China,
{jiangbirong, edxun}@blcu.edu.cn
[2] University of Melbourne, Australia
jianzhong.qi@unimelb.edu.au

Abstract. We study the problem of extracting terms from research papers, which is an important step towards building knowledge graphs in research domain. Existing terminology extraction approaches are mostly domain dependent. They use domain specific linguistic rules, supervised machine learning techniques or a combination of the two to extract the terms. Using domain knowledge requires much human effort, e.g., manually composing a set of linguistic rules or labeling a large corpus, and hence limits the applicability of the existing approaches. To overcome this limitation, we propose a new terminology extraction approach that makes use of no knowledge from any specific domain. In particular, we use the title words and the keywords in research papers as the seeding terms and word2vec to identify similar terms from an open-domain corpus as the candidate terms, which are then filtered by checking their occurrence in the research papers. We repeat this process using the newly found terms until no new candidate term can be found. We conduct extensive experiments on the proposed approach. The results show that our approach can extract the terms effectively, while being domain independent.

Keywords: terminology extraction, word2vec, statistical approach

1 Introduction

We study the problem of *terminology extraction* from a corpus of research papers, i.e., we extract the terms from the research papers. Here, a "*term*" refers to a word or compound words representing a concept of a specific *domain*, e.g., in chemistry, "alcohol" is a term that refers to an organic compound in which the hydroxyl functional group is bound to a saturated carbon atom[3]. Terminology extraction has various applications in text mining such as semantic analysis and machine translation. For instance, terminology extraction is a first step of building a knowledge graph from a large corpus, which has gained much

** This work is done when Birong is a visiting student at the University of Melbourne.
[3] http://goldbook.iupac.org/A00204.html

popularity in recent years in both the research community (e.g., YAGO2 [8]) and the industry (e.g., Google Knowledge Graph[4]).

Existing terminology extraction approaches [5, 7, 9, 16, 18] are using rule based techniques, supervised machine learning techniques, or a combination of these two types of techniques, all of which rely on some domain knowledge, e.g., manually composed linguistic rules or labeled corpuses. Acquiring such domain knowledge requires much human effort (e.g., manually labeling a large corpus) and limits the applicability of the existing approaches.

To overcome this limitation, we propose a novel approach that does not require any domain knowledge. This approach extracts terms from research papers (which can be of any domain) based on the following observations:

1. The title and the keywords of a research paper usually contains the key terms in the paper.
2. Related terms of the same domain tend to appear in similar context.
3. While a large manually labeled corpus of a particular domain is difficult to obtain, a massive open-domain corpus is relatively simple to acquire (e.g., Wikipedia provides free downloads of its full content[5]).

Following these observations, we extract terms as follows. We start with forming a set of seeding terms consisting of the title words and the keywords (after removing the stop words) of the research papers. We then identify words with similar context to that of the seeding terms from a massive open-domain corpus as the candidate terms. These candidate terms will be filtered by checking their occurrence in the research papers, the remaining of which will be added to the set of seeding terms. We repeat this process until no more candidate terms can be found and the set of seeding terms will be returned.

In our approach, we make use of *word2vec* [13, 14] to identify the words with similar context to that of the seeding terms from a massive open-domain corpus. Here, word2vec is an open-sourced tool released by Google for computing vector representations of words. It takes a text corpus as input and outputs a numeric vector for each word. Using these word vectors, given a word w, word2vec can compute and produce a list of words that are similar to w (e.g., having similar context) and their cosine similarity values. For example, when "france" is input, "spain" would be one of the similar words in the produced list, and the cosine similarity may be 0.678515. We train a word2vec model with a massive open-domain corpus (detailed in the experimental section), feed it with the seeding terms, and take the produced lists of similar words as our candidate terms. We observe that, even though word2vec is not trained on the corpus of research papers, it can still produce most of the terms appearing in the research papers. This is because of the massiveness of the open-domain training corpus that covers many of the concepts in various domains.

