Using compound lists for German decompounding in a back-off scenario

Pedro Bispo Santos
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Dept. of Computer Science, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de
santos@ukp.informatik.tu-darmstadt.de

Abstract

Lexical resources like GermaNet offer compound lists of reasonable size. These lists can be used as a prior step to existing decompounding algorithms, wherein decompounding algorithms would function as a back-off mechanism. We investigate whether the use of compound lists can enhance dictionary and corpus-based decompounding algorithms. We analyze the effect of using an initial decompounding step based on a compound list derived from GermaNet with a gold standard in German. The obtained results show that applying information from GermaNet can significantly improve all tested decompounding approaches across all metrics. Precision and recall increases statistically significant by .004-.018 and .011-.022 respectively.

1 Introduction

Compounds are words composed of at least two other lexemes and are a frequent linguistic phenomenon which can be found in several languages. English, Greek, Turkish, German, and Scandinavian languages are examples of languages which have compounds. In some languages, compounds can make part of a significant part of the corpus.[1]

Some compounds consist of two lexemes without any further modification, other require a linking element. *doorbell* and *toothbrush* are examples that do not require any change regarding their lexemes. However, this is not the case for every compound. *Verkehrszeichen*(*Verkehr+s+zeichen*, Engl = *traffic sign*) is a compound in German different from the ones presented before in English, as they require a linking element. The Greek word for cardboard box $\chi\alpha\rho\tau\acute{o}\kappa\upsilon\tau\sigma$ ($\chi\alpha\rho\tau\acute{\iota}$+$\kappa\upsilon\tau\acute{\iota}$) is a compound, for which both lexemes are modified as parts of the compound.

Although some compounds contain two other words, they may not be decompounded depending on the application. *Löwenzahn* consists of the terms *Löwe* and *Zahn*, however, this compound should not be split, since the compound itself has a different meaning from its constituents. This and the previous examples show why decompounding is not a straightforward problem to tackle.

Decompounding is of great importance for NLP tasks as its application as a preprocessing step improves results for several tasks. Monz and Rijke (2002) apply decompounding to information retrieval in German and Dutch and obtain an improvement of 25% for German and 70% for Dutch regarding average precision. Koehn and Knight (2003) obtain a performance gain of .039 BLEU in the German-English noun phrase translation task. Adda-Decker et al. (2000) apply decompounding to speech recognition and obtain a drop on the out of vocabulary word rate from 4.5% to 4.0%. These are just some examples of works in the literature that apply decompounding to other tasks. An improvement of decompounding methods might lead to further improvement of these tasks.

Lexical resources like GermaNet (Hamp and Feldweg, 1997) offer related German nouns, verbs, and adjectives semantically by grouping lexical units that express the same concept into synsets and by defining semantic relations between these synsets. Since version 8.0, GermaNet also offers a compound list indicating nouns that are compounds and how they should be split. In this work we tackle the question whether a prior decompounding step with a compound list improves results for existing decompounding algorithms. The existing algorithms are then used as a back-off solution.

[1] Schiller (2005) shows that for a large German newspaper corpus, 5.5% of 9,3 million tokens were identified as compounds.

2 Decompounding algorithms

Decompounding algorithms found in the literature can be divided in two categories: **lexicon-based** algorithms and **corpus-based** algorithms. Some of the **lexicon-based** algorithms base their lexicon on a corpus, although they do not use further information from the corpus. Additional information could be frequencies in monolingual corpora or words alignment in parallel corpora.

Among the **lexicon-based** algorithms there are works like the one from (Monz and Rijke, 2002), which used the CELEX lexical database for Dutch[2] and a tagger-based lexicon for German. The algorithm splits recursively a word from the right to left, as long as the remaining part of the word is also a word, so *Autobahnraststätte* would be split in *(Auto+(bahn+(rast+stätte)))*. They evaluated their results, and got reasonable results for Dutch and German when considering all nouns, more than 70% for micro/macro average precision/recall, but the results were not that good when evaluating only the complex nouns.

Corpus-based algorithms can then be divided in **monolingual** and **bilingual corpora** approaches. Among the **monolingual corpus** approaches there is the work from (Holz and Biemann, 2008) which filters splitting candidates by checking the minimal morpheme frequency in a corpus for each constituent. After this filtering process, it computes the geometrical mean of the constituent frequencies for each candidate and the one with the highest value is selected as the possible candidate. They use two corpora for evaluation, one from the CELEX lexical database for German and one manually constructed. The results were between 50%-70% of precision for both datasets, 1%-16% of recall for the CELEX database, and 36%-68% for the manually generated dataset.

Alfonseca et al. (2008) generates the candidates using a lexicon built from a corpus and then chooses the candidate by using a SVM classifier, wherein each training instance has different kinds of frequency-based features computed from a corpus. Weighted finite state transducers trained on a corpus are used by (Marek, 2006; Schiller, 2005) to split compound words.

