A Ranking Approach to Keyphrase Extraction

Xin Jiang[a], Yunhua Hu[b], Hang Li[b]

[a] *School of Mathematical Sciences, Peking University, No. 5 Summer Palace Street, Beijing, China 100871, jiangxin@pku.edu.cn*
[b] *Microsoft Research Asia, No. 49 Zhichun Road, Haidian District, Beijing, China 100190, {yuhu,hangli}@microsoft.com*

Abstract

This paper addresses the issue of automatically extracting keyphrases from document. Previously, this problem was formalized as classification and learning methods for classification were utilized. This paper points out that it is more essential to cast the keyphrase extraction problem as ranking and employ a learning to rank method to perform the task. As example, it employs Ranking SVM, a state-of-art method of learning to rank, in keyphrase extraction. Experiments conducted on three datasets show that Ranking SVM significantly outperforms the baseline methods of classification, indicating that it is better to exploit learning to rank techniques in keyphrase extraction.

Key words: Keyphrase extraction, Learning to rank, Ranking SVM

1. Introduction

In this paper we address automatic extraction of keyphrases from document. By keyphrases of a document we mean the words and phrases that can precisely and compactly represent the content of the document. Keyphrases are useful for various applications such as document summarization [5, 33], document retrieval [14, 20], document categorization and clustering [12]. In digital library the keyphrases of a scientific paper can help users to get a rough sense of the paper. In web search the keyphrases of a web page can serve as metadata for indexing and retrieving the web page [10, 16]. In Web 2.0 applications keyphrases can be used as candidates for people to tag web pages.

Many documents do not have manually assigned keyphrases. Ideally, we would use a tool to automatically assign keyphrases to the documents. In general, keyphrases of a document do not necessarily appear in the document. In this paper, for simplicity we only consider the cases in which we identify keyphrases from the body of the document. Keyphrase extraction consists of two steps: candidate phrase identification and keyphrase selection.

Much work was conducted on keyphrase extraction [1, 8, 11, 15, 16, 22, 25, 26, 27, 28, 29, 30, 31], with many of them leveraging machine learning techniques. The problem was basically formalized as classification in which a classifier was trained and used to categorize phrases as *keyphrases* or *non-keyphrases*. Documents as well as their keyphrases assigned by authors or annotators were utilized as training data. Classification methods such as decision tree and Naive Bayes were employed.

Recently, machine learning technologies for ranking called 'learning to rank' have been intensively studied (e.g., [2, 3, 4, 9, 13, 18, 32]). Different from classification and regression, the

goal of learning to rank is to learn a function that can rank objects according to their degree of preferences, importance, or relevance defined in an application. So far, learning to rank methods were mainly applied to document retrieval. To the best of our knowledge, it was not applied to keyphrase extraction.

We point out that keyphrase extraction is by nature a ranking problem rather than a classification problem, and it is better to employ a learning to rank method for keyphrase extraction than a classification method. The reasons are as follows. First, it is more natural to consider the likelihood of a phrase's being a keyphrase in a relative sense than in an absolute sense. Second, by employing the ranking approach we can avoid the difficulty of making hard decisions on keyphrases or non-keyphrases, which plagues the classification approach. Third, information (features) for determining whether a phrase is a keyphrase is also relative.

In this paper, we employ Ranking SVM [13] as our learning to rank method to perform keyphrase extraction. Specifically, Ranking SVM takes ranked phrase pairs as examples, indicating which phrases are more likely to be keyphrases than other phrases. It creates a linear ranking model by means of SVM, uses the linear model for ranking the candidate phrases of a new document, and selects the top ranked phrases as keyphrases.

We conducted experiments to verify the effectiveness of our approach with three datasets. The first dataset is composed of research papers and their keyphrases assigned by the authors. The second dataset consists of web pages and their associated tags provided by internet users. The third dataset is a set of web pages in the TREC .Gov corpus and their keyphrases labeled by human annotators. Experimental results show that Ranking SVM statistically significantly outperforms the baseline classification methods of Naive Bayes and SVM on all the three datasets. The experimental results strongly indicate that it is better to employ a ranking approach to keyphrase extraction than the conventional classification approach.

Although the idea of using ranking for keyphrase extraction is simple, we know of no existing work which studied the problem. We think, therefore, that our work reported in this paper represents an enhancement on the state-of-the-art for this important topic.

The rest of the paper is organized as follows. Section 2 introduces related work on keyphrase extraction and learning to rank. Section 3 describes our method of keyphrase extraction using Ranking SVM. Experimental results are reported in Section 4. Finally, Section 5 concludes this paper and discusses the future work.

2. Related Work

2.1. Keyphrase Extraction

Many methods have been proposed for keyphrase extraction. Most of them are based on machine learning techniques.

Turney [27] proposed viewing keyphrase extraction as classification. In this approach, phrases are extracted from documents and are labeled as keyphrases or non-keyphrases. The documents and labeled phrases are then used as training data for creating a keyphrase classifier. Two learning methods are applied: the decision tree learning algorithm of C4.5 [24] and a genetic algorithm called GenEx. Features such as phrase frequency, position in document are utilized in the classifiers. The classifiers are then used to categorize phrases of a new document as keyphrases or non-keyphrases. Experimental results show that GenEx can achieve better performance than C4.5.

