Technical Report for SF1 Keyphrases Extraction

Jia Guo

January 28, 2011

Document History		
Date	Author	Content
2011-01-26	Jia Guo	Initial version, extract kpe part from the TR of taxonomy generation.
2011-01-28	Jia Guo	Describe the algorithms for keyphrase extraction and named entities recognization by support vector machine.

Abstract

Automatic keyphrase extraction from documents is a task with many applications in information retrieval and natural language processing. Previously, Several keyphrase extraction methods have been proposed based on different techniques. In this paper a keyphrase extraction algorithm based on the one-class support vector machine is proposed. In order to determine whether a phrase is a keyphrase or not, the following features of a phrase in a given collection are adopted: context terms including prefix and suffix in whole collection, the first two and last two terms of this phrase. We extract the keyphrases in collection level, so once a keyphrase found, it will be identitied in all documents it appears compare to the tf*idf approach. *TODO evaluation description*

Contents

1 Introduction

Keyphrases provide a brief summary of a documents contents. As large document collections such as digital libraries and enterprise information center become widespread, the value of such summary information increases. Keyphrases are particularly useful because they can be interpreted individually and independently of each other. They can be used in information retrieval systems as descriptions of the documents returned by a query, as the basis for search indexes, as a way of browsing a collection, and as a document clustering technique. In addition, keyphrases can help users get a feel for the content of a collection, provide sensible entry points into it, show how queries can be extended, facilitate document skimming by visually emphasizing important phrases; and offer a powerful means of measuring document similarity.

Keyphrases are usually chosen manually. In many academic contexts, authors assign keyphrases to documents they have written. Professional indexers often choose phrases from a predefined controlled vocabulary relevant to the domain at hand. However, the great majority of documents come without keyphrases, and assigning them manually is a tedious process that requires knowledge of the subject matter. Automatic extraction techniques are potentially of great benefit. There are two fundamentally different approaches to the problem of automatically generating keyphrases for a document: keyphrase assignment and keyphrase extraction. Both use machine learning methods, and require for training purposes a set of documents with keyphrases already attached. Keyphrase assignment seeks to select the phrases from a controlled vocabulary that best describe a document. The training data associates a set of documents with each phrase in the vocabulary, and builds a classifier for each phrase. A new document is processed by each classifier, and assigned the keyphrase of any model that classifies it positively. The only keyphrases that can be assigned are ones that have already been seen in the training data.

Keyphrase extraction, the approach used here, does not use a controlled vocabulary, but instead chooses keyphrases from a collection of documents itself. It also does not employs any lexical techniques in baseline approach.

This paper describes the keyphrase extraction algorithm which will be used in our enterprise search engine sf1-r: Kpe. It uses the one-class support vector machine to identity positive instances(keyphrases) from negative ones(non-keyphrases) by training on *Wikipedia* corpus. Then Kpe uses SVM classifier to categorizing named entities including people names, locations and orgnizations.

In SF1-R, keyphrases extraction and indexing lay the basis for many data mining features such as search result clustering, taxonomy generation, query recommendation, query autofill, document similarity measuring etc. Keyphrase extraction could be differentiated as document level and collection level ones. The former usually identifies keyphrases using single document based statistics and contextual features around the phrase while the latter extracts keyphrases that is significant in the collection level based on collection level statistics. SF1-R prefers a collection level keyphrase extraction because of the following reasons.

- In enterprise search engine, users seek the needed information against a document collection.
- Keyphrase is used in query recommendation, query auto-completion and other data mining algorithms which surely should be statistical significant

in the collection level.

Specifically, the task of keyphrase extraction in SF1-R could be stated as: Given a collection of documents $C = d_1, d_2, ..., d_n$, extract the set of keyphrases $L = l_1, l_2, ..., l_m$ from C that best captures the content and topics in C.

In this paper:

1. We make an approach to extract correct keyphrases including named entities from Wikipedia in multi-language for training purpose.
2. We apply the one-class SVM to identity keyphrases in large collections.
3. We categorize extracted keyphrases to named entities.

2 Background and Related Work

The state-of-the-art system for keyphrases extraction is KEA [8]. It uses Naïve Bayes classifier and few heuristics. The best results reported by KEA team show about 18% of Precision in extracting keyphrases from generic web pages. Use of domain specific vocabularies may improve the result up to 28.3% for Recall and 26.1% for Precision.

Turney suggested another approach which uses GenEx algorithm [5]. The GenEx algorithm is based on a combination of parameterized rules and genetic algorithms. The approach provides nearly the same precision and recall as KEA. In a more recent work [6], the author applies web-querying techniques to get additional information from the Web as background knowledge to improve the results. This method has a disadvantage: mining the Web for information and parsing the responses is a time and resource consuming operation. This is problematic for Digital Libraries with millions of documents. In this approach the author measures the results by the ratio of average number of correctly found phrases vs. total number of extracted phrases.