Parallel corpora algorithms (Brown, 2002) are based on the idea that compounds in languages like German have their lexemes separated in their corresponding translation when translated to English. The work from (Koehn and Knight, 2003) uses both monolingual and parallel corpora in their work to learn morphological rules for compound splitting.

[2]http://wwwlands2.let.kun.nl/members/software/celex.html

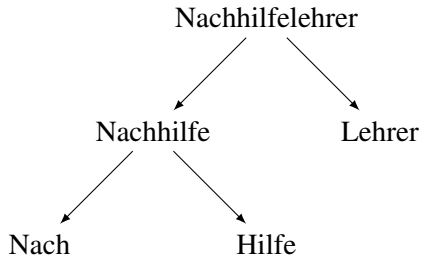

Figure 1: Decompounding of German term *Nachhilfelehrer* (Eng: Private tutor).

However, sometimes these methods might overlap. The work from (Monz and Rijke, 2002) relies on using lexical resources, but the German lexicon it uses for evaluation is based on a corpus. Brown (2002) uses a bilingual dictionary in its evaluation, which is derived from a parallel corpus.

Since some lexical resources offer compounds lists for languages like German. These compounds lists are specify how a compound must be split and the levels of decomposition, as Figure 1 shows. The hypothesis raised by this work is that these compound lists can be used as a prior decompounding step to improve the performance of **lexicon-based** and **corpus-based** algorithms.

3 Evaluation

The lexical resource GermaNet (Hamp and Feldweg, 1997; Henrich and Hinrichs, 2011) provides a list of compounds with their lexemes. This compound list was semi-automatically generated. A decompounding algorithm was run first, and then human annotators manually corrected the compounds which were wrongly split.

In this paper we present a system that uses this list as a primary source for decompounding and falls back to existing decompounding approaches if a word is not covered by this list. We analyze whether list-based decompounding improves existing decompounding algorithms.

Figure 2 illustrates our classification of the evaluated decompounding algorithms: **lexicon-based**, **corpus-based** and **compound list-based** algorithms. We use **lexicon** and **corpus** based algorithms as a back-off strategy for the GermaNet

Word	Split	Prefix String	Prefix Class	Suffix String	Suffix Class
Holzhaus	Holz-Haus	Holzhaus	4	suahzloH	4
Berggipfel	Berg-gipfel	Berggipfel	4	lefpiggreB	6
Hintergedanke	Hinter-gedanke	Hintergedanke	6	eknadegretniH	7

Table 1: Training set example for the prefix and suffix trie-based classifiers (Holz and Biemann, 2008)

compound list based algorithm.

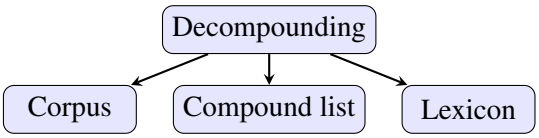

Figure 2: Decompounding algorithms used for evaluation

We use the lexicon-based decompounding API **JWord Splitter**[3]. It performs a dictionary look-up from left to right, and repeats this process if the remaining part of the word is not included in the dictionary. After JWordSplit finds words in both parts (left and right), it creates a split and stops.

This algorithm can generate several splitting candidates. A splitting candidate is a candidate to a possible decomposition. To judge which candidate will be the one selected, a ranking function is responsible for assigning scores to each candidate. We have ranked it by the geometric mean of the unigram frequencies from its constituents. This is based on the idea that the more frequent a candidate is, the more likely it is to be the correct decomposition

$$(\prod_{p_i \in C} count(p_i))^{\frac{1}{n}} \quad (1)$$

wherein C is a decomposition candidate, p_i is a constituent from the candidate and n is the number of constituents the candidate has. This frequency based metric is presented by Koehn and Knight (2003).

ASV Toolbox[4] is a modular collection of tools for the exploration of written language data. This toolbox offers solutions for language detection, POS-tagging, base form reduction, named entity recognition, terminology extraction and so on. It implements a decomposition algorithm which uses an information retrieval data structure called Compact Patrica Tree (CPT). It creates two CPTs (Holz and Biemann, 2008) from a specific corpus, one storing the suffixes for each word and another one storing the prefix, as Table 1 shows. More information about the construction of the CPTs can be found in (Witschel and Biemann, 2005).

A compound list-based decompounding algorithm is also implemented. This decompounding algorithm only splits a word if it is present in the compound list. If it is not there, then it supposes the word is not a compound. The GermaNet compound list[5] is chosen as the compound list for this list-based decompounder. This GermaNet list is also used as the prior step to JWordSplitter and ASV Toolbox in order to prove our hypothesis and check whether there is an improvement.

4 Results

The corpus created by (Marek, 2006) is used as 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 split. The compounds were obtained from the issues 01/2000 to 13/2004 of the German computer magazine c't[6], in a semi-automatic approach. Human annotators reviewed the list to identify and correct possible errors.