Kea is a tool for keyphrase extraction based on Naive Bayes [8, 30]. In one version of Kea [30], only two features are used: TF-IDF (term frequency-inverse document frequency),

and position of the first occurrence. The numerical values of the features are discretized and used to build the Naive Bayes model. In extraction, candidate phrases are ranked according to their probabilities of being keyphrases, and top-ranked phrases are treated as keyphrases. Experimental results show that Kea can achieve a performance comparable to GenEx. Frank et al. [8] extended the Kea model by adding another feature called keyphrase-frequency, which is the frequency of a phrase's being keyphrase in all the documents in the corpus. This feature is effective in domain-specific keyphrase extraction. Turney [28] further improved the Kea model by replacing this domain-specific feature with a number of new features based on co-occurrence measures.

Hulth [15, 16] tried three approaches to candidate phrase identification and employed a rule induction system to classify the candidate phrases. Wang et al. [29] used neural network and the back propagation algorithm in keyphrase extraction. For other related work, see [1, 11, 22, 25, 31]

To evaluate the accuracy of the keyphrase extraction methods, measures such as precision and recall are used. Human annotated keyphrases are usually used as positive examples. Turney [26] and Jones and Paynter [19] also proposed ways of human evaluation on the keyphrases extracted by GenEx and Kea.

2.2. Learning to Rank

Ranking is the central problem for many information retrieval applications, such as document retrieval and collaborative filtering. Recently a new research area is emerging in machine learning, which is called learning to rank. Learning to rank aims at automatically creating a model (function) that can perform ranking on instances, using training data and machine learning techniques. Many learning to rank methods have been developed and applied to information retrieval.

There is one typical approach to learning to rank, referred to as the pairwise approach. This approach formalizes the ranking problem as that of classifying instance pairs into two categories: *correctly ranked* and *incorrectly ranked*. Herbrich et al. [13] proposed a pairwise method called Ranking SVM. Ranking SVM employs Support Vector Machines (SVM) in classifying the instance pairs. Joachims [18] applied Ranking SVM to document retrieval, where documents pairs are generated from click-through data. Cao et al. [3] adapted Ranking SVM to document retrieval by modifying the hinge loss function of SVM to better meet the requirements of IR. Other pairwise methods include RankBoost [9], RankNet [2], etc. In this paper, we take Ranking SVM as an example method for ranking.

3. Ranking SVM for Keyphrase Extraction

In this section, we first describe the motivation on employing a learning to rank method in keyphrase extraction. Then we introduce our method of using Ranking SVM. Finally, we list the features used in the Ranking SVM model.

3.1. Motivation

Keyphrase extraction has been previously formalized as classification. Although this approach has certain advantages such as theoretical soundness, it still has some drawbacks. It is more essential to formalize the problem as ranking and employ a learning to rank method to carry out the task. There are three major reasons.

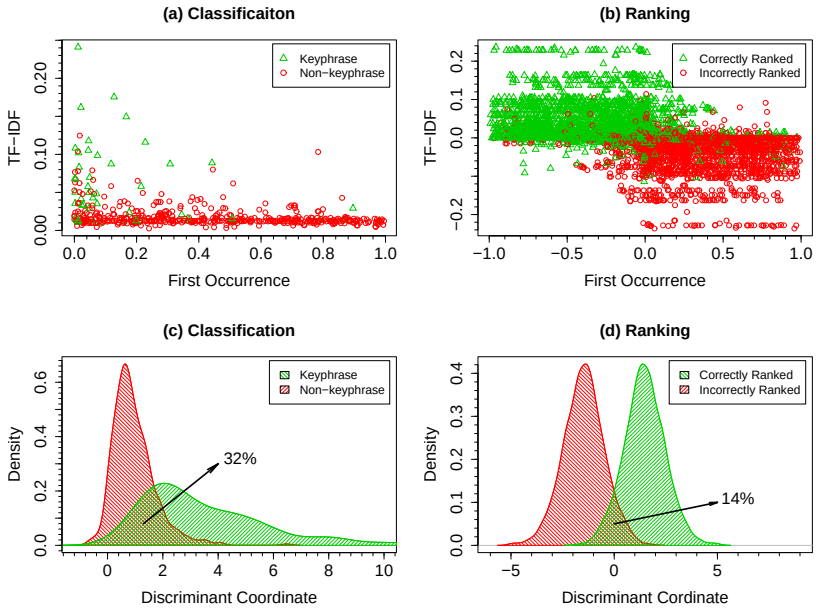

Figure 1: (a) and (b) show the scatterplots of instances over two features in the case of classification and ranking, respectively. (c) and (d) show the distribution of the data projected on the directions found by linear discriminant analysis. All the data come from ten documents with human-labeled keyphrases randomly selected from TREC .Gov dataset.

First, keyphrases can only be identified in a relative sense. That is to say, it is usually easy to determine whether a phrase is more like a keyphrase than the other phrase than to determine whether a phrase is a keyphrase or not. This is even true for humans.

Second, the setting of classification does not fit well with keyphrase extraction. Classification methods need to make hard decisions in keyphrase extraction. However, this is difficult because there are always many boundary cases in the task (even for humans). Moreover, the number of keyphrases is usually much smaller than the number of non-keyphrases, as pointed out by Turney [27] and shown in Figure 1. The highly skewed sample distribution makes the learning hard. For instance, a classifier of predicting all the instances to be non-keyphrases can still have a high classification accuracy. To conquer this problem, techniques such as sampling from the training data or modifying the loss function have been tried. However, they do not seem to form fundamental solutions.

