How Document Pre-processing affects Keyphrase Extraction Performance

Florian Boudin and **Hugo Mougard** and **Damien Cram**
LINA - UMR CNRS 6241, Université de Nantes, France
firstname.lastname@univ-nantes.fr

Abstract

The SemEval-2010 benchmark dataset has brought renewed attention to the task of automatic keyphrase extraction. This dataset is made up of scientific articles that were automatically converted from PDF format to plain text and thus require careful preprocessing so that irrevelant spans of text do not negatively affect keyphrase extraction performance. In previous work, a wide range of document preprocessing techniques were described but their impact on the overall performance of keyphrase extraction models is still unexplored. Here, we re-assess the performance of several keyphrase extraction models and measure their robustness against increasingly sophisticated levels of document preprocessing.

1 Introduction

Recent years have seen a surge of interest in automatic keyphrase extraction, thanks to the availability of the SemEval-2010 benchmark dataset (Kim et al., 2010). This dataset is composed of documents (scientific articles) that were automatically converted from PDF format to plain text. As a result, most documents contain irrelevant pieces of text (e.g. muddled sentences, tables, equations, footnotes) that require special handling, so as to not hinder the performance of keyphrase extraction systems. In previous work, these are usually removed at the preprocessing step, but using a variety of techniques ranging from simple heuristics (Wang and Li, 2010; Treeratpituk et al., 2010; Zervanou, 2010) to sophisticated document logical structure detection on richly-formatted documents recovered from Google Scholar (Nguyen and Luong, 2010). Under such conditions, it may prove difficult to draw firm conclusions about which keyphrase extraction model performs best, as the impact of preprocessing on overall performance cannot be properly quantified.

While previous work clearly states that efficient document preprocessing is a prerequisite for the extraction of high quality keyphrases, there is, to our best knowledge, no empirical evidence of how preprocessing affects keyphrase extraction performance. In this paper, we re-assess the performance of several state-of-the-art keyphrase extraction models at increasingly sophisticated levels of preprocessing. Three incremental levels of document preprocessing are experimented with: raw text, text cleaning through document logical structure detection, and removal of keyphrase sparse sections of the document. In doing so, we present the first consistent comparison of different keyphrase extraction models and study their robustness over noisy text. More precisely, our contributions are:

- We show that performance variation across keyphrase extraction systems is, at least in part, a function of the (often unstated) effectiveness of document preprocessing.
- We empirically show that supervised models are more resilient to noise, and point out that the performance gap between baselines and top performing systems is narrowing with the increase in preprocessing effort.

Proceedings of the 2nd Workshop on Noisy User-generated Text,
pages 92–99, Osaka, Japan, December 11 2016.

- We compare the previously reported results of several keyphrase extraction models with that of our re-implementation, and observe that baseline performance is underestimated because of the inconsistence in document preprocessing.
- We release both a new version of the SemEval-2010 dataset[1] with preprocessed documents and our implementation of the state-of-the-art keyphrase extraction models[2] using the `pke` toolkit (Boudin, 2016) for use by the community.

2 Dataset and Preprocessing

The SemEval-2010 benchmark dataset (Kim et al., 2010) is composed of 244 scientific articles collected from the ACM Digital Library (conference and workshop papers). The input papers ranged from 6 to 8 pages and were converted from PDF format to plain text using an off-the-shelf tool[3]. The only preprocessing applied is a systematic dehyphenation at line breaks[4] and removal of author-assigned keyphrases. Scientific articles were selected from four different research areas as defined in the ACM classification, and were equally distributed into training (144 articles) and test (100 articles) sets. Gold standard keyphrases are composed of both author-assigned keyphrases collected from the original PDF files and reader-assigned keyphrases provided by student annotators.

Long documents such as those in the SemEval-2010 benchmark dataset are notoriously difficult to handle due to the large number of keyphrase candidates (i.e. phrases that are eligible to be keyphrases) that the systems have to cope with (Hasan and Ng, 2014). Furthermore, noisy textual content, whether due to format conversion errors or to unusable elements (e.g. equations), yield many spurious keyphrase candidates that negatively affect keyphrase extraction performance. This is particularly true for systems that make use of core NLP tools to select candidates, that in turn exhibit poor performance on degraded text. Filtering out irrelevant text is therefore needed for addressing these issues.

