Counting What Counts: Decompounding for Keyphrase Extraction

Nicolai Erbs[◇‡], **Pedro Bispo Santos**[◇], **Torsten Zesch**[§], **Iryna Gurevych**[◇‡]

◇ UKP Lab, Technische Universität Darmstadt
‡ UKP Lab, German Institute for Educational Research
§ Language Technology Lab, University of Duisburg-Essen
http://www.ukp.tu-darmstadt.de

Abstract

A core assumption of keyphrase extraction is that a concept is more important if it is mentioned more often in a document. Especially in languages like German that form large noun compounds, frequency counts might be misleading as concepts "hidden" in compounds are not counted. We hypothesize that using decompounding before counting term frequencies may lead to better keyphrase extraction. We identified two effects of decompounding: (i) enhanced frequency counts, and (ii) more keyphrase candidates. We created two German evaluation datasets to test our hypothesis and analyzed the effect of additional decompounding for keyphrase extraction.

1 Introduction

Most approaches for automatic extraction of keyphrases are based on the assumption that the more frequent a term or phrase is mentioned, the more important it is. Consequently, most extraction algorithms apply some kind of normalization, e.g. lemmatization or noun chunking (Hulth, 2003; Mihalcea and Tarau, 2004), in order to arrive with accurate counts. However, especially in Germanic languages the frequent use of noun compounds has an adverse effect on the reliability of frequency counts. Consider for example a German document that talks about *Lehrer* (Engl.: *teacher*) without ever mentioning the word "Lehrer" at all, because it is always part of compounds like *Deutschlehrer* (Engl.: *German teacher*) or *Gymnasiallehrer* (Engl.: *grammar school teacher*). Thus, we argue that the problem can be solved by splitting noun compounds in meaningful parts, i.e. by performing decompounding. Figure 1 give an example for decompounding

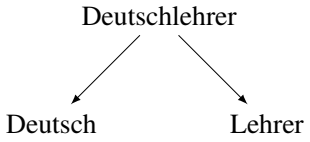

Figure 1: Decompounding of German term *Deutschlehrer* (Engl.: *German teacher*).

in German. The compound *Deutschlehrer* consists of the parts *Deutsch* (Engl.: *German*) and *Lehrer* (Engl.: *teacher*).

In this paper, we propose a comprehensive decompounding architecture and analyze the performance of four state-of-the-art algorithms. We then perform experiments on three German datasets, of which two have been created particularly for these experiments, in order to analyze the impact of decompounding on standard keyphrase extraction approaches. Decompounding has previously been successfully used in other applications, e.g. in machine translation (Koehn and Knight, 2003), information retrieval (Hollink et al., 2004; Alfonseca et al., 2008b; Alfonseca et al., 2008a), speech recognition (Ordelman, 2003), and word prediction (Baroni et al., 2002). Hasan and Ng (2014) have shown that infrequency errors are a major cause for lower keyphrase extraction results . To the best of our knowledge, we are the first to examine the influence of decompounding on keyphrase extraction.

2 Decompounding

Decompounding is usually performed in two steps: (i) a splitting algorithm creates candidates, and (ii) a ranking function decides which candidates are best suited for splitting the compound. For example, *Aktionsplan* has two splitting candidates: *Aktion(s)+plan* (Engl.: *action plan*) and *Akt+ion(s)+plan* (Engl.: *nude ion plan*).[1] After

[1]The additional 's' is a linking morpheme (Langer, 1998)

Proceedings of the ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction, pages 10–17,
Beijing, China, July 30, 2015.

generating the candidates, the ranking function assigns a score to each splitting candidate, including the original compound. We will now take a closer look on possible splitting algorithms and ranking functions.

2.1 Splitting algorithms

Left-to-Right grows a window over the input from left to right. When a word from a dictionary is found a split is generated. The algorithm is then applied recursively to the rest of the input.

