COLING 2010: Posters Volume
Beijing, China, August 2010, pp. 365--373.

Conundrums in Unsupervised Keyphrase Extraction: Making Sense of the State-of-the-Art

Kazi Saidul Hasan and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
{saidul,vince}@hlt.utdallas.edu

Abstract

State-of-the-art approaches for unsupervised keyphrase extraction are typically evaluated on a single dataset with a single parameter setting. Consequently, it is unclear how effective these approaches are on a new dataset from a different domain, and how sensitive they are to changes in parameter settings. To gain a better understanding of state-of-the-art unsupervised keyphrase extraction algorithms, we conduct a systematic evaluation and analysis of these algorithms on a variety of standard evaluation datasets.

1 Introduction

The keyphrases for a given document refer to a group of phrases that *represent* the document. Although we often come across texts from different domains such as scientific papers, news articles and blogs, which are labeled with keyphrases by the authors, a large portion of the Web content remains untagged. While keyphrases are excellent means for providing a concise summary of a document, recent research results have suggested that the task of automatically identifying keyphrases from a document is by no means trivial. Researchers have explored both supervised and unsupervised techniques to address the problem of automatic keyphrase extraction. Supervised methods typically recast this problem as a binary classification task, where a model is trained on annotated data to determine whether a given phrase is a keyphrase or not (e.g., Frank et al. (1999), Turney (2000; 2003), Hulth (2003), Medelyan et al. (2009)). A disadvantage of supervised approaches is that they require a lot of training data and yet show bias towards the domain on which they are trained, undermining their ability to generalize well to new domains. Unsupervised approaches could be a viable alternative in this regard.

The unsupervised approaches for keyphrase extraction proposed so far have involved a number of techniques, including language modeling (e.g., Tomokiyo and Hurst (2003)), graph-based ranking (e.g., Zha (2002), Mihalcea and Tarau (2004), Wan et al. (2007), Wan and Xiao (2008), Liu et al. (2009a)), and clustering (e.g., Matsuo and Ishizuka (2004), Liu et al. (2009b)). While these methods have been shown to work well on a particular domain of text such as short paper abstracts and news articles, their effectiveness and portability across different domains have remained an unexplored issue. Worse still, each of them is based on a set of assumptions, which may only hold for the dataset on which they are evaluated.

Consequently, *we have little understanding of how effective the state-of the-art systems would be on a completely new dataset from a different domain.* A few questions arise naturally. How would these systems perform on a different dataset with their original configuration? What could be the underlying reasons in case they perform poorly? Is there any system that can generalize fairly well across various domains?

We seek to gain a better understanding of the state of the art in unsupervised keyphrase extraction by examining the aforementioned questions. More specifically, we compare five unsupervised keyphrase extraction algorithms on four corpora with varying domains and statistical characteristics. These algorithms represent the major directions in this research area, including Tf-Idf and four recently proposed systems, namely, TextRank (Mihalcea and Tarau, 2004), SingleRank (Wan and Xiao, 2008), ExpandRank (Wan and Xiao, 2008), and a clustering-based approach (Liu et al., 2009b). Since none of these systems (except TextRank) are publicly available, we re-implement all of them and make them freely available for research purposes.[1] To our knowledge, this is the *first* attempt to compare the performance of state-of-the-art unsupervised keyphrase extraction systems on multiple datasets.

2 Corpora

Our four evaluation corpora belong to different domains with varying document properties. Table 1 provides an overview of each corpus.

The **DUC-2001** dataset (Over, 2001), which is a collection of 308 news articles, is annotated by Wan and Xiao (2008). We report results on all 308 articles in our evaluation.

The **Inspec** dataset is a collection of 2,000 abstracts from journal papers including the paper title. Each document has two sets of keyphrases assigned by the indexers: the *controlled keyphrases*, which are keyphrases that appear in the *Inspec* thesaurus; and the *uncontrolled keyphrases*, which do not necessarily appear in the thesaurus. This is a relatively popular dataset for automatic keyphrase extraction, as it was first used by Hulth (2003) and later by Mihalcea and Tarau (2004) and Liu et al. (2009b). In our evaluation, we use the set of 500 abstracts designated by these previous approaches as the test set and its set of uncontrolled keyphrases. Note that the average document length for this dataset is the smallest among all our datasets.