In this study, we concentrate our effort on re-assessing keyphrase extraction performance on three increasingly sophisticated levels of document preprocessing described below.

Level 1: We process each input file with the Stanford CoreNLP suite (Manning et al., 2014)[5]. We use the default parameters and perform tokenization, sentence splitting and Part-Of-Speech (POS) tagging.

Level 2: Similarly to (Nguyen and Luong, 2010; Lopez and Romary, 2010), we retrieve[6] the original PDF files from the ACM Digital Library. We then extract the enriched[7] textual content of the PDF files using an Optical Character Recognition (OCR) system[8], and perform document logical structure detection using ParsCit (Kan et al., 2010)[9]. We use the detected logical structure to remove author-assigned keyphrases and select only relevant elements: title, headers, abstract, introduction, related work, body text[10] and conclusion. We finally apply a systematic dehyphenation at line breaks and run the Stanford CoreNLP suite.

Level 3: As pointed out by (Treeratpituk et al., 2010; Nguyen and Luong, 2010; Wang and Li, 2010; Eichler and Neumann, 2010; El-Beltagy and Rafea, 2010), considering only the keyphrase dense parts of the scientific articles allows to improve keyphrase extraction performance. Accordingly we follow previous work and further abridge the input text from level 2 preprocessed documents to the following: title, headers, abstract, introduction, related work, background and conclusion. Here, the idea is to achieve the best compromise between search space (number of candidates) and maximum performance (recall).

[1] `https://github.com/boudinfl/semeval-2010-pre`
[2] `https://github.com/boudinfl/pke`
[3] pdftotext, `http://www.foolabs.com/xpdf/`
[4] Valid hyphenated forms may have their hyphen removed by this process.
[5] Use use Stanford CoreNLP v3.6.0.
[6] To ensure consistency, articles were manually collected.
[7] Additional information such as font format or spacial layout is also extracted.
[8] We use Omnipage v18, `http://www.nuance.com/omnipage`
[9] We use ParsCit v110505.
[10] We further filter out tables, figures, captions, equations, notes, copyright and references.

Table 1 shows the average number of sentences and words along with the maximum possible recall for each level of preprocessing. The maximum recall is obtained by computing the fraction of the reference keyphrases that occur in the documents. We observe that the level 2 preprocessing succeeds in eliminating irrelevant text by significantly reducing the number of words (-19%) while maintaining a high maximum recall (-2%). Level 3 preprocessing drastically reduce the number of words to less than a quarter of the original amount while interestingly still preserving high recall.

	Lvl 1	**Lvl 2**	**Lvl 3**
Avg. sentences	399	347	101
Avg. words	9 772	7 874	1 922
Max. recall	83.9%	81.8%	70.9%

Table 1: Statistics computed at the different levels of document preprocessing on the training set.

3 Keyphrase Extraction Models

We re-implemented five keyphrase extraction models : the first two are commonly used as baselines, the third is a resource-lean unsupervised graph-based ranking approach, and the last two were among the top performing systems in the SemEval-2010 keyphrase extraction task (Kim et al., 2010). We note that two of the systems are supervised and rely on the training set to build their classification models. Document frequency counts are also computed on the training set[11]. Stemming[12] is applied to allow more robust matching. The different keyphrase extraction models are briefly described below:

TF×IDF: we re-implemented the TF×IDF n-gram based baseline computed by the task organizers. We use 1, 2, 3-grams as keyphrase candidates and filter out those shorter than 3 characters, containing words made of only punctuation marks or one character long[13].

Kea (Witten et al., 1999): keyphrase candidates are 1, 2, 3-grams that do not begin or end with a stopword[14]. Keyphrases are selected using a naïve bayes classifier[15] with two features: TF×IDF and the relative position of first occurrence.

TopicRank (Bougouin et al., 2013): keyphrase candidates are the longest sequences of adjacent nouns and adjectives. Lexically similar candidates are clustered into topics and ranked using TextRank (Mihalcea and Tarau, 2004). Keyphrases are produced by extracting the first occurring candidate of the highest ranked topics.