In contrast, learning to rank methods can more naturally tackle the problem. When applied to keyphrase extraction, a learning to rank method tries to learn and utilize a ranking function that sorts phrases based on their degree of being keyphrases. In other words, it directly treats keyphrases in a relative sense in the framework.

Third, the information (features) that helps make keyphrase or non-keyphrase decisions is relative to documents. Classification methods use absolute feature values, while (pairwise) ranking methods use differences of feature values. Difference in feature values is a kind of normalization and is more meaningful across documents. For example, it is hard to say how likely a phrase is a keyphrase based on the absolute value of its TF-IDF feature. In contrast, it is safer to say that if the TF-IDF value of a phrase is larger than that of the other phrase, then the former is more

likely to be a keyphrase than the latter. As a result, the features in a ranking method have more discriminative power for keyphrase extraction than the features in a classification method.

In this paper, we employ the linear Ranking SVM method. As baseline, we also use the linear SVM method. The two methods actually use the same set of features, but different feature values (the former absolute values, the latter differences between values). To test the discriminative abilities of the feature values, we conducted analysis on some randomly selected data. Figure 1 (a)(b) shows the plots of the instances in two feature space (*First occurrence* and *TF-IDF*) for the classification and ranking data. (For the ranking data, 'correctly ranked' is viewed as positive class, and 'incorrectly ranked' is viewed as negative class. See Section 3.2 for details.) From the figures, we can see that the distribution patterns of the data are very different. It would be easier for the ranking approach to separate the positive and negative classes. Note that there are far more instances in ranking than in classification, due to its use of instance pairs.

We also conducted a linear discriminant analysis on the data using all the features. Linear discriminant analysis provides a way of presenting data in a lower dimension space [6]. Specifically for binary case, it gives a direction on which the projected data has the maximal between-class variance as well as the minimal within-class variance, i.e., the direction that optimizes the class separability. Figure 1 (c)(d) shows the distributions of data projected on the discriminant directions for classification and ranking. The overlap between the densities of classes is greater in classification than in ranking (the estimated overlapped areas for classification and ranking are 32% and 14% respectively), which means the instances are more separable in the ranking case than in the classification case. The reason seems to be that the ranking approach implicitly conducts normalization on feature values and makes the feature values more discriminative. This strongly indicates that ranking is more likely to achieve better performance than classification in keyphrase extraction. (As will be seen later, this is true).

3.2. Method

The first step of keyphrase extraction is to identify candidate phrases from the document. We use the existing NLP technologies to extract all the base noun phrases in the document as candidate phrases. We then conduct stemming (using Poter's stemmer [23]) and normalization on the candidate phrases. Next, we create a feature vector for each candidate phrase.

In training, human judges label candidate phrases as keyphrases or non-keyphrases for each of the training documents. We train a keyphrase extraction model by using the labeled training data (feature vectors of phrases) and machine learning techniques. In extraction, given a new document and its associated candidate phrases we use the trained model to select keyphrases from them.

Traditionally, a classifier is employed and the phrases classified as positive are selected as keyphrases, as explained above. In this paper, we instead employ a ranking model, specifically, Ranking SVM, to sort the candidate phrases and to select the top ranked phrases as keyphrases.

Let $X \subseteq \mathbb{R}^p$ denote the input feature space, and $Y = \{r_1, r_2, \ldots, r_m\}$ denote the output rank space, where p is number of features, $\{r_1, r_2, \ldots, r_m\}$ is a set of ranks, and m is number of ranks. In this paper, usually $m = 2$. There exists a total order among the ranks: $r_m \succ r_{m-1} \succ \ldots \succ r_1$, where $\succ$ denotes preference relation. Let x and y denote elements of X and Y respectively. In this paper, we assume to use a linear ranking function f:

$$f(x) = \langle \omega, x \rangle \tag{1}$$

where ω is vector of coefficients and $\langle \cdot, \cdot \rangle$ denotes inner product.

In keyphrase selection, the ranking function f determines the preference relations between feature vectors of phrases in a given document

$$f(x_i) > f(x_j) \Leftrightarrow \langle \omega, x_i - x_j \rangle > 0 \Leftrightarrow y_i \succ y_j \tag{2}$$

We then select the top k ranked phrases as the keyphrases of the documents.

Suppose that we have a collection of M documents $\mathcal{D} = \{d_1, d_2, \ldots, d_M\}$, and each document d_k has N_k candidate phrases. The training data is then given by $\{(x_{k,i}, y_{k,i}) \mid k = 1, 2, \ldots, M; i = 1, 2, \ldots, N_k\}$, where $x_{k,i}$ denotes the feature vector of the ith candidate phrase of the kth document and $y_{k,i}$ denotes the corresponding rank. The goal of learning is to train a ranking function f^* that can best rank the phrases in a new document.