JWord Splitter[2] performs a dictionary look-up from left to right, but continues this process if the remainder of the word is not right), it creates a split and stops. **Banana Splitter**[3] searches for the word from the right to the left, and if there is more than one possibility, the one with the longest split on the right side is taken as candidate. **Data Driven** counts the number of words in a dictionary, which contain a split at this position as prefix or suffix for every position in the input. A split is made at the position with the largest difference between prefix and suffix counts (Larson et al., 2000). **ASV Toolbox**[4] uses a trained Compact Patricia Tree to recursively split parts from the beginning and end of the word (Biemann et al., 2008). Unlike the other algorithms, it generates only a single split candidate at each recursive step. For that reason, it does not need a ranker. It is also the only supervised (using lists of existing compounds) approach tested.

2.2 Ranking functions

As stated earlier, the ranking functions are as important as the splitting algorithms, since a ranking function is responsible for assigning scores to each possible decompounding candidate. For the ranking functions, Alfonseca et al. (2008b) use a geometric mean of unigram frequencies (Equation 1), and a mutual information function (Equation 2).

$$r_{Freq}() = \left(\prod_{i}^{N} f(w_i)\right)^{\frac{1}{N}} \quad (1)$$

$$r_{M.I.}() = \begin{cases} -f(c)\log f(c) & \text{if } N = 1 \\ \frac{1}{N-1}\sum_{i}^{N-1} \log \frac{bigr(w_i, w_{i+1})}{f(w_i)f(w_{i+1})} \end{cases} \quad (2)$$

[2] github.com/danielnaber/jwordsplitter

[3] niels.drni.de/s9y/pages/bananasplit.html

[4] wortschatz.uni-leipzig.de/~cbiemann/software/toolbox/

Splitter	Ranker	P_{comp}	R_{comp}	P_{split}
Left-to-right	Freq.	.64	.58	.71
	M.I.	.26	.08	.33
JWord Splitter	Freq.	.67	.63	.79
	M.I.	.59	.20	.73
Banana Splitter	Freq.	.70	.40	.83
	M.I.	.66	.16	.81
Data Driven	Freq.	.49	.18	.70
	M.I.	.40	.04	.58
ASV ToolBox		**.80**	**.75**	**.87**

Table 1: Evaluation results of state-of-the-art decompounding systems.

In these equations, N is the number of fragments the candidate has, w is the fragment itself, $f(w)$ is the relative unigram frequency for that fragment w, $bigr(w_i, w_j)$ is the relative bigram frequency for the fragment w_i and w_j, c is the compound itself without being split.

2.3 Decompounding experiments

For evaluation, we use the corpus created by Marek (2006) as a gold standard to evaluate the performance of the decompounding methods. This corpus contains a list of 158,653 compounds, stating how each compound should be decompounded. The compounds were obtained from the issues 01/2000 to 13/2004 of the German computer magazine c't[5] in a semi-automatic approach. Human annotators reviewed the list to identify and correct possible errors. For calculating the required frequencies, we use the Web1T corpus[6] (Brants and Franz, 2006).

Koehn and Knight (2003) use a modified version of precision and recall for evaluating decompounding performance. Following Santos (2014), we decided to apply these metrics for measuring the splitting algorithms, and ranking the functions' performance. The following counts were used for evaluating the experiments on the compound level: *correct split (cs)*, a split fragment which was correctly identified and *wrong split (ws)*, a split fragment which was wrongly identified. P_{comp} and R_{comp} evaluate decompounding on the level of compounds, and we propose to use $P_{split} = \frac{cs}{cs + ws}$ to evaluate on the level of splits.

As we focus in this work on the influence of decompounding on improving the accuracy of fre-

[5] www.heise.de/ct/

[6] German version (see https://catalog.ldc.upenn.edu/LDC2009T25).

Dataset	peDOCS	MedForum	Pythag.
Number of doc.	2,644	102	60
∅ doc. length	14,016	135	277
Median doc. length	809	104	68
# keyphrases	30,051	853	622
∅ key / doc.	11.37	8.41	10 .37
∅ tokens / key	1.15	1.07	1.30
∅ characters / key	13.27	10.28	12 .22

Table 2: Corpus statistics of datasets.

quency counts, P_{split} is the best metric in our case. We can see in Table 1 that the ASV Toolbox splitting algorithm is the best performing system in respect to P_{split}. Thus, we select it as the decompounding algorithm in our keyphrase extraction experiments described in the next section.

