CorePhrase: Keyphrase Extraction for Document Clustering

Khaled M. Hammouda[1], Diego N. Matute[2], and Mohamed S. Kamel[3]

[1] Department of Systems Design Engineering
[2] School of Computer Science
[3] Department of Electrical and Computer Engineering
Pattern Analysis and Machine Intelligence (PAMI) Research Group
University of Waterloo
Waterloo ON N2L3G1, Canada
{hammouda,dnmatute,mkamel}@pami.uwaterloo.ca

Abstract. The ability to discover the topic of a large set of text documents using relevant keyphrases is usually regarded as a very tedious task if done by hand. Automatic keyphrase extraction from multi-document data sets or text clusters provides a very compact summary of the contents of the clusters, which often helps in locating information easily. We introduce an algorithm for topic discovery using keyphrase extraction from multi-document sets and clusters based on frequent and significant shared phrases between documents. The keyphrases extracted by the algorithm are highly accurate and fit the cluster topic. The algorithm is independent of the domain of the documents. Subjective as well as quantitative evaluation show that the algorithm outperforms keyword-based cluster-labeling algorithms, and is capable of accurately discovering the topic, and often ranking it in the top one or two extracted keyphrases.

1 Introduction

The abundance of information in text form has been both a blessing and a plague. There is always a need to summarize information into compact form that could be easily absorbed. The challenge is to extract the essence of text documents and present it in a compact form that identifies their topic(s). *Keyphrase extraction*, which is a text mining task, extracts highly relevant phrases from documents.

Turney [1] lists over a dozen applications that utilizes keyphrase extraction. For example, providing mini-summaries of large documents, highlighting keyphrases in text, text compression, constructing human-readable keyphrase-based indexes, interactive query refinement by suggesting improvements to queries, document clustering, and document classification.

Keyphrase extraction algorithms fall into two categories: keyphrase extraction from single documents, which is often posed as a supervised learning task; and keyphrase extraction from a set of documents, which is an unsupervised learning task that tries to *discover* the topics rather than *learn* from examples.

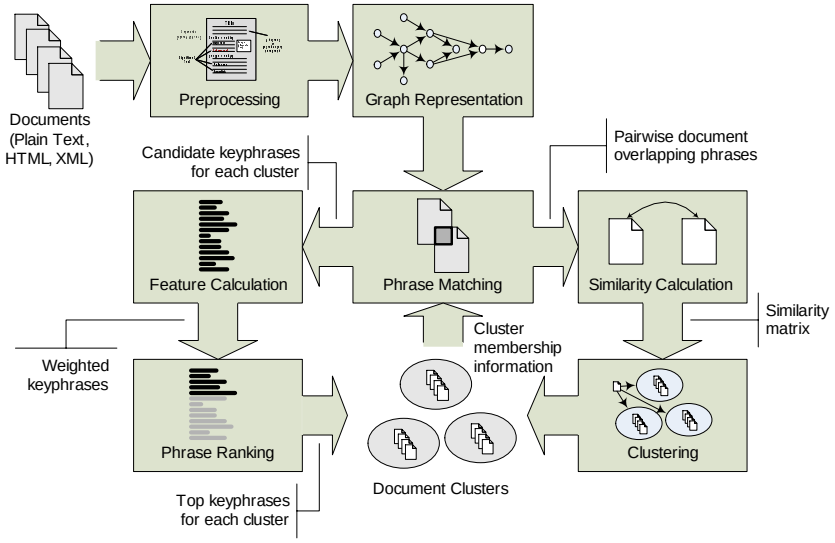


Fig. 1. CorePhrase Keyphrase Extraction System

Extraction vs. Construction There are two ways to finding relevant keyphrases in text: either to *extract* them from existing phrases in the text, or to automatically *construct* them [2]. Construction of keyphrases is regarded as a more intelligent way of summarizing text, but in practice is more difficult.

