Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 137 (2018) 102–108

Procedia Computer Science

www.elsevier.com/locate/procedia

SEMANTiCS 2018 – 14th International Conference on Semantic Systems

Adapted TextRank for Term Extraction: A Generic Method of Improving Automatic Term Extraction Algorithms

Ziqi Zhang*, Johann Petrak, Diana Maynard

University of Sheffield, 211 Portobello, Sheffield, S1 4DP, UK

Abstract

Automatic Term Extraction is a fundamental Natural Language Processing task often used in many knowledge acquisition processes. It is a challenging NLP task due to its high domain dependence: no existing methods can consistently outperform others in all domains, and good ATE is very much an unsolved problem. We propose a generic method for improving the ranking of terms extracted by a potentially wide range of existing ATE methods. We re-design the well-known TextRank algorithm to work at corpus level, using easily obtainable domain resources in the form of seed words or phrases, to compute a score for a word from the target dataset. This is used to refine a candidate term's score computed by an existing ATE method, potentially improving the ranking of real terms to be selected for tasks such as ontology engineering. Evaluation shows consistent improvement on 10 state of the art ATE methods by up to 25 percentage points in average precision measured at top-ranked K candidates.

Keywords: automatic term extraction; NLP; terminology; ontology engineering

1. Introduction

Automatic Term Extraction (ATE) recognises *terms* (either single words or word collocations representing domain-specific concepts) in a collection of domain-specific, usually unstructured texts (target corpus). It is a fundamental step for many complex text-based knowledge acquisition tasks, such as ontology engineering [6], glossary construction [22], and improving information access [18]. Despite continued research effort over the years, good ATE is still very much considered an unsolved problem [24]. In this work, we introduce a novel method addressing two limitations in the state of the art. First, no single ATE method consistently performs well in all domains [24]. We therefore investigate a generic method that can potentially *improve the accuracy of a wide range of existing ATE methods*. Second, while

* Corresponding author. Tel.: +44-114-222-2657.
E-mail address: ziqi.zhang@sheffield.ac.uk, johann.petrak@sheffield.ac.uk, d.maynard@sheffield.ac.uk

the majority of ATE methods are unsupervised, we argue that it is often possible to use existing lexical resources or easily generate such resources to support ATE.

Based on these principles, we introduce **AdaText**[1], a generic method that revises the TextRank algorithm [16] to apply it to an ATE method to improve its performance. Given a target corpus, our method firstly selects a subset of words extracted from the corpus, based on their semantic relatedness to a set of seed words or phrases considered relevant to a domain but not necessarily representative of the terms from the target corpus. It then applies an adapted TextRank algorithm to create a graph for these words, and computes a corpus-level TextRank score for each selected word. These scores are then used to revise the score of a term candidate previously computed by an ATE method. AdaText is extensively evaluated with 10 state-of-the-art ATE algorithms on two well-known ATE datasets. We show it can consistently improve these methods by up to 25 percentage points in average precision measured at top K ranked candidate terms.

2. Related Work

A typical ATE method consists of two sub-processes: *extracting candidate terms* using linguistic processors, followed by *candidate ranking and selection (i.e., filtering)* using algorithms that exploit word statistics. *Linguistic processors* often make use of domain-specific lexico-syntactic patterns to capture term formation and collocation, such as noun phrases, n-grams, or sequences based on their Part-of-Speech (PoS) tag chains [15]. *Candidate ranking and selection* then computes scores for candidate terms using their statistics gathered from the corpus, to indicate their likelihood of being a term, and then classifies the candidates into terms and non-terms often based on heuristics such as a score threshold, or a selection of the top-ranked candidate terms [24].