Koehn and Knight (2003) use a variation of precision and recall for evaluating decompounding performance:

$$\mathrm{P}_{comp} = \frac{\mathrm{cc}}{\mathrm{cc} + \mathrm{wfc}} \quad (2)$$

$$\mathrm{R}_{comp} = \frac{\mathrm{cc}}{\mathrm{cc} + \mathrm{wfc} + \mathrm{wnc}} \quad (3)$$

wherein **correct compound (cc)** is a compound which was correctly split, **wrong faulty compound (wfc)**, a compound which was wrongly split and **wrong non compound (wnc)**, a compound which was not split.

Table 2 shows that although GermaNet list approach's precision is very high. However, its recall is quite low, since it misses too many compounds

[3]https://github.com/danielnaber/jwordsplitter
[4]http://wortschatz.uni-leipzig.de/~cbiemann/software/toolbox/
[5]http://www.sfs.uni-tuebingen.de/lsd/compounds.shtml
[6]http://www.heise.de/ct/

Algorithm	R_{comp}	P_{comp}
GermaNet list	.083	.917
ASV Toolbox	.755	.799
ASV Toolbox with GermaNet list	**.766**†	**.803**†
JWord	.766	.799
JWord with GermaNet list	**.780**†	**.808**†

Table 2: Evaluation results. † indicates a statistical significant difference according to McNemar's Test.

which are not in the list. It is very hard to obtain a list-based decompounder with a good recall when applied to such datasets since it is impossible to obtain a list with every possible compound from the German language. The results show an improvement of the decompounding methods by the usage of compound lists in recall and precision with a statistical significance according to McNemar's (McNemar, 1947) Test, proving our hypothesis.

Using a list as a prior step could improve cases like *Badezimmer* (*Bad+zimmer*, Engl = bathroom), which is not split by ASV Toolbox and JWord original implementations. The reason is that *Badezimmer* by itself is a very frequent word since both approaches rely on corpus frequency. *Nordwestdeutschland* (*Nord+west+deutschland*, Engl = Germany northwest) is another case which the dictionary-based extension correctly solves. ASVToolbox splits only in two parts the compound, nord-west+deutschland, and JWord Splitter splits as nord+west+deutsch+land.

However, some cases could not be solved for none of the approaches. Cases like *kartenaufbau* (*karte+auf+bau*) are split like *karten+aufbau* by ASV Toolbox and JWord Splitter with and without compound list. GermaNet list does not contain this compound in its compound list, so no method was able to deal with this case. That is the case also for *ausdrucken* (*aus+drucken*), which is considered as not being a compound for every approach. Most of the cases which have a preposition as modifier were the cases which could not be solved by any of the decompounding algorithms.

5 Conclusion and Future Work

This paper raised the hypothesis of whether compound lists improve the performance of decompounding algorithms. We evaluated three different types of decompounding algorithms. Each algorithm was implemented and tested with a German gold standard containing more than 150,000 compounds. The results show that the best outcome is achieved by using a compound list as a prior step to existing decompounding algorithms, and then relying on the original algorithm as a back-off solution if the word is not found in the compound list.

For future work we want to test the algorithms on a dataset containing compounds as well as non-compounds. The reason for that is that we cannot evaluate false positives, in other words, non-compounds that are should not be split, but are. These cases need also to be considered.

References

Martine Adda-Decker, Gilles Adda, and Lori Lamel. 2000. Investigating text normalization and pronunciation variants for german broadcast transcription. In *Sixth International Conference on Spoken Language Processing*, pages 266–269.

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008. German Decompounding in a Difficult Corpus. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 128–139.

Ralf D. Brown. 2002. Corpus-driven splitting of compound words. In *Proceedings of the Ninth International Conference on Theoretical and Methodological Issues in Machine Translation*.

Birgit Hamp and Helmut Feldweg. 1997. Germanet - a lexical-semantic net for german. In *Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15.

Verena Henrich and Erhard Hinrichs. 2011. Determining Immediate Constituents of Compounds in GermaNet. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 420–426, Hissar, Bulgaria.

Florian Holz and Chris Biemann. 2008. Unsupervised and knowledge-free learning of compound splits and periphrases. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 117–127.

Philipp Koehn and Kevin Knight. 2003. Empirical Methods for Compound Splitting. In *10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 187–193.

Torsten Marek. 2006. Analysis of german compounds using weighted finite state transducers. *Bachelor thesis, University of Tübingen*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Christof Monz and Maarten Rijke. 2002. Shallow morphological analysis in monolingual information retrieval for dutch, german, and italian. In *Second Workshop of the Cross-Language Evaluation Forum*, pages 262–277.

Anne Schiller. 2005. German compound analysis with wfsc. In *5th International Workshop on Finite-State Methods and Natural Language Processing*, pages 239–246.

Hans Friedrich Witschel and Chris Biemann. 2005. Rigorous dimensionality reduction through linguistically motivated feature selection for text categorization. In *Proceedings of NODALIDA*.