Single Document Keyphrase Extraction Using Neighborhood Knowledge

Xiaojun Wan and Jianguo Xiao

Institute of Computer Science and Technology
Peking University, Beijing 100871, China
{wanxiaojun, xiaojianguo}@icst.pku.edu.cn

Abstract

Existing methods for single document keyphrase extraction usually make use of only the information contained in the specified document. This paper proposes to use a small number of nearest neighbor documents to provide more knowledge to improve single document keyphrase extraction. A specified document is expanded to a small document set by adding a few neighbor documents close to the document, and the graph-based ranking algorithm is then applied on the expanded document set to make use of both the local information in the specified document and the global information in the neighbor documents. Experimental results demonstrate the good effectiveness and robustness of our proposed approach.

Introduction

A keyphrase is defined as a meaningful and significant expression consisting of one or more words in a document. Appropriate keyphrases can serve as a highly condensed summary for a document, and they can be used as a label for the document to supplement or replace the title or summary, or they can be highlighted within the body of the document to facilitate users' fast browsing and reading. Moreover, document keyphrases have been successfully used in the following IR and NLP tasks: document indexing (Gutwin et al., 1999), document classification (Krulwich and Burkey, 1996), document clustering (Hammouda et al., 2005) and document summarization (Berger and Mittal, 2000).

Keyphrases are usually manually assigned by authors, especially for journal or conference articles. However, the vast majority of documents (e.g. news articles, magazine articles) do not have keyphrases, therefore it is beneficial to automatically extract a few keyphrases from a given document to deliver the main content of the document. Here, keyphrases are selected from within the body of the input document, without a predefined list (i.e. controlled vocabulary). Though keyphrase extraction is an important research topic in the NLP and IR field, it has received less attention than it deserves. Most previous works focus on keyphrase extraction for journal or conference articles, while this paper focus on keyphrase extraction for news articles because news article is one of the most popular document genres on the web and most news articles have no author-assigned keyphrases.

Existing methods conduct the keyphrase extraction task using only the information contained in the specified document, including the phrase's TFIDF, position and other syntactic information in the document. One common assumption of existing methods is that the documents are independent of each other. And the keyphrase extraction task is conducted separately without interactions for each document. However, some topic-related documents actually have mutual influences and contain useful clues which can help to extract keyphrases from each other. For example, two documents about the same topic "earthquake" would share a few common phrases, e.g. "earthquake", "victim", and they can provide additional knowledge for each other to better evaluate and extract salient keyphrases from each other. Therefore, given a specified document, we can retrieve a few documents topically close to the document from a large corpus through search engines, and these neighbor documents are deemed beneficial to evaluate and extract keyphrases from the document because they can provide more knowledge and clues for keyphrase extraction from the specified document.

This study proposes to construct an appropriate knowledge context for a specified document by leveraging a few neighbor documents close to the specified document. The neighborhood knowledge can be used in the keyphrase extraction process and help to extract salient keyphrases from the document. In particular, the graph-based ranking algorithm is employed for single document keyphrase extraction by making use of both the word relationships in the specified document and the word relationships in the neighbor documents, where the former relationships reflect the local information existing in the specified document and the latter relationships reflect the global information existing in the neighborhood.

Experiments have been performed on a dataset consisting of 308 news articles and human-annotated keyphrases, and the results demonstrate the good effectiveness of the proposed approach. The use of the neighborhood knowledge can significantly improve the performance of single document keyphrase extraction. We also investigate how the size of the neighborhood influences the keyphrase extraction performance and it is encouraging that a small number of neighbor documents can improve the performance.

Related Work

The methods for keyphrase (or keyword) extraction can be roughly categorized into either unsupervised or supervised. In this study, we focus on unsupervised methods.

Unsupervised methods usually involve assigning a saliency score to each candidate phrases by considering various features. Krulwich and Burkey (1996) use heuristics to extract keyphrases from a document. The heuristics are based on syntactic clues, such as the use of italics, the presence of phrases in section headers, and the use of acronyms. Barker and Cornacchia (2000) propose a simple system for choosing noun phrases from a document as keyphrases. Muñoz (1996) uses an unsupervised learning algorithm to discover two-word keyphrases. The algorithm is based on Adaptive Resonance Theory (ART) neural networks. Steier and Belew (1993) use the mutual information statistics to discover two-word keyphrases. Tomokiyo and Hurst (2003) use pointwise KL-divergence between multiple language models for scoring both phraseness and informativeness of phrases. More recently, Mihalcea and Tarau (2004) propose the TextRank model to rank keywords based on the co-occurrence links between words. Such algorithms make use of "voting" or "recommendations" between words to extract keyphrases.