The **NUS Keyphrase Corpus** (Nguyen and Kan, 2007) includes 211 scientific conference papers with lengths between 4 to 12 pages. Each paper has one or more sets of keyphrases assigned by its authors and other annotators. We use all the 211 papers in our evaluation. Since the number of annotators can be different for different documents and the annotators are not specified along with the annotations, we decide to take the union of all the gold standard keyphrases from all the sets to construct one single set of annotation for each paper. As Table 1 shows, each NUS paper, both in terms of the average number of tokens (8291) and candidate phrases (2027) per paper, is more than five times larger than any document from any other corpus. Hence, the number of candidate keyphrases that can be extracted is potentially large, making this corpus the most challenging of the four.

Finally, the **ICSI meeting corpus** (Janin et al., 2003), which is annotated by Liu et al. (2009a), includes 161 meeting transcriptions. Following Liu et al., we remove topic segments marked as 'chitchat' and 'digit' from the dataset and use all the remaining segments for evaluation. Each transcript contains three sets of keyphrases produced by the same three human annotators. Since it is possible to associate each set of keyphrases with its annotator, we evaluate each system on this dataset three times, once for each annotator, and report the average score. Unlike the other three datasets, the gold standard keys for the ICSI corpus are mostly unigrams.

3 Unsupervised Keyphrase Extractors

A generic unsupervised keyphrase extraction system typically operates in three steps (Section 3.1), which will help understand the unsupervised systems explained in Section 3.2.

3.1 Generic Keyphrase Extractor

Step 1: Candidate lexical unit selection The first step is to filter out unnecessary word tokens from the input document and generate a list of potential keywords using heuristics. Commonly used heuristics include (1) using a stop word list to remove non-keywords (e.g., Liu et al. (2009b)) and (2) allowing words with certain part-of-speech tags (e.g., nouns, adjectives, verbs) to be considered candidate keywords (Mihalcea and Tarau (2004), Liu et al. (2009a), Wan and Xiao (2008)). In all of our experiments, we follow Wan and Xiao (2008) and select as candidates words with the following Penn Treebank tags: NN, NNS, NNP, NNPS, and JJ, which are obtained using the Stanford POS tagger (Toutanova and Manning, 2000).

[1]See `http://www.hlt.utdallas.edu/~saidul/code.html` for details.

	Corpora			
	DUC-2001	*Inspec*	NUS	ICSI
Type	News articles	Paper abstracts	Full papers	Meeting transcripts
# Documents	308	500	211	161
# Tokens/Document	876	134	8291	1611
# Candidate words/Document	312	57	3271	453
# Candidate phrases/Document	207	34	2027	296
# Tokens/Candidate phrase	1.5	1.7	1.6	1.5
# Gold keyphrases	2484	4913	2327	582
# Gold keyphrases/Document	8.1	9.8	11.0	3.6
U/B/T/O distribution (%)	17/61/18/4	13/53/25/9	27/50/16/7	68/29/2/1
# Tokens/Gold keyphrase	2.1	2.3	2.1	1.3

Table 1: Corpus statistics for the four datasets used in this paper. A candidate word/phrase, typically a sequence of one or more adjectives and nouns, is extracted from the document initially and considered a potential keyphrase. The U/B/T/O distribution indicates how the gold standard keys are distributed among unigrams, bigrams, trigrams, and other higher order n-grams.

Step 2: Lexical unit ranking Once the candidate list is generated, the next task is to rank these lexical units. To accomplish this, it is necessary to build a representation of the input text for the ranking algorithm. Depending on the underlying approach, each candidate word is represented by its syntactic and/or semantic relationship with other candidate words. The relationship can be defined using co-occurrence statistics, external resources (e.g., neighborhood documents, Wikipedia), or other syntactic clues.

Step 3: Keyphrase formation In the final step, the ranked list of candidate words is used to form keyphrases. A candidate phrase, typically a sequence of nouns and adjectives, is selected as a keyphrase if (1) it includes one or more of the top-ranked candidate words (Mihalcea and Tarau (2004), Liu et al. (2009b)), or (2) the sum of the ranking scores of its constituent words makes it a top scoring phrase (Wan and Xiao, 2008).