To summarize, we make the following contributions in this paper:

[4] http://www.google.com.au/insidesearch/features/search/knowledge.html
[5] http://en.wikipedia.org/wiki/Wikipedia:Database_download

- We analyze the key characteristics of term occurrence in research papers that may allow us to overcome the limitation of the domain dependent terminology extraction algorithms.
- Based on the analysis we propose a novel domain independent term extraction algorithm. This algorithm uses the title words and the keywords in research papers as the seeding terms and word2vec to identify similar terms from an open-domain corpus as the candidate terms. It filters the candidate terms by checking their occurrence in the research papers. The surviving candidate terms are added to the seeding terms and the above process repeats until there is no new candidate terms.
- We conduct extensive experiments on the proposed algorithm, and the results confirm the effectiveness of the proposed algorithm in extracting terms from research papers.

The rest of the paper is organized as follows. In Section 2 we review related studies. In Section 3 we present the proposed algorithm. In Section 4 we discuss the experimental results and in Section 5 we conclude the paper.

2 Related Work

Existing terminology extraction approaches [5, 7, 9, 16, 18] can be categorized into three groups: rule-based, statistical, and a combination of the two. We briefly review each group of the studies. A more detailed survey of the studies on terminology extraction can be found in [15].

Rule-based approaches use linguistic features to identify candidate terms. Various linguistic rules [4, 5, 7] have been devised for term extraction. For example, a simple rule is that terms are mostly nouns or noun phrases in the form of *[noun, noun]* or *[adjective, noun]*. This rule has been applied in systems such as the PHRASE system [4]. Evans et al. [5] combine the simple noun phrases and compose more complex rules for term extraction. Daille et al. [2] identify the most common linguistic rules from human produced terminological data banks. Their study confirms that terms are mainly short noun phrases in the form of *[noun, noun]* and *[adjective, noun]*. They call terms in such form the *base-terms*. Using morphological or syntactic variations, more complex and longer terms can be formed from the base-terms. Other than word level structural rules for terms, sentence level structural rules have also been proposed. For example, Xun and Li [20] use rules such as "*[noun] is ...*" or "*[noun] refers to ...*" to identify the terms and their definitions at the same time.

Using the linguistic rules, a terminology extraction algorithm generally works in four steps [15]:

- Perform Part-Of-Speech (POS) tagging on the corpus for terminology extraction.
- Identify and extract candidate terms from the corpus using the basic linguistic rules.
- Collapse variations of the same term into one unique form.

– Apply more sophisticated linguistic rules to further filter the candidate terms.

After the four steps a list of candidate terms will be produced. Depending on the quality of the linguistic rules and the domain of the corpus, the probability of these candidate terms being true terms varies. A further refinement step is required to identify the true terms, which is usually done by human experts manually.

Statistics-based approaches use statistical features [19] to determine the possibility (i.e., the *termhood* [15]) of a word (or compound words) being a term. For example, Salton et al. [16] extract all 2-word combinations as candidate terms, and use *tf-idf* to measure their termhood. Jones et al. [9] extract all N-word combinations as candidate terms, and use term *length* and *frequency* to measure the termhood. Krenn [11] also uses frequency as the termhood measure and empirically shows that it is a reliable measure. More sophisticated measures have been used, such as the Z-score [3], the C-value [6], and the Multiword Expression Distance [1]. The main disadvantage of statistics-based approaches is that they need a large training data set to learn the parameters for determining whether a candidate term is indeed a term, which is usually manually labeled and requires much human effort.

Hybrid approaches combine rule-based techniques and statistical techniques. A common practice is to use linguistic rules to first identify a set of candidate terms and then filter them by statistical termhood measures. For example, Earl [4] first extracts noun phrases as the candidate terms and then filters them by the frequency of the noun elements. Justeson and Katz [10] also first extract the candidate terms using linguistic rules, and then rank the candidates by frequency. Semantic and contextual information is also used in the hybrid approaches to help rank the candidate terms. For example, Maynard and Ananiadou [12] use *context-factor* to incorporate the semantic and contextual information. Velardi et al. [18] use *domain relevance* and *domain consensus* to achieve similar purpose.

Table 1. Similar Word List

Word	Cosine similarity
spain	0.678515
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

Word2vec [13, 14] is an open-sourced tool released by Google for computing vector representations of words using deep learning techniques. It takes a text corpus as input and computes a numeric vector for each word. Using these vectors, given a word w, word2vec can compute and produce a list of words that are similar to w (e.g., having similar context) and their cosine similarity values. For example, given "france" as the input, a list as shown in Table 1 may be produced.[6] If the training corpus is labeled properly, word2vec can also handle phrases, i.e., computing phrase vectors as well as similar phrases. In this study, we train a word2vec model with a massive open-domain corpus (detailed in the experimental section), feed it with the seeding terms, and take the produced lists of similar words as our candidate terms. We observe that, even though word2vec is not trained on the corpus of research papers, it can still produce most of the terms appearing in the research papers. This is because of the massiveness of the open-domain training corpus that covers many of the concepts in various fields. We will detail how word2vec is used in the following section.

