DFKI KeyWE: Ranking keyphrases extracted from scientific articles

Kathrin Eichler
DFKI - Language Technology
Berlin, Germany
kathrin.eichler@dfki.de

Günter Neumann
DFKI - Language Technology
Saarbrücken, Germany
neumann@dfki.de

Abstract

A central issue for making the content of a scientific document quickly accessible to a potential reader is the extraction of keyphrases, which capture the main topic of the document. Keyphrases can be extracted automatically by generating a list of keyphrase candidates, ranking these candidates, and selecting the top-ranked candidates as keyphrases. We present the KeyWE system, which uses an adapted nominal group chunker for candidate extraction and a supervised ranking algorithm based on support vector machines for ranking the extracted candidates. The system was evaluated on data provided for the SemEval 2010 Shared Task on Keyphrase Extraction.

1 Introduction

Keyphrases capture the main topic of the document in which they appear and can be useful for making the content of a document quickly accessible to a potential reader. They can be presented to the reader directly, in order to provide a short overview of the document, but can also be processed further, e.g. for text summarization, document clustering, question-answering or relation extraction. The task of extracting keyphrases automatically can be performed by generating a list of keyphrase candidates, ranking these candidates, and selecting the top-ranked candidates as keyphrases. In the KeyWE system, candidates are generated based on an adapted nominal group chunker described in section 3 and ranked using the SVM^{rank} algorithm (Joachims, 2006), as described in section 4. The used features are specified in section 5. In section 6, we present the results achieved on the test data provided for the SemEval 2010 Shared Task on Keyphrase Extraction[1] by selecting as keyphrases the top 5, 10, and 15 top-ranked candidates, respectively.

2 Related work

The task of keyphrase extraction came up in the 1990s and was first treated as a supervised learning problem in the GenEx system (Turney, 1999). Since then, the task has evolved and various new approaches have been proposed. The task is usually performed in two steps: 1. candidate extraction (or generation) and 2. keyphrase selection. The most common approach towards candidate extraction is to generate all n-grams up to a particular length and filter them using stopword lists. Lately, more sophisticated candidate extraction methods, usually based on additional linguistic information (e.g. POS tags), have been proposed and shown to produce better results (e.g. Hulth (2004)). Liu et al. (2009) restrict their candidate list to verb, noun and adjective words. Kim and Kan (2009) generate regular expression rules to extract simplex nouns and nominal phrases. As the majority of technical terms is in nominal group positions[2], we assume that the same holds true for keyphrases and apply an adapted nominal group chunker to extract keyphrase candidates.

The selection process is usually based on some supervised learning algorithm, e.g. Naive Bayes (Frank et al., 1999), genetic algorithms (Turney, 1999), neural networks (Wang et al., 2005) or decision trees (Medelyan et al., 2009). Unsupervised approaches have also been proposed, e.g. by Mihalcea and Tarau (2004) and Liu et al. (2009). However, as for the shared task, annotated training data was available, we opted for an approach based on supervised learning.

[1] http://semeval2.fbk.eu/semeval2.php?location=tasks#T6

[2] Experiments on 100 manually annotated scientific abstracts from the biology domain showed that 94% of technical terms are in nominal group position (Eichler et al., 2009).

3 Candidate extraction

Rather than extracting candidates from the full text of the article, we restrict our search for candidates to the first 2000 characters starting with the abstract[3]. We also extract title and general terms for use in the feature construction process. From the reduced input text, we extract keyphrase candidates based on the output of a nominal group chunker.

This approach is inspired by findings from cognitive linguistics. Talmy (2000) divides the concepts expressed in language into two subsystems: the grammatical subsystem and the lexical subsystem. Concepts associated with the grammatical subsystem provide a structuring function and are expressed using so-called closed-class forms (function words, such as conjunctions, determiners, pronouns, and prepositions, but also suffixes such as plural markers and tense markers). Closed-class elements (CCEs) provide a scaffolding, across which concepts associated with the lexical subsystem (i.e. nouns, verbs, adjectives and adverbs) can be draped (Evans and Pourcel, 2009). Spurk (2006) developed a nominal group (NG) chunker that makes use of this grammatical subsystem. Using a finite list of CCEs and learned word class models for identifying verbs and adverbs, a small set of linguistically motivated extraction patterns is stated to extract NGs. The rules are based on the following four types of occurrences of NGs in English: 1. at the sentence beginning, 2. within a determiner phrase, 3. following a preposition and 4. following a verb. Not being trained on a particular corpus, the chunker works in a domain-independent way. In addition, it scales well to large amounts of textual data.