Supervised machine learning algorithms have been proposed to classify a candidate phrase into either keyphrase or not. GenEx (Turney, 2000) and Kea (Frank et al., 1999; Witten et al., 1999) are two typical systems, and the most important features for classifying a candidate phrase are the frequency and location of the phrase in the document. More linguistic knowledge has been explored by Hulth (2003). Statistical associations between keyphrases have been used to enhance the coherence of the extracted keyphrases (Turney, 2003). Song et al. (2003) present an information gain-based keyphrase extraction system called KPSpotter. Medelyan and Witten (2006) propose KEA++ that enhances automatic keyphrase extraction by using semantic information on terms and phrases gleaned from a domain-specific thesaurus. Nguyen and Kan (2007) focus on keyphrase extraction in scientific publications by using new features that capture salient morphological phenomena found in scientific keyphrases.

All the above methods make use of only the information contained in the specified document. The use of neighbor documents to improve single document keyphrase extraction has not been investigated yet.

Other related works include web page keyword extraction (Kelleher and Luz, 2005), advertising keywords finding (Yih et al., 2006). It is noteworthy that collaborative techniques have been successfully used in the tasks of information filtering (Xue et al., 2005), document summarization (Wan et al., 2007) and web mining (Wong et al., 2006).

Proposed Approach

Overview

Given a specified document d_0 for keyphrase extraction, the proposed approach first finds a few neighbor documents for document d_0. The neighbor documents are topically close to the specified document and they construct the neighborhood knowledge context for the specified document. In other words, document d_0 is expanded to a small document set D which provides more knowledge and clues for keyphrase extraction from d_0. Given the expanded document set, the proposed approach adopts the graph-based ranking algorithm to incorporate both the word relationships in d_0 (local information) and the word relationships in neighbor documents (global information) for keyphrase extraction from d_0. Figure 1 gives the framework of the proposed approach.

1. ***Neighborhood Construction**: Expand the specified document d_0 to a small document set $D=\{d_0, d_1, d_2,\ldots d_k\}$ by adding k neighbor documents. The neighbor documents $d_1, d_2, \ldots, d_k$ can be obtained by using document similarity search techniques;*
2. ***Keyphrase Extraction**: Given document d_0 and the expanded document set D, perform the following steps to extract keyphrases for d_0:*
   a) ***Neighborhood-level Word Evaluation**: Build a global affinity graph G based on all candidate words restricted by syntactic filters in all the documents of the expanded document set D, and employ the graph-based ranking algorithm to compute the global saliency score for each word.*
   b) ***Document-level Keyphrase Extraction**: For the specified document d_0, evaluate the candidate phrases in the document based on the scores of the words contained in the phrases, and finally choose a few phrases with highest scores as the keyphrases of the document.*

Figure 1: The framework of the proposed approach

For the first step in the above framework, different similarity search techniques can be adopted to obtain neighbor documents close to the specified document. The number *k* of the neighbor documents influences the keyphrase extraction performance and will be investigated in the experiments.

For the second step in the above framework, substep a) aims to evaluate all candidate words in the expanded document set based on the graph-based ranking algorithm. The global affinity graph aims to reflect the neighborhood-level co-occurrence relationships between all candidate words in the expanded document set. The saliency scores of the words are computed based on the global affinity graph to indicate how much information about the main topic the words reflect. Substep b) aims to evaluate the candidate phrases in the specified document based on the neighborhood-level word scores, and then choose a few salient phrases as the keyphrases of the document.

Neighborhood Construction

Given a specified document d_0, neighborhood construction aims to find a few nearest neighbors for the document from a text corpus or on the Web. The k neighbor documents d_1, d_2, …, d_k and the specified document d_0 build the expanded document set $D=\{d_0, d_1, d_2, …, d_k\}$ for d_0, which can be considered as the expanded knowledge context for document d_0.