In Ranking SVM, we convert the training data of labeled phrases into that of ordered phrase pairs, while preserving their preference relations:

$$\mathcal{T}' = \{(x_{k,i} - x_{k,j}, z_{k,ij}) \mid k = 1, 2, \ldots, M; i, j = 1, 2, \ldots, N_k\}$$

where $(x_{k,i} - x_{k,j})$ denotes difference between feature vectors of ith and jth candidate phrases in kth document, and $z_{k,ij}$ denotes their preference relationships: $z_{k,ij} = +1$ if $y_{k,i} \succ y_{k,j}$ and $z_{k,ij} = -1$ if $y_{k,i} \prec y_{k,j}$. In the simplest case, there are only two ranks (keyphrases and non-keyphrases) and we create a pair from each of the ranks. Next, we train a SVM model on the training set $\mathcal{T}'$, which can be formalized as the following optimization problem:

$$\begin{aligned} &\min_{\omega, \xi_{k,ij}} \frac{1}{2}\|\omega\|^2 + C \sum_{i,j,k} \xi_{k,ij} \\ &\text{s.t. } z_{k,ij}\langle \omega, x_{k,i} - x_{k,j} \rangle \geq 1 - \xi_{k,ij},\ \xi_{k,ij} \geq 0,\ \forall i, j, k \end{aligned} \tag{3}$$

where ω denotes weights of ranking function and $\xi_{k,ij}$ denotes slack variables, and C denotes parameter. Problem (3) is a Quadratic Programming problem and can be solved by the standard SVM techniques.

Suppose the optimal solution to (3) is obtained and denoted as ω^*, then the resulting ranking function becomes

$$f^*(x) = \langle \omega^*, x \rangle \tag{4}$$

$f^*(x)$ can then be used in keyphrase selection.

3.3. Features

We explain the features used in the Ranking SVM model. In this paper we try to use as many existing features as possible.

TF-IDF

TF (term frequency) represents number of times the phrase appears in the document. DF (document frequency) represents number of documents containing the phrase in the corpus. TF-IDF score is computed as

$$\text{TF-IDF} = \frac{TF}{N_d} \times \log \frac{N_c}{DF} \tag{5}$$

where N_d is number of words in document and N_c is number of documents in the corpus. In our experiments, we used TF, IDF, and TF-IDF as three features. Intuitively, the more frequently a phrase occurs the more likely the phrase is a keyphrase.

Phrase length

The feature is simply number of words contained in the phrase.

First occurrence

The feature is calculated as number of words that precede the phrase's first appearance in the document, divided by total number of words of the document. Intuitively, the earlier the phrase occurs in the document, the more likely the phrase is a keyphrase.

Phrase distribution

The feature is proposed in [34]. The intuition is that a keyphrase tends to be uniformly distributed in the document, and entropy can be used to measure this uniformity. Suppose a document is divided into n parts. Then the probability of a phrase appearing in the ith part is $p_i = \mathrm{TF}_i/\mathrm{TF}$, where TF_i is frequency of phrase in ith section and TF is frequency in whole document. Entropy of phrase is defined as

$$entropy = -\sum_{i=1}^{n} p_i \log p_i \tag{6}$$

In this paper, we set $n = 10$.

Is in title or not

The feature takes two values on whether the phrase occurs in the title of the document. Usually the phrases appearing in title are likely to be keyphrases.

Maximum and minimum word frequencies

The features represent the maximum and minimum frequencies of words in the phrase. This is because a keyphrase often contains high frequency words.

4. Experiments

We conducted experiments on keyphrase extraction using three datasets, and examined the performances of Ranking SVM as well as the two baseline classification methods: Kea (Naive Bayes) and SVM. For Kea we used the program that is publicly available[1]. The tool SVM^{light}[2] [17] was utilized for Ranking SVM and SVM. The default parameter setting in SVM^{light} was used, except that the cost factors in SVM were tuned to balance the class proportion (For boosting the performances of SVM this is very necessary).

In learning, each dataset was separated into training data and test data. The methods were trained with the training data and then were evaluated with the test data. Note that the ways of using Ranking SVM and the baseline methods are similar: a score is assigned to each candidate phrase, the phrases are sorted on the scores, and the top n phrases with highest scores are selected as keyphrases.

[1] `http://www.nzdl.org/Kea/`

[2] `http://svmlight.joachims.org/`

Table 1: Statistics on the datasets

Document type	Source	No. of documents	Ave. No. ± Dev.		Perc. of keyphrases	
			No. of words per document	No. of keyphrases per document	In text	In candidate phrases
Research Paper	Academic search	341	7,222±4,074	4.8±2.2	94.4%	86.9%
Web page	Social Tagging	600	3,813±5,872	17.9±6.6	64.8%	57.0%
Web page	.Gov	300	438±155	5.14±1.44	100%	100%

4.1. Evaluation Measures

The performances of Ranking SVM and the two baseline methods were evaluated in terms of the following measures.

Precision@n

Suppose that each candidate phrase has a binary label: positive or negative. For document d_k, there are N_k candidate phrases sorted by their scores in descending order. We denote the label of phrase at position i as $y_{k,(i)}$. Precision at n measures rate of positive examples at top n position:

$$P_k(n) = \frac{\sum_{i=1}^{n} y_{k,(i)}}{n} \tag{7}$$

where $y_{k,(i)}$ takes values of 0 and 1, representing negative or positive examples. For a collection of documents, we average $P_k(n)$ over all the documents.

Mean Average Precision (MAP)

First, average precision of document d_k is given by

$$AP_k = \frac{\sum_{i=1}^{N_k} y_{k,(i)} P_k(i)}{\sum_{i=1}^{N_k} y_{k,(i)}} \tag{8}$$

MAP is defined as AP_k averaged over all documents.

Kendall's Tau

Kendall's tau is a statistics to measure the degree of correspondence between two rankings:

$$\tau_k = \frac{2P}{\frac{1}{2}N_k(N_k-1)} - 1 \tag{9}$$

where P denotes number of concordant pairs in two rankings, and $\frac{1}{2}N_k(N_k-1)$ is total number of pairs. The value of τ ranges from -1 (perfect inversion) to $+1$ (perfect agreement).