3.2 The Five Keyphrase Extractors

As mentioned above, we re-implement five unsupervised approaches for keyphrase extraction. Below we provide a brief overview of each system.

3.2.1 Tf-Idf

Tf-Idf assigns a score to each term t in a document d based on t's frequency in d (term frequency) and how many other documents include t (inverse document frequency) and is defined as:

$$\mathrm{tfidf}_t = \mathrm{tf}_t \times log(D/D_t) \tag{1}$$

where D is the total number of documents and D_t is the number of documents containing t.

Given a document, we first compute the Tf-Idf score of each candidate word (see Step 1 of the generic algorithm). Then, we extract all the longest n-grams consisting of candidate words and score each n-gram by summing the Tf-Idf scores of its constituent unigrams. Finally, we output the top N n-grams as keyphrases.

3.2.2 TextRank

In the TextRank algorithm (Mihalcea and Tarau, 2004), a text is represented by a graph. Each vertex corresponds to a word type. A weight, w_{ij}, is assigned to the edge connecting the two vertices, v_i and v_j, and its value is the number of times the corresponding word types co-occur within a window of W words in the associated text. The goal is to (1) compute the score of each vertex, which reflects its *importance*, and then (2) use the word types that correspond to the highest-scored vertices to form keyphrases for the text. The score for v_i, $S(v_i)$, is initialized with a default value and is computed in an iterative manner until convergence using this recursive formula:

$$S(v_i) = (1-d) + d \times \sum_{v_j \epsilon Adj(v_i)} \frac{w_{ji}}{\sum_{v_k \epsilon Adj(v_j)} w_{jk}} S(v_j) \tag{2}$$

where $Adj(v_i)$ denotes v_i's neighbors and d is the damping factor set to 0.85 (Brin and Page, 1998). Intuitively, a vertex will receive a high score if it has many high-scored neighbors. As noted before, after convergence, the $T\%$ top-scored vertices are selected as keywords. Adjacent keywords are then collapsed and output as a keyphrase.

According to Mihalcea and Tarau (2004), TextRank's best score on the *Inspec* dataset is achieved when only nouns and adjectives are used to create a uniformly weighted graph for the text under consideration, where an edge connects two word types only if they co-occur within a window of two words. Hence, our implementation of TextRank follows this configuration.

3.2.3 SingleRank

SingleRank (Wan and Xiao, 2008) is essentially a TextRank approach with three major differences. First, while each edge in a TextRank graph (in Mihalcea and Tarau's implementation) has the same weight, each edge in a SingleRank graph has a weight equal to the number of times the two corresponding word types co-occur. Second, while in TextRank only the word types that correspond to the top-ranked vertices can be used to form keyphrases, in SingleRank, we do not filter out any low-scored vertices. Rather, we (1) score each candidate keyphrase, which can be any longest-matching sequence of nouns and adjectives in the text under consideration, by summing the scores of its constituent word types obtained from the SingleRank graph, and (2) output the N highest-scored candidates as the keyphrases for the text. Finally, SingleRank employs a window size of 10 rather than 2.

3.2.4 ExpandRank

ExpandRank (Wan and Xiao, 2008) is a TextRank extension that exploits neighborhood knowledge for keyphrase extraction. For a given document d, the approach first finds its k nearest neighboring documents from the accompanying document collection using a similarity measure (e.g., cosine similarity). Then, the graph for d is built using the co-occurrence statistics of the candidate words collected from the document itself and its k nearest neighbors.

Specifically, each document is represented by a term vector where each vector dimension corresponds to a word type present in the document and its value is the Tf-Idf score of that word type for that document. For a given document d_0, its k nearest neighbors are identified, and together they form a larger document set of k+1 documents, $D = \{d_0, d_1, d_2, ..., d_k\}$. Given this document set, a graph is constructed, where each vertex corresponds to a candidate word type in D, and each edge connects two vertices v_i and v_j if the corresponding word types co-occur within a window of W words in the document set. The weight of an edge, $w(v_i, v_j)$, is computed as follows:

$$w(v_i, v_j) = \sum_{d_k \epsilon D} sim(d_0, d_k) \times freq_{d_k}(v_i, v_j) \qquad (3)$$

where $sim(d_0, d_k)$ is the cosine similarity between d_0 and d_k, and $freq_{d_k}(v_i, v_j)$ is the co-occurrence frequency of v_i and v_j in document d_k. Once the graph is constructed, the rest of the procedure is identical to SingleRank.