The neighbor documents can be obtained by using the technique of document similarity search. Document similarity search is to find documents similar to a query document in a text corpus and return a ranked list of similar documents to users. The effectiveness of document similarity search relies on the function for evaluating the similarity between two documents. In this study, we use the widely-used cosine measure to evaluate document similarity and the term weight is computed by TFIDF. The similarity $sim_{doc}(d_i,d_j)$, between documents d_i and d_j, can be defined as the normalized inner product of the two term vectors $\vec{d}_i$ and $\vec{d}_j$:

$$sim_{\text{doc}}(d_i,d_j) = \frac{\vec{d}_i \cdot \vec{d}_j}{\|\vec{d}_i\| \times \|\vec{d}_j\|} \tag{1}$$

In the experiments, we simply use the cosine measure to compute the pairwise similarity value between the specified document d_0 and the documents in the corpus, and then choose k documents (different from d_0) with the largest similarity values as the nearest neighbors for d_0. Finally, there are totally k+1 documents in the expanded document set. For the document set $D=\{d_0, d_1, d_2, …, d_k\}$, the pairwise cosine similarity values between documents are calculated and recorded for later use. The efficiency of document similarity search can be significantly improved by adopting some index structure in the implemented system, such as K-D-B tree, R-tree, SS-tree, SR-tree and X-tree (Böhm & Berchtold, 2001).

The use of neighborhood information is worth more discussion. Because neighbor documents might not be sampled from the same generative model as the specified document, we probably do not want to trust them so much as the specified document. Thus a confidence value is associated with every document in the expanded document set, which reflects out belief that the document is sampled from the same underlying model as the specified document. When a document is close to the specified one, the confidence value is high, but when it is farther apart, the confidence value will be reduced. Heuristically, we use the cosine similarity between a document and the specified document as the confidence value. The confidence values of the neighbor documents will be incorporated in the keyphrase extraction algorithm.

Keyphrase Extraction

a) Neighborhood-Level Word Evaluation

Like the PageRank algorithm (Page et al., 1998), the graph-based ranking algorithm employed in this study is essentially a way of deciding the importance of a vertex within a graph based on global information recursively drawn from the entire graph. The basic idea is that of "voting" or "recommendation" between the vertices. A link between two vertices is considered as a vote cast from one vertex to the other vertex. The score associated with a vertex is determined by the votes that are cast for it, and the score of the vertices casting these votes.

Formally, given the expanded document set D, let $G=(V, E)$ be an undirected graph to reflect the relationships between words in the document set. V is the set of vertices and each vertex is a candidate word[1] in the document set. Because not all words in the documents are good indicators of keyphrases, the words added to the graph are restricted with syntactic filters, i.e., only the words with a certain part of speech are added. As in Mihalcea and Tarau (2004), the documents are tagged by a POS tagger, and only the nouns and adjectives are added into the vertex set[2]. E is the set of edges, which is a subset of $V\times V$. Each edge e_{ij} in E is associated with an affinity weight $aff(v_i,v_j)$ between words v_i and v_j. The weight is computed based on the co-occurrence relation between the two words, controlled by the distance between word occurrences. The co-occurrence relation can express cohesion relationships between words. Two vertices are connected if the corresponding words co-occur at least once within a window of maximum w words, where w can be set anywhere from 2 to 20 words. The affinity weight $aff(v_i,v_j)$ is simply set to be the count of the controlled co-occurrences between the words v_i and v_j in the whole document set as follows:

$$aff(v_i,v_j) = \sum_{d_p \in D} sim_{doc}(d_0,d_p) \times count_{d_p}(v_i,v_j) \tag{2}$$

where $count_{d_p}(v_i,v_j)$ is the count of the controlled co-occurrences between words v_i and v_j in document d_p, and $sim_{doc}(d_0,d_p)$ is the similarity factor to reflect the confidence value for using document d_p $(0 \le p \le k)$ in the expanded document set.