In order to use the chunker for keyphrase extraction, we manually analysed annotated keyphrases in scientific texts, and, based on the outcome of the evaluation, made some adaptations to the chunker, which take care of the fact that the boundaries of a keyphrase do not always coincide with the boundaries of a NG. In particular, we remove determiners, split NGs on conjunctions, and process text within parentheses separately from the main text. An evaluation on the provided training data showed that the adapted chunker extracts 80% of the reader-annotated keyphrases found in the text.

[3]This usually covers the introductory part of the article and is assumed to contain most of the keyphrases. Partial sentences at the end of this input are cut off.

4 Candidate ranking

The problem of ranking keyphrase candidates can be formalized as follows: For a document d and a collection of n keyword candidates C = $c_1 ... c_n$, the goal is to compute a ranking r that orders the candidates in C according to their degree of keyphraseness in d.

The problem can be transformed into an ordinal regression problem. In ordinal regression, the label assigned to an example indicates a rank (rather than a nominal class, as in classification problems). The ranking algorithm we use is SVM^{rank}, developed by Joachims (2006). This algorithm learns a linear ranking function and has shown to outperform classification algorithms in keyphrase extraction (Jiang et al., 2009).

The target (i.e. rank) value defines the order of the examples (i.e. keyphrase candidates). During training, the target values are used to generate pairwise preference constraints. A preference constraint is included for all pairs of examples in the training file, for which the target value differs. Two examples are considered for a pairwise preference constraint only if they appear within the same document.

The model that is learned from the training data is then used to make predictions on the test examples. For each line in the test data, the model predicts a ranking score, from which the ranking of the test examples can be recovered via sorting.

For ranking the candidates, they are transformed into vectors based on the features described in section 5.

During training, the set of candidates is made up of the annotated reader and author keywords as well as all NG chunks extracted from the text. These candidates are mapped to three different ranking values: All annotated keywords are given a ranking value of 2; all extracted NG chunks that were annotated somewhere else in the training data are given a ranking value of 1; all other NG chunks are assigned a ranking value of 0. Giving a special ranking value to chunks annotated somewhere else in the corpus is a way of exploiting domain-specific information about keyphrases. Even though not annotated in this particular document, a candidate that has been annotated in some other document of the domain, is more likely to be a keyphrase than a candidate that has never been annotated before (cf. Frank et al. (1999)).

5 Features

We used two types of features: term-specific features and document-specific features. Term-specific features cover properties of the candidate term itself (e.g. term length). Document-specific features relate properties of the candidate to the text, in which it appears (e.g. frequency of the term in the document). Our term-specific features concern the following properties:

- **Term length** refers to the length of a candidate in number of tokens. We express this property in terms of five boolean features: *has1token*, *has2tokens*, *has3tokens*, *has4tokens*, *has5orMoreTokens*. The advantage over expressing term length as a numeric value is that using binary features, we allow the algorithm to learn that candidates of medium lengths are more likely to be keyphrases than very short or very long candidates.

- The **MSN score** of a candidate refers to the number of results retrieved when querying the candidate string using the MSN search engine[4]. The usefulness of MSN scores for technical term extraction has been shown by Eichler et al. (2009). We normalize the MSN scores based on the number of digits of the score and store the normalized value in the feature *normalizedMsn*. We also use a binary feature *isZeroMsn* expressing whether querying the candidate returns no results at all.

- **Special characters** can indicate whether a candidate is (un)likely to be a keyphrase. We use two features concerning special characters: *containsDigit* and *containsHyphen*.

- **Wikipedia** has shown to be a valuable source for extracting keywords (Medelyan et al., 2009). We use a feature *isWikipediaTerm*, expressing whether the term candidate corresponds to an entry in Wikipedia.

In addition, we use the following document-specific features:

- **TFIDF**, a commonly used feature introduced by Salton and McGill (1983), relates the frequency of a candidate in a document to its frequency in other documents of the corpus.

- **Term position** relates the position of the first appearance of the candidate in the document to the length of the document. In addition, our feature *appearsInTitle* covers the fact that candidates appearing in the document title are very likely to be keyphrases.

- **Average token count** measures the average occurrence of the individual (lemmatized) tokens of the term in the document. Our assumption is that candidates with a high average token count are more likely to be keyphrases.

