Witt, Nils; Milz, Tobias; Seifert, Christin

Conference Paper

Most Important First – Keyphrase Scoring for Improved Ranking in Settings With Limited Keyphrases

Suggested Citation: Witt, Nils; Milz, Tobias; Seifert, Christin (2018) : Most Important First – Keyphrase Scoring for Improved Ranking in Settings With Limited Keyphrases, In: Soldatova, Larisa et al. (Ed.): Discovery Science, 21st International Conference, DS 2018, Limassol, Cyprus, October 29–31, 2018, Proceedings, ISBN 978-3-030-01771-2, Springer, Cham, pp. 373-385, https://doi.org/10.1007/978-3-030-01771-2_24

This Version is available at:
http://hdl.handle.net/11108/398

Most Important First – Keyphrase Scoring for Improved Ranking in Settings With Limited Keyphrases

Nils Witt[1], Tobias Milz[2], and Christin Seifert[3]

[1] ZBW - Leibniz Information Centre for Economics, Kiel, Germany
n.witt@zbw.eu,
[2] University of Passau, Passau, Germany
tobias.milz@uni-passau.de
[3] University of Twente, Enschede, The Netherlands
c.seifert@utwente.nl

Abstract. Automatic keyphrase extraction attempts to capture keywords that accurately and extensively describe the document while being comprehensive at the same time. Unsupervised algorithms for extractive keyphrase extraction, i.e. those that filter the keyphrases from the text without external knowledge, generally suffer from low precision and low recall. In this paper, we propose a scoring of the extracted keyphrases as post-processing to rerank the list of extracted phrases in order to improve precision and recall particularly for the top phrases. The approach is based on the tf-idf score of the keyphrases and is agnostic of the underlying method used for the initial extraction of the keyphrases. Experiments show an increase of up to 14% at 5 keyphrases in the F1-metric on the most difficult corpus out of 4 corpora. We also show that this increase is mostly due to an increase on documents with very low F1-scores. Thus, our scoring and aggregation approach seems to be a promising way for robust, unsupervised keyphrase extraction with a special focus on the most important keyphrases.

1 Introduction

Automatic text summarization is applied in Natural Language Processing and Information Retrieval to provide a quick overview of longer texts. More specifically, automatic keyphrase extraction methods are employed to allow human readers to quickly assess relevant concepts in the text. As was pointed out by Miller on average people can hold 7 (±2) items in their short-term memory [19]. This indicates that people who read keyphrase lists that exceed 5 to 9 keyphrases forget the first items of the list as they reach the end. Thus 5 keyphrases per document is an optimal number regarding human perception and it is worth optimizing keyphrase extraction methods towards this threshold. In this paper, we investigate a method-agnostic approach that enhances the important first part of a keyphrase list. That is, given a long (e.g. 20 keyphrases) ranked list of keyphrases extracted by some keyphrase extraction method, our tf-idf-based approach reorganizes that list such that more suitable keywords are at the top of the list. The tf-idf value has been shown to be an informative feature for keywords [9]. Thus, we apply tf-idf-based scoring on extracted keyphrases a-posteriori, assuming that words with a high tf-idf value are more likely to be high quality keywords or part of high quality keyphrases (i.e. a short sequence of words)[4]. Concretely, the contributions of this paper are the following:

1. We propose a tf-idf-based scoring and re-ranking of keyphrases which is agnostic to the underlying keyphrase extraction method.
2. In experiments on four different corpora, we show that tf-idf-based scoring can enhance the precision and recall of well-known keyphrase extraction algorithms.

The source code and the data that were used to conduct the experiments are publicly available[5]. After reviewing related work we explain the details on keyphrase scoring. Then we report on the experimental setup (section 4) and results (section 5). Finally, we discuss and conclude our work in section 6.

2 Related Work

Due to the rapid growth of available information, the ability to automatically generate summarized short texts has become a valuable tool for many Natural Language Processing tasks. Summarization approaches aim to generate sentences, keyphrases or keywords that condense the information provided by a document. Summaries that are extracted directly from the document and abstractive summaries that are created based on the content of the document with words not necessarily appearing in the document, are two main concepts of these approaches [15]. This paper focuses on extractive summaries and in particular *Rake* [21] and *TextRank* [18]. *TextRank* similarly to Wan and Xiao [24] and Liu et. al [11] searches for POS tag combinations in the document to identify possible keyphrase candidates. Other systems use different NLP methods and heuristics such as the removal of stop words [14], finding matching n-grams in Wikipedia articles [7] or extracting n-grams with specific syntactic patterns [26, 10, 16]. As these methods often produce too many and poor candidates for long documents, a second step is required to separate those candidates that are more likely keyphrases. Previous approaches [6, 22, 26] applied supervised binary classification techniques to select the keyphrases from the candidates. Binary classification, however, yields the problem that a candidate is simply deemed as either worthy or not worthy and their relative importance is not compared to the other candidates. As a result, other approaches adapted a ranking based system such as the unsupervised graph-based approach implemented by *TextRank*, *CollabRank* [23] and *TopicRank* [2]. Here, each candidate is represented as a node in

[4] Throughout the document we will use the unifying term *keyphrase* to refer to keywords as well as keyphrases as defined in the Introduction.