3 The Term Extraction Algorithm

Given a set of research papers P of the *same domain*, we aim to extract all *terms* in the research papers. Here, the research papers need to be of the same domain, but we do not have any assumption on of which specific domain the research paper should be; a "term" refers to a word or compound words representing a concept of the domain.

To gain insight to the problem of extracting terms from research papers we conducted an empirical study on a corpus of 200 medical research papers. We have the following observations.

- *60.13% of the nouns in the title are terms; 83.61% of the noun phrases in the title are terms (after removing the stop words).*
- *96.72% of the keywords are terms.*
- *64.44% of the terms not in either the titles or keywords appear in similar context (in the abstract or body) to that of the terms in the titles or keywords.*

These observations motivate us to use the nouns and noun phrases in the title and the keywords as a set of seeding terms, find the words or phrases in the abstract and body parts of the research papers that are similar to the seeding terms, and return them together with the seeding terms as the answer set. To facilitate the process of finding the words or phrases similar to the seeding terms, we make use of *word2vec* [13, 14].

We propose to extract terms from a set of research papers P as follows.

1. We use a POS tagging tool[7] to identify the noun phrases from the titles of the research papers in P.

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

[7] There are many existing POS tagging tools, e.g., the Stanford POS Tagger [17].

Algorithm 1: Term Extraction(P, k, α)

```
1  $N \leftarrow$ the noun phrases in the titles of $P$;
2  $K \leftarrow$ the keywords in the titles of $P$;
3  $W \leftarrow N \cup K$;
4  foreach $w \in W$ do
5      Compute $ranking\_score(w)$;
6  Sort $W$ by the ranking scores;
7  $T \leftarrow$ top-$k$ words/phrases in $W$;
8  $has\_new\_term \leftarrow true$;
9  while $has\_new\_term$ do
10     $has\_new\_term \leftarrow false$;
11     $L \leftarrow \emptyset$;
12     $C \leftarrow \emptyset$;
13     foreach $t \in T$ do
14         $L_t \leftarrow$ words/phrases similar to $t$ returned by word2vec;
15         $L \leftarrow L \cup \{L_t\}$;
16     foreach $L_t \in L$ do
17         foreach $w \in L_t$ do
18             if $w$ appears in at least $\alpha$ lists in $L$ then
19                 $C \leftarrow C \cup \{w\}$;
20     foreach $c \in C$ do
21         if $c$ appears in $P$ then
22             $T \leftarrow T \cup \{c\}$;
23             $has\_new\_term \leftarrow true$;
   return: $T$
```

2. Together with the keywords of the research papers in P, we add the noun phrases to a set W.
3. We use the top-k words/phrases in W as a set of seeding terms T, where k is an algorithm parameter that will be chosen empirically. To choose the top-k words/phrases, we rank the words/phrases in W by their length (number of words contained by a phrase) and frequency in P. As suggested by existing studies such as [9, 11], these two measures have been reliable in identifying the true terms. Given a word/phrase $w \in W$, we use the following empirical equation to compute its ranking score:

   $$ranking\_score(w) = \log(f_w) \times l_w$$

   Here, f_w denotes the frequency of w and l_w denotes the length of w. Intuitively, words/phrases of larger frequency or length have higher probability of being true terms.
4. For each seeding term $t \in T$, we use word2vec to compute a list of words/phrases that are similar to t.
5. If a word/phrase appears in at least α lists returned by word2vec, we add it to the set of candidate terms C. Here, α is an algorithm parameter that will be chosen empirically.
6. For each candidate term $c \in C$, we scan the research papers in P and see if it can be found in the papers. If yes, we add it to the set of seeding terms T. Otherwise, we simply drop c.
7. We repeat Step 4 through Step 6 until no more terms can be found. Then we return T as the answer set.

