Automatic Term Extraction in Technical Domain using Part-of-Speech and Common-Word Features

Nisha Ingrid Simon
Dalhousie University
Halifax, Nova Scotia
nsimon@dal.ca

Vlado Kešelj
Dalhousie University
Faculty of Computer Science
Halifax, Nova Scotia
vlado@cs.dal.ca

ABSTRACT

Extracting key terms from technical documents allows us to write effective documentation that is specific and clear, with minimum ambiguity and confusion caused by nearly synonymous but different terms. For instance, in order to avoid confusion, the same object should not be referred to by two different names (e.g. *"hydraulic oil filter"*). In the modern world of commerce, clear terminology is the hallmark of successful RFPs (Requests for Proposal) and is therefore a key to the growth of competitive organizations. While Automatic Term Extraction (ATE) is a well-developed area of study, its applications in the technical domain have been sparse and constrained to certain narrow areas such as the biomedical research domain. We present a method for Automatic Term Extraction (ATE) for the technical domain based on the use of part-of-speech features and common words information. The method is evaluated on a C programming language reference manual as well as a manual of aircraft maintenance guidelines, and has shown comparable or better results to the reported state of the art results.

CCS CONCEPTS

• **Information systems** → **Data mining**; **Ontologies**;

KEYWORDS

Terminology extraction, POS tagging, Natural language processing, Text mining

ACM Reference Format:
Nisha Ingrid Simon and Vlado Kešelj. 2018. Automatic Term Extraction in Technical Domain using Part-of-Speech and Common-Word Features . In *DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209280.3229100

DocEng'18, August 28–31, 2018, Halifax, Canada

ACM ISBN 978-1-4503-5769-2/18/08…$15.00
https://doi.org/10.1145/3209280.3229100

1 INTRODUCTION

Automated Term Extraction (ATE) from technical documents is an important problem since extracting key terms from technical documents allows us to write high-quality documentation that is specific and clear, with minimum ambiguity. For instance, in order to avoid confusion, the same object should not be referred to by two different names. In the world of business, clear terminology is the hallmark of successful RFPs. ATE in general is an important area of study because it has applications in IR (Information Retrieval) such as text summarization, text categorization, opinion mining and document indexing.

In creative writing, authors make their text more vivid by choosing different synonymous words for the same concept. However, this would cause confusion in technical writing. For example, if a technician is following assembly instructions, referring to the same part with different names could be confusing and would increase the chance of error in assembly. Thus, a more uniform and standardized terminology is highly regarded in the technical domain, and a tool that can detect terminology and help a writer to create a better document would be beneficial. A similar service would also be useful to reviewers and assessors of technical proposals and similar documents.

The vocabulary of technical terms is constantly growing, especially in specialized areas such as computer science, engineering and medicine [9]. With the pace of knowledge acquisition that is required to maintain currency in modern technical fields, it is helpful for a user to have access to a method of quickly extracting part of an index that is comprised of the key terms of a document.

Much work has been done on ATE, however the state of the art performance is still low. Major factors that affect ATE performance measures are the length of documents, a lack of structural consistency (for instance, lack of a defined index or abstract), topic changes, and the presence of uncorrelated topics in the same text [11].

1.1 Objective

Our hypothesis is that Automatic Term Extraction (ATE) can be successfully performed in the technical domain to extract keyphrases using part-of-speech information with additional information about commonality of some words. We build and evaluate a system for automated term extraction based on part-of-speech and common-word feature information, and evaluate it on the C programming language reference manual

[18]. As a gold standard we used the manually prepared indexes of terms found at the end of this document. In addition we apply our methods on a manual of aircraft maintenance guidelines [12], and compare our results to that obtained from the C programming language reference manual. Results are then compared to those obtained by TBXTools statistical [20].

1.2 Contributions

The novelty of this paper lies in the domain to which ATE is applied. While ATE has been used in extremely specific areas such as the biological domain [2], Archaeology or Chemistry [4], or in general areas such as newspapers [5], to the best of our knowledge, ATE has not been used earlier in the technical domain on documents such as C programming reference manuals or aircraft maintenance guidelines. Also most of the earlier research focuses on extracting a limited set of the most important keywords by rank, while our research is aimed at generating all valid index terms from a document.