[5] https://doi.org/10.5281/zenodo.1435518

a graph and its importance is recursively computed based on the number of connections the node has and how important the connected candidates are. Among these systems *TextRank* has established itself as the most popular graph-based ranking system and was also adapted in topic-based clustering concepts such as *TopicalPageRank* [13] and *CommunityCluster* [7]. Both systems apply *TextRank* multiple times (once for each topic in the document) and add to the importance of the topic to the computation. More recently, topic-based keyphrase extraction was done using topic modeling to find clusters of co-occurring words, which were used to construct candidate keyphrases [5]. Those candidates were then ranked according to several different properties, with the best-performing one being *purity* which prefers keyphrases consisting of words that are frequent in a given topic and rare in other topics. Sequence-to-sequence models based on recurrent neural networks have shown to perform very well not just on the task of keyphrase extraction but also on the more challenging task of keyphrase prediction, which includes finding keyphrases that do not appear in the text [17]. Similar results were achieved using convolutional neural networks [27], but due to the concurrent nature of convolutional neural networks the training time could be reduced by a factor of 5-6.

For our experiments we focus on fast, unsupervised methods with solid implementations as they are not constrained to extract only those keyphrases they saw during training. Since these algorithms do not rely on training data, they also have a larger domain of application. We also include the tf-idf baseline as still it is still a comparative baseline, despite its simplicity. Textrank remains a very important method and is still used by the community [12, 8, 17]. Rake was chosen as is it was able to outperform Textrank while scaling much better on longer documents (see figure 1.7 in [21]).

3 Approach

In this section, we explain how we assign scores to keyphrases irrespective of the algorithm that extracted it. An overview of the approach is shown in figure 1.

3.1 Keyphrase Scoring

We follow the idea described in section 1 to create ranked keyphrase lists per document. The keyphrases in those lists can come from one or more keyphrase extracting algorithms. The lists are supposed to have the property that, on average, the highest ranked keyphrases are the ”best” keyphrases, followed by the second highest ranked keyphrase, etc. Moreover, we act on the assumption that the gold standard keyphrases are ”good”, which allows us to formulate our expectations towards the ranked lists more formally: For any given document, the probability of a higher ranked keyphrase of being in the set of gold standard keyphrases is higher than the probability of a keyphrase with a lower rank:

$$rank(kp) \propto P(kp \in GS), \tag{1}$$

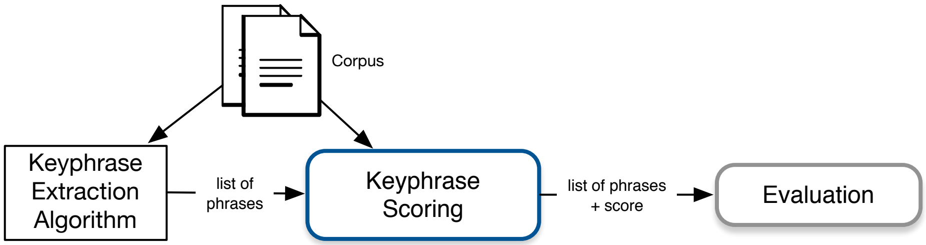


Fig. 1: Overview of keyphrase scoring approach and its evaluation. Standard evaluation measures precision, recall and F1-measure are compared before and after scoring and reranking.

where $P(kp \in GS)$ is the probability of keyphrase kp being in the set of gold standard keyphrases GS and $rank(kp)$ is the rank of the keyphrase. Since tf-idf offers only scores for single words rather than phrases we need a mechanism by which the score of a phrase can be computed in order to rank a set of keyphrases. We use a simple weighting approach:

$$score(kp) = \left(\sum_{i=1}^{|\,t\,|} ts_i \right) \cdot (1 - \alpha \cdot |\,t\,|), \qquad (2)$$

where $|\,t\,|$ is the number of tokens in the keyphrase and ts_i is the token score (i.e. the tf-idf value of the token) of the token at position i. The parameter α determines how long phrases are penalized. In our experiments, we set $\alpha = \frac{1}{10}$ meaning that keyphrases with 10 tokens are always assigned a keyphrase score of 0.0 and keyphrases with more than 10 tokens get a negative score. This property might seem undesirable, but is reasonable as most gold standard keyphrases in the corpora used have less than 6 words (see table 1). Therefore it is reasonable to penalize long extracted keyphrases in this scenario. Different keyphrase extraction algorithms can be used to extract keyphrases for a corpus. The quality of those extractions can be evaluated using gold standard keyphrases GS obtaining precision, recall and F1. For keyphrase scoring the score values (in our case tf-idf) are calculated on the corpus and used to rank or rerank the output of the keyphrase extraction step. The reranked lists are then evaluated in a similar fashion against the gold standard GS.