4.2. Experiment with Research Paper Data

In this experiment, we randomly selected 341 research papers with author-given keyphrases from an academic search engine. Statistics on the dataset are shown in Table 1.

During the experiment, candidate phrases were extracted from the documents and their features were calculated as described in Section 3.3. The candidate phrases were assigned binary labels by matching with the author-given keyphrases. For Ranking SVM, pairs of candidate phrases as well as their preference relations were generated, such that each pair is composed of one keyphrases and one non-keyphrases, and the keyphrase is preferred to the non-keyphrase.

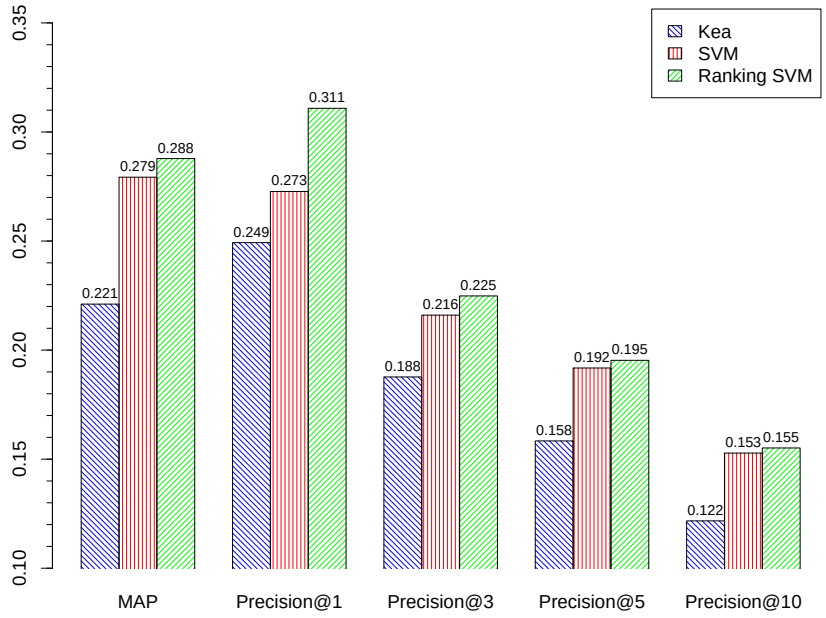

Figure 2: Keyphrase extraction accuracies on research paper data

For Kea the same dataset was used as SVM. Note that in the tool of Kea only two features are utilized.

We divided the documents into three even subsets and conducted a 3-fold cross-validation experiment. The experimental results averaged over three trials are reported in Figure 2.

From the figure, we can see that Ranking SVM outperforms SVM, and significantly outperforms Kea in terms of all measures. We conducted sign test. The results show that the improvements of Ranking SVM over Kea in terms of all measures are statistically significant (p-value ¡ 0.05). Furthermore, the improvement of Ranking SVM over SVM in terms of MAP is statistically significant. The improvements on Kea come from the fact that more effective features are used in Ranking SVM. We conducted an additional experiment and found that when the same features are used as in Kea, the performance of Ranking SVM (and also SVM) is still better. This demonstrates that the use of Support Vector Machine is better than the use of Naive Bayes in keyphrase extraction. Since Ranking SVM uses the same information as SVM does, the better performance of it indicates that the ranking approach is more powerful than the classification approach for the task.

4.3. Experiment with Social Tagging Data

In this experiment, data was collected from a social bookmark web site. We randomly selected URL's from the URL list provided at the site and crawled their source web pages. Each web page is accompanied by a list of tags (single words) added by users at the site. In addition, frequency information of tags is also available, which reflects the popularity of the tags.

We took advantage of the data to construct the training data and test data for keyphrase extraction. The web pages were transformed to plain texts. For the classification methods, the tagged

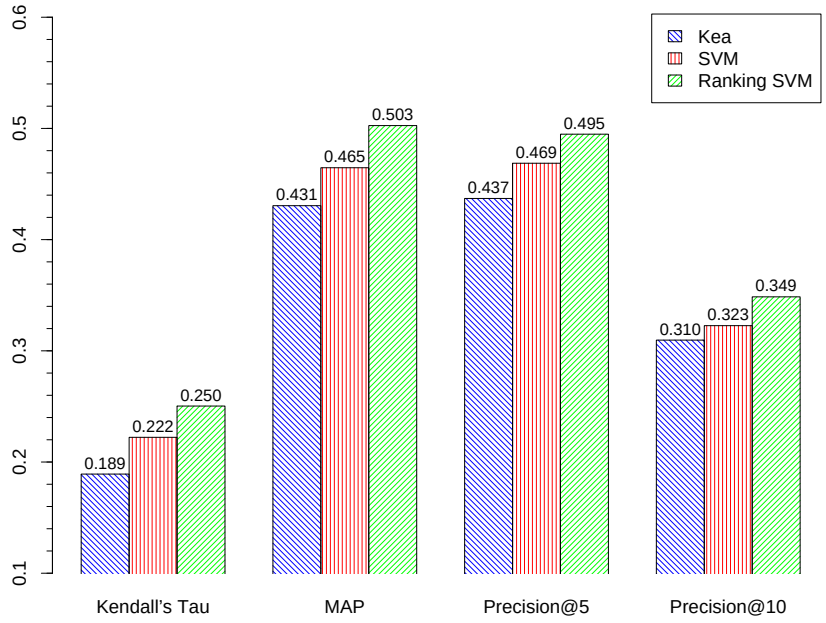

Figure 3: Keyphrase (tag) extraction accuracies on social tagging data

candidate words were treated as keywords and the non-tagged candidate words were treated as non-keywords. For the ranking method, preference relations between any two candidate words were produced by comparing their frequencies, such that candidate words with higher frequencies are preferred to candidate words with lower frequencies.