Keyphrases vs. Keywords A *keyphrase* is "a sequence of one or more words that is considered highly relevant", while a *keyword* is "a single word that is highly relevant." An arbitrary combination of keywords does not necessarily constitute a keyphrase; neither do the constituents of a keyphrase necessarily represent individual keywords.

In this paper we present a highly accurate method for extracting keyphrases from multi-document sets or clusters, with no prior knowledge about the documents. The algorithm is called **CorePhrase**, and is based on finding a set of core phrases from a document cluster.

CorePhrase works by extracting a list of candidate keyphrases by intersecting documents using a graph-based model of the phrases in the documents. This is facilitated through a powerful phrase-based document indexing model [3].

Features of the extracted candidate keyphrases are then calculated, and phrases are ranked based on their features. The top phrases are output as the descriptive topic of the document cluster. Results show that the extracted keyphrases are highly relevant to the topic of the document set. Figure 1 illustrates the different components of the keyphrase extraction system.

The work presented here assumes that: (1) keyphrases exist in the text and are not automatically generated; (2) the algorithm *discovers* keyphrases rather than *learns* how to extract them; and (3) if used with text clustering, the algo-

rithm is not concerned with how the clusters are generated; it extracts keyphrases from already clustered documents.

The paper is organized as follows. Section 2 discusses related work in keyphrase extraction. The CorePhrase algorithm is presented in section 3. Experimental results are presented and discussed in section 4. Finally we discuss some conclusions and outline future work in section 5.

2 Related Work

Two popular single-document keyphrase extraction algorithms are: *Extractor* by Turney [1], and *Kea* by Frank *et al* [2]. Extractor uses machine learning to extract keyphrases from individual documents, which employs a genetic algorithm to tune its parameters. Evaluation is based on the number of extracted keyphrases that match human generated keyphrases, which is claimed to achieve human-like performance, but could be biased towards the trained data. Kea is a single document summarizer, which employs a Bayesian supervised learning approach to build a model out of training data, then applies the model to unseen documents for keyphrase extraction.

Methods using keyword-based cluster labeling include that proposed by Neto *et al* [4], which uses Autoclass-based clustering with top frequent keywords for cluster labels. In addition to keyword-based cluster summarization, it is also used to perform single document summarization, using a variation of the popular tf×idf weighting scheme to score phrases.

An information theoretic based approach for phrase extraction from multi-documents has been proposed by Bakus *et al* [5]. The method finds hierarchical combinations of statistically significant phrases.

Mani and Bloedorn suggested a method for multi-document summarization based on a graph representation based on concepts in the text [6]. Also another system for topic identification is TopCat [7]. It uses a series of natural language processing, frequent itemset analysis, and clustering steps to identify the topics in a document collection.

3 The CorePhrase Algorithm

CorePhrase works by first constructing a list of candidate keyphrases, scoring each candidate keyphrase according to some criteria, ranking the keyphrases by score, and finally selecting a number of the top ranking keyphrases for output.

3.1 Extraction of Candidate Keyphrases

Candidate keyphrases naturally lie at the *intersection* of the document cluster. The CorePhrase algorithm compares every pair of documents to extract matching phrases. This process of matching every pair of documents is inherently $O(n^2)$. However, by using a proven method of document phrase indexing

graph structure, known as the Document Index Graph (DIG), the algorithm can achieve this goal in near-linear time [3].

In essence, what the DIG model does is to keep a cumulative graph representing currently processed documents: $G_i = G_{i-1} \cup g_i$, where g_i is the subgraph representation of a new document. Upon introducing a new document, its subgraph is matched with the existing cumulative graph to extract the matching phrases between the new document and all previous documents. That is, the list of matching phrases between document $\mathbf{d}_i$ and previous documents is given by $M_i = g_i \cap G_{i-1}$. The graph maintains complete phrase structure identifying the containing document and phrase location, so cycles can be uniquely identified. This process produces complete phrase-matching output between every pair of documents in near-linear time, with arbitrary length phrases. Figure 2 illustrates the process of phrase matching between two documents. In the figure, the two subgraphs of two documents are matched to get the list of phrases shared between them.