KP-Miner (El-Beltagy and Rafea, 2010): keyphrase candidates are sequences of words that do not contain punctuation marks or stopwords. Candidates that appear less than three times or that first occur beyond a certain position are removed[16]. Candidates are then weighted using a modified TF×IDF formula that account for document length.

WINGNUS (Nguyen and Luong, 2010): keyphrase candidates are simplex nouns and noun phrases detected using a set of rules described in (Kim and Kan, 2009). Keyphrases are then selected using a naïve bayes classifier with the optimal set of features found on the training set[17]: TF×IDF, relative position of first occurrence and candidate length in words.

[11] For more reliable estimates, we rely on level 2 counts when experimenting with level 3.

[12] We use the Porter stemmer in nltk.

[13] This filtering process is also applied to the other models.

[14] We use the stoplist in nltk, `http://www.nltk.org`

[15] We use the Multinomial Naive Bayes classifier from scikit-learn with default parameters, `http://scikit-learn.org`

[16] To better see the impact of preprocessing, we do not consider the cutoff parameter in our experiments. The least allowable seen frequency parameter is set to 2 which is the optimal value found on the training set.

[17] The optimal set of features in (Nguyen and Luong, 2010) also include the term frequency of substrings, but we observed a significant drop in performance when this feature is included.

Each model uses a distinct keyphrase candidate selection method that provides a trade-off between the highest attainable recall and the size of set of candidates. Table 2 summarizes these numbers for each model. Syntax-based selection heuristics, as used by TopicRank and WINGNUS, are better suited to prune candidates that are unlikely to be keyphrases. As for KP-miner, removing infrequent candidates may seem rather blunt, but it turns out to be a simple yet effective pruning method when dealing with long documents. For details on how candidate selection methods affect keyphrase extraction, please refer to (Wang et al., 2014).

Model	**Lvl 1**		**Lvl 2**		**Lvl 3**	
TF×IDF	80.2%	7 837	78.2%	6 958	67.8%	2 270
Kea	80.2%	3 026	78.2%	2 502	67.8%	912
TopicRank	70.9%	742	69.2%	627	57.8%	241
KP-Miner	64.0%	724	61.8%	599	48.7%	212
WINGNUS	75.2%	1 355	73.0%	1 007	63.0%	403

Table 2: Maximum recall and average number of keyphrase candidates for each model.

Apart from TopicRank that groups similar candidates into topics, the other models do not have any redundancy control mechanism. Yet, recent work has shown that up to 12% of the overall error made by state-of-the-art keyphrase extraction systems were due to redundancy (Hasan and Ng, 2014; Boudin, 2015). Therefore as a post-ranking step, we remove redundant keyphrases from the ranked lists generated by all models. A keyphrase is considered redundant if it is included in another keyphrase that is ranked higher in the list.

4 Experiments

4.1 Experimental settings

We follow the evaluation procedure used in the SemEval-2010 competition and evaluate the performance of each model in terms of f-measure (F) at the top N keyphrases[18]. We use the set of combined author- and reader-assigned keyphrases as reference keyphrases. Extracted and reference keyphrases are stemmed to reduce the number of mismatches.

4.2 Results

The performances of the keyphrase extraction models at each preprocessing level are shown in Table 3. Overall, we observe a significant increase of performance for all models at levels 3, confirming that document preprocessing plays an important role in keyphrase extraction performance. Also, the difference of f-score between the models, as measured by the standard deviation σ_1, gradually decreases with the increasing level of preprocessing. This result strengthens the assumption made in this paper, that performance variation across models is partly a function of the effectiveness of document preprocessing.

Somewhat surprisingly, the ordering of the two best models reverses at level 3. This showcases that rankings are heavily influenced by the preprocessing stage, despite the common lack of details and analysis it suffers from in explanatory papers. We also remark that the top performing model, namely KP-Miner, is unsupervised which supports the findings of (Hasan and Ng, 2014) indicating that recent unsupervised approaches have rivaled their supervised counterparts in performance.