2 RELATED WORK

There has been much work on Automatic Term Extraction based on document features and statistical approaches [16, 21]. Automatic keyphrase extraction was defined by Turney [22] as "the automatic selection of important topical phrases from within the body of the document". A human-generated keyphrase was considered to be the same as a machine-generated keyphrase if they had the same sequence of *stems*. A method to create a back-of-the-book index for *Stargazers* text using lexical classes was described by Da Sylva [6]. This process involved noun phrase extraction (including lemmatization and part-of-speech tagging), text segmentation, candidate term weighting and index compilation. Other related work includes material by Ferrari et al. [7]. TBXTools [20] was presented by Oliver and Vàzquez as a tool that was built in Python to extract keyterms from controlled corpora. There is also an instance where ATE has been used on only the abstracts of scientific articles, as opposed to the entire documents [14]. Tf-Idf has been used to extract keyterms [10]. An overview of the advances in Natural Language Processing was presented by Hirschberg and Manning [13] where it is observed that "simple methods using words, part-of-speech (POS) sequences... or simple templates can often achieve notable results when trained on large quantities of data". Methods for ATE and comparison of various approaches were discussed in Astrakhansev et al. [1, 2]

3 METHODOLOGY

ATE can be considered a supervised learning classification task [8] where a candidate phrase is either a keyphrase or it is not. A certain aspect of ATE can be seen as a search for collocations [19]. One way of identifying collocations and keyterms is by using part-of-speech tags. A method of selecting the most frequent bigrams and passing them through a part of speech filter of "likely phrase" sequence patterns

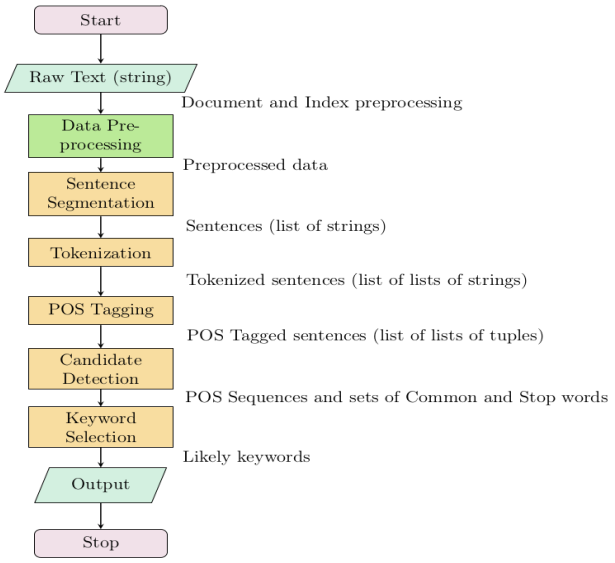

Figure 1: Methodology

was proposed by Justeson [15]. For example, if some likely part-of-speech patterns for selecting terms are adjective-noun or noun-noun, the filter would search for patterns *JJ NN* or *NN NN* in the text.

Our overall methodology is shown in Figure 1. We used a C programming language reference manual as our first dataset. The index of each C programming reference document was used as the gold standard to measure how well the system performed. Data pre-processing forms an important part of the process. After individual words are tagged, POS sequences are extracted. POS tagging is then performed in order to obtain candidate terms. A list of stop words (common words i.e. domain-specific stop words) and calculation of the frequency of word occurrence are then used to find likely keywords.

Our second data set was a manual of aircraft maintenance guidelines [12]. As a gold standard, the glossary of the document was used to validate candidate terms. A major difference between this dataset and the C data set was that the former contained a large number of acronyms. The NLTK toolkit version 2.0.4 (Natural Language Toolkit) described by Bird et al. [3] was used to process the data. TBXTools statistical [20] was then used to compare the results of our system to candidate terms generated by TBXTools .

3.1 Data Pre-processing

The HTML or PDF format of the file was converted to a plain text format. The text was converted completely to lowercase to remove duplicates that were based solely on case differences. Non-alphanumeric characters and "words" that were composed solely of numbers were excluded, since the index of each document contained keyterms that were composed solely of letters. A Python script was run on the Index file to remove the section names, to avoid inadvertently repeating

the keywords that were found based strictly on POS tag information. Extraneous material such as acknowledgments and example code snippets were removed from the document since otherwise, a search for POS tags of the form <*NN*> <*NN*> would find invalid terms.

To ensure consistent matching between terms as they appeared in the text of the document and terms in the index, index terms which used commas to alter the order of words were pre-processed by having their commas removed, and the order of words "flipped" so as to make them grammatically correct. For instance, *"array elements, accessing"* was converted to *"accessing array elements"*.

3.2 POS Tagging

Part of speech tagging was performed using the default Maximum Entropy Penn Treebank POS tagger of the NLTK toolkit. POS tagging uses POS tag sequences as a "within collection" (as opposed to external) syntactic feature, in order to identify candidate keyphrases.

Sequences of POS tags in the index were used to find the most appropriate expressions to use as filters on the body of the C reference manual document.