3.2 Keyphrase Extraction Ensembles

The introduced keyphrase scoring provides a unified, comparable score for all phrases, independent of the respective extraction algorithm. Thus, this score can be used to combine the output from different keyphrases extraction methods (as depicted in figure 2), similarly to the idea of bagging in machine learning [3]. Therefore we also measure the performance of multiple keyphrase extraction methods combined to see whether the overall performance can be enhanced

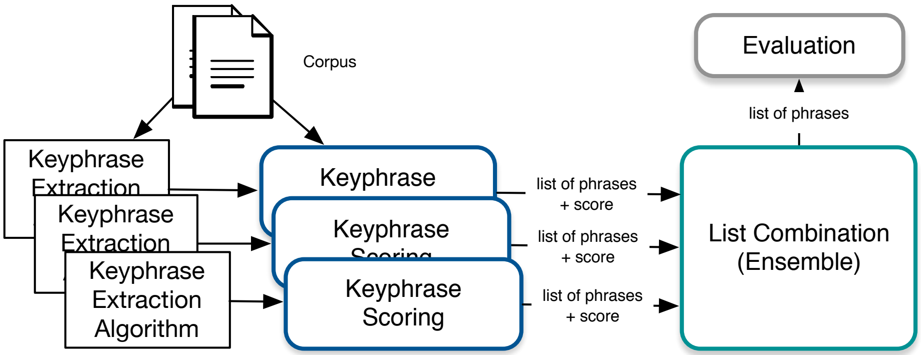


Fig. 2: Overview of the ensemble approach and its evaluation. Multiple keyphrase extraction methods combined using our keyphrase scoring approach.

in comparison to the individual methods. The keyphrase extraction ensemble works as follows: Given a document, k different keyphrase extraction methods can be applied, resulting in k result lists. Each of the keyphrases in the list might or might not have an algorithm-specific score, depending on the extraction method used. We remove duplicates from those lists, score each of the keyphrases as described in the previous section and create a unified result list containing keyphrases ordered by descending score.

4 Experimental Setup

In this section, we describe the data sets and the base keyphrase algorithms as well as the evaluation methodology for the experiments.

4.1 Data Sets

To evaluate our approach we used four corpora containing abstracts of scientific publications. Although the datasets are homogeneous as they all contain abstracts of scientific publications, the corresponding keyphrases exhibit vastly different characteristics. They not only differ in the average number of keyphrases per document but also in the average number of words per keyphrase (see table 1). SemEval for example contains keyphrases that are as long as a whole sentence (e.g. *controllable size reduction with high resolution towards the observation of size- and quantum effects*), also there are unsuitable keywords (e.g. *defense simulation platform is discussed in*) which makes this corpus very challenging. The Scopus corpus is challenging for another reason. There are documents whose keyphrases mostly or only consist of abbreviations (e.g. *ARPA, CSPs, ERP, IaaS, NIST, PaaS, SaaS*) that are not mentioned in the text, leading to zero scores on all performance metrics for that document. KP20k stands out as it contains much more abstracts than the other corpora of which we only used 100,000 randomly sampled documents due to computational constraints.

Table 1: Data Set Overview. The abbreviation KP refers to keyphrases. |KP|denotes the number of words in a keyphrase, σ the standard deviation.

Corpus	**Type**	**#Docs**	**# KP**	**∅ KP/doc** [σ]	**∅ |KP|**[σ]
Inspec [10]	abstracts	2000	19275	9.64 [4.80]	2.3 [0.44]
SemEval2017 [1]	abstracts	493	5846	11.90 [7.44]	3.03 [1.31]
Scopus[6]	abstracts	745	3385	4.54 [1.34]	2.16 [0.65]
KP20k [17]	abstracts	570,809	3,017,637	5.29 [3.77]	2.05 [0.63]

4.2 Keyphrase Extraction Algorithms

In this section we briefly describe the three keyphrase extraction algorithms that are used for our experiments.