3 Experiments

3.1 Datasets

For our evaluation, we could not rely on English datasets, as there is only very little compounding and thus the expected effect of decompounding is small. German is a good choice, as it is infamous for its heavy compounding, e.g. the well-known *Donaudampfschifffahrtskapitän* (Engl.: *captain of a steam ship on the river Danube*). For German keyphrase extraction, we can use the peDOCS datasets described in Erbs et al. (2013) and we created two additional datasets consisting of summaries of lesson transcripts (Pythagoras) and posts from a medical forum (MedForum). Table 2 summarizes their characteristics.

peDOCS consists of peer-reviewed articles, dissertations, and books from the educational domain published by researchers. The gold standard for this dataset was compiled by professional indexers and should thus be of high quality. We present two novel keyphrase datasets consisting of German texts. **MedForum** is composed of posts from a medical forum.[7] To our knowledge, it is the first dataset with keyphrase annotations from user-generated data in German. Two German annotators with university degrees identified a set of keyphrases for every document and following Nguyen and Kan (2007), the union of both sets are the final gold keyphrases. The **Pythagoras** dataset contains summaries of lesson transcripts compiled in the Pythagoras project.[8] Two annotators identified keyphrases after a training phase with discussion of three documents. As in the MedForum dataset, the gold standard consists of the union of lemmatized keyphrases by both annotators. All datasets contain a unranked list of keyphrases.

The peDOCS dataset is by far the largest of the sets, since it has been created over the course of several years. MedForum and Pythagoras contain fewer documents but each document is annotated by a fixed pair of human annotators. The average number of keyphrases is highest for peDOCS and lowest for MedForum. The length of the document also influences the number of keyphrases as short documents have fewer keyphrase candidates. Keyphrases in all three datasets are on average very short. The example in Figure 1 gives an example of a rather specific keyphrase which, however, consists of only one token. We believe that keyphrase extraction approaches benefit from decompounding more in cases of short documents. Longer documents provide more statistical data which reduces the need for additional statistical data obtained with decompounding.

3.2 Experimental Setup

For preprocessing, we rely on components from the DKPro Core framework (Eckart de Castilho and Gurevych, 2014) and on DKPro Lab (de Castilho and Gurevych, 2011) for building experimental pipelines. We use the Stanford Segmenter[9] for tokenization, TreeTagger (Schmid, 1994; Schmid, 1995) for lemmatization and part-of-speech tagging. Finally, we perform stopword removal and decompounding as described in Section 2. It should be noted that in most preprocessing pipelines, decompounding should be the last step, as it heavily influences POS-tagging. We extract all lemmas in the document as keyphrase candidates and rank them according to basic ranking approaches based on frequency counts and the position in the document. We do not use more sophisticated extraction approaches, as we want to examine the influence of decompounding as directly as possible. However, it has been shown that frequency-based heuristics are a very strong baseline (Zesch and Gurevych, 2009), and even supervised keyphrase extraction methods such as KEA (Witten et al., 1999) use term frequency and position as the most important features and will be

[7] www.medizin-forum.de/

[8] www.dipf.de/en/research/projects/pythagoras

[9] nlp.stanford.edu/software/segmenter.shtml

heavily influenced by decompounding.

We evaluate the following ranking methods: **tf-idf**$_{\text{constant}}$ ranks candidates according to their term frequency $f(t, d)$ in the document. **tf-idf** decreases the impact of words that occur in most documents. The term frequency count is normalized with the inverse document frequency in the test collection (Salton and Buckley, 1988).

$$\text{tf-idf} = f(t, d) \log \frac{|D|}{|d \in D : t \in d|} \quad (3)$$

In this formula $|D|$ is the number of documents and $|d \in D : t \in d|$ is the number of documents mentioning term t. As some document collections may be too small to allow computing reliable frequency estimates, we also evaluated **tf-idf**$_{web}$. Again, the document frequency is approximated by the frequency counts from the Web1T corpus. We take the **position** of a candidate as a baseline. The closer the keyword is to the beginning of the text, the higher it is ranked. This is not dependent on frequency counts, but decompounding can also have an influence if a compound that appears early in the document is split into parts that are now also possible keyphrase candidates. We test each of the ranking methods with (w) and without (w/o) decompounding.