- **Point-wise mutual information** (PMI, Church and Hanks (1989)) is used to capture the semantic relatedness of the candidate to the topic of the document. A similar feature is introduced by Turney (2003), who, in a first pass, ranks the candidates based on a base feature set, and then reranks them by calculating the statistical association between the given candidate and the top K candidates from the first pass. To avoid the two-pass method, rather than calculating inter-candidate association, we calculate the association of each candidate to the terms specified in the General Terms section of the paper. Like Turney, we calculate PMI based on web search results (in our case, using MSN). The feature *maxPmi* captures the maximum PMI score achieved with the lemmatized candidate and any of the general terms.

6 Results and critical evaluation

Table 1 presents the results achieved by applying the KeyWE system on the data set of scientific articles provided by the organizers of the shared task along with two sets of manually assigned keyphrases for each article (reader-assigned and author-assigned keyphrases). Our model was trained on the trial and training data (144 articles) and evaluated on the test data set (100 articles). The evaluation is based on stemmed keyphrases, where stemming is performed using the Porter stemmer (Porter, 1980).

Since SVM^{rank} learns a linear function, one can analyze the individual features by studying the learned weights. Roughly speaking, a high positive (negative) weight indicates that candidates with this feature should be higher (lower) in the

[4] http://de.msn.com/

Top	Set	P	R	F
5	reader	24.40%	10.13%	14.32%
	combined	29.20%	9.96%	14.85%
10	reader	19.80%	16.45%	17.97%
	combined	23.30%	15.89%	18.89%
15	reader	17.40%	21.68%	19.31%
	combined	20.27%	20.74%	20.50%

Table 1: Results on the two keyword sets: reader (reader-assigned keyphrases) and combined (reader- and author-assigned keyphrases)

ranking. In our learned model, the four most important features (i.e. those with the highest absolute weight) were *containsDigit* (-1.17), *isZeroMsn* (-1.12), *normalizedMsn* (-1.00), and *avgTokenCount* (+0.97). This result confirms that web frequencies can be used as a valuable source for ranking keyphrases. It also validates our assumption that a high average token count indicates a good keyphrase candidate. The *maxPMI* feature turned out to be of minor importance (-0.16). This may be due to the fact that we used the terms from the General Terms section of the paper to calculate the association scores, which may be too general for this purpose.

Acknowledgments

We thank Angela Schneider for her adaptations to the chunker and helpful evaluations. The research project DiLiA is co-funded by the European Regional Development Fund (ERDF) in the context of Investitionsbank Berlins ProFIT program under grant number 10140159. We gratefully acknowledge this support.

References

K. W. Church and P. Hanks. 1989. Word association norms, mutual information and lexicography. In *Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*.

K. Eichler, H. Hemsen, and G. Neumann. 2009. Unsupervised and domain-independent extraction of technical terms from scientifc articles in digital libraries. In *Proceedings of the LWA Information Retrieval Workshop*, TU Darmstadt, Germany.

V. Evans and S. Pourcel. 2009. *New Directions in Cognitive Linguistics*. John Benjamins Publishing Company.

E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*.

A. Hulth. 2004. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction*. Ph.D. thesis, Department of Computer and Systems Sciences, Stockholm University.

X. Jiang, Y. Hu, and H. Li. 2009. A ranking approach to keyphrase extraction. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

T. Joachims. 2006. Training linear svms in linear time. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*.

S. N. Kim and M. Y. Kan. 2009. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the ACL/IJCNLP Multiword Expressions Workshop*.

F. Liu, D. Pennell, F. Liu, and Y. Liu. 2009. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of the Conference of the NAACL, HLT*.

O. Medelyan, E. Frank, and I.H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the International Conference of Empirical Methods in Natural Language Processing (EMNLP)*.

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the EMNLP*.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

G. Salton and M. J. McGill. 1983. *Introduction to modern information retrieval*. McGraw-Hill.

C. Spurk. 2006. Ein minimal überwachtes Verfahren zur Erkennung generischer Eigennamen in freien Texten. Diplomarbeit, Saarland University, Germany.

L. Talmy. 2000. *Towards a cognitive semantics*. MIT Press, Cambridge, MA.

P. D. Turney. 1999. Learning to extract keyphrases from text. Technical report, National Research Council, Institute for Information Technology.

P. D. Turney. 2003. Coherent keyphrase extraction via web mining. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*.

J.-B. Wang, H. Peng, and J.-S. Hu. 2005. Automatic keyphrases extraction from document using backpropagation. In *Proceedings of 2005 international conference on Machine Learning and Cybernetics*.