The **tf-idf** keyphrase extraction is based on POS tags, following Wen and Xiao [25]. We determined the 12 most common POS tags in gold standard keyphrases among all corpora using the NLTK POS tagger[7]. In order to generate candidate keyphrases we determine the POS tag of each word in a given text and extract word sequences where all POS tags are "good". A sequence like *[bad, good, bad, good, good, bad]* generates two keyphrase candidates. One of length 1 corresponding to the word at position two and one of length 2 corresponding to the words at positions three and four. Finally, the candidates are ranked by the mean of their tf-idf values of the individual words.

Rake is based on the observation that keyphrases rarely contain stop words and punctuation. Therefore all sequences in a text not containing stop words or punctuation are identified and treated as candidate keyphrases. Then a matrix is constructed where the co-occurrence of words within a keyphras is counted. Finally each keyphrase is scored based on the co-occurrence scores of its individual words. The phrases with the highest scores are the keyphrases of the document. We used the stopword list introduced in [20] and the implementation provided by NLTK[8]

Textrank builds a graph of lexical units (e.g. words). Only words passing a syntactic filter (e.g. nouns and adjectives only) are added to the graph. These words are connected based on co-occurrence in a sliding window. Once the graph is built PageRank [4] is used to determine the importance of each node in the graph. In a post-processing step sequences of adjacent keywords are merged into keyphrases and their scores are added for the final ranking. We used the stopword list introduced in [20] and the jgtextrank[9].

[6] https://www.kaggle.com/neelshah18/scopusjournal/data

[7] https://www.nltk.org/

[8] https://pypi.org/project/rake-nltk/

[9] https://github.com/jerrygaoLondon/jgtextrank

Table 2: The effect of keyphrase scoring on one example. The top cell contains an original abstract. The cell below contains the expert-assigned ground-truth keyphrases. The bottom row contains the extracted keyphrases from multiple algorithms. Entries written in italic indicate a match to a ground-truth keyphrase. Note: This example was chosen as it clearly shows the positive effect of our scoring approach. But there are also examples where the scoring has no effect.

<table>
<tr><th colspan="4">Abstract</th></tr>
<tr><td colspan="4">A comparison theorem for the iterative method with the preconditioner (I + S/sub max/)
A.D. Gunawardena et al. (1991) have reported the modified Gauss-Seidel method with a preconditioner (I + S). In this article, we propose to use a preconditioner (I + S/sub max/) instead of (I + S). Here, S/sub max/ is constructed by only the largest element at each row of the upper triangular part of A. By using the lemma established by M. Neumann and R.J. Plemmons (1987), we get the comparison theorem for the proposed method. Simple numerical examples are also given.</td></tr>
<tr><th colspan="4">Ground-truth keyphrases</th></tr>
<tr><td colspan="4">iterative method, preconditioner, modified Gauss-Seidel method, comparison theorem</td></tr>
<tr><th colspan="4">Extracted keyphrases</th></tr>
<tr><th>Rake</th><th>Rake$_s$</th><th>Textrank</th><th>Textrank$_s$</th></tr>
<tr><td>1. upper triangular part</td><td><i>1. preconditioner</i></td><td><i>1. iterative method</i></td><td><i>1. preconditioner</i></td></tr>
<tr><td>2. simple numerical examples</td><td><i>2. comparison theorem</i></td><td><i>2. modified Gauss-Seidel method</i></td><td><i>2. comparison theorem</i></td></tr>
<tr><td>3. Seidel method</td><td><i>3. iterative method</i></td><td>3. method</td><td><i>3. iterative method</i></td></tr>
<tr><td>4. proposed method</td><td>4. upper triangular part</td><td>4. R.J. Plemmons</td><td><i>4. modified Gauss-Seidel method</i></td></tr>
<tr><td>5. modified Gauss</td><td>5. lemma established</td><td>5. M. Neumann</td><td>5. upper triangular part</td></tr>
</table>