3.3 Evaluation metrics

For the keyphrase experiments, we compare results in terms of precision and recall of the top-5 keyphrases (P@5), Mean Average Precision (MAP), and R-precision (R-p).[10] MAP is the average precision of extracted keyphrases from 1 to the number of extracted keyphrases, which can be much higher than ten. R-precision[11] is the ratio of true positives in the set of extracted keyphrases when as many keyphrases as there are gold keyphrases are extracted.[12]

4 Results and discussion

In order to assess the influence of decompounding on keyphrase extraction, we evaluate the selected extraction approaches with (w/) and without (w/o) decompounding. The final evaluation results will be influenced by two factors:

[10]Using the top-5 keyphrases reflects best the average number of keyphrases in our evaluation datasets and is common practice in related work (Kim et al., 2013).

[11]This is commonly in information retrieval and first used for keyphrase identification in Zesch and Gurevych (2009)

[12]Refer to Buckley and Voorhees (2000) for an overview of evaluation measures and their characteristics.

Method	Δ P@5	Δ R@5	Δ R-p.	Δ MAP
Position	.000	.000	.000	.000
tf-idf$_{\text{constant}}$	.039	.030	.022	.012
tf-idf	.031	.024	.025	.015
tf-idf$_{web}$	.035	.021	.024	.012

Table 3: Difference of results with decompounding on the MedForum dataset.

Enhanced frequency counts: As we have discussed before, the frequency counts will be more accurate, which should lead to higher quality keyphrases being extracted. This affects frequency-based rankings.

More keyphrase candidates: The number of keyphrase candidates might increase, as it is possible that some of the parts created by the decompounding were not mentioned in the document before. This is the special case of a enhanced frequency count going up from 0 to 1.

We perform experiments to investigate the influence of both effects, first, the enhanced frequency counts, and second, the newly introduced keyphrase candidates.

4.1 Enhanced frequency counts

In order to isolate the effect, we limit the list of keyphrase candidates to those that are already present in the document without decompounding. We selected the MedForum dataset for this analysis, because it contains many compounds and has the shortest documents which we believe is best suited for an additional decompounding step.

Table 3 shows improvements of evaluation results for keyphrase extraction approaches on the MedForum datasets. The improvement is measured as the difference of evaluation metrics of using extraction approaches with decompounding compared to not using any decompounding. This table does not show absolute numbers, instead it shows the increase of performance. Absolute values are not comparable to other experimental settings, because all gold keyphrases that do not appear in the text as lemmas are disregarded. We can thus analyze the effect of enhanced frequency counts in isolation. Results show that for tf-idf$_{\text{constant}}$, tf-idf, and tf-idf$_{web}$ our decompounding extension increases results on the MedForum dataset considering only candidates that are extracted without decompounding. Decompounding does not affect results for the position baseline as it is not based on frequency counting. For the frequency-based approaches, the effect is rather

Dataset	Decompounding w/o	w	Δ
peDOCS	.614	.632	.018
MedForum	.592	.631	.038
Pythagoras	.624	.625	.002

Table 4: Maximum recall for keyphrase extraction with and without decompounding for the datasets.

small in general, however consistent across all metrics and methods. The decompounding extension, however, has the effect of adding further keyphrase candidates.