Recent works by A. Hulth et al. considered domain [5] and linguistic [3] knowledge to search for relevant keyphrases. In particular, [5] used a thesaurus to capture domain knowledge. But the recall value reported in this work is very low, namely 4-6%. The approach proposed in [3] introduced a heuristic related to part-of-speech usage, and proposed training based on the three standard *Kea* features plus one linguistic feature. Authors reported fairly good results (F-Measure up to 33.9%). However, it is hard to compare their results with others because of the strong specificity of the used data set: short abstract with on average 120 tokens where around 10% of all words in the proposed set were keyphrases.

A recent interesting work about the application of linguistic knowledge to the specific problem is reported in [2]. The authors used WordNet and ““lexical chains““ based on synonyms and antonyms. Then they applied decision trees as an ML part on about 50 journal articles as the training set and 25 documents as the testing set. They reported high precision, up to 45%, but did not mention recall. This makes difficult any comparison with other techniques. Other ML technique, least square SVM [7] shows 21.0% precision and 23.7% recall in the analysis of web mined scientific papers. Also in this case the described experiments are limited to a very small testing dataset of 40 manually collected papers.

Our work contributes along four dimensions: 1) using the large high quality multi-language corpus *Wikipedia* as the training set. 2) propose new context based features for ML approach. 3) no linguistic knowledge needed, easier to expand to more languages. 4) support named entities recognization by the advantage of *Wikipedia* corpus, using SVM classifier.

3 Wikipedia Dataset

The online encyclopedia Wikipedia is tantamount to a huge controlled vocabulary whose structure and features resemble those of thesauri, which are commonly used as indexing vocabularies[4]. [1] introduces a method to do the named entity identification and disambiguation. The author introduce a simple way to extract the (entity, category) pairs from Wikipedia pages with some basic filtering to discard meta-categories in English version. Versions in other languages like Chinese and Korean, could be targeted in a similar manner.

All titles in Wikipedia entity pages are candidates keyphrases. This is obvious that each Wikipedia entity page describes one meaningful thing with the representation of its title.

```
After graduation, she went to [[New York City]] and
played Ashley Linden on [[Texas (TV series)|Texas]]
from [[1981]] to [[1982]].
```

In Wikitext, the references to other Wikipedia articles are within pairs of double square brackets. If a reference contains a vertical bar then the text at the left of the bar is the name of the referred article (e.g. Texas (TV Series)), while the text at the right of the bar (e.g., Texas) is the surface form that is displayed (also referred to as the anchor text of the link). Otherwise, the surface form shown in the text is identical to the title of the Wikipedia article referred (e.g., New York City). We also can extract context terms of these references(titles) from Wikitext. For "New York City", the left context in this short snippet is "to" while the right one is "and".

4 Previous Algorithm

We applied a rule-based algorithm to identify keyphrases from all candidate n-grams in previous version. We used some context based features which is extracted from Wikipedia dataset. Our rule-based algorithm contains following features, we indicate the candidate n-gram as $\overline{ABCD}$, and $x = \overline{ABC}$, $y = \overline{BCD}$, $xy = \overline{ABCD}$:

1. mutual information in new documents collection which we want to extract keyphrases from:

$$MI(xy) = \frac{f(xy)}{f(x) * f(y)} \tag{1}$$

2. log-likelihood ratio:

$$\begin{aligned} loglikelihood(xy) = & ll(\frac{k_1}{n_1}, k_1, n_1) \\ & + ll(\frac{k_2}{n_2}, k_2, n_2) \\ & - ll(\frac{k_1+k_2}{n_1+n_2}, k_1, n_1) \\ & - ll(\frac{k_1+k_2}{n_1+n_2}, k_2, n_2) \end{aligned} \tag{2}$$

where $k_1 = f(x, y)$, $n_1 = f(x, *)$, $k_2 = f(\neg x, y)$, $n_2 = f(\neg x, *)$ and

$$ll(p, k, n) = klog(p) + (n - k)log(1 - p) \tag{3}$$

3. biased contextual entropy

4.1 Biased Contextual Entropy

Candidate phrases with high score of mutual information and loglikelihood doesnt necessarilly mean the keyness of those candidates. There are two main cases of false positives. First, some phrases are incomplete. For example, chain monte is not a keyphrase because it is only part of a complete phrase of Markov Chain Monte Carlo. Second, some phrases are very common phrases across domains, for example every day. To the first issue, the weighted contextual entropy is used to measure whether a candidate is a complete phrase or not. The basic idea is that if a candidate is a complete lexical pattern, then its adjacent words on left and right will have greater variety then those incomplete lexical patterns. For example, for the incomplete candidate chain monte, the possible left adjacent words occur in the collection maybe only Monte, while there maybe a lot of different words appearing in the left position of the complete lexical pattern Markov chain monte carlo. The contextual entropy is defined as follows:

$$CE(x) = -\sum_{t \in C} \frac{f(t)}{TF} log \frac{f(t)}{TF} \tag{4}$$

where C is the set of contextual terms of the candidate x. TF is the overall context occurrence frequency. CE indicates the variety of the contextual terms associated with the candidate phrase. However, as we stated earlier, there is a great variety of words that could possibly occur in the context. Also, besides content words, there are also possible functional delimiters that appears in the left or right contextual positions of the candidate. Different groups of words have different weight for the contribution of the contextual entropy. For example, the occurrence of different numbers like one, two maybe not as strong as the different occurrence of other content words for indicating the completeness of a lexical pattern. So, special bias or weighting should be considered for those delimiters which may indicate good or bad candidates in different cases. Motivated by the above observation, we define a biased contextual entropy:

$$BCE(x) = -\sum_{t \in C} \frac{w_g * f(t)}{TF} log \frac{f(t)}{TF} \tag{5}$$

where g represents the term group occurred in the context while w_g is the weight for the term group. Currently, in the implementation, we differentiate several term groups like number, different functional words as propositions etc and weight such groups in an empirical way. Actually, w_g can be estimated by Wikitext as we as we had discussed in section 3, each keyphrase in Wikitext can be extracted with its left and right context. For a extracted keyphrase $L|ABCD|R$ where L is the left context while R is the right one in Wikitext, the probability of w_g for left context could be calculated as

$$w_g(left) = Prob_{left}(L) * Prob_{left*}(LA) * Prob_u(A) * Prob_b(AB) \quad (6)$$

$w_g(right)$ will be calculated in a similar manner with $CD|R$. The $Prob$ can be made by the maximum likelihood estimator on Wikipedia corpus.

5 One-Class SVM

In previous algorithm 4, the dicision whether one candidate n-gram is a keyphrases made by rules of empirical value on MI, log-likelihood and BCE formula. This section will introduce a machine learning approach to study the "rules" automatically from positive examples.

Traditional machine learning approaches like SVM are based on training using both positive and negative examples, as the basic SVM paradigm suggests. We have been interested, however, in information retrieval using only positive examples for training. This is important in many applications. Consider, for example, trying to classify sites of "interest" to a web surfer where the only information available is the history of the users activities. One can envisage identifying typical positive examples by such tracking, but it would be hard to identify representative negative examples. Of course, the absence of negative information entails a price, and one should not expect as good results as when they are available.

As we had described above, Wikipedia titles with their references could be treated as keyphrases, they're positive examples. However, negative examples are hard to get. There're two reasons, 1) some candidate keyphrases are not tagged in wikitext, editors may set the reference for entities appears at the first time of articles while leave it as a normal text in following ones. 2) it is a huge number of non-keyphrases n-grams which is hard to cover all of them in machine learning approach.

By the reasons listed above, we decide to use one-class support vector machine to identity keyphrases on new documents by the positive examples from Wikipedia dataset.

5.1 Features Selection

To apply the one-class SVM in 5. We should extract some features for positive example keyphrases in Wikipedia dataset for training, and extract the same ones for n-grams in new documents for prediction. The features used in BCE model can be used here. Each items in BCE formula can be thought as a single feature 4.1 with the common statistical mutual information and log-likelihood ratio.

1. $Prob_{left}(L)$
2. $Prob_{left*}(LA)$
3. $Prob_{u}(A)$
4. $Prob_{b}(AB)$
5. mutual information
6. log-likelihood

//TODO more specific features

5.2 Parameters chose

//TODO one-class SVM parameters

6 Named Entities Recognization on extracted Keyphrases

After one keyphrase extracted, we may also need to know if it is a named entity or not. SVM classifier is a good choice for categorization problem esspecially on text processing. Section 5 has already introduced an SVM approach to identify keyphrases from documents. However, one-class SVM can not handle multi-class categorization problem and those features can not be used totally to do the NER tasks. We can take all keyphrases in Wikipedia dataset as the training set, test set is keyphases extracted and use the traditional text categorization features with context support as following ones, for keyphrase $\overline{ABCD}$:

1. all left context appears in collection for $\overline{ABCD}$, $List_of_Left$.
2. all right context appears in collection for $\overline{ABCD}$, $List_of_Right$.
3. the prefix two terms $\overline{AB}$
4. the suffix two terms $\overline{CD}$

All these features can be easily got in Wikipedia training set as we had already mentioned. Then we predict the keyphrases' category of named entities by C-SVM algorithm.

References

[1] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL*, volume 2007, pages 708–716, 2007.

[2] G. Ercan and I. Cicekli. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714, 2007.

[3] A. Hulth, J. Karlgren, A. Jonsson, H. Bostrom, and L. Asker. Automatic keyword extraction using domain knowledge. *Computational Linguistics and Intelligent Text Processing*, pages 472–482, 2010.

[4] O. Medelyan, I. Witten, and D. Milne. Topic indexing with Wikipedia. In *Proceedings of the AAAI WikiAI workshop*, 2008.

[5] P. Turney. Learning to extract keyphrases from text. *Arxiv preprint cs/0212013*, 2002.

[6] P. Turney. Mining the web for lexical knowledge to improve keyphrase extraction: Learning from labeled and unlabeled data. *Arxiv preprint cs/0212011*, 2002.

[7] J. Wang and H. Peng. Keyphrases extraction from web document by the least squares support vector machine. In *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 293–296. IEEE, 2005.

[8] I. Witten, G. Paynter, E. Frank, C. Gutwin, and C. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.