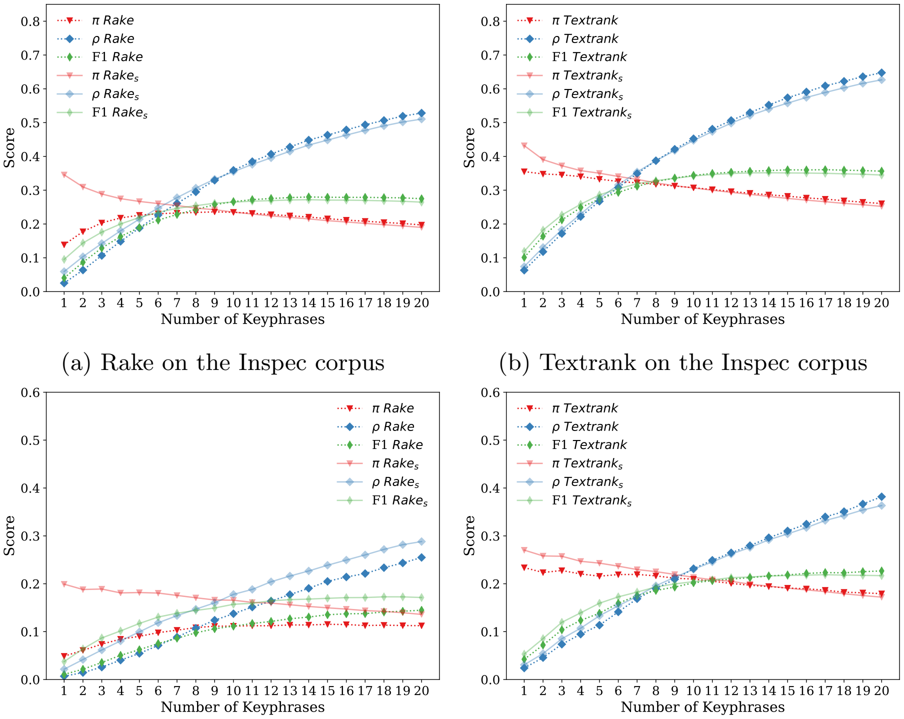
4.3 Evaluation Method

In user-facing applications the quality of the complete keyphrase list is more important than the quality of the individual keyphrases. Therefore, we evaluate the quality of keyphrase lists, similar to evaluations in previous work [9]. To simplify further discussion, we introduce the term n-*sublist*, which is a list of the first n elements of a larger list. For example the 2-sublist of the list $(1, 2, 3)$ is $(1, 2)$. We expected that the 1-sublists exhibit the highest precision scores at a low recall score (because in scenarios where multiple gold standard keyphrases are given, single keyphrases cannot reach high recall scores). When assessing longer keyphrase sublists (e.g. the 2-sublists, 3-sublists etc.) the precision is expected to decline, due to the lower precision of lower ranked keyphrases, while the recall increases, as more extracted keyphrases match the gold standard keyphrases. For each document and each algorithm, we then compute precision, recall and F1 for each n-sublist ($n \in [1, 20]$ in the experiments). Measures are macro-averaged, that means, we calculate the measure for each document and then average over the total number of documents.

5 Results

In this section, we provide results on the influence of the keyphrase scoring and show the effect of combining the scored output of different extraction algorithms.

Table 2 shows keyphrases extracted from an example document. We can see that Rake initially does not find a ground-truth keyphrase, but after scoring



(a) Rake on the Inspec corpus

(b) Textrank on the Inspec corpus

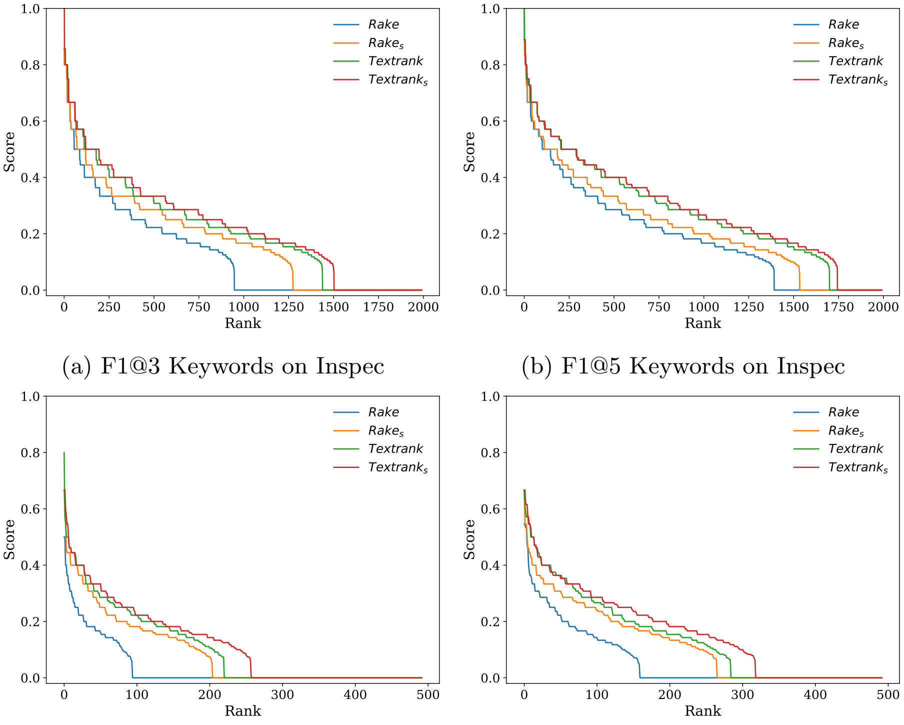
(c) Rake on the SemEval2017 corpus

(d) Textrank on the SemEval2017 corpus

Fig. 3: Precision (π), Recall (ρ) and F1-score of Textrank and Rake compared to the tf-idf-based reranking on the Inspec and the SemEval2017 corpora.

there are three matches. Similarly, Textrank initially finds two ground-truth keyphrases but after scoring it finds all four ground-truth keyphrases. Figure 3 compares precision (π), recall (ρ) and $F1$ for the scored and unscored versions of Rake and TextRank on the Inspec corpus and on the SemEval2017 corpus. The tf-idf-based scoring increases the precision of Rake significantly for up to five keywords. The precision of Textrank is also enhanced but the effect is significantly smaller.