3.2.5 Clustering-based Approach

Liu et al. (2009b) propose to cluster candidate words based on their semantic relationship to ensure that the extracted keyphrases *cover* the entire document. The objective is to have each cluster represent a unique aspect of the document and take a representative word from each cluster so that the document is covered from all aspects.

More specifically, their algorithm (henceforth referred to as KeyCluster) first filters out the stop words from a given document and treats the remaining unigrams as candidate words. Second, for each candidate, its relatedness with another candidate is computed by (1) counting how many times they co-occur within a window of size W in the document and (2) using Wikipedia-based statistics. Third, candidate words are clustered based on their relatedness with other candidates. Three clustering algorithms are used of which spectral clustering yields the best score. Once the clusters are formed, one representative word, called an exemplar term, is picked from each cluster. Finally, KeyCluster extracts from the document all the longest n-grams starting with zero or more adjectives and ending with one or more nouns, and if such an n-gram includes one or more exemplar words, it is selected as a keyphrase. As a post-processing step, a frequent word list generated from Wikipedia is used to filter out the frequent unigrams that are selected as keyphrases.

4 Evaluation

4.1 Experimental Setup

TextRank and SingleRank setup Following Mihalcea and Tarau (2004) and Wan and Xiao (2008), we set the co-occurrence window size for TextRank and SingleRank to 2 and 10, respectively, as these parameter values have yielded the best results for their evaluation datasets.

ExpandRank setup Following Wan and Xiao (2008), we find the 5 nearest neighbors for each document from the remaining documents in the same corpus. The other parameters are set in the same way as in SingleRank.

KeyCluster setup As argued by Liu et al. (2009b), Wikipedia-based relatedness is computationally expensive to compute. As a result, we follow them by computing the *co-occurrence-based* relatedness instead, using a window of size 10. Then, we cluster the candidate words using spectral clustering, and use the frequent word list that they generously provided us to post-process the resulting keyphrases by filtering out those that are frequent unigrams.

4.2 Results and Discussion

In an attempt to gain a better insight into the five unsupervised systems, we report their performance in terms of precision-recall curves for each of the four datasets (see Figure 1). This contrasts with essentially all previous work, where the performance of a keyphrase extraction system is reported in terms of an F-score obtained via a particular parameter setting on a particular dataset. We generate the curves for each system as follows. For Tf-Idf, SingleRank, and ExpandRank, we vary the number of keyphrases, N, predicted by each system. For TextRank, instead of varying the number of predicted keyphrases, we vary T, the percentage of top-scored vertices (i.e., unigrams) that are selected as keywords at the end of the ranking step. The reason is that TextRank only imposes a ranking on the unigrams but not on the keyphrases generated from the high-ranked unigrams. For KeyCluster, we vary the number of clusters produced by spectral clustering rather than the number of predicted keyphrases, again because KeyCluster does not impose a ranking on the resulting keyphrases. In addition, to give an estimate of how each system performs in terms of F-score, we also plot curves corresponding to different F-scores in these graphs.

Tf-Idf Consistent with our intuition, the precision of Tf-Idf drops as recall increases. Although it is the simplest of the five approaches, Tf-Idf is the best performing system on all but the *Inspec* dataset, where TextRank and KeyCluster beat Tf-Idf on just a few cases. It clearly outperforms all other systems for NUS and ICSI.