The ranking algorithms are considered the most important and complicated process in an ATE method [4] as they are often how an ATE method distinguishes itself from others. These are often based on unithood (the collocation strength of the lexical components) and termhood (the degree to which a term is relevant to a domain). Methods based on unithood include word association measures such as the χ^2 test and mutual information [7]; while some well-known termhood-based methods include CValue [2], and more recently topic-modelling based techniques [11]. Many use a combination of both unithood and termhood [17, 12]. Traditionally, ATE methods are unsupervised, though supervised machine learning has been adopted recently [4, 13].

ATE is also closely related to keyword extraction [21, 8], where TextRank [16] is one of the most influential methods. It creates a graph of words based on their collocation in a document, and computes a score using the PageRank algorithm applied to the graph. The key difference of that task is that it aims to extract only a few (e.g. 10−15) keyphrases within individual document context, while ATE extracts terms that are representative for the entire corpus, and thus depends on corpus-level statistics.

Research has shown that the performance of state of the art ATE methods is always domain dependent, such that a high-performing method in one task does not always perform well in another [4, 25]. We thus argue that aiming to develop a 'one-size-fit-all' ATE method for any domain is impractical. It may be more realistic and useful to develop generic methods to improve performance in any domain when coupled with an existing ATE method. Meanwhile, although most ATE methods are unsupervised, it is often possible to identify existing lexical resources from a domain (e.g. gazetteers), or to generate such resources from widely available domain corpora (e.g. author-supplied keywords for publications in a subject area). Such resources are not necessarily representative of the target corpus and may contain noise. However, they may be relevant to the task and could be used to support ATE.

3. Methodology

Formally, let D denote the domain specific target corpus. ATE aims to extract a set of candidate terms $T = \{t_1, t_2, ..., t_m\}$ from D, compute a score for each candidate $ate(t_i)$, then rank T by this score and select a subset to be considered as true terms. Let S be a set of seed words or phrases, which can be either taken from existing domain lexicons, or generated in an unsupervised way from available corpora (the creation of S is domain-specific and is de-

[1] **Ada**pted **Text**Rank for Automatic **T**erm Extraction

scribed in Section 4). Note that S does not need to be representative of the target corpus (i.e., S and T can be disjoint). We define $words(X)$ as a function returning distinct words (with stopwords removed) from X, and X can be a single candidate term t_i (which can be a single word, or contain multiple words), a document d_i, or sets of them. We further define $W = words(D)$ as the set of distinct words from the corpus.

AdaText selects a subset $W_{sub} \subset W$ based on the semantic relatedness of a word $w_j \in W$ with entries in S (Section 3.1). Next, it creates a corpus level graph of W_{sub} based on a collocation window size win, and runs PageRank to derive a 'corpus level' TextRank score of the words (Section 3.2). Finally, for each candidate $t_i \in T$, we aggregate its score computed by the ATE method $ate(t_i)$ with the TextRank score of its composing words (Section 3.3).

3.1. Seeding

Let $s_k \in S$ denote an entry in the seed set. We compute pair-wise relatedness $rel(s_k, w_j)$ for all pairs $(s_k, w_j) \in S \times W$ using word embeddings for s_k and w_j with the cosine similarity function between two vectors. For the word embeddings, we use the GloVe embeddings pre-trained on the general-purpose Common Crawl data[2], but others could be investigated. For multi-word expressions, we calculate the averaged embedding vectors of all words, following a method which was previously used in [9, 23] on compositional embeddings. We then calculate the cosine similarity from the average vectors. Words not present in the embeddings vocabulary are ignored and will have a similarity score of zero with other words.

Next, for each $s_k \in S$, we apply a threshold min to return a subset of W satisfying $rel(s_k, w_j) > min$, and concatenate the selected subsets for all entries in the seed set to create W_{sub}.