For instance, for only one keyphrase on the Inspec corpus, Rake is below the baseline (baseline 0.24 π, Rake 0.14 π) but the scored version of Rake is considerably better than the baseline ($Rake_s$ 0.35 π). The already better-than-baseline performance (0.36 π) of Textrank is enhanced (0.43 π). At 5 keyphrases the situation is similar. But here the performance of Rake is above the baseline and the performance gain due to the scoring is smaller (Rake +0.04 π, Textrank +0.02 π). Figure 4 shows the ranked F1-scores on the Inspec and SemEval2017. There we can see that our method increases the performance mostly on documents



(a) F1@3 Keywords on Inspec

(b) F1@5 Keywords on Inspec

(c) F1@3 Keywords on SemEval2017

(d) F1@5 Keywords on SemEval2017

Fig. 4: Individual results ordered by F1-scores.

with mediocre to low scores. The performance on documents where to score is already good is not affected as much.

In general we can say that as the number of keyphrases increases the positive effect of the reranking diminishes due to the fact that scoring the list of keywords has no influence anymore if the whole list is used. This behaviour is also observable for the other corpora. Figures are omitted here due to space constraints, but the snapshots of performance curves at 1, 5, 10 and 20 keyphrases are provided in table 3. In this table, it can also be seen, that Textrank always outperforms Rake.

Table 3 also shows the performance of the ensemble method. Performance of the ensemble is consistently better than $Rake_s$ but worse than $Textrank_s$. This means the ensemble method is not able to incorporate the additional keyphrases provided by Rake to enhance the performance of Textrank. Figure 5 depicts the performance of the ensemble versus the best performing algorithm $Textrank_s$ on the Inspec corpus.

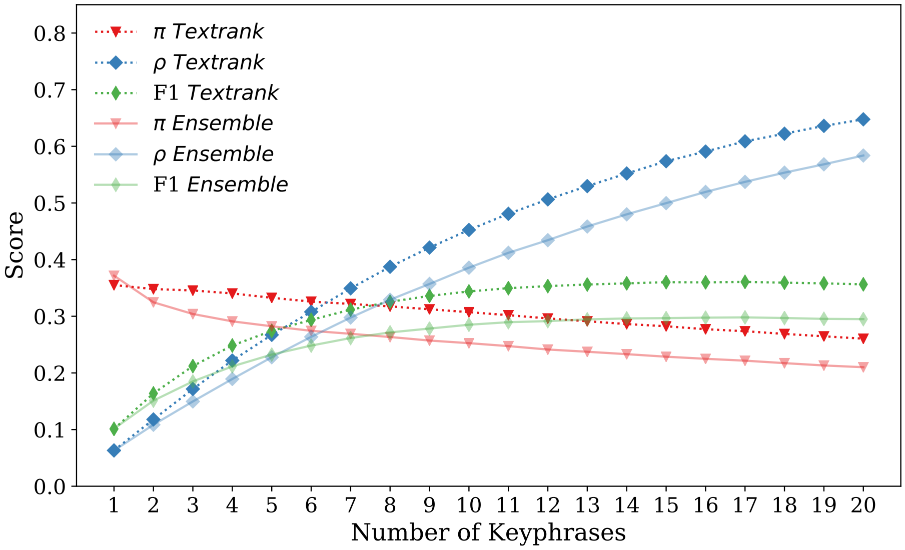


Fig. 5: Precision (π), Recall (ρ) and F1-score of textrank compared to an ensemble of Rake and Textrank on the Inspec corpus.

6 Discussion

The results show that the reranking approach has a significant effect on the precision of Textrank and Rake in the range of 1 to 5 keyphrases. In general, the effect size is strongest when only a single keyphrase is extracted and declines as the number of extracted keyphrases increases. Similarly, recall benefits from the reranking but the absolut effect size is much smaller. The experiments also show that the ensemble is not able to retain the performance of the strongest individual algorithm. Instead it consistently performs better than the weaker algorithm (Rake) and worse than the stronger algorithm (Textrank). Also it must be noted that from these results one cannot conclude that *in general* keyphrases with a higher tf-idf value are better keyphrases than keyphrases with lower tf-idf values. Instead, one can only state that the probability of being a gold standard keyphrase is proportional to tf-idf value. Moreover, the way keyphrase-function may differ depends on the scenario. We chose a simple linear method (as shown in equation 2) which favors keyphrases with 2-5 tokens. Preferences for longer or shorter keyphrases can be steered with the parameter α in equation 2, which was set to $\alpha = \frac{1}{10}$ in our experiments. However, its influence on the quality of the result list would need to be investigated with a parameter study in the future.