In an attempt to quantify the performance variation across preprocessing levels, we compute the standard deviation σ_2 for each model. Here we see that unsupervised models are more sensitive to noisy input, as revealed by higher standard deviations. We found two main reasons for this. First, using multiple discriminative features to rank keyphrase candidates adds inherent robustness to the models. Second, the supervision signal helps models to disregard noise.

In Table 4, we compare the outputs of the five models by measuring the percentage of valid keyphrases that are retrieved by all models at once for each preprocessing level. By these additional results, we aim

[18] Scores are computed using the evaluation script provided by the SemEval-2010 organizers.

	Model	Lvl 1	Lvl 2	Lvl 3	σ_2
Unsup.	TopicRank	12.2	12.5	$14.5^{\alpha,\beta}$	1.25
	TF×IDF	16.0	16.4	$19.3^{\alpha,\beta}$	1.80
	KP-Miner	20.2	19.8	$\mathbf{22.5}^{\alpha,\beta}$	1.46
Sup.	Kea	19.2	19.3	21.2^{α}	1.13
	WINGNUS	**20.5**	**20.3**	21.8^{β}	0.82
	σ_1	3.51	3.26	3.22	

Table 3: F-scores computed at the top 10 extracted keyphrases for the unsupervised (Unsup.) and supervised (Sup.) models at each preprocessing level. We also report the standard deviation across the five models for each level (σ_1) and the standard deviation across the three levels for each model (σ_2). α and β indicate significance at the 0.05 level using Student's t-test against level 1 and level 2 respectively.

to assess whether document preprocessing smoothes differences between models. We observe that the overlap between the outputs of the different models increases along with the level of preprocessing. This suggests that document preprocessing reduces the effect that the keyphrase extraction model in itself has on overall performance. In other words, the singularity of each model fades away gradually with increase in preprocessing effort.

	Lvl 1	Lvl 2	Lvl 3
% valid keyphrases	19.9%	23.1%	25.1%

Table 4: Percentage of valid keyphrases found by all five keyphrase extraction models at each preprocessing level.

4.3 Reproducibility

Being able to reproduce experimental results is a central aspect of the scientific method. While assessing the importance of the preprocessing stage for five approaches, we found that several results were not reproducible, as shown in Table 5.

Model	Ori.	Lvl 1	Lvl 2	Lvl 3
TopicRank	12.1	+0.1	+0.4	+2.4
TF×IDF	14.4	+1.6	+2.0	+4.9
KP-Miner	23.2	−3.2	−3.4	−0.7
WINGNUS	24.7	−4.2	−4.4	−2.8

Table 5: Difference in f-score between our re-implementation and the original scores reported in (Hasan and Ng, 2014; Bougouin et al., 2013).

We note that the trends for baselines and high ranking systems are opposite: compared to the published results, our reproduction of top systems under-performs and our reproduction of baselines over-performs. We hypothesise that this stems from a difference in hyperparameter tuning, including the ones that the preprocessing stage makes implicit. Competitors have strong incentives to correctly optimize hyperparameters, to achieve a high ranking and get more publicity for their work while competition organizers might have the opposite incentive: too strong a baseline might not be considered a baseline anymore.

We also observe that with this leveled preprocessing, the gap between baselines and top systems is much smaller, lessening again the importance of raw scores and rankings to interpret the shared task results and emphasizing the importance of understanding correctly the preprocessing stage.

5 Additional experiments

In the previous sections, we provided empirical evidence that document preprocessing weighs heavily on the outcome of keyphrase extraction models. This raises the question of whether further improvement might be gained from a more aggressive preprocessing. To answer this question, we take another step beyond content filtering and further abridge the input text from level 3 preprocessed documents using an unsupervised summarization technique. More specifically, we keep the title and abstract intact, as they are the two most keyphrase dense parts within scientific articles (Nguyen and Luong, 2010), and select only the most content bearing sentences from the remaining contents. To do this, sentences are ordered using TextRank (Mihalcea and Tarau, 2004) and the less informative ones, as determined by their TextRank scores normalized by their lengths in words, are filtered out.

Finding the optimal subset of sentences from already shortened documents is however no trivial task as maximum recall linearly decreases with the number of sentences discarded. Here, we simply set the reduction ratio to 0.865 so that the average maximum recall on the training set does not lose more than 5%. Table 6 shows the reduction in the average number of sentences and words compared to level 3 preprocessing.