While the frequency of nouns appeared prominent among the index terms e.g. *arithmetic operators, operator precedence*, verbs came a close second e.g. *accessing array elements, initializing arrays*.

3.3 Candidate Detection

Candidate words were selected based on POS patterns. The goal of this step was to avoid "incorrect" keyphrases while using pruning to create the smallest possible number of candidates [11]. This pruning is necessary because reducing the number of candidates increases the value of the precision measure.

3.4 Stop words and Common words

The text was compared to a list of common words and stopwords and any terms from the text that also appeared in the list of stopwords were excluded. Frantzi et al. [9] used a method of selecting high frequency words from a sample of their corpus as stop words. A large number of words in the document were in fact stopwords.

Five different types of these stopwords and common words were used. First, we used a basic list of stopwords provided by NLTK, enhanced by words of occurrence frequency of less than three from the text. Second, a list of stopwords from the Brown corpus *editorial* category was used. Next a set of common words that were prevalent in the technical domain was manually created by inspection. In addition a list of words that could be considered stopwords but were actually valid key terms was also created. This Operators list was made of common words that could be considered stopwords in a generic domain, but were in fact specialized terms and therefore valid keywords in the technical domain. Also the least frequent words in a particular document were used as another stop word list. Occurrence frequencies of three and ten were used as thresholds for the least frequent words list. Stop words were extracted after candidate terms were found, so as not to prematurely exclude valid key terms.

Table 1: Experimental Results

Exp.	Precision	Recall	F-Measure
ATE C data	0.1969	0.3596	0.2548
TBXTools C data	0.0668	0.2905	0.1086
ATE Aircraft	0.0275	0.8485	0.0533
TBXTools Aircraft	0.0069	0.2675	0.0134

4 EXPERIMENTS

The observed performance is shown in Table 1. "C data" indicates the C language document, "Aircraft" indicates the aircraft maintenance guidelines manual, and "TBXTools" describes the values when the TBXTools software [20] was used on the C language document and the aircraft maintenance guidelines manual. "ATE" indicates our system.

5 DISCUSSION

Evaluation metrics for ATE were laid out by Kim at al. [17] and Hasan et al. [11]. This approach involved mapping the keyphrases in the "gold standard" document (i.e. the index file) to the key phrases that were output by the system, using an exact match. The mapping was then scored using precision, recall and F-measure values. The "gold standard" is a prebuilt list of reference terms that provides "reproducibility of results, tunability of parameters, and comparison between different methods on one dataset" according to Astrakhansev [2]. Using the Odds ratio test we obtained a score of greater than one, in favor of our system when compared to TBXTools statistical.

All keyphrases that were generated were used in our calculation of performance measures, unlike other research experiments that placed a cutoff on the number of generated keyphrases [8, 22].

The use of more relaxed POS sequences e.g. <N.> <N.>* resulted in more candidates being identified, but only a small number of these candidates were actual index terms. Using more restrictive POS sequences e.g. <NN> <NN> provided better matches between the text and index terms. Precision and recall were improved by removing code snippets and pre-processing the data.

The default POS tagger in the NLTK software was lacking in accuracy when it came to certain terms. Also, there was a discrepancy between tags in the text file and in the index file due to ambiguity of word usage in various sentences as opposed to the index file. It should be noted that the text file consisted of whole paragraphs while the index file comprised of index terms that were listed one term per line.

State of the art performance in ATE is still low. Hasan et al. [11] explain that state of the art values for Precision in general vary between 0.27 and 0.35, Recall varies between 0.28 and 0.66, while F-measure falls between 0.27 and 0.45.

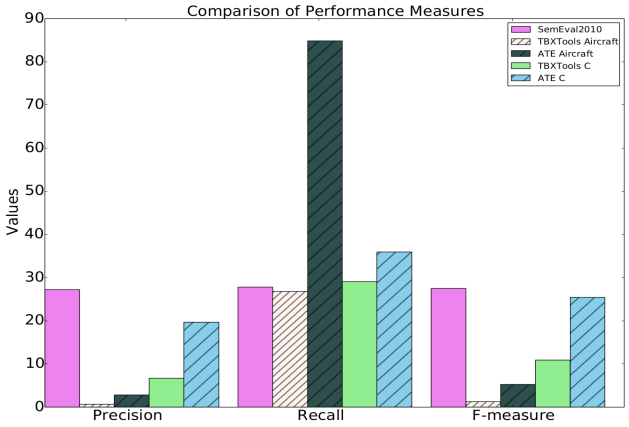

Figure 2: Performance Measures Comparison

For performance measures on scientific papers in particular (SemEval 2010), which corresponds most closely to our dataset, the corresponding values are 0.27, 0.28 and 0.28 respectively. Our results are comparable to the previously mentioned state of the art performance values as indicated by Figure 2.