Table 3: Performance of base keyphrase extraction algorithms, their scored version and the ensemble-based keyphrase extractor.

	Algo	1 Phrase			5 Phrases			10 Phrase			20 Phrases		
		π	ρ	$F1$	π	ρ	$F1$	π	ρ	$F1$	π	ρ	$F1$
Inspec	tf-idf	0.24	0.04	0.06	0.15	0.11	0.12	0.12	0.17	0.13	0.09	0.22	0.12
	RK	0.14	0.03	0.04	0.23	0.19	0.19	0.23	0.36	0.27	0.20	0.53	0.27
	TR	0.36	0.06	0.10	0.33	0.27	0.27	**0.31**	**0.45**	**0.34**	**0.26**	**0.65**	**0.36**
	RK_s	0.35	0.06	0.10	0.27	0.21	0.22	0.24	0.35	0.26	0.19	0.51	0.26
	TR_s	**0.43**	**0.07**	**0.12**	**0.35**	**0.28**	**0.29**	**0.31**	**0.45**	**0.34**	0.25	0.63	0.34
	ENS_s	0.37	0.06	0.10	0.28	0.23	0.23	0.25	0.39	0.29	0.21	0.58	0.29
SemEval17	tf-idf	0.24	0.02	0.04	0.16	0.08	0.10	0.12	0.12	0.11	0.08	0.17	0.10
	RK	0.05	0.01	0.01	0.09	0.05	0.06	0.11	0.14	0.11	0.11	0.25	0.14
	TR	0.23	0.02	0.04	0.22	0.11	0.14	**0.21**	**0.23**	**0.20**	**0.18**	**0.38**	**0.23**
	RK_s	0.20	0.02	0.04	0.18	0.10	0.12	0.17	0.18	0.16	0.14	0.29	0.17
	TR_s	**0.27**	**0.03**	**0.05**	**0.24**	**0.13**	**0.16**	**0.21**	**0.23**	**0.20**	0.17	0.36	0.22
	ENS_s	0.24	**0.03**	0.04	0.19	0.10	0.13	0.18	0.20	0.17	0.15	0.32	0.19
Scopus	tf-idf	0.11	0.04	0.06	0.05	0.09	0.06	0.03	0.13	0.05	0.02	0.16	0.04
	RK	0.02	0.01	0.01	0.04	0.08	0.05	0.05	0.20	0.07	0.05	0.36	0.08
	TR	0.08	0.03	0.04	0.09	0.18	0.11	**0.08**	**0.33**	**0.13**	**0.07**	**0.49**	**0.11**
	RK_s	0.10	0.04	0.06	0.09	0.17	0.11	0.07	0.26	0.10	0.05	0.36	0.08
	TR_s	**0.15**	**0.06**	**0.08**	**0.11**	**0.22**	**0.14**	**0.08**	0.32	**0.13**	0.06	0.44	0.10
	ENS_s	0.11	0.05	0.06	0.09	0.18	0.11	0.07	0.27	0.11	0.05	0.39	0.09
KP20k	tf-idf	0.10	0.04	0.05	0.05	0.09	0.06	0.03	0.12	0.05	0.02	0.15	0.04
	RK	0.01	0.01	0.01	0.03	0.07	0.04	0.04	0.17	0.06	0.04	0.31	0.07
	TR	0.09	0.04	0.05	0.08	0.17	0.10	0.07	0.29	0.11	**0.06**	**0.43**	**0.10**
	RK_s	0.09	0.04	0.05	0.08	0.16	0.10	0.06	0.24	0.09	0.05	0.33	0.08
	TR_s	**0.12**	**0.05**	**0.07**	**0.10**	**0.20**	**0.12**	**0.08**	**0.30**	**0.12**	**0.06**	0.41	**0.10**
	ENS_s	0.10	0.04	0.06	0.08	0.16	0.10	0.06	0.26	0.10	0.05	0.37	0.08

7 Summary

We presented a framework that allows to rank a list or a set of keyphrases based on the tf-idf values of their individual tokens. Moreover, the framework is agnostic to the method applied to extract the keyphrases. In fact, it is also able to deal with keyphrases extracted by multiple methods, regardless whether these methods rank the keyphrases they extract or not. This property provides a normalized, common score for all keyphrases and thus allows to combine results from different algorithms. For two keyphrase extraction algorithms, we showed that the keyphrases with high tf-idf values are more likely to be gold standard keyphrases. Thus, they are – on average – more informative keyphrases for end users. The results could be reproduced on four different corpora. We also showed a method to merge multiple keyphrase extraction algorithms into a single one, although it failed to achieve the top performance of the best individual method. Future work includes finding and investigating other keyphrase scoring functions and more extensive experiments with more keyphrase extraction algorithms to aggregate.