Since this method outputs matching phrases for each new document, it is essential to keep a *master list*, M, of unique matched phrases, which will be used as the list of candidate keyphrases. The following simple procedure keeps this list updated:

1: {calculate M_i for document $\mathbf{d}_i$}
   $M_{ij} = \{p_{ij}: 1 < j < i\}$: matching phrases between $\mathbf{d}_i$ and $\mathbf{d}_j$
   $M_i = \{M_{ij}\}$: matching phrases of $\mathbf{d}_i$
2: **for each** phrase p_{ij} in M_i **do**
3: **if** phrase p_{ij} is in master list M **then**
4: add feature vector $\mathbf{p}_i$ to p_{ij} in M
5: add feature vector $\mathbf{p}_j$ to p_{ij} in M if not present
6: **else**
7: add p_{ij} to M
8: add feature vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ to p_{ij} in M
9: **end if**
10: **end for**
11: **for each** *unique* phrase p_k in M **do**
12: calculate averages of feature vectors associated with p_k
13: **end for**

3.2 Phrase Features

Quantitative features are needed to judge the quality of the candidate keyphrases. Each candidate keyphrase p is assigned the following features:

df: **document frequency**; the number of documents in which the phrase appeared, normalized by the total number of documents.

$$df = \frac{\mid \text{documents containing } p \mid}{\mid \text{all documents} \mid}$$

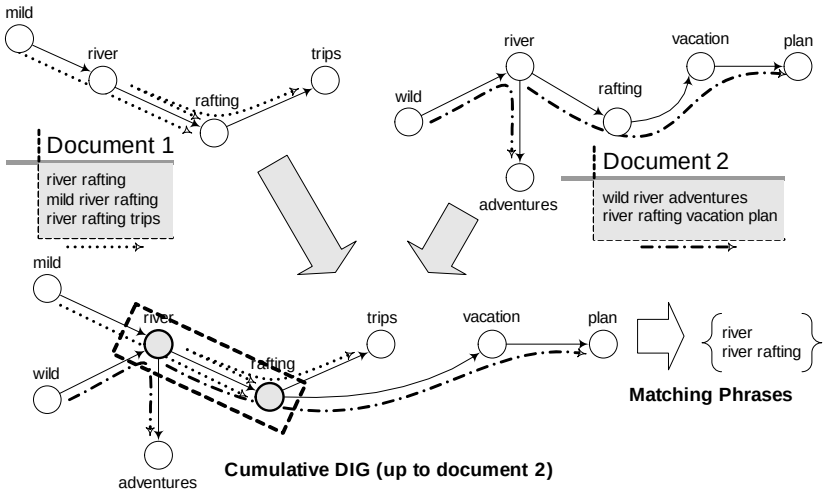


Fig. 2. Phrase Matching Using Document Index Graph

w**:** average **weight**; the average weight of the phrase over all documents. The weight of a phrase in a document is calculated using structural text cues. Examples: title phrases have maximum weight, section headings are weighted less, while body text is weighted lowest.

pf**:** average **phrase frequency**; the average number of times this phrase has appeared in one document, normalized by the length of the document in words.

$$pf = \arg avg \left[\frac{| \text{ occurrences of } p |}{| \text{ words in document } |} \right]$$

d**:** average phrase **depth**; the location of the first occurrence of the phrase in the document.

$$d = \arg avg \left[1 - \frac{| \text{ words before first occurrence } |}{| \text{ words in document } |} \right]$$

Those features will be used to rank the candidate phrases. In particular, we want phrases that appear in more documents (high df; *i.e.* high *support*), have higher weights (high w), higher frequencies (high pf), and shallow depth (low d).

3.3 Phrase Ranking

Phrase features are used to calculate a *score* for each phrase. Phrases are then ranked by score, and a number of the top phrases are selected as the ones describing the topic of the cluster. There are two phrase scoring formulas used, as well as two methods of assigning the score to the candidate phrases, for a total of four variants of the CorePhrase algorithm.