Discussion. Algorithm 1 summarizes the procedure above. In the algorithm, word2vec is trained on an open-domain corpus which does not require any labeling effort. Meanwhile, the algorithm does not require any domain specific knowledge and is fully automatic. Therefore, we achieve an algorithm that avoids the drawbacks of the existing approaches. Note that a potential limitation is that when word2vec is trained on an open-domain corpus, it may not contain all the terms that appear in P and thus cannot help extract those terms. This limitation can be easily overcome by adding the documents in P to the corpus used to train word2vec.

4 Experimental Study

We implement the proposed algorithm in Perl and conduct experiments on a Desktop computer with a 3.0GHz CPU and 3GB memory.

We test the algorithm on a set of 200 medical research papers, where a total of 1704 unique terms with 9780 occurrences have been labeled by domain experts as the gold standard. We measure the term extraction accuracy and recall:

$$Accuracy = \frac{number\ of\ unique\ true\ terms\ extracted}{total\ number\ of\ unique\ terms\ extracted} \times 100\%$$

$$Recall = \frac{number\ of\ unique\ true\ terms\ extracted}{total\ number\ of\ unique\ true\ terms} \times 100\%$$

To the best of our knowledge, there is no existing domain independent terminology extraction algorithms. Hence, no baseline algorithm is used in the experiments.

We train a word2vec model with an open-domain corpus containing 500,000 documents from science and technology periodicals. The corpus has approximately 20 billion words.

The proposed algorithm takes in the 200 medical research papers, extracts the nouns and noun phrases from the titles and the keywords as the seeding terms, and use word2vec to identify candidate terms. We only use the words/phrases that appear in at least 2 word2vec similar word list as the candidate terms. We filter the candidate terms by checking their respective number of appearance in the research papers and only return a candidate term as a term if it appears for at least once.

4.1 The Effect of the Number of Initial Seeding Terms

We first verify the effect of the number of initial seeding terms k. We expect higher recall and lower accuracy as more initial seeding terms are used. Table 2 shows the result where we vary k from 10 to 40. We use the top 15 similar words returned by word2vec for each seeding term as the candidate terms in this set of experiments. As expected, the more initial seeding terms are used, the higher the recall will be. This is natural because more initial seeding terms will bring more candidate terms and eventually more terms. Also, the recall increases as more iterations of the algorithm are performed since more candidate terms will be find by word2vec (for conciseness we only show the result of the first two iterations). Meanwhile, the accuracy drops as k increases because more nouns and noun phrases that are less likely to be true terms are used as the seeding terms, and the candidate terms similar to them are less likely to be the true terms.

Table 2. Varying Initial Seeding Term Set Size

k	10	20	30	40
Recall (iteration 1)	56.69%	66.90%	71.83%	74.30%
Recall (iteration 2)	62.68%	72.89%	76.06%	77.82%
Accuracy (iteration 1)	40.21%	36.54%	32.64%	30.35%
Accuracy (iteration 2)	36.73%	33.50%	30.07%	28.14%

4.2 The Effect of the Size of the Word2vec Similar Word Lists

Given an input word w, word2vec can return a list all words/phrases appearing in the corpus sorted by their similarity to w. The words/phrases appearing at the later part of the list are not very similar to w. Therefore, we limit the size of the word2vec similar word lists and only use the words/phrases that are most similar to the seeding terms as the candidate terms.

Let the size of the word2vec similar word list be l. Table 3 shows the accuracy and recall where we vary l from 15 to 240. We use 40 initial seeding terms in this set of experiments. We can see that, the more words returned by word2vec are used as the candidate terms, the higher the recall will be. Also, the recall increases as more iterations of the algorithm are run. After 2 iterations, using 240 similar words for each seeding terms will achieve a recall of 90.14%. This is very close to the recall upper bound 91.20% (We scan the open-domain corpus where word2vec is trained on and find that 1554 among the 1704 unique terms are contained in the open-domain corpus. This means that using word2vec, the recall upper bound is $\frac{1554}{1704} \times 100\% = 91.20\%$). Adding more iterations will not significantly increase the recall and thus we terminate the algorithm after 2 iterations. Meanwhile, the accuracy drops as the recall increases, which is also expected because currently the proposed algorithm has a very simple candidate term checking procedure and has a relatively small pruning power in filtering the candidate terms. When more similar words are considered as the candidate terms, more will pass the filtering stage and be returned as terms. How to achieve a better accuracy while maintaining a good recall value is our next step of study.