4.2 More keyphrase candidates

The second effect of decompounding is that new terms are introduced that cannot be found in the original document. Table 4 shows the maximum recall for lemmas with and without decompounding on all German datasets. The maximum recall is obtained by assuming that given a list of candidates the best possible set of keyphrases are extracted. Keyphrase extraction with decompounding increases the maximum recall on all datasets by up to 3.8% points. It must be noted that the increase is due to more keyphrase candidates extracted, which increases the importance of the final ranking. The increase is higher for MedForum while it is lower for Pythagoras. Pythagoras comprises summaries of lesson transcripts for students in the ninth grade, thus teachers are less likely to use complex words which need to be decompounded. The smaller increase for peDOCS compared to MedForum is due to longer peDOCS documents. The longer a document is, the more likely a part in a compound also appears as an isolated token which limits the increase of maximum recall. peDOCS shows to have a higher maximum recall compared to collections with shorter documents because documents with more tokens also have more candidates. MedForum comprises forum data, which contains both medical terms and informal description of such terms. Furthermore, gold keyphrases were assigned to assist others in searching. This leads to having documents containing terms like *Augenschmerzen* (Engl.: *eye pain*) for which the gold keyphrase *Auge* (Engl.: *eye*) was assigned.

4.3 Combined results

Previously, we analyzed the effects of decompounding in isolation, now we analyze the combination of enhanced frequency counts and more keyphrase candidates on the overall results. Table 5 shows the complete results for the German datasets, described keyphrase extraction methods, and with and without decompounding.

For the peDOCS dataset, we see a negative effect of decompounding. Only the position baseline and tf-idf$_{\text{constant}}$ benefit from decompounding in terms of mean average precision (MAP), while they yield lower results in terms of the other evaluation metrics. The improvement of the position baseline in terms of MAP might be to several correctly extracted keyphrases beyond the top-5 extracted keyphrases. We have previously discussed that peDOCS has on average the longest documents and most likely contains all gold keyphrases multiple times in the document text. For this reason, frequency-based approaches do not benefit from additional frequency information obtained from compounds. Many compounds are composed of common words, which already appear in the document. On the contrary, more common keyphrases are weighted higher, which hurts results in the case of peDOCS with highly-specialized and longer keyphrases. Depending on the task, this might be an undesired behavior.[13]

The only dataset for which the decompounding yields higher results is the MedForum dataset. Results improve with decompounding for tf-idf$_{\text{constant}}$ and tf-idf. As can be seen in Table 4, enhanced frequency counts improve results, and yield a higher maximum recall. Contrary to the other tf-idf configurations, results for tf-idf$_{web}$ decrease with decompounding. This leads to the observation that, besides the effect of enhanced ranking and more keyphrase candidates, a third effect influences results of keyphrase extraction methods: The ranking of additional keyphrase candidates obtained from decompounding. These candidates might appear infrequently in isolation and are ranked high if external document frequencies (*df* values) are used. Compound parts which do not appear in isolation[14]—hence, no good keyphrases—are ranked high in case of tf-idf$_{web}$ because their document frequency from the web is very low. In case of classic tf-idf they are ranked low because they are normalized with doc-

[13] When searching for documents, highly-specialized keyphrases might be better suited, while common keyphrases might be better suited for clustering of documents.

[14] The verb *begießen* (Engl.: *to water*) can be split into the verb *gießen* (Engl.: *to pour*) and the prefix *be* which does not appear as an isolated word.