The graph is built based on the whole document set and it can reflect the global information in the neighborhood, which is called *Global Affinity Graph*. We use an affinity matrix M to describe G with each entry corresponding to the weight of an edge in the graph. $M = (M_{i,j})_{|V|\times|V|}$ is defined as follows:

$$M_{i,j} = \begin{cases} aff(v_i,v_j), & \text{if } v_i \text{ links with } v_j \text{ and } i \ne j; \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Then M is normalized to $\widetilde{M}$ as follows to make the sum of each row equal to 1:

[1] The original words are used without stemming.
[2] The corresponding POS tags of the candidate words include "JJ", "NN", "NNS", "NNP", "NNPS". We used the Stanford log-linear POS tagger (Toutanova and Manning, 2000) in this study.

$$\tilde{M}_{i,j} = \begin{cases} M_{i,j} \Big/ \sum_{j=1}^{|V|} M_{i,j}, & \text{if } \sum_{j=1}^{|V|} M_{i,j} \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Based on the global affinity graph G, the saliency score $WordScore(v_i)$ for word v_i can be deduced from those of all other words linked with it and it can be formulated in a recursive form as in the PageRank algorithm:

$$WordScore(v_i) = \mu \cdot \sum_{all\, j \neq i} WordScore(v_j) \cdot \tilde{M}_{j,i} + \frac{(1-\mu)}{|V|} \quad (5)$$

And the matrix form is:

$$\vec{\lambda} = \mu \tilde{M}^T \vec{\lambda} + \frac{(1-\mu)}{|V|} \vec{e} \quad (6)$$

where $\vec{\lambda} = [WordScore(v_i)]_{|V| \times 1}$ is the vector of word saliency scores. $\vec{e}$ is a vector with all elements equaling to 1. μ is the damping factor usually set to 0.85, as in the PageRank algorithm.

The above process can be considered as a Markov chain by taking the words as the states and the corresponding transition matrix is given by $\mu \tilde{M}^T + \frac{(1-\mu)}{|V|} \vec{e}$. The stationary probability distribution of each state is obtained by the principal eigenvector of the transition matrix. For implementation, the initial scores of all words are set to 1 and the iteration algorithm in Equation (5) is adopted to compute the new scores of the words. Usually the convergence of the iteration algorithm is achieved when the difference between the scores computed at two successive iterations for any words falls below a given threshold (0.0001 in this study).

b) Document-Level Keyphrase Extraction

After the scores of all candidate words in the document set have been computed, candidate phrases (either single-word or multi-word) are selected and evaluated for the specified document d_0. The candidate words (i.e. nouns and adjectives) of d_0, which is a subset of V, are marked in the text of document d_0, and sequences of adjacent candidate words are collapsed into a multi-word phrase. The phrases ending with an adjective is not allowed, and only the phrases ending with a noun are collected as candidate phrases for the document. For instance, in the following sentence: "*Mad/JJ cow/NN disease/NN has/VBZ killed/VBN 10,000/CD cattle/NNS*", the candidate phrases are "*Mad cow disease*" and "*cattle*". The score of a candidate phrase p_i is computed by summing the neighborhood-level saliency scores of the words contained in the phrase.

$$PhraseScore(p_i) = \sum_{v_j \in p_i} WordScore(v_j) \quad (7)$$

All the candidate phrases in document d_0 are ranked in decreasing order of the phrase scores and the top m phrases are selected as the keyphrases of d_0. m ranges from 1 to 20 in this study.

Empirical Evaluation

Evaluation Setup

To our knowledge, there was no gold standard news dataset with assigned keyphrases for evaluation. So we manually annotated the DUC2001 dataset (Over, 2001) and used the annotated dataset for evaluation in this study. The dataset was originally used for document summarization. It consisted of 309 news articles collected from TREC-9, in which two articles were duplicate (i.e. d05a\FBIS-41815 and d05a\FBIS-41815~), so the actual document number was 308. The articles could be categorized into 30 news topics and the average length of the documents was 740 words. Two graduate students were employed to manually label the keyphrases for each document. At most 10 keyphrases could be assigned to each document. The annotation process lasted two weeks. The Kappa statistic for measuring inter-agreement among annotators was 0.70. And then the annotation conflicts between the two subjects were solved by discussion. Finally, 2488 keyphrases were labeled for the dataset. The average keyphrase number per document was 8.08 and the average word number per keyphrase was 2.09. In the experiments, the DUC2001 dataset was considered as the corpus for document expansion in this study, which could be easily expanded by adding more documents. Each specified document was expanded by adding k documents (different from the specified document) most similar to the document.