First Scoring Formula. The score of each phrase p is:

$$\text{score}(p) = (w \cdot pf) \times -\log(1 - df) \quad (1)$$

The equation is derived from the tf×idf term weighting measure; however, we are rewarding phrases that appear in more documents (high df) rather than punishing those phrases. Notice also that the first scoring formula does not take the *depth* feature into account. We will refer to the variant of the algorithm that uses this formula as CorePhrase-1.

Second Scoring Formula. By examining the distribution of the values of each feature in a typical corpus (see Table 1 for details), it was found that the *weight* and *frequency* features usually have low values compared to the *depth* feature. To take this fact into account, it was necessary to "expand" the *weight* and *frequency* features by taking their square root, and to "compact" the *depth* by squaring it. This helps even out the feature distributions and prevents one feature from dominating the score equation. The formula is given in equation 2, and is used by the variant which we will refer to as CorePhrase-2.

$$\text{score}(p) = (\sqrt{w \cdot pf} \cdot d^2) \times -\log(1 - df) \quad (2)$$

Word weight-based score assignment. A *modified* score assignment scheme based on word weights is also used:

- First, assign *initial* scores to each phrase based on phrase scoring formulas given above.
- Construct a list of unique individual words out of the candidate phrases.
- For each word: add up all the scores of the phrases in which this word appeared to create a word weight.
- For each phrase: assign the *final* phrase score by adding the individual word weights of the constituent words and average them.

The variants of the algorithm that use this method will be referred to as CorePhrase-1M and -2M, based on the equation used to assign the phrase scores.

4 Experimental Results

4.1 Experimental Setup

Two data sets were used for evaluation, which are listed in Table 1[4]. The first one is a collection of web documents representing six topics formed by submitting six different queries to the google search engine, and is used as a realistic evaluation in scenarios involving the online keyphrase extraction. The second data set is a collection of web documents from intranet web sites at the University of Waterloo, and serves in evaluating scenarios involving data-mining an intranet information system. Table 1 also lists the characteristics of each data set.

[4] The data sets are available for download at: `http://pami.uwaterloo.ca/~hammouda/webdata/`

Table 1. Data sets used for multi-document keyphrase extraction

class	# docs.	average words/doc.	candidate phrases	feature averages			
				df	*w*	*pf*	*d*
Data Set 1: Total **151** docs.							
canada transportation	22	893	24403	0.0934	0.2608	0.0007	0.4907
winter weather canada	23	636	5676	0.0938	0.2530	0.0019	0.5903
snowboarding skiing	24	482	990	0.0960	0.3067	0.0033	0.5215
river fishing	23	443	485	0.1129	0.2965	0.0042	0.5763
river rafting	29	278	599	0.0931	0.3680	0.0057	0.5612
black bear attacks	30	620	1590	0.0859	0.3593	0.0023	0.6024
Data Set 2: Total **161** docs.							
Co-operative Education	54	251	1379	0.0511	0.2672	0.0082	0.5693
Career Services	52	508	4245	0.0473	0.2565	0.0031	0.5000
Health Services	23	329	351	0.1454	0.2707	0.0057	0.5170
Campus Network	32	510	14200	0.0810	0.2569	0.0020	0.5198

A keyword-based extraction algorithm was used as a baseline for comparison. The algorithm extracts the centroid vector of a cluster represented as a set of keywords and selects the top frequent keywords in the cluster. This method is considered representative of most cluster labeling methods.

4.2 Evaluation Measures

Two extrinsic evaluation measures are used to assess the performance of CorePhrase. The first measure is called *overlap*, which measures the similarity between each extracted keyphrase to the predefined topic phrase of the cluster. The similarity is based on how many terms are shared between the two phrases. The overlap between an extracted keyphrase p_i and the topic phrase p_t is defined as:

$$\text{overlap}(p_i, p_t) = \frac{\mid p_i \cap p_t \mid}{\mid p_i \cup p_t \mid} \tag{3}$$

To evaluate the top k keyphrases as a set, we take the average overlap of the whole set. This measure is essentially telling us how well the top keyphrases, as a set, *fit* the reference topic.