We used 600 web pages in this experiment. Statistics on the dataset are shown in Table 1.

A 3-fold cross validation experiment was conducted to test the performances of the three methods: Kea, SVM, and Ranking SVM. Experimental results are reported in Figure 3.

From the figure, we can see that Ranking SVM outperforms the baseline classification methods in terms of Kendall's Tau, MAP, and precisions. The differences are all statistically significant in sign test (p-value ¡ 0.05). We conclude that Ranking SVM makes proper use of the tagging information and achieves significant performance enhancement in keyphrase (tag) extraction.

4.4. Experiment with TREC .Gov Data

In this experiment, we randomly selected 300 web pages from the TREC .Gov dataset and asked human annotators to label the keyphrases in them. Statistics on the dataset are shown in Table 1.

The human-labeled data was used for both training and testing. Moreover, two annotation methods, referred to as pointwise method and pairwise method, were exploited. For the pointwise method, 50 candidate phrases were extracted from each document, and the annotators were asked to pick up keyphrases from the candidates (mark '1' for keyphrase and '0' for non-keyphrase). For the pairwise method, 250 pairs of phrases were extracted from each document, and the annotators were asked to specify the preference relation for each pair, i.e., identify which phrase is more like a keyphrase (mark '¿' as better, '¡' as worse, '=' as equal, and '?' as unknown). A

tool with graphical user interface was developed to facilitate the annotating work. Six annotators participated in the annotation task and separated into two groups. Each group annotated half of the documents using the pointwise method and the other half using the pairwise method. Each document in the data has six records: three pointwise annotations by one group and three pairwise annotations by the other group. (This was to make each annotator to label a document only once, either pointwise or pairwise, to reduce the possibilities of bringing bias into annotation).

We measured the level of agreements among the annotators. We used Fleiss' kappa, a commonly used measure, to asses the degree of agreement between the annotators [7]:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{10}$$

where $\bar{P}$ represents average rate of agreement, and $\bar{P}_e$ is degree of agreement achieved by chance. κ typically takes a value between 0 and 1, and $\kappa = 1$ means a complete agreement and $\kappa < 0$ means no agreement.

Table 2: Inter-annotator agreement measured by Fleiss' kappa statistics

	Pointwise method	Pairwise method
Group1	0.425	0.409
Group2	0.463	0.379

Table 2 shows the agreements for both two groups and two annotation methods. We can see that there exist certain agreements between the annotators for the two methods. We think that the agreements at this level are acceptable (cf., [21]). The pointwise method has slightly higher κ values than the pairwise method. Note that the results on the two methods are based on different data, and thus are not directly comparable.

In dataset creation, we combined the annotations by majority voting. For each instance (phrase or pair of phrases), we chose the label given by at least two annotators. The other instances were discarded. Then we trained SVM and Ranking SVM using the pointwise data, in the same way as previous experiments. We also used the pairwise data to train Ranking SVM (note that only Ranking SVM can be trained with pairwise data).

A 3-fold cross validation experiment was conducted to test the performances of the four methods: Kea, SVM, Ranking SVM trained on pointwise data (denoted as "Ranking SVM (Pointwise)") and Ranking SVM trained on pairwise data (denoted as "Ranking SVM (Pairwise)"). The performances were evaluated by using both pointwise annotated data in terms of MAP and precision, and the pairwise annotated data in terms of Kendall's tau. The results are given in Figure 4.

As shown in the figure, when evaluated on pairwise data, Ranking SVM (Pairwise) performs significantly better than the other methods in terms of Kendall's tau. Furthermore, Ranking SVM (Pointwise) significantly outperforms the other methods in terms of MAP and precisions, when evaluated on pointwise data. The improvements are all statistically significant in sign test (p-value ¡ 0.05). We conclude that again Ranking SVM can outperform SVM, no matter the training data is pointwise or pairwise.

4.5. Discussion

We conducted analysis on the reason that Ranking SVM performs better than SVM and Naive Bayes. It seems that the feature values in Ranking SVM have more discriminative power than those in the other methods, as discussed in Section 3.1.

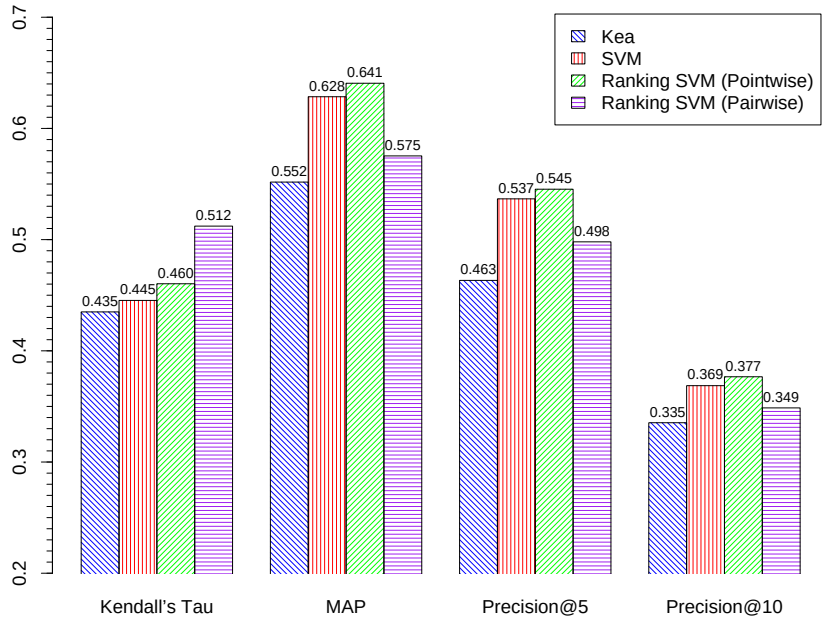

Figure 4: Keyphrase extraction accuracies on TREC .Gov dataset with pointwise and pairwise annotated data