3.2. Corpus Level TextRank

The original TextRank algorithm creates a graph for each individual document by assigning each distinct word from the document as a node and placing an edge between two words if they appear within each other's context, which is defined as a window of win words. We adapt this for corpus-level in two ways. First, the nodes on the graph consist of distinct words extracted from the entire corpus, but we only use W_{sub} instead of all words. By focusing on words that are semantically related to the seed set, we should get a more accurate reflection of their relevance to the domain. Second, an edge is created for two nodes if they appear in each other's context within any document. We keep the remaining part of TextRank, i.e., the use of context window and the PageRank algorithm, unchanged. Thus at the end of this process, each word w_j on the graph is assigned a TextRank score, denoted as $tr(w_j)$.

3.3. Combining TextRank and ATE scores

The final refined score of a candidate term t_i is a non-linear[3] combination of its score computed by an ATE algorithm and the TextRank scores of its composing words:

$$score(t_i) = (1.0 + \frac{\sum_{w_i \in words(t_i)} tr(w_i)}{|words(t_i)|}) \times ate(t_i) \quad (1)$$

4. Experiment Settings

We first run a base ATE method over a corpus to obtain a ranking of candidate terms, and then run AdaText to compute TextRank scores of words, refining the original term scores to obtain a new ranking.

Base ATE methods. We select the following state of the art ATE methods, which include some best performing ones based on previous studies, and are chosen to represent different categories including unithood based, termhood based, hybrid, and learning based methods. We summarise these methods below and refer readers to their original papers for algorithmic details.

[2] http://nlp.stanford.edu/data/glove.840B.300d.zip
[3] Although we also experimented with linear combination, empirically results are worse.

Table 1. The GloVe word embeddings coverage on the datasets.

Dataset	candidate terms	multi-word candidate terms	OOV words	candidate terms containing OOVs
GENIA	38,850	38,637	1,512	6,682
ACLv2	5,659	4,101	110	203

- modified *TFIDF* [24] adapts the classic document-specific TFIDF (term frequency, inverse document frequency) used in information retrieval to work at corpus level by replacing term frequency in each document with total frequency in the corpus
- *CValue* [2] focuses on multi-word terms by computing a score that is based on the frequency of a candidate and its length, then adjusted by the frequency of longer candidates that contain it
- *Basic* [5] modifies CValue by promoting nested candidate terms, often used for creation of longer terms
- *RAKE* [20] is based on a similar principle to Basic but uses a graph based model
- *Weirdness (WD)* [1] compares frequency of a candidate term in the target domain-specific corpus with a reference corpus
- *LinkProbability (LP)* [4] uses Wikipedia as a reference corpus and normalises the frequency of a candidate term as a hyperlink caption by its total frequency in Wikipedia pages
- χ^2 [14] measures the degree to which a candidate term co-occurs with some 'frequent' terms are biased
- *GlossEx* [17] linearly combines a modified Weirdness score with a notion of 'term cohesion', which measures the degree to which the composing words tend to occur together as a candidate other than appearing individually
- *Positive Unlabeled (PU)* learning [4] follows a bootstrapping approach to train a classifier using features such as CValue scores of candidate terms

For all methods, we use their implementation from the JATE[4] and ATR4S[5] libraries. The two libraries support different linguistic processors to extract candidate terms. We configured them to use the domain-specific PoS tag sequence patterns previously reported in [24] for candidate term extraction.

We also test a method purely based on semantic relatedness between the term and the domain lexicon S. Let S'_{t_i} be the subset of S where each entry has a non-zero relatedness score with t_i, computed as before, then:

$$avgrel(t_i) = \frac{\sum_{s_k \in S'_{t_i}} rel(s_k, t_i)}{|S'_{t_i}|} \tag{2}$$

This is similar to the idea of 'key concept relatedness' introduced in [4], who used words *within* the target corpus extracted in a controlled way as reference lexicon for computing semantic relatedness.

AdaText parameters. We implemented AdaText based on an existing TextRank implementation[6], and experiment with two parameters: the semantic relatedness threshold *min* starting from 0.5 with an increment of 0.05 up to 0.85, and the context window *win* using the default value of 5, and 10 which is quite large given the size of each document.