The second evaluation measure is called *precision*[5], which gives an indication of how high the single keyphrase that best fits the topic is ranked. The best keyphrase is defined as the first keyphrase, in the top k, that has maximum overlap with the reference topic. Thus, the precision for the set of top k phrases ($\mathbf{p}^k$) with respect to the reference topic p_t is defined as:

$$\text{precision}(\mathbf{p}^k, p_t) = \text{overlap}(p_{\max}, p_t) \cdot \left[1 - \frac{\text{rank}(p_{\max}) - 1}{k}\right] \tag{4}$$

where $p_{\max} \in \mathbf{p}^k$ is the first phrase with maximum overlap in the top k phrases; and $\text{rank}(p_{\max})$ is its rank in the top k. In other words, precision tells us how *high* in the ranking the best phrase appears. The lower the best phrase comes in the ranking, the lower the precision.

[5] This is not the precision measure usually used in the information retrieval literature.

Table 2. Keyphrase Extraction Results – Top 10 Keyphrases

	CorePhrase-1	CorePhrase-2	CorePhrase-1M	CorePhrase-2M
			canada transporation	
1	canada transport	canada transport	transport canada	canada transport
2	panel recommend	canada transport act	canada transport	transport canada
3	transport associ	transport act	road transport	transport act
4	transport associ canada	transport associ	transport issu	transport issu
5	associ canada	panel recommend	govern transport	recommend transport
6	canada transport act	unit state	surfac transport	transport polici canada transport
7	transport act	transport associ canada tac	public transport	canadian transport
8	road transport	associ canada tac	transport public	transport public
9	transport infrastructur	canada tac	transport infrastructur	public transport
10	transport associ canada tac	public privat sector	transport passeng	transport infrastructur
			winter weather canada	
1	winter storm	sever weather	new hampshir new	environment assess environment
2	winter weather	winter weather	new jersei new	program legisl
3	environ canada	winter storm	new mexico new	program hunt
4	sever weather	weather warn	new hampshir new jersei new	fund program
5	weather warn	sever winter	new jersei new mexico new	environment link fund program
6	freez rain	sever weather warn	new hampshir new jersei new mexico	environment assess environment link fund
7	new brunswick	sever winter weather	new hampshir	environment link
8	heavi snowfal	new brunswick	hampshir new	environment assess environment link
9	winter weather warn	environ canada	carolina new hampshir new	assess environment
10	warn issu	cold winter	carolina new	environment assess
			campus network	
1	campu network	campu network	network network	network network
2	uw campu network	uw campu network	network uw network	network level network
3	uw campu	uw campu	network level network	network uw network
4	roger watt	network connect	uw network	network subscrib network
5	roger watt ist	level network	network uw	level network level network
6	watt ist	high speed	network subscrib network	level network
7	ip address	uw resnet	network assign network	network level
8	ip network	connect uw	network uw campu network	network assign network
9	high speed	area campu network	network level	extern network level network level network
10	request registr	switch rout	level network level network	network level network rout

4.3 Discussion

Table 2 shows the results of keyphrase extraction by the CorePhrase algorithm variants for three of the classes (two classes from data set 1, and one class from data set 2)[6]. The phrases in the results are shown in stemmed form, with stop words removed.

A more concrete evaluation based on the quantitative measures, overlap and precision, is given in Table 3, which is illustrated also in Figure 3 (in the figure, only CorePhrase-2 and CorePhrase-2M are shown). For each of the four variants of the CorePhrase algorithm, in addition to the baseline keyword centroid algorithm, we report the overlap and precision. The average overlap is taken over the top 10 keyphrases/keywords of each cluster, with the maximum overlap value (best phrase) also shown.

The keyphrases extracted by the variants of the CorePhrase[7] algorithm were very close to the reference topic, which is a subjective verification of the algorithm correctness. We leave it to the reader to judge the quality of the keyphrases.