We manually checked the keyphrases extracted by Ranking SVM and the other methods. It appears that Ranking SVM can do a better job than the other methods. We randomly selected six documents from the three datasets and listed the extracted keyphrases in Table 3.

From the table, we see that Ranking SVM can find a large number of keyphrases in its top 10 positions. Moreover, Ranking SVM can rank the keyphrases at higher positions than the other methods.

We conducted error analysis on Ranking SVM. We found there are several typical errors in Ranking SVM. (Note that these errors can hurt other methods as well). (1) Some less useful phrases were identified as candidate phrases, which affected the ranking results. For instance, in the first example in Table 3, the phrase "kWh/m3" is not useful for keyphrase extraction but included in the candidate list. Hence a kind of filtering on phrase candidates may be necessary. (2) The linear ranking function currently used may not be powerful enough. In a linear model, a phrase with one or two strong features is easily to be boosted to a high position when the features have high weights. In the second example of Table 3, the Ranking SVM model gives a large weight to the feature *Is in title or not*, and thus the words "query" and "complex" which appear in the document title are incorrectly ranked high. (3) For web pages, simply converting them to plain texts would cause loss of information. Making more effective use of information on the web pages such as formatting and structural information would be necessary. (4) Some top ranked non-keyphrases were actually good but not labeled as keyphrases by human annotators (Note that keyphrase annotation is hard for humans as well).

Table 3: Examples of the top 10 keyphrases extracted from six documents

Title	A Small Scale Seawater Reverse Osmosis System with Excellent Energy Efficiency over a Wide Operating Range (a research paper)
Author's keyphrases	Wind, Solar, Photovoltaic, PV, Seawater, Reverse osmosis, Energy recovery, Renewable energy, Desalination
Kea	pump, Clark pump, **Desalination**, kWh/m3, energy, **seawater**, membranes, pressure, recovery ratio, product water
SVM	**Reverse osmosis**, osmosis system, **seawater**, energy efficiency, wide operating range, excellent energy efficiency, **Desalination**, seawater reverse osmosis, small scale, small scale seawater
Ranking SVM	**Reverse osmosis**, **Desalination**, osmosis system, **seawater**, energy efficiency, wide operating range, excellent energy efficiency, small scale, CREST, kWh/m3
Title	KCNQ/M Currents in Sensory Neurons: Significance for Pain Therapy (a research paper)
Author's keyphrases	M-current, dorsal root ganglion, neuropathic pain, retigabine, KCNQ, nociceptors
Kea	Sensory Neurons, **KCNQ**, Neurons, Sensory, DRG neurons, KCNQ2/3, **retigabine**, DRGs, channels, rats
SVM	**M-current**, sensory neuron, **KCNQ**, M channel, neuron, Passmore, pain, KCNQ2/3, **retigabine**, KCNQ/M Channels
Ranking SVM	**M-current**, sensory neuron, **KCNQ**, neuron, M channel, KCNQ2/3, DRG neuron, **retigabine**, neuronal excitability, **neuropathic pain**
Title	DevMaster.net - A Beginner's Guide to Creating a MMORPG (a web page from social tagging data)
URL	`http://www.devmaster.net/articles/building-mmorpg/`
Tags and tagging frequencies	mmorpg (32), games (23), programming (22), gamedev (19), development (18), game (12), tutorial (9), design (7), gamedesign (5), server (4)
Kea	**MMORPG**, 3D, Database, DevMaster.net, **Game**, player, Forums, Engines, Beginner, **sever**
SVM	**MMORPG**, Beginner, need, **Server**, **Game**, player, Client, **Design**, people, use
Ranking SVM	**MMORPG**, **Server**, **Game**, Client, **Design**, **programming**, step, experience, Beginner, post
Title	Pivotal Blabs: HasFinder – It's Now Easier than ever to create complex, re-usable SQL queries (a web page from social tagging data)
URL	`http://pivots.pivotallabs.com/users/nick/blog/articles/284`
Tags and tagging frequencies	rails (93), activerecord (79), plugin(65), rubyonrails(38), finder(37), sql(35), ruby(28), plugins(25), finders(6), development(5), gems(5), hasfinder(5), associations (4), has_finder (4)
Kea	**HasFinder**, Pivotal, GMT, **ActiveRecord**, Nick, Blabs, complex, queries, **SQL**, **has_finder**
SVM	**HasFinder**, query, complex, GMT, **has_ finder**, Nick, Pivotal, **SQL**, work, **plugin**
Ranking SVM	**HasFinder**, **has_finder**, query, complex, **plugin**, **ActiveRecord**, condition, Find, **rail**, model
Title	Adobe Acrobat Instructions (a web page from .Gov archive)
URL	`http://seer.cancer.gov/acro.html`
Annotated keyphrases	Adobe Acrobat Reader, Acrobat, Adobe Acrobat Download, Acrobat Reader, browser
Kea	**Acrobat**, **browser**, PDF, installation, adobe, Adobe Acrobat Instructions, Internet Browser, exit, hard disk, **Acrobat Reader**
SVM	**Acrobat**, instruction, Internet Browsers, PDF file, **browser**, PDF, button, **Acrobat Reader**, **Adobe Acrobat Download**, installation
Ranking SVM (Pointwise)	**Acrobat**, **browser**, Internet Browsers, PDF file, instruction, **Adobe Acrobat Download**, **Acrobat Reader**, installation, button, PDF
Title	Types of Radiation Exposure (a web page from .Gov archive)
URL	`http://orise.orau.gov/reacts/guide/injury.htm`
Annotated keyphrases	radioactive material, Radiation, Contamination, External Irradiation, Radiation Exposure, Incorporation, radiation injury
Kea	**Radiation**, radioactive, **Contamination**, **radioactive materials**, Exposure, Irradiation, body, External, Types of Radiation, **External Irradiation**
SVM	exposure, **Radiation**, **radioactive material**, Total Body Irradiation, Radiation Accident Patient, Hospital Emergency Care, body, **Radiation Exposure**, Biological Effects, **Incorporation**
Ranking SVM (Pointwise)	**Radiation**, exposure, **radioactive material**, Total Body Irradiation, **Radiation Exposure**, Radiation Accident Patient, body, **External Irradiation**, **Incorporation**, Hospital Emergency Care