Dataset	Method	Decompounding Precision@5 w/o	w/	Δ	Recall@5 w/o	w/	Δ	R-precision w/o	w/	Δ	MAP w/o	w/	Δ
peDOCS	*Upper bound*	.856	.864	.012	.393	.403	.010	.614	.632	.018	.614	.632	.018
	Position	.096	.068	-.028	.042	.030	-.012	.092	.080	-.012	.083	.086	.003
	tf-idf$_{\text{constant}}$	.170	.160	-.010	.075	.070	-.004	.127	.125	-.002	.123	.123	.001
	tf-idf	.137	.117	-.020	.060	.051	-.009	.107	.088	-.019	.112	.099	-.014
	tf-idf$_{web}$	**.188**	.168	-.020	**.083**	.074	-.009	**.139**	.126	-.013	**.139**	.129	-.010
MedForum	*Upper bound*	.867	.890	.023	.397	.422	.025	.592	.631	.038	.592	.631	.038
	Position	.082	.073	-.010	.049	.043	-.006	.101	.090	-.011	.142	.130	-.012
	tf-idf$_{\text{constant}}$	.149	.161	.012	.089	.096	.007	.144	.145	.001	.165	.162	-.003
	tf-idf	.235	**.282**	.047	.140	**.168**	.028	.210	**.234**	.025	.203	**.210**	.007
	tf-idf$_{web}$	.231	.165	-.067	.138	.098	-.040	.223	.159	-.064	.206	.180	-.027
Pythagoras	*Upper bound*	.941	.942	.001	.344	.344	.001	.624	.625	.002	.624	.625	.002
	Position	.030	.023	-.007	.014	.011	-.003	.044	.022	-.022	.106	.075	-.031
	tf-idf$_{\text{constant}}$	.137	.087	-.050	.066	.042	-.024	.143	.103	-.040	.153	.121	-.032
	tf-idf	.150	.150	.000	.072	.072	.000	.113	.114	.001	.141	.136	-.005
	tf-idf$_{web}$	**.187**	.100	-.087	**.090**	.048	-.042	**.205**	.102	-.103	**.191**	.136	-.055

Table 5: Results for keyphrase extraction approaches without (w/o) and with (w/) decompounding.

ument frequencies from a corpus where decompounding has been applied. In case of tf-idf$_{web}$, no decompounding has been applied. The effect of the poor ranking of newly introduced keyphrase candidates needs to be investigated further by conducting a manual analysis of the decompounding performance and the creation of non-words.

For the Pythagoras dataset, keyphrase extraction approaches yield similar results as for peDOCS. Decompounding decreases results, only results for tf-idf stay stable. As seen earlier (see Table 4), decompounding does not raise the maximum recall much (only by .002). As before in the case of the MedForum dataset, tf-idf$_{web}$ is influenced negatively by the decompounding extension. Results for tf-idf$_{web}$ decrease by .103 in terms of R-precision, which is a reduction of more than 50%. The ranking of keyphrases is hurt by many keyphrases, which appear as parts of compounds. They are ranked high because they infrequently appear as separate words. Considering the characteristics of keyphrases in Pythagoras, we see that keyphrases are rather long with 12.22 characters per keyphrase. This leads to the observation that the style of the keyphrases has an effect on the applicability of decompounding. Datasets with more specific keyphrases are less likely to benefit from decompounding.

5 Conclusions and future work

We presented a decompounding extension for keyphrase extraction. We created two new datasets to analyze these effects and showed that decompounding has the potential to increase results for keyphrase extraction on shorter German documents. We identified two effects of decompounding relevant for keyphrase extraction: (i) enhanced frequency counts, and (ii) more keyphrase candidates. We find that the first effect slightly increases results when updating the term frequencies, while including the second effect in the evaluation, reduces results for two of three datasets. We thus conclude that the effect of decompounding for keyphrases extraction requires further analysis, but may be a useful feature for supervised systems (Berend and Farkas, 2010).

In the future, we propose to further analyze characteristics of good keyphrases and whether they often are compounds. We see the potential for better decompounding approaches as any improvements on this task may have positive effects on keyphrase extraction. We would also like to investigate other effects that make tasks like keyphrase extraction especially hard. Named entity disambiguation might improve results further as some concepts are mentioned frequently in a text but always with another surface form. We make our experimental framework available to the community to foster future research.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, by the Klaus Tschira Foundation under project No. 00.133.2008, and by the German Institute for Educational Research (DIPF) We thank the anonymous reviewers for their helpful comments.

References

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008a. Decompounding Query Keywords from Compounding Languages. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 253–256, Stroudsburg, PA, USA. Association for Computational Linguistics.

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008b. German Decompounding in a Difficult Corpus. In *Computational Linguistics and Intelligent Text Processing*, volume 4919 of *Lecture Notes in Computer Science*, pages 128–139. Springer Berlin Heidelberg.

Marco Baroni, Johannes Matiasek, and H Trost. 2002. Predicting the Components of German Nominal Compounds. *ECAI*, pages 1–12.