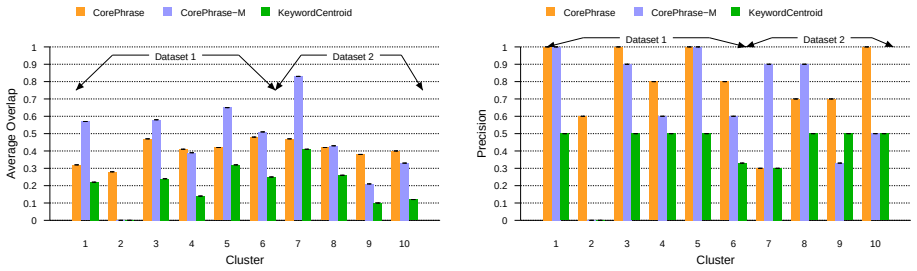
The first observation is that CorePhrase performs consistently better than the keyword centroid method. This is attributed to the keyphrases being in greater overlap with the reference topic than the naturally-shorter keywords. An interesting observation also is that CorePhrase-M, which is based on weighted

[6] Due to paper size limitation. A full list that can be obtained from: `http://pami.uwaterloo.ca/~hammouda/corephrase/results.html`

[7] Throughout this discussion the name CorePhrase will refer to both CorePhrase-1 and CorePhrase-2, while CorePhrase-M will refer to both CorePhrase-1M and CorePhrase-2M; unless otherwise specified.

Table 3. Performance of the CorePhrase algorithm

class	CorePhrase-1 overlap (avg,max)	CorePhrase-1 precision	CorePhrase-2 overlap (avg,max)	CorePhrase-2 precision	CorePhrase-1M overlap (avg,max)	CorePhrase-1M precision	CorePhrase-2M overlap (avg,max)	CorePhrase-2M precision	Keyword Centroid overlap (avg,max)	Keyword Centroid precision
Dataset 1										
canada transportation	(0.45,1.00)	1.00	(0.32,1.00)	1.00	(0.47,1.00)	1.00	(0.57,1.00)	1.00	(0.22,0.50)	0.5
winter weather canada	(0.22,0.67)	0.60	(0.28,0.67)	0.60	(0.00,0.00)	0.00	(0.00,0.00)	0.00	(0.00,0.00)	0.0
snowboarding skiing	(0.37,1.00)	1.00	(0.47,1.00)	1.00	(0.58,1.00)	0.90	(0.58,1.00)	0.90	(0.24,0.50)	0.5
river fishing	(0.41,1.00)	0.90	(0.41,1.00)	0.80	(0.43,1.00)	0.60	(0.39,1.00)	0.60	(0.14,0.50)	0.5
river rafting	(0.38,1.00)	1.00	(0.42,1.00)	1.00	(0.68,0.67)	0.90	(0.65,0.67)	1.00	(0.32,0.50)	0.5
black bear attacks	(0.45,1.00)	0.80	(0.48,1.00)	0.80	(0.47,1.00)	0.60	(0.51,1.00)	0.60	(0.25,0.33)	0.33
data set 1 average	(0.38,0.95)	0.88	(0.39,0.95)	0.87	(0.44,0.78)	0.67	(0.45,0.78)	0.68	(0.20,0.39)	0.39
Dataset 2										
co-operative education	(0.38,1.00)	0.20	(0.47,1.00)	0.30	(0.55,1.00)	0.80	(0.83,1.00)	0.90	(0.41,0.50)	0.3
career services	(0.37,1.00)	0.70	(0.42,1.00)	0.70	(0.58,1.00)	0.90	(0.43,1.00)	0.90	(0.26,0.50)	0.5
health services	(0.28,1.00)	0.70	(0.38,1.00)	0.70	(0.32,1.00)	0.50	(0.21,0.33)	0.33	(0.10,0.50)	0.5
campus network	(0.23,1.00)	1.00	(0.40,1.00)	1.00	(0.38,0.67)	0.20	(0.33,0.50)	0.50	(0.12,0.50)	0.5
data set 2 average	(0.31,1.00)	0.65	(0.42,1.00)	0.68	(0.46,0.92)	0.60	(0.45,0.71)	0.66	(0.22,0.46)	0.45
overall average	**(0.35,0.97)**	**0.79**	**(0.40,0.97)**	**0.79**	**(0.45,0.83)**	**0.64**	**(0.45,0.75)**	**0.67**	**(0.21,0.42)**	**0.41**