TextRank The TextRank curves show a different progression than Tf-Idf: precision does not drop as much when recall increases. For instance, in case of DUC and ICSI, precision is not sensitive to changes in recall. Perhaps somewhat surprisingly, its precision increases with recall for *Inspec*, allowing it to even reach a point (towards the end of the curve) where it beats Tf-Idf. While additional experiments are needed to determine the reason for this somewhat counter-intuitive result, we speculate that this may be attributed to the fact that the TextRank curves are generated by progressively increasing T (i.e., the percentage of top-ranked vertices/unigrams that are used to generate keyphrases) rather than the number of predicted keyphrases, as mentioned before. Increasing T does not necessarily imply an increase in the number of predicted keyphrases, however. To see the reason, consider an example in which we want TextRank to extract the keyphrase "advanced machine learning" for a given document. Assume that TextRank ranks the unigrams "advanced", "learning", and "machine" first, second, and third, respectively in its ranking step. When $T = \frac{2}{n}$, where n denotes the total number of candidate unigrams, only the two highest-ranked unigrams (i.e., "advanced" and "learning") can be used to form keyphrases. This implies that "advanced" and "learning" will each be predicted as a keyphrase, but "advanced machine learning" will not. However, when $T = \frac{3}{n}$, all three unigrams can be used to form a keyphrase, and since TextRank collapses unigrams adjacent to each other in the text to form a keyphrase, it will correctly predict "advanced machine learning" as a keyphrase. Note that as we increase T from $\frac{2}{n}$ to $\frac{3}{n}$, recall increases, and so does precision, since