For evaluation of keyphrase extraction results, the automatic extracted keyphrases were compared with the manually labeled keyphrases. The words in a keyphrase were converted to their corresponding basic forms using word stemming before comparison. The precision $p=count_{correct}/count_{system}$, recall $r=count_{correct}/count_{human}$, F-measure ($F=2pr/(p+r)$) were used as evaluation metrics, where $count_{correct}$ was the total number of correct keyphrases extracted by the system, and $count_{system}$ was the total number of automatic extracted keyphrases, and $count_{human}$ was the total number of human-labeled keyphrases.

Evaluation Results

The proposed approach (i.e. ExpandRank) is compared with the baseline methods relying only on the specified document (i.e. SingleRank and TFIDF). The SingleRank baseline uses the graph-based ranking algorithm to compute the word scores for each single document based on the local graph for the specified document. The TFIDF baseline computes the word scores for each single document based on the word's TFIDF value in the specified document. The two baselines do not make use of the neighborhood knowledge.

Table 1 gives the comparison results of the baseline methods and the proposed ExpandRank methods with different neighbor numbers (k=1, 5, 10). In the experiments, the keyphrase number m is typically set to 10 because at most 10 keyphrases can be manually labeled for each document, and the co-occurrence window size *w* is also simply set to 10.

Table 1. Keyphrase Extraction Results

System	Precision	Recall	F-measure
TFIDF	0.232	0.281	0.254
SingleRank	0.247	0.303	0.272
ExpandRank (k=1)	0.264	0.325	0.291
ExpandRank (k=5)	0.288	0.354	0.317
ExpandRank (k=10)	0.286	0.352	0.316

Seen from Table 1, the ExpandRank methods with different neighbor numbers can always outperform the baseline methods of SingleRank and TFIDF over all three metrics. The results demonstrate the good effectiveness of the proposed method.

In order to investigate how the size of the neighborhood influences the keyphrase extraction performance, we conduct experiments with different values of the neighbor number *k*. Figure 2 shows the performance curves for the ExpandRank method. In the figure, *k* ranges from 0 to 15. Note that when *k*=0, the ExpandRank method degenerates into the baseline SingleRank method. We can see from the figure that the performance of ExpandRank (i.e. *k*>0) can always outperform the baseline SingleRank method (i.e. *k*=0), no matter how many neighbor documents are used. We can also see that the performance of ExpandRank first increases and then decreases with the increase of *k*. The trend demonstrates that very few or very many neighbors will deteriorate the results, because very few neighbors cannot provide sufficient knowledge and very many neighbors may introduce noisy knowledge. Seen from the figure, it is not necessary to use many neighbors for ExpandRank, and the neighbor number can be set to a comparable small number (i.e. 5), which will improve the computational efficiency and make the propose approach more applicable.

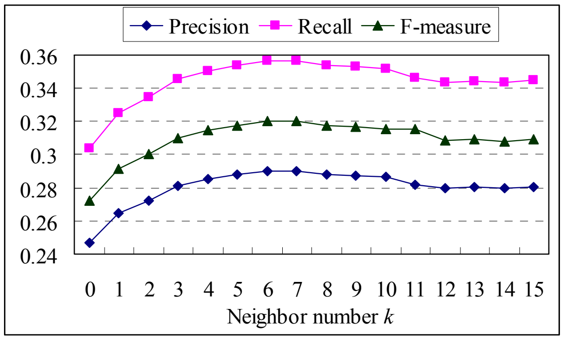

Figure 2: ExpandRank (*m*=10, *w*=10) performance vs. neighbor number *k*

In order to investigate how the co-occurrence window size influences the keyphrase extraction performance, we conduct experiments with different window size *w*. Figures 3 and 4 show the performance curves for ExpandRank when *w* ranges from 2 to 20. In Figure 3 the neighbor number is set to 5 and in Figure 4 the neighbor number is set to 10. We can see from the figures that the performances are almost not affected by the window size, except when *w* is set to 2.