Table 3. Varying the Size of the Word2vec Similar Word Lists

l	15	30	60	120	240
Recall (iteration 1)	74.30%	76.41%	80.63%	83.45%	83.08%
Recall (iteration 2)	77.82%	81.69%	86.27%	88.03%	90.14%
Accuracy (iteration 1)	30.35%	28.48%	26.83%	25.49%	25.92%
Accuracy (iteration 2)	28.14%	25.58%	23.45%	21.62%	21.94%

4.3 The Effect of the Pruning Threshold of the Candidate Terms

We explore the following two ways to increase the term extraction accuracy.

Table 4. Varying the Number of Word2vec Similar Word Lists that a Candidate Term Needs to Appear in

α	2	3	4	5	6
Recall (iteration 2)	77.82%	76.41%	76.06%	75.35%	75.35%
Accuracy (iteration 2)	28.14%	29.51%	30.02%	30.19%	30.36%

(1) Requiring the words/phrases to appear in more than 2 word2vec similar word lists so that they can be added to the candidate terms. In this set of ex-

periments we set the word2vec similar word list size at 15 and the number of initial seeding terms at 40. We vary the number of word2vec similar word lists a word/phrase needs to appear in, i.e., α, so as to be treated as a candidate term from 2 to 6. Table 4 shows the accuracy and recall values. Intuitively, as α increases, the accuracy increases while the recall decreases. We notice that the increase and the decrease are at a similar rate. This can guide us to choose an appropriate value of α given a recall or accuracy value we want to achieve.

Table 5. Varying the Number of Research Papers that a Candidate Term Needs to Appear in

p	1	2	3	4	5
Recall (iteration 2)	62.68%	49.30%	39.44%	30.99%	26.41%
Accuracy (iteration 2)	36.73%	40.84%	42%	40.03%	40%

(2) Requiring the candidate terms to appear in more than one research papers. In this set of experiments we set the word2vec similar word list size at 15 and the number of initial seeding terms at 10. We vary the number of research papers a candidate terms needs to appear in (denoted by p) so as to be returned as a term from 1 to 5. Table 5 shows the accuracy and recall values. It shows that, as p increases, the accuracy increases while the recall decreases. We notice that the recall decreases much faster than the accuracy increases. Further, accuracy starts to also decrease when the number p is larger than or equal to 4. This suggests that the value of p should be kept at a small value (e.g., 1 or 2).

4.4 Error Case Study

We conduct an error case analysis on the experimental results and make the following observations.

1. As shown in Section 4.2, our algorithm is able to achieve a recall of 90.14%. Among the terms that are not extracted, i.e., the false negatives, 89% are not contained in the open-domain corpus used to train the word2vec model. For example, "OVX rat" (a female rat whose ovaries have been removed) is term that is not contained in the training corpus and is missed by the algorithm. These false negatives can be recovered by adding the test corpus into the training corpus. The rest 10.3% of the false negatives (18 out of a total of 1704 terms labeled in the test corpus) are missed by the proposed algorithm because they have extremely low number of occurrences in the word2vec training corpus. As described in the experiments, we constrain the number of similar words returned by word2vec to be used as the candidate terms. If a word has a low occurrence and is not ranked as the top similar word to the seeding terms, then it may not be found the proposed algorithm. For example, "2,3-dimethyl-2,3-butanediol" is such a term.

2. As discussed in the subsections above, the accuracy of the proposed algorithm is relatively low due to a relatively simple strategy used to filter the candidate terms. We find that the false positives are mainly the words used as terms in the open-domain corpus while not being viewed as terms in the research papers, such as "portable transformer rectifier", "animal migration", and "average life expectancy". To filter such false positives, more sophisticated techniques such as the transfer learning techniques are to be explored to integrate the knowledge in the open-domain corpus into our target domain more systematically.

5 Conclusions and Future Work

We proposed a novel approach for extracting terms from research papers. We use the title words and the keywords in research papers as the seeding terms and word2vec to identify similar terms from an open-domain corpus as the candidate terms, which are then filtered by checking their occurrence in the research papers. We repeat this process using the newly found terms until no new candidates terms can be found. Compared with the existing approaches, our approach has the advantage of not requiring any domain knowledge, i.e., being domain independent, which is an important advantage considering the amount of effort needed to acquire and prepare the domain knowledge. As shown in the experimental study, our approach can extract the terms effectively. We analyze the false positives and false negatives in our experimental results and observe that: (i) the false positives are mainly the words used as terms in the open-domain corpus while not being viewed as terms in the research papers; (ii) the false negatives are the terms that have extremely low number of occurrences in the open-domain corpus. These observations are expected as the open-domain corpus used to train the word2vec model is very different from the corpus of research papers. Our future study is to build upon the current results and design algorithms to work together with word2vec to reduce the false negatives as well as algorithms to filter the false positives.