Performance measure. One frequently used approach for measuring ATE performance is **Precision at top *K* (P@K)** ranked candidate terms, instead of the standard Precision, Recall, and F1. This is because in practice, real terms are often infrequent, and the expected number of correct terms is unknown a priori. In reality, domain experts would only verify a subset of the candidate terms from the ranked list, and given the large number of candidate terms, only top K are considered. Thus we calculate P@K, for five different Ks (50, 100, 500, 1k, 2k).

Datasets. We use two standard datasets: **GENIA** [10], which contains 2,000 semantically annotated Medline abstracts containing some 434k words, and **ACLv2** [19], which comprises 300 abstracts from the publications indexed by ACL, containing some 32k words. Gold standard terms are provided with the two datasets as separate lists, each containing some 33k and 3k terms. Table 1 shows the GloVe word embeddings coverage on this dataset.[7]

[4] https://github.com/ziqizhang/jate
[5] https://github.com/ispras/atr4s
[6] https://github.com/summanlp/textrank
[7] Note that the two ATE libraries extract slightly different sets of candidate terms on the same dataset. The statistics here are shown based on the JATE library. The statistics on the ATR4S library have the same trend.

Table 2. Result of the 10 base ATE methods on all datasets. The highest figures on each dataset under each evaluation metric are in **bold**.

	Basic	LP	PU	CValue	GlossEx	RAKE	TFIDF	Weirdness	χ^2	AvgRel
					ACLv2					
P@50	**.84**	.72	.82	.62	.44	.18	.64	.40	.58	.30
P@100	.72	.69	**.82**	.69	.46	.15	.65	.50	.62	.34
P@500	.56	.56	.60	**.67**	.34	.29	.53	.36	.48	.39
P@1,000	.49	.51	.43	**.56**	.36	.29	.47	.40	.45	.35
P@2,000	.39	.39	.40	**.45**	.38	.32	.43	.40	.41	.33
AvgP@K	.60	.57	**.61**	.60	.40	.25	.54	.41	.52	.34
					GENIA					
P@50	.80	.38	.74	.86	**.88**	.68	.68	.78	.66	.16
P@100	.74	.51	.69	**.83**	.82	.63	.65	.74	.69	.28
P@500	.64	.70	.65	**.80**	.58	.56	.74	.78	.71	.36
P@1,000	.57	.69	.61	**.78**	.53	.52	.77	.77	.71	.44
P@2,000	.49	.66	.58	.74	.47	.44	**.77**	.74	.67	.42
AvgP@K	.65	.59	.65	**.80**	.66	.57	.72	.76	.69	.33

Seed sets. For GENIA, we create a seed set of 5,502 entries by extracting the named entities (e.g. gene, protein domain) from the Entity Relations Supporting Task dataset in the BioNLP Shared Task 2011.[8] Of these, only 25 match candidate terms exactly. For ACL, we extract noun phrases occurring at least twice in the titles of papers published since 2000 at ACL, NAACL, and EACL, to create a seed set of 1,301 entries, none of which exactly matches any candidate term.

5. Results and Discussion

Base ATE results. Table 2 shows that the performance of the base ATE methods can vary significantly depending on datasets. There is no single, consistently winning method on all five Ks. Considering only the average (Avg P@K), PU is the best performing method on the ACL corpus, however it becomes the fourth worst performing on the GENIA corpus.

AdaText results. Figure 1 shows the percentage of words retained under different *min* thresholds to create the TextRank graph. Figure 2 shows the results obtained by AdaText under different *min* thresholds when applied to each ATE method on the two datasets. We present only the average P@K to focus on the general trend and avoid repeating information due to the number of Ks.