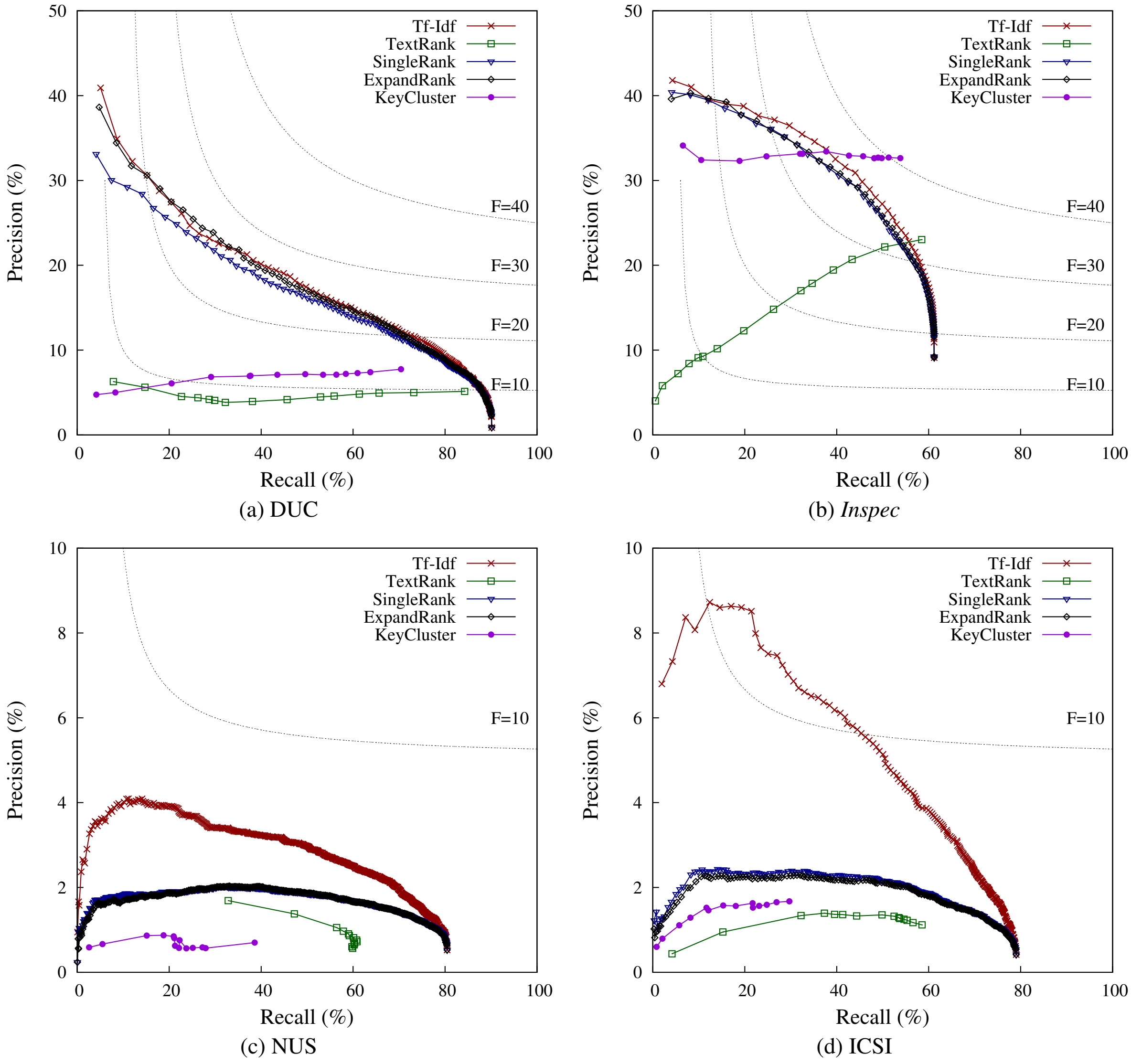

Figure 1: Precision-recall curves for all four datasets

“advanced” and “learning” are now combined to form one keyphrase (and hence the number of predicted keyphrases decreases). In other words, it is possible to see a simultaneous rise in precision and recall in a TextRank curve. A natural question is: why does is happen only for *Inspec* but not the other datasets? The reason could be attributed to the fact that *Inspec* is composed of abstracts: since the number of keyphrases that can be generated from these short documents is relatively small, precision may not drop as severely as with the other datasets even when all of the unigrams are used to form keyphrases.

On average, TextRank performs much worse compared to Tf-Idf. The curves also prove TextRank’s sensitivity to T on *Inspec*, but not on the other datasets. This certainly gives more insight into TextRank since it was evaluated on *Inspec* only for T=33% by Mihalcea and Tarau (2004).

SingleRank SingleRank, which is supposed to be a simple variant of TextRank, surprisingly exhibits very different performance. First, it shows a more intuitive nature: precision drops as recall increases. Second, SingleRank outperforms TextRank by big margins on all the datasets. Later, we will examine which of the differences between them is responsible for the differing performance.

	DUC		*Inspec*		NUS		ICSI	
	Parameter	F	Parameter	F	Parameter	F	Parameter	F
Tf-Idf	$N = 14$	27.0	$N = 14$	36.3	$N = 60$	6.6	$N = 9$	12.1
TextRank	$T = 100\%$	9.7	$T = 100\%$	33.0	$T = 5\%$	3.2	$T = 25\%$	2.7
SingleRank	$N = 16$	25.6	$N = 15$	35.3	$N = 190$	3.8	$N = 50$	4.4
ExpandRank	$N = 13$	26.9	$N = 15$	35.3	$N = 177$	3.8	$N = 51$	4.3
KeyCluster	$m = 0.9n$	14.0	$m = 0.9n$	40.6	$m = 0.25n$	1.7	$m = 0.9n$	3.2

Table 2: Best parameter settings. N is the number of predicted keyphrases, T is the percentage of vertices selected as keywords in TextRank, m is the number of clusters in KeyCluster, expressed in terms of n, the fraction of candidate words.

ExpandRank Consistent with Wan and Xiao (2008), ExpandRank beats SingleRank on DUC when a small number of phrases are predicted, but their difference diminishes as more phrases are predicted. On the other hand, their performance is indistinguishable from each other on the other three datasets. A natural question is: why does ExpandRank improve over SingleRank only for DUC but not for the other datasets? To answer this question, we look at the DUC articles and find that in many cases, the 5-nearest neighbors of a document are on the same topic involving the same entities as the document itself, presumably because many of these news articles are simply updated versions of an evolving event. Consequently, the graph built from the neighboring documents is helpful for predicting the keyphrases of the given document. Such topic-wise similarity among the nearest documents does not exist in the other datasets, however.

KeyCluster As in TextRank, KeyCluster does not always yield a drop in precision as recall improves. This, again, may be attributed to the fact that the KeyCluster curves are generated by varying the number of clusters rather than the number of predicted keyphrases, as well as the way keyphrases are formed from the exemplars. Another reason is that the frequent Wikipedia unigrams are excluded during post-processing, making KeyCluster more resistant to precision drops. Overall, KeyCluster performs slightly better than TextRank on DUC and ICSI, yields the worst performance on NUS, and scores the best on *Inspec* when the number of clusters is high. These results seem to suggest that KeyCluster works better if more clusters are used.