References

1. Augenstein, I., Das, M., Riedel, S., Vikraman, L., McCallum, A.: Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. arXiv preprint arXiv:1704.02853 (2017)
2. Bougouin, A., Boudin, F., Daille, B.: Topicrank: Graph-based topic ranking for keyphrase extraction. pp. 543–551 (10 2013)
3. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (Aug 1996)
4. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer networks and ISDN systems 30(1-7), 107–117 (1998)
5. Danilevsky, M., Wang, C., Desai, N., Ren, X., Guo, J., Han, J.: Automatic construction and ranking of topical keyphrases on collections of short documents. In: Proceedings of the 2014 SIAM International Conference on Data Mining. pp. 398–406. SIAM (2014)
6. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., Nevill-Manning, C.G.: Domain-specific keyphrase extraction. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. pp. 668–673. IJCAI '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
7. Grineva, M., Grinev, M., Lizorkin, D.: Extracting key terms from noisy and multi-theme documents. In: Proceedings of the 18th International Conference on World Wide Web. pp. 661–670. WWW '09, ACM, New York, NY, USA (2009)
8. Hasan, K.S., Ng, V.: Conundrums in unsupervised keyphrase extraction: Making sense of the state-of-the-art. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. pp. 365–373. COLING '10, Association for Computational Linguistics, Stroudsburg, PA, USA (2010)
9. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: A survey of the state of the art. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1262–1273 (2014)
10. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing. pp. 216–223. EMNLP '03, Association for Computational Linguistics, Stroudsburg, PA, USA (2003)
11. Liu, F., Pennell, D., Liu, F., Liu, Y.: 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. pp. 620–628. NAACL '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
12. Liu, Z., Chen, X., Zheng, Y., Sun, M.: Automatic keyphrase extraction by bridging vocabulary gap. In: Proceedings of the Fifteenth Conference on Computational Natural Language Learning. pp. 135–144. Association for Computational Linguistics (2011)
13. Liu, Z., Huang, W., Zheng, Y., Sun, M.: Automatic keyphrase extraction via topic decomposition. In: Proceedings of the conference on empirical methods in natural language processing. pp. 366–376. Association for Computational Linguistics (2010)

14. Liu, Z., Li, P., Zheng, Y., Sun, M.: Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1. pp. 257–266. EMNLP '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
15. Mani, I.: Advances in Automatic Text Summarization. MIT Press, Cambridge, MA, USA (1999)
16. Medelyan, O., Frank, E., Witten, I.H.: Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3. pp. 1318–1327. EMNLP '09, Association for Computational Linguistics, Stroudsburg, PA, USA (2009)
17. Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y.: Deep keyphrase generation. arXiv preprint arXiv:1704.06879 (2017)
18. Mihalcea, R., Tarau, P.: Textrank: Bringing order into texts. In: Proc. Conf. on Empirical Methods in Natural Language Processing. Barcelona, Spain (2004)
19. Miller, G.A.: The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological review 63(2), 81 (1956)
20. Ren, X., El-Kishky, A., Wang, C., Tao, F., Voss, C.R., Han, J.: Clustype: Effective entity recognition and typing by relation phrase-based clustering. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 995–1004. ACM (2015)
21. Rose, S., Engel, D., Cramer, N., Cowley, W.: Automatic Keyword Extraction from Individual Documents, pp. 1–20. John Wiley & Sons, Ltd (2010)
22. Turney, P.: Learning to extract keyphrases from text (01 1999)
23. Wan, X., Xiao, J.: Collabrank: Towards a collaborative approach to single-document keyphrase extraction. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008). pp. 969–976. Coling 2008 Organizing Committee, Manchester, UK (August 2008)
24. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2. pp. 855–860. AAAI'08, AAAI Press (2008)
25. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: AAAI. vol. 8, pp. 855–860 (2008)
26. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: Kea: Practical automatic keyphrase extraction. In: Proceedings of the Fourth ACM Conference on Digital Libraries. pp. 254–255. ACM, New York, NY, USA (1999)
27. Zhang, Y., Fang, Y., Weidong, X.: Deep keyphrase generation with a convolutional sequence to sequence model. In: Systems and Informatics (ICSAI), 2017 4th International Conference on. pp. 1477–1485. IEEE (2017)