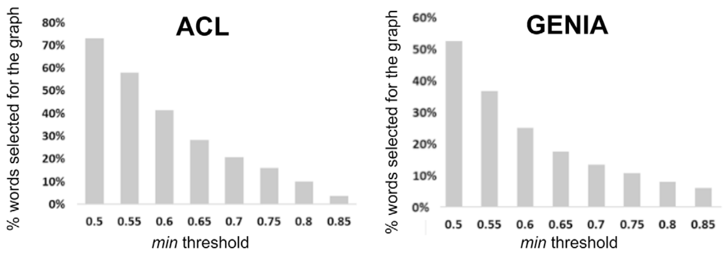

Fig. 1. Different threshold *min* values and the corresponding percentage of words selected to create the TextRank graph.

Overall, we notice that in the majority of cases, AdaText is able to further improve any base ATE method, sometimes quite significantly. This suggests that it is a very robust method. In terms of the *effect of the min* threshold, we notice that in many cases and particularly on the ACL corpus, a low (e.g. 0.5) or high value (0.85) can harm its

[8] http://2011.bionlp-st.org/home/entity-relations

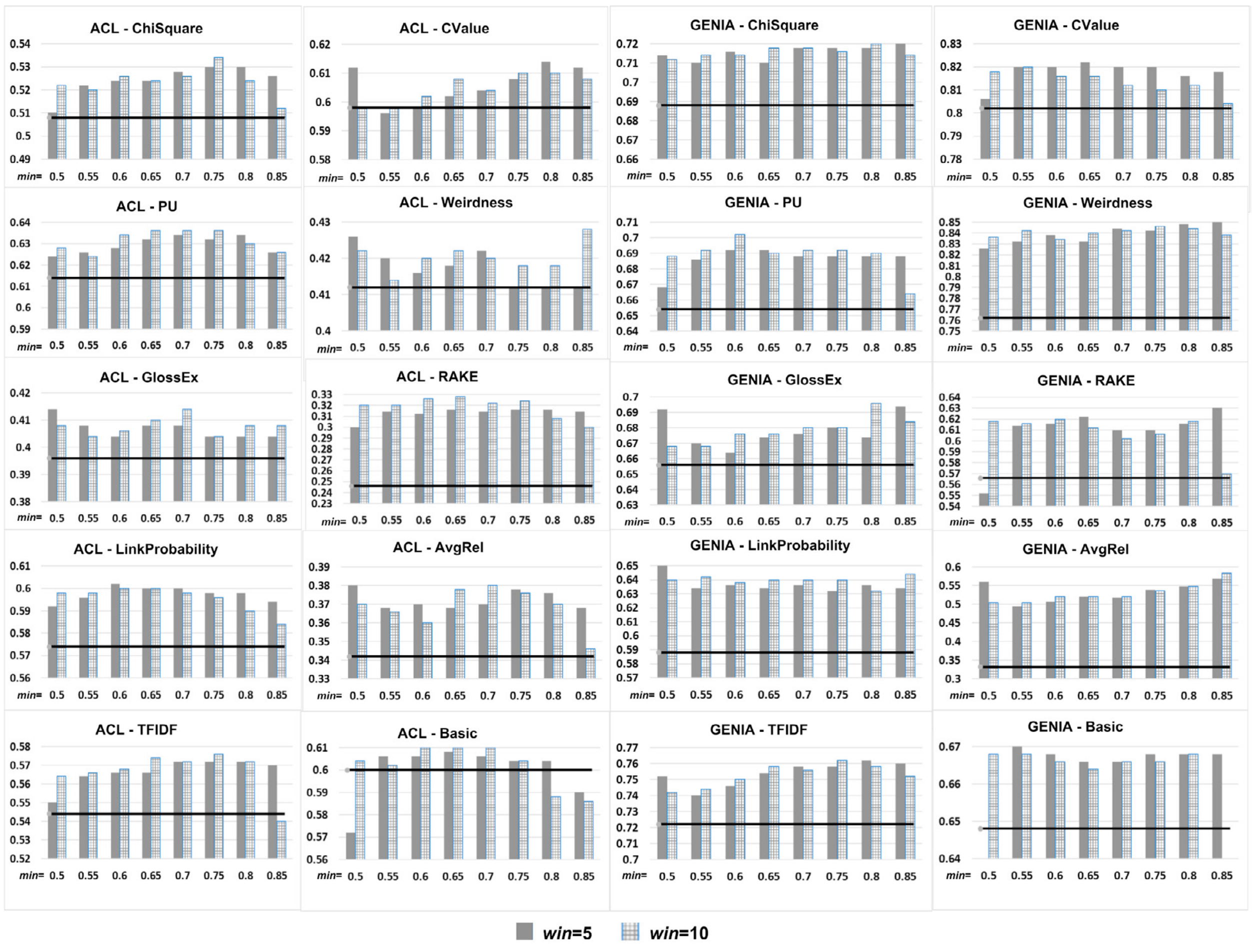

Fig. 2. Results by AdaText compared against the base ATE methods. *y*-axis: average P@K for all five K's considered; *x*-axis: the different *min* threshold values. The horizontal dark line indicates the result obtained by each base ATE method. The columns/bars indicate the results obtained by AdaText when coupled with the corresponding ATE method.

performance. Relating to Figure 1, these are when we could have selected too many or too few words for graph creation. The first case could have caused inclusion of irrelevant words, hence 'diluting' the scores of words calculated by TextRank as more nodes and edges are created. The second may create isolated small graphs biased towards the local focus words, as well as reducing the percentage of candidate terms containing a word from the graph to below 50% or much less (e.g., on ACLv2, at $min = 0.8$ this is 61%, dropping to 27% with $min = 0.85$).

The effect of the *varying win* parameter is much weaker. The difference between *win* =5 and 10 is at best only noticed on some individual methods, and cannot be generalised. There is no consistent pattern as to which performs better than another.