Best parameter settings Table 2 shows for each system the parameter values yielding the best F-score on each dataset. Two points deserve mention. First, in comparison to SingleRank and ExpandRank, Tf-Idf outputs fewer keyphrases to achieve its best F-score on most datasets. Second, the systems output more keyphrases on NUS than on other datasets to achieve their best F-scores (e.g., 60 for Tf-Idf, 190 for SingleRank, and 177 for ExpandRank). This can be attributed in part to the fact that the F-scores on NUS are low for all the systems and exhibit only slight changes as we output more phrases.

Our re-implementations Do our duplicated systems yield scores that match the original scores? Table 3 sheds light on this question.

First, consider KeyCluster, where our score lags behind the original score by approximately 5%. An examination of Liu et al.'s (2009b) results reveals a subtle caveat in keyphrase extraction evaluations. In *Inspec*, not all gold-standard keyphrases appear in their associated document, and as a result, none of the five systems we consider in this paper can achieve a recall of 100. While Mihalcea and Tarau (2004) and our re-implementations use *all* of these gold-standard keyphrases in our evaluation, Hulth (2003) and Liu et al. address this issue by using as gold-standard keyphrases only those that appear in the corresponding document when computing recall.[2] This explains why our KeyCluster score (38.9) is lower than the original score (43.6). If we follow Liu et al.'s way of computing recall, our re-implementation score goes up to 42.4, which lags behind their score by only 1.2.

Next, consider TextRank, where our score lags behind Mihalcea and Tarau's original score by more than 25 points. We verified our implementation against a publicly available implementation

[2] As a result, Liu et al. and Mihalcea and Tarau's scores are not directly comparable, but Liu et al. did not point this out while comparing scores in their paper.

	Dataset	F-score	
		Original	Ours
Tf-Idf	DUC	25.4	25.7
TextRank	*Inspec*	36.2	10.0
SingleRank	DUC	27.2	24.9
ExpandRank	DUC	31.7	26.4
KeyCluster	*Inspec*	43.6	38.9

Table 3: Original vs. re-implementation scores

of TextRank[3], and are confident that our implementation is correct. It is also worth mentioning that using our re-implementation of SingleRank, we are able to match the best scores reported by Mihalcea and Tarau (2004) on *Inspec*.

We score 2 and 5 points less than Wan and Xiao's (2008) implementations of SingleRank and ExpandRank, respectively. We speculate that document pre-processing (e.g., stemming) has contributed to the discrepancy, but additional experiments are needed to determine the reason.

SingleRank vs. TextRank Figure 1 shows that SingleRank behaves very differently from TextRank. As mentioned in Section 3.2.3, the two algorithms differ in three major aspects. To determine which aspect is chiefly responsible for the large difference in their performance, we conduct three "ablation" experiments. Each experiment modifies exactly one of these aspects in SingleRank so that it behaves like TextRank, effectively ensuring that the two algorithms differ only in the remaining two aspects. More specifically, in the three experiments, we (1) change SingleRank's window size to 2, (2) build an unweighted graph for SingleRank, and (3) incorporate TextRank's way of forming keyphrases into SingleRank, respectively. Figure 2 shows the resultant curves along with the SingleRank and TextRank curves on *Inspec* taken from Figure 1b. As we can see, the way of forming phrases, rather than the window size or the weight assignment method, has the largest impact on the scores. In fact, after incorporating TextRank's way of forming phrases, SingleRank exhibits a remarkable drop in performance, yielding a curve that resembles the TextRank curve. Also note that SingleRank achieves better recall values than TextRank. To see the reason, recall that TextRank requires that every word of a gold keyphrase must appear among the top-