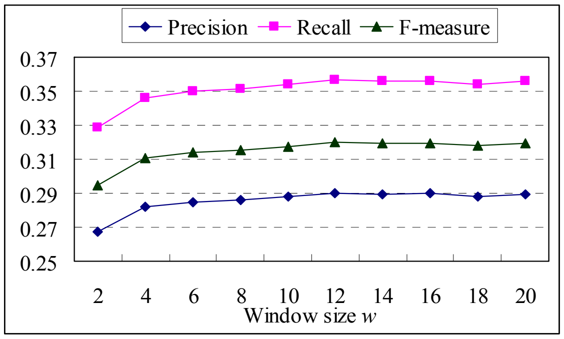

Figure 3: ExpandRank (*k*=5, *m*=10) performance vs. window size *w*

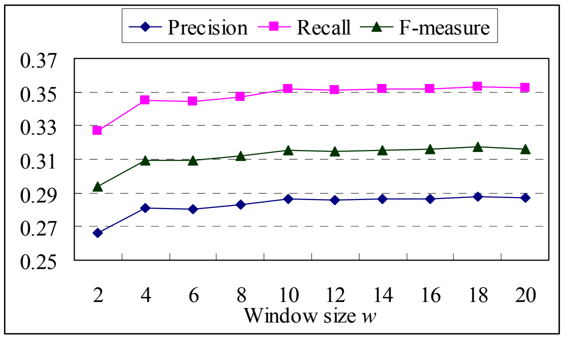

Figure 4: ExpandRank (*k*=10, *m*=10) performance vs. window size *w*

In the above experiments, the keyphrase number is set to 10. We further conduct experiments with different keyphrase number *m* to investigate how the keyphrase number influences the keyphrase extraction performance. Figures 5 and 6 show the performance curves for ExpandRank when *m* ranges from 1 to 20. In Figure 5 the neighbor number is set to 5 and in Figure 6 the neighbor number is set to 10. We can see from the figures that the precision values decrease with the increase of *m*, and the recall values increases with the increase of *m*, while the F-measure values first increase and then tend to decrease with the increase of *m*.

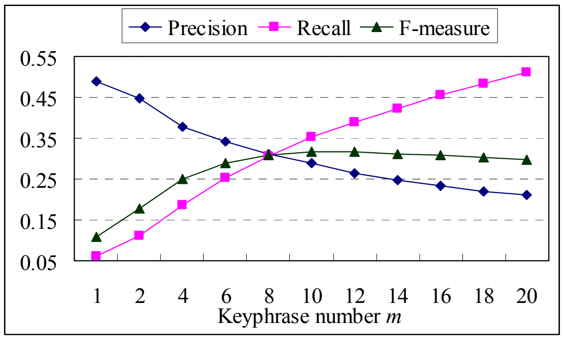

Figure 5: ExpandRank (*k*=5, *w*=10) performance vs. keyphrase number *m*

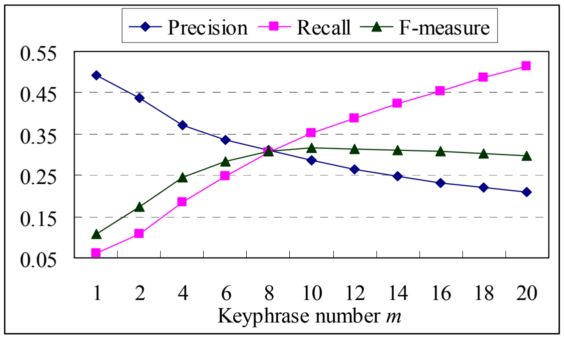

Figure 6: ExpandRank (*k*=10, *w*=10) performance vs. keyphrase number *m*

It is noteworthy that the proposed approach has higher computational complexity than the baseline approach because it involves more documents, and we can improve its efficiency by collaboratively conducting single document keyphrase extractions in a batch mode. Suppose there are multiple documents to be extracted separately, we can group the documents into clusters, and for each cluster, we can use all other documents as the neighbors for a specified document. Thus the mutual influences between all documents can be incorporated into the keyphrase extraction algorithm and all the words and phrases in the documents of a cluster are evaluated collaboratively, resulting in keyphrase extraction for all the single documents in a batch mode.