5. Conclusions and future work

In this paper, we have proposed using a learning to rank method, Ranking SVM, to extract keyphrases from document. We have discussed that the ranking approach is more essential for keyphrase extraction than the traditional classification approach. Experimental results based on three datasets have verified that Ranking SVM can significantly outperform the baseline methods of Naive Bayes and SVM.

As future work, we plan to apply other learning to rank methods to keyphrase extraction such as RankBoost [9] and AdaRank [32]. In addition, we plan to test other features; for example, features from other sources such as anchor texts and tags are likely to be useful for keyphrase extraction from web pages.

References

[1] K. Barker and N. Cornacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 40–52, 2000.

[2] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89–96, 2005.

[3] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2006.

[4] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, 2007.

[5] E. D'Avanzo and B. Magnini. A keyphrase-based approach to summarization: the lake system at duc-2005. In *Proceedings of the Fifth Document Understanding Conference*, 2005.

[6] R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[7] J. L. Fleiss. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5), 1971.

[8] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *Proceedings of 16th international joint conference on Artificial Intelligence*, pages 668–673, 1999.

[9] Y. Freund, R. D. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[10] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27(1-2):81–104, 1999.

[11] Y. HaCohen-Kerner, Z. Gross, and A. Masa. Automatic extraction and learning of keyphrases from scientific articles. *Lecture Notes in Computer Science*, 3406:657–669, 2005.

[12] J. Han, T. Kim, and J. Choi. Web document clustering by using automatic keyphrase extraction. In *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, pages 56–59, 2007.

[13] R. Herbrich, T. Graepel, , and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

[14] A. Hliaoutakis, K. Zervanou, E. G. Petrakis, and E. E. Milios. Automatic document indexing in large medical collections. In *Proceedings of the international workshop on Healthcare Information and Knowledge Management*, pages 1–8, 2006.

[15] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical Methods in Matural Language Processing*, pages 216–223, 2003.

[16] A. Hulth. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction*. PhD thesis, Department of Computer and Systems Sciences, Stockholm University, Sweden, 2004.

[17] T. Joachims. Aking large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[18] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 133–142, 2002.

[19] S. Jones and G. W. Paynter. Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications. *Journal of the American Society for Information Science and Technology*, 53(8):653–677, 2002.

[20] S. Jones and M. S. Staveley. Phrasier: a system for interactive document retrieval using keyphrases. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 160 – 167, 1999.

[21] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[22] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411, 2004.

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

[24] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[25] M. Song, I.-Y. Song, and X. Hu. Kpspotter: a flexible information gain-based keyphrase extraction system. In *Proceedings of the 5th ACM international workshop on Web Information and Data Management*, pages 50–53, 2003.

[26] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[27] P. D. Turney. Learning to extract keyphrases from text. Technical Report ERB-1057, National Research Council, Institute for Information Technology, 2000.

[28] P. D. Turney. Mining the web for lexical knowledge to improve keyphrase extraction: Learning from labeled and unlabeled data. Technical Report ERB-1096, National Research Council, Institute for Information Technology, 2002.

[29] J.-B. Wang, H. Peng, and J.-S. Hu. Automatic keyphrases extraction from document using backpropagation. In *Proceedings of 2005 international conference on Machine Learning and Cybernetics*, pages 3770–3774, 2005.

[30] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: practical automatic keyphrase extraction. In *Proceedings of Digital Libraries 99: The 4th ACM conference on Digital Libraries*, pages 254–255, 1999.

[31] Y.-F. B. Wu, Q. Li, R. S. Bot, and X. Chen. Finding nuggets in documents: a machine learning approach. *Journal of the American Society for Information Science and Technology*, 67(6), 2006.

[32] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 391–398, 2007.

[33] H. Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pages 113–120, 2002.

[34] L. Zhang, Y. Pan, and T. Zhang. Focused named entity recognition using machine learning. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 281–288, 2004.