Variant terms were included in our candidate detection as otherwise, as Bougouin et al state, "variations in the extracted keyphrases that might be judged as correct cannot be taken into account" [4].

As expected, precision varies inversely with recall. This trade-off becomes inevitable as returning more candidates improves recall but lowers precision, while pruning the number of candidate terms improves precision while lowering recall.

6 CONCLUSION

We have evaluated and demonstrated a method for the automatic creation of a list of the most relevant keyterms from examples of technical documents. The results compare favorably to the state of the art, although the state of the art performance itself is still relatively low in terms of precision and recall. While most indexers tend to focus mostly on noun phrases, we have endeavored to include verb phrases as well, since it has been noted that index terms are more complex than mere sequences of nouns [16]. In our calculation of performance measures, we used all the keyphrases that were generated, unlike experiments that placed a cutoff on the number of generated keyphrases.

Future work will include the addition of external features e.g. data from Wikipedia article titles or other sources, in order to provide more domain-specific background information. An alternate POS tagger will be used to find POS sequences, and the results will be compared with that of the default POS tagger provided by NLTK, in order to determine if this new tagger increases the accuracy of POS tagging.

REFERENCES

[1] Nikita Astrakhantsev. 2016. ATR4S: Toolkit with State-of-the-Art Automatic Terms Recognition methods in Scala. *Language Resources and Evaluation* (2016), 1–20.

[2] NA Astrakhantsev, Denis G Fedorenko, and D Yu Turdakov. 2015. Methods for Automatic Term Recognition in Domain-Specific Text Collections: A Survey. *Programming and Computer Software* 41, 6 (2015), 336–349.

[3] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, Inc.

[4] Adrien Bougouin, Sabine Barreaux, Laurent Romary, Florian Boudin, and Béatrice Daille. 2016. Termith-eval: a French Standard-based Resource for Keyphrase Extraction Evaluation. In *Language Resources and Evaluation Conference (LREC)*.

[5] Damien Cram and Béatrice Daille. 2016. Termsuite: Terminology Extraction with Term Variant Detection. *ACL 2016* (2016), 13.

[6] Lyne Da Sylva and Frédéric Doll. 2005. A Document Browsing Tool: Using Lexical Classes to Convey Information. In *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 307–318.

[7] Alessio Ferrari, Felice dell'Orletta, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2014. Measuring and Improving the Completeness of Natural Language Requirements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 23–38.

[8] Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific Keyphrase Extraction. In *16th International Joint Conference on Artificial Intelligence (IJCAI 99)*, Vol. 2. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 668–673.

[9] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. 2000. Automatic Recognition of Multi-word terms:. the C-value/NC-value method. *International Journal on Digital Libraries* 3, 2 (2000), 115–130.

[10] Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, 365–373.

[11] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic Keyphrase Extraction: A Survey of the State of the Art. In *ACL (1)*. 1262–1273.

[12] Lindley R. Higgins. 1990. *Maintenance Engineering Handbook*. McGraw-Hill.

[13] Julia Hirschberg and Christopher D Manning. 2015. Advances in Natural Language Processing. *Science* 349, 6245 (2015), 261–266.

[14] Anette Hulth. 2003. Improved Automatic Keyword Extraction given more Linguistic Knowledge. In *Proceedings of the 2003 conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 216–223.

[15] John S Justeson and Slava M Katz. 1995. Technical Terminology: some Linguistic Properties and an Algorithm for Identification in text. *Natural Language Engineering* 1, 01 (1995), 9–27.

[16] Su Nam Kim and Min-Yen Kan. 2009. Re-examining Automatic Keyphrase Extraction Approaches in Scientific Articles. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*. Association for Computational Linguistics, 9–16.

[17] Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 21–26.

[18] Sandra Loosemore, Richard M. Stallman, Roland McGrath, Andrew Oram, and Ulrich Drepper. 1989. *The GNU C Library Reference Manual*. GNU.

[19] Christopher D Manning, Hinrich Schütze, et al. 1999. *Foundations of Statistical Natural Language Processing*. Vol. 999. MIT Press.

[20] Antoni Oliver and Mercè Vàzquez. 2015. TBXTools: a Free, Fast and Flexible Tool for Automatic Terminology Rxtraction. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*. 473–479.

[21] Sifatullah Siddiqi and Aditi Sharan. 2015. Keyword and Keyphrase Extraction Techniques: a Literature Review. *International Journal of Computer Applications* 109, 2 (2015).

[22] Peter D Turney. 2000. Learning Algorithms for Keyphrase Extraction. *Information retrieval* 2, 4 (2000), 303–336.