Within the range of min = $[0.6, 0.75]$, AdaText brings consistent improvement in all cases, ranging from about 1 to 25 percentage points, depending on datasets and the base ATE methods. It is worth noting that AdaText is able to improve both the best and worst performing ATE methods on both datasets. Interestingly, on the GENIA dataset, AdaText managed to improve the second best performing base method, Weirdness, to 84.6 (76.2 + 8.2), outpeforming the best base method, CValue (80.2 + 2). However, the patterns of change in performance due to varying *min* within $[0.6, 0.75]$ are not strong, as we notice only a few cases of relatively sharp fluctuations out of the 20 cases. This suggests that overall, the performance of AdaText can be quite robust as long as the *min* is not at the low and high end of the spectrum.

6. Conclusion

In this work, we have introduced a method for improving existing ATE methods. We show that it improves all methods tested, sometimes quite significantly. However, the main limitation of our work is the lack of understanding of the relation between the threshold used for selecting words on the TextRank graph (*min*) and the performance of the base ATE methods, and how we can optimize this parameter automatically. Future work will explore this problem along with several other questions, including: whether and how the size and source of the seed lexicon affects performance; how to adapt TextRank to a graph of both words and phrases, and how this affects results.

References

[1] Ahmad, K., Gillam, L., Tostevin, L., 1999. University of surrey participation in trec 8: Weirdness indexing for logical document extrapolation and retrieval (wilder), in: Proc. of TREC1999.

[2] Ananiadou, S., 1994. A methodology for automatic term recognition, in: Proc. of COLING1994, ACL, Stroudsburg, PA, USA. pp. 1034–1038.

[3] Astrakhantsev, N., 2015. Methods and software for terminology extraction from domainspecific text collection, in: Ph.D. thesis. Institute for System Programming of Russian Academy of Sciences.

[4] Astrakhantsev, N., 2016. Atr4s: Toolkit with state-of-the-art automatic terms recognition methods in scala. arXiv preprint arXiv:1611.07804.