Conclusion and Future Work

This paper proposes a novel approach to single document keyphrase extraction by leveraging the neighborhood knowledge of the specified document. In future work, other keyphrase extraction algorithms will be integrated into the proposed framework, and we will use more test data for evaluation to validate the robustness of the proposed approach.

Acknowledgements

This work was supported by the National Science Foundation of China (No.60703064) and the Research Fund for the Doctoral Program of Higher Education of China (No.20070001059).

References

Berger, A., and Mittal, V. 2000. OCELOT: A system for summarizing Web Pages. In *Proceedings of SIGIR2000*.

Barker, K., and Cornacchia, N. 2000. Using nounphrase heads to extract document keyphrases. In *Canadian Conference on AI*.

Böhm, C., and Berchtold, S. 2001. Searching in high-dimensional spaces-index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3): 322-373.

Frank, E.; Paynter, G. W.; Witten, I. H.; Gutwin, C.; and Nevill-Manning, C. G. 1999. Domain-specific keyphrase extraction. *Proceedings of IJCAI-99*, pp. 668-673.

Gutwin, C.; Paynter, G. W.; Witten, I. H.; Nevill-Manning, C. G.; and Frank, E. 1999. Improving browsing in digital libraries with keyphrase indexes. *Journal of Decision Support Systems*, 27, 81-104.

Hammouda, K. M.; Matute, D. N.; and Kamel, M. S. 2005. CorePhrase: keyphrase extraction for document clustering. In *Proceedings of MLDM2005*.

Hulth, A. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP2003*.

Kelleher, D., and Luz, S. 2005. Automatic hypertext keyphrase detection. In *Proceedings of IJCAI2005*.

Krulwich, B., and Burkey, C. 1996. Learning user information interests through the extraction of semantically significant phrases. In *AAAI 1996 Spring Symposium on Machine Learning in Information Access*.

Medelyan, O., and Witten, I. H. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of JCDL2006*.

Mihalcea, R., and Tarau, P. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP2004*.

Muñoz, A. 1996. Compound key word generation from document databases using a hierarchical clustering ART model. *Intelligent Data Analysis*, 1(1).

Nguyen, T. D., and Kan, M.-Y. 2007. Keyphrase extraction in scientific publications. In *Proceedings of ICADL2007*.

Over, P. 2001. Introduction to DUC-2001: an intrinsic evaluation of generic news text summarization systems. In *Proceedings of DUC2001*.

Page, L.; Brin, S.; Motwani, R.; and Winograd, T. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report*, Stanford Digital Libraries.

Song, M.; Song, I.-Y.; and Hu, X. 2003. KPSpotter: a flexible information gain-based keyphrase extraction system. In *Proceedings of WIDM2003*.

Steier, A. M., and Belew, R. K. 1993. Exporting phrases: A statistical analysis of topical language. In *Proceedings of Second Symposium on Document Analysis and Information Retrieval*, pp. 179-190.

Tomokiyo, T., and Hurst, M. 2003. A language model approach to keyphrase extraction. In *Proceedings of ACL Workshop on Multiword Expressions*.

Toutanova, K., and Manning, C. D. 2000. Enriching the knowledge sources used in a maximum entropy Part-of-Speech tagger. In *Proceedings of EMNLP/VLC-2000*.

Turney, P. D. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2:303-336.

Turney, P. D. 2003. Coherent keyphrase extraction via web mining. In *Proc. of IJCAI-03*, pages 434–439.

Wan, X.; Yang, J.; and Xiao, J. 2007. Single document summarization with document expansion. In *Proceedings of AAAI2007*

Witten, I. H.; Paynter, G. W.; Frank, E.; Gutwin, C.; and Nevill-Manning, C. G. 1999. KEA: Practical automatic keyphrase extraction. *Proceedings of Digital Libraries 99 (DL'99)*, pp. 254-256.

Wong, T.-L.; Lam, W.; and Chan, S.-K. 2006. Collaborative information extraction and mining from multiple web documents. In *Proceedings of SDM2006*.

Xue, G.-R.; Lin, C.; Yang, Q.; Xi, W.; Zeng, H.-J.; Yu, Y.; and Chen, Z. 2005. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of SIGIR2005*.

Yih, W.-T.; Goodman, J.; and Carvalho, V. R. 2006. Finding advertising keywords on web pages. In *Proceedings of WWW2006*.