[3]`http://github.com/sharethis/textrank`

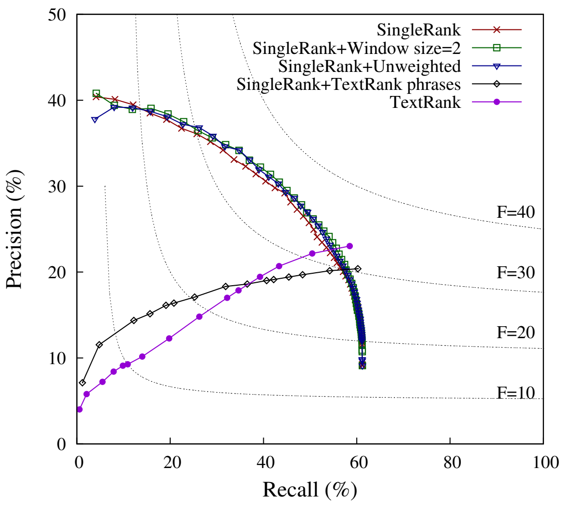

Figure 2: Ablation results for SingleRank on *Inspec*

ranked unigrams. This is a fairly strict criterion, especially in comparison to SingleRank, which does not require all unigrams of a gold keyphrase to be present in the top-ranked list. We observe similar trends for the other datasets.

5 Conclusions

We have conducted a systematic evaluation of five state-of-the-art unsupervised keyphrase extraction algorithms on datasets from four different domains. Several conclusions can be drawn from our experimental results. First, to fully understand the strengths and weaknesses of a keyphrase extractor, it is essential to evaluate it on multiple datasets. In particular, evaluating it on a single dataset has proven inadequate, as good performance can sometimes be achieved due to certain statistical characteristics of a dataset. Second, as demonstrated by our experiments with TextRank and SingleRank, post-processing steps such as the way of forming keyphrases can have a large impact on the performance of a keyphrase extractor. Hence, it may be worthwhile to investigate alternative methods for extracting candidate keyphrases (e.g., Kumar and Srinathan (2008), You et al. (2009)). Finally, despite the large amount of recent work on unsupervised keyphrase extractor, our results indicated that Tf-Idf remains a strong baseline, offering very robust performance across different datasets.

Acknowledgments

We thank the three anonymous reviewers for their comments. Many thanks to Anette Hulth and Yang Liu for providing us with the *Inspec* and ICSI datasets; Rada Mihalcea, Paco Nathan, and Xiaojun Wan for helping us understand their algorithms/implementations; and Peng Li for providing us with the frequent word list that he and his co-authors used in their paper. This work was supported in part by NSF Grant IIS-0812261.

References

Brin, Sergey and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1–7):107–117.

Frank, Eibe, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 668–673.

Hulth, Anette. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Janin, Adam, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The ICSI meeting corpus. In *Proceedings of 2003 IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 364–367.

Kumar, Niraj and Kannan Srinathan. 2008. Automatic keyphrase extraction from scientific documents using n-gram filtration technique. In *Proceedings of the Eighth ACM Symposium on Document Engineering*, pages 199–208.

Liu, Feifan, Deana Pennell, Fei Liu, and Yang Liu. 2009a. Unsupervised approaches for automatic keyword extraction using meeting transcripts. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 620–628.

Liu, Zhiyuan, Peng Li, Yabin Zheng, and Maosong Sun. 2009b. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 257–266.

Matsuo, Yutaka and Mitsuru Ishizuka. 2004. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169.

Medelyan, Olena, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1318–1327.

Mihalcea, Rada and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Nguyen, Thuy Dung and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the International Conference on Asian Digital Libraries*, pages 317–326.

Over, Paul. 2001. Introduction to DUC-2001: An intrinsic evaluation of generic news text summarization systems. In *Proceedings of the 2001 Document Understanding Conference*.

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

Toutanova, Kristina and Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language processing and Very Large Corpora*, pages 63–70.

Turney, Peter. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2:303–336.

Turney, Peter. 2003. Coherent keyphrase extraction via web mining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 434–439.

Wan, Xiaojun and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 855–860.

Wan, Xiaojun, Jianwu Yang, and Jianguo Xiao. 2007. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559.

You, Wei, Dominique Fontaine, and Jean-Paul Barthès. 2009. Automatic keyphrase extraction with a refined candidate set. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 576–579.

Zha, Hongyuan. 2002. 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.