Gábor Berend and Richárd Farkas. 2010. SZTERGAK: Feature Engineering for Keyphrase Extraction. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 186–189, Stroudsburg, PA, USA.

Chris Biemann, Uwe Quasthoff, Gerhard Heyer, and Florian Holz. 2008. ASV Toolbox: a Modular Collection of Language Exploration Tools. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 1760–1767, Paris. European Language Resources Association (ELRA).

Thorsten Brants and Alex Franz. 2006. Web 1T 5-Gram Version 1. In *Linguistic Data Consortium*, Philadelphia.

Chris Buckley and Ellen M. Voorhees. 2000. Evaluating Evaluation Measure Stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '00*, pages 33–40, New York, New York, USA.

Richard Eckart de Castilho and Iryna Gurevych. 2011. A Lightweight Framework for Reproducible Parameter Sweeping in Information Retrieval. In *Proceedings of the 2011 Workshop on Data Infrastructures for Supporting Information Retrieval Evaluation*, DESIRE '11, pages 7–10, New York, NY, USA. ACM.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A Broad-coverage Collection of Portable NLP Components for Building Shareable Analysis Pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT at COLING 2014*, pages 1–11.

Nicolai Erbs, Iryna Gurevych, and Marc Rittberger. 2013. Bringing Order to Digital Libraries: From Keyphrase Extraction to Index Term Assignment. *D-Lib Magazine*, 19(9/10):1–16.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. *Proceedings of the Association for Computational Linguistics (ACL), Baltimore, Maryland: Association for Computational Linguistics*.

Vera Hollink, Jaap Kamps, Christof Monz, and Maarten de Rijke. 2004. Monolingual Document Retrieval for European Languages. *Information Retrieval*, 7(1/2):33–52.

Anette Hulth. 2003. Improved Automatic Keyword Extraction given more Linguistic Knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic Keyphrase Extraction from Scientific Articles. *Language Resources and Evaluation*, 47:723–742.

Philipp Koehn and Kevin Knight. 2003. Empirical Methods for Compound Splitting. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 187–193, Stroudsburg, PA, USA. Association for Computational Linguistics.

Stefan Langer. 1998. Zur Morphologie und Semantik von Nominalkomposita. In *Tagungsband der 4. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS)*, pages 83–97.

Martha Larson, Daniel Willett, Joachim Koehler, and Gerhard Rigoll. 2000. Compound Splitting and Lexical Unit Recombination for Improved Performance of a Speech Recognition System for German Parliamentary Speeches. In *Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP)*, pages 945–948.

Torsten Marek. 2006. Analysis of German Compounds using Weighted Finite State Transducers. *Bachelor thesis, University of Tübingen*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of Empirical Methods for Natural Language Processing*, pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase Extraction in Scientific Publications. In *Proceedings of International Conference on Asian Digital Libraries*, volume 4822 of *Lecture Notes in Computer Science*, pages 317–326.

R. J. F. Ordelman. 2003. *Dutch Speech Recognition in Multimedia Information Retrieval*. Ph.D. thesis, University of Twente, Enschede, Enschede, October.

Gerard Salton and Christopher Buckley. 1988. Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5):513–523.

Pedro Bispo Santos. 2014. Using compound lists for german decompounding in a back-off scenario. In *Workshop on Computational, Cognitive, and Linguistic Approaches to the Analysis of Complex Words and Collocations (CCLCC 2014)*, pages 51–55.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK.

Helmut Schmid. 1995. Improvements in Part-of-Speech Tagging with an Application to German. In *Proceedings of the ACL SIGDAT-Workshop*, volume 21, pages 1–9.

Ian H Witten, Gordon W Paynter, and Eibe Frank. 1999. KEA: Practical Automatic Keyphrase Extraction. In *Proceedings of the 4th ACM Conference on Digital Libraries*, pages 254–255.

Torsten Zesch and Iryna Gurevych. 2009. Approximate Matching for Evaluating Keyphrase Extraction. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing*, pages 484–489.