	Lvl 4	**Δ Lvl 3**
Avg. sentences	71	−30.0%
Avg. words	1 470	−23.5%
Max. recall	65.9%	−7.0%

Table 6: Statistics computed at level 4 of document preprocessing on the training set. We also report the relative difference with respect to level 3 preprocessing.

The performances of the keyphrase extraction models at level 4 preprocessing are shown in Table 7. We note that two models, namely TopicRank and TF×IDF, lose on performance. These two models mainly rely on frequency counts to rank keyphrase candidates, which in turn become less reliable at level 4 because of the very short length of the documents. Other models however have their f-scores once again increased, thus indicating that further improvement is possible from more reductive document preprocessing strategies.

	Model	**Lvl 4**	**Δ Lvl 3**
Unsup.	TopicRank	13.7	−0.8
	TF×IDF	18.5	−0.8
	KP-Miner	23.2	+0.7
Sup.	Kea	21.7	+0.5
	WINGNUS	22.5	+0.7

Table 7: F-scores computed at the top 10 extracted keyphrases for the unsupervised (Unsup.) and supervised (Sup.) models at level 4 preprocessing. We also report the difference in f-score with level 3 preprocessing.

6 Conclusion

In this study, we re-assessed the performance of several keyphrase extraction models and showed that performance variation across models is partly a function of the effectiveness of the document preprocessing. Our results also suggest that supervised keyphrase extraction models are more robust to noisy input.

Given our findings, we recommend that future works use a common preprocessing to evaluate the interest of keyphrase extraction approaches. For this reason we make the four levels of preprocessing used in this study available for the community.

References

Florian Boudin. 2015. Reducing over-generation errors for automatic keyphrase extraction using integer linear programming. In *Proceedings of the ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction*, pages 19–24, Beijing, China, July. Association for Computational Linguistics.

Florian Boudin. 2016. pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016: System Demonstrations*, Osaka, Japan, December.

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 543–551, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Kathrin Eichler and Günter Neumann. 2010. Dfki keywe: Ranking keyphrases extracted from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 150–153, Uppsala, Sweden, July. Association for Computational Linguistics.

Samhaa R. El-Beltagy and Ahmed Rafea. 2010. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 190–193, Uppsala, Sweden, July. Association for Computational Linguistics.

Kazi Saidul Hasan and Vincent Ng. 2014. 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)*, pages 1262–1273, Baltimore, Maryland, June. Association for Computational Linguistics.

Min-Yen Kan, Minh-Thang Luong, and Thuy Dung Nguyen. 2010. Logical structure recovery in scholarly articles with rich document features. *Int. J. Digit. Library Syst.*, 1(4):1–23, October.

Su Nam Kim and Min-Yen Kan. 2009. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 9–16, Singapore, August. Association for Computational Linguistics.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26, Uppsala, Sweden, July. Association for Computational Linguistics.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 248–251, Uppsala, Sweden, July. Association for Computational Linguistics.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 404–411, Barcelona, Spain, July. Association for Computational Linguistics.

Thuy Dung Nguyen and Minh-Thang Luong. 2010. Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 166–169, Uppsala, Sweden, July. Association for Computational Linguistics.

Pucktada Treeratpituk, Pradeep Teregowda, Jian Huang, and C. Lee Giles. 2010. Seerlab: A system for extracting keyphrases from scholarly documents. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 182–185, Uppsala, Sweden, July. Association for Computational Linguistics.

Letian Wang and Fang Li. 2010. Sjtultlab: Chunk based method for keyphrase extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 158–161, Uppsala, Sweden, July. Association for Computational Linguistics.

Rui Wang, Wei Liu, and Chris Mcdonald. 2014. How preprocessing affects unsupervised keyphrase extraction. In *Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 8403*, CICLing 2014, pages 163–176, New York, NY, USA. Springer-Verlag New York, Inc.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA. ACM.

Kalliopi Zervanou. 2010. Uvt: The uvt term extraction system in the keyphrase extraction task. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 194–197, Uppsala, Sweden, July. Association for Computational Linguistics.