[5] Bordea, G., Buitelaar, P., Polajnar, T., 2013. Domain-independent term extraction through domain modelling, in: Proc. of the Conference on Terminology and Artificial Intelligence.

[6] Brewster, C., Iria, J., Zhang, Z., Ciravegna, F., Guthrie, L., Wilks, Y., 2007. Dynamic iterative ontology learning, in: Proc. of RANLP'07, Borovets, Bulgaria.

[7] Church, K., Hanks, P., 1990. Word association norms, mutual information, and lexicography. Comput. Linguist. 16, 22–29.

[8] Hasan, K.S., Ng, V., 2010. Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art, in: Proceedings of COLING10 Posters Volume, Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 365–373.

[9] Iyyer, M., Manjunatha, V., Boyd-Graber, J., Daume, H., 2015. Deep unordered composition rivals syntactic methods for text classification, in: Association for Computational Linguistics. URL: `docs/2015_acl_dan.pdf`.

[10] Kim, J., Ohta, T., Tateisi, Y., Tsujii, J., 2003. GENIA corpus - a semantically annotated corpus for bio-textmining, in: ISMB (Supplement of Bioinformatics), pp. 180–182.

[11] Li, S., Li, J., Song, T., Li, W., Chang, B., 2013. A novel topic model for automatic term extraction, in: Proc. of SIGIR'13, ACM, New York, NY, USA. pp. 885–888.

[12] Lossio-Ventura, J., Jonquet, C., Roche, M., Teisseire, M., 2014. Biomedical terminology extraction: A new combination of statistical and web mining approaches, in: Journées d'Analyse statistique des Données Textuelles, Paris, France. pp. 421–432.

[13] Maldonado, A., Lewis, D., 2016. Self-tuning ongoing terminology extraction retrained on terminology validation decisions, in: Proc. of Conference on Terminology and Knowledge Engineering.

[14] Matsuo, Y., Ishizuka, M., 2003. Keyword extraction from a single document using word co-occurrence statistical information. International Journal on Artificial Intelligence Tools 13, 157–169.

[15] Maynard, D., Funk, A., Peters, W., 2009. Using Lexico-Syntactic Ontology Design Patterns for ontology creation and population, in: WOP 2009 – ISWC Workshop on Ontology Patterns, Washington, USA.

[16] Mihalcea, R., Tarau, P., 2004. TextRank: Bringing order into texts, in: Proc. of EMNLP'04.

[17] Park, Y., Byrd, R., Boguraev, B., 2002. Automatic glossary extraction: Beyond terminology identification, in: Proc. of COLING'02, Association for Computational Linguistics. pp. 1–7.

[18] Peñas, A., Verdejo, F., Gonzalo, J., 2001. Corpus-based terminology extraction applied to information access, in: Proceedings of the Corpus Linguistics.

[19] QasemiZadeh, B., Schumann, A., 2016. The acl rd-tec 2.0: A language resource for evaluating term extraction and entity recognition methods., in: Proc. of LREC'16.

[20] Rose, S., Engel, D., Cramer, N., Cowley, W., 2010. Automatic keyword extraction from individual documents. John Wiley and Sons.

[21] Turney, P., 2000. Learning algorithms for keyphrase extraction. Inf. Retr. 2, 303–336.

[22] Velardi, P., Navigli, R., D'Amadio, P., 2008. Mining the web to create specialized glossaries. IEEE Intelligent Systems 23, 18–25.

[23] Wieting, J., Bansal, M., Gimpel, K., Livescu, K., 2016. Towards universal paraphrastic sentence embeddings, in: International Conference on Learning Representations.

[24] Zhang, Z., Gao, J., Ciravegna, F., 2016. Jate 2.0: Java automatic term extraction with apache solr, in: Proc. of LREC'16.

[25] Zhang, Z., Iria, J., Brewster, C., Ciravegna, F., 2008. A comparative evaluation of term recognition algorithms, in: Proc. of LREC'08, Marrakech, Morocco.