(a) Average Overlap (b) Precision

Fig. 3. CorePhrase Accuracy Comparison

words for phrase-scoring, and the keyword centroid follow the same trend. This is due to the link between the phrase scores and their constituent word scores.

Variants of the algorithm that use the depth feature are consistently better in terms of both overlap and precision. This is attributed to the fact that some common phrases usually appear at the end of each document (such as "last updated", "copyright", *etc.*). If depth information is ignored, these phrases make their way up the rank (*e.g.* the phrase "roger watt" in **campus network** cluster, which is the name of the network maintainer that appears at the end of each document.)

CorePhrase is somewhat better than its word-weighted counterpart (CorePhrase-M) in extracting the best phrase and ranking it among the top 10, where it achieves 97% overlap on average for the best phrase. The word-weighted variant achieves 83% maximum overlap on average for the best phrase.

When the top 10 keyphrases are considered as a *set*, the word-weighted variant achieves better average overlap performance in (45% for CorePhrase-M against 40% for CorePhrase). This is attributed to CorPhrase-M extracting heavily weighted words that often overlap with the topic, but not necessarily are the best descriptive phrases. (An example is the **winter weather canada** cluster.)

A final observation is that CorePhrase consistently achieves better precision than CorePhrase-M (79% for CorePhrase against 67% for CorePhrase-M.) This means that CorePhrase does not only find the best keyphrase, but ranks it higher than CorePhrase-M.

To summarize these findings: (a) CorePhrase is more accurate than keyword-based algorithms; (b) using phrase depth information achieves better performance; (c) using word-weights usually produces a better *set* of phrases; however, ignoring the word-weights usually produces better descriptive phrases; and (d) CorePhrase is able to identify the reference topic in the top few keyphrases.

5 Conclusions and Future Work

We presented the CorePhrase algorithm for accurately extracting descriptive keyphrases from text clusters. It is domain independent, and achieves high accuracy in identifying the topic of a document cluster compared with keyword-based cluster labeling algorithms.

The algorithm can be used to label clusters accurately, and to summarize clusters for other purposes such as calculating cluster-to-cluster or document-to-cluster similarity.

Future directions include using more features for the candidate phrases, based on heuristics of what constitutes a good phrase. Other ranking schemes are being investigated. Also the set of top keyphrases could be enhanced by removing spurious permutations and sub-phrases that appear in other phrases.

References

1. Turney, P.D.: Learning algorithms for keyphrase extraction. Information Retrieval **2** (2000) 303–336
2. Frank, E., Paynter, G., Witten, I., Gutwin, C., Nevill-Manning, C.: Domain-specific keyphrase extraction. In: 16th International Joint Conference on Artificial Intelligence (IJCAI-99), Stockholm, Sweden (1999) 668–673
3. Hammouda, K., Kamel, M.: Efficient phrase-based document indexing for web document clustering. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 1279–1296
4. Neto, J., Santos, A., Kaestner, C., Freitas, A.: Document clustering and text summarization. In: Proc. 4th International Conference Practical Applications of Knowledge Discovery and Data Mining (PADD-2000), London, UK (2000) 41–55
5. Bakus, J., Kamel, M., Carey, T.: Extraction of text phrases using hierarchical grammar. In: Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI 2002), Calgary, Canada (2002) 319–324
6. Mani, I., Bloedorn, E.: Multi-document summarization by graph search and matching. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-97), Providence, RI (1997) 622–628
7. Clifton, C., Cooley, R., Rennie, J.: TopCat: data mining for topic identification in a text corpus. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 949–964