Acknowledgment This work is supported by the Natural Science Foundation of China under Projects No. 61300081, No. 61170162 and No. 2012BAH16F00, and Melbourne School of Engineering Early Career Researcher Grant under Project Reference Number 4180-E55.

References

1. Bu, F., Zhu, X., Li, M.: Measuring the non-compositionality of multiword expressions. In: COLING. pp. 116–124 (2010)
2. Daille, B., Habert, B., Jacquemin, C., Royauté, J.: Empirical Observation of Term Variations and Principles for their Description. Terminology 3(2), 197–258 (1996)
3. Dennis, S.F.: The construction of a thesaurus automatically from a sample of text. In: Proceedings of the Symposium on Statistical Association Methods for Mechanized Documentation. pp. 61–148 (1965)

4. Earl, L.L.: Experiments in automatic extracting and indexing. Information Storage and Retrieval 6(4), 313 – 330 (1970)
5. Evans, D.A., Zhai, C.: Noun-phrase analysis in unrestricted text for information retrieval. In: Proceedings of the 34th Annual Meeting on Association for Computational Linguistics (ACL). pp. 17–24 (1996)
6. Frantzi, K.T., Ananiadou, S.: Extracting nested collocations. In: COLING. pp. 41–46 (1996)
7. Gianluca, R.B., Rossi, G.D., Pazienza, M.T.: Inducing terminology for lexical acquisition. In: EMNLP (1997)
8. Hoffart, J., Suchanek, F.M., Berberich, K., Lewis-Kelham, E., de Melo, G., Weikum, G.: Yago2: Exploring and querying world knowledge in time, space, context, and many languages. In: Proceedings of the 20th International Conference Companion on World Wide Web (WWW). pp. 229–232 (2011)
9. Jones, L.P., Gassie, Jr., E.W., Radhakrishnan, S.: Index: The statistical basis for an automatic conceptual phrase-indexing system. Journal of the American Society for Information Science 41(2), 87–97 (1990)
10. Justeson, J.S., Katz, S.M.: Technical terminology: some linguistic properties and an algorithm for identification in text. Natural Language Engineering 1, 9–27 (1995)
11. Krenn, B.: Empirical implications on lexical association measures. In: Proceedings of the 9th EURALEX International Congress (2000)
12. Maynard, D., Ananiadou, S.: Identifying contextual information for multi-word term extraction. In: 5th International Congress on Terminology and Knowledge Engineering. pp. 212–221 (1999)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS). pp. 3111–3119 (2013)
14. Mikolov, T., Yih, W., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics on Human Language Technologies (HLT-NAACL). pp. 746–751 (2013)
15. Pazienza, M., Pennacchiotti, M., Zanzotto, F.: Terminology extraction: An analysis of linguistic and statistical approaches. In: Sirmakessis, S. (ed.) Knowledge Mining, Studies in Fuzziness and Soft Computing, vol. 185, pp. 255–279. Springer Berlin Heidelberg (2005)
16. Salton, G., Yang, C.S., Yu, C.T.: A theory of term importance in automatic text analysis. Journal of the American Society for Information Science 26(1), 33–44 (1975)
17. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL). pp. 173–180 (2003)
18. Velardi, P., Missikoff, M., Basili, R.: Identification of relevant terms to support the construction of domain ontologies. In: Proceedings of the ACL Workshop on Human Language Technology and Knowledge Management. pp. 5:1–5:8 (2001)
19. Wen, Z., Zhang, R., Ramamohanarao, K., Qi, J., Taylor, K.L.: MASCOT: fast and highly scalable SVM cross-validation using gpus and ssds. In: ICDM. pp. 580–589 (2014)
20. Xun, E., Li, C.: Applying terminology definition pattern and multiple features to identify technical new term and its definition. Journal of Computer Research and Development 46(1), 62–68 (2009)