Annotated Suffix Tree Method for German Compound Splitting

Anna Shishkova[1] and Ekaterina Chernyak[1]

National Research University – Higher School of Economics, Moscow, Russia,
asshishkova_1@edu.hse.ru, echernyak@hse.ru

Abstract. The paper presents an unsupervised and knowledge-free approach to compound splitting. Although the research is focused on German compounds, the method is expected to be extensible to other compounding languages. The approach is based on the annotated suffix tree (AST) method proposed and modified by Mirkin et al. To the best of our knowledge, *annotated* suffix trees have not yet been used for compound splitting. The main idea of the approach is to match all the substrings of a word (suffixes and prefixes separately) against an AST, determining the longest and sufficiently frequent substring to perform a candidate split. A simplification considers only the suffixes (or prefixes) and splits a word at the beginning of the selected suffix (the longest and sufficiently frequent one). The results are evaluated by precision and recall.

Keywords: compound splitting, annotated suffix tree, German language

1 Introduction

The Germanic languages, in particular German, often use a large number of compounds, i.e. words consisting of several parts. Such word combinations result in larger vocabulary. Moreover, the word base may eventually expand further even without the formation of completely new words, only through compounding the existing ones.

This especially complicates machine translation and search engine development. For examlpla, without compound splitting it is necessary to store all the words *Sommer* (*summer*), *Winter* (*winter*), *Wetter* (*weather*), *Märchen* (*fairy tale*), *Sommerwetter* (*summer weather*), *Winterwetter* (*winter weather*), *Wintermärchen* (*winter's tale*) to translate them correctly. In contrast, a system equipped with compound splitting tool needs only four words (*Sommer*, *Winter*, *Wetter*, *Märchen*) to represent the entire group. What is more, such system will easily translate the previously unseen word, *Sommermärchen*. In terms of search engines, compounds prevent search result diversity. For instance, to find results containing *Regenmantel* (*raincoat*) by query *Regenschirm* (*umbrella*) it is useful to split the compounds as well.

Section 2 reviews different approaches to this problem. Due to the space limitation, the review is brief and incomplete yet sufficient to provide a general outline of the problem. Section 3 introduces the proposed algorithm for compound splitting. Section 4 is devoted to the experiment results. Section 5 concludes.

2 Related Work

This paper combines two aspects: German compound splitting and annotated suffix tree method. To correctly split a compound, it is necessary to study both areas.

2.1 Compound Splitting

In [3], F. Holz and C. Biemann point out that the approaches to compound splitting can be divided into knowledge-intensive and knowledge-free ones. According to [3], knowledge-intensive splitters are based on handcrafted rules or use supervised learning on training sets of splitted words, while knowledge-free approaches are not language-specific and create a compound splitter by analyzing a raw text corpus.

The work [5] is focused on German, Dutch and Italian compounds. As a knowledge-free approach it can be applied to any language having compound words. The authors provide the reader with an algorithm that parses sequential combinations of the first letters (referred to as prefixes) one by one and accommodates them with a list of possible splitting candidates (all non-compound words) established beforehand. If a match is found, the current prefix is considered as the first part of splitting. Parsing continues until there are tokens left or splitting fails, unable to match a new prefix with the list. The authors evaluate their method on a set of around 700 complex nouns, achieving 60% precision and 50% recall.

However, as mentioned in [3], this approach fails if the compound includes words having an independent root and an auxiliary part. F. Holz and C. Biemann give an example word *Prüfungsvorbereitungsstress* (stress that occurs when one prepares for an exam). The correct splitting should contain the word *Prüfung* but there is an independent German root *Prüf*, while *ung* is a nonsensical auxiliary part. After marking *Prüf* as the first part of splitting, the analysis becomes unsuccessful.

Building upon previous research, F. Holz and C. Biemann [3] develop a more sophisticated knowledge-free algorithm. During the preprocessing, a word list is collected from a large raw corpus and marked with frequencies. Then, all possible splits are generated and the program checks whether the precomputed word list contains the proposed items. The split is chosen based on the lengths of the components and other measures, such as the largest geometric mean of partial frequencies.

Next, the authors improve the results by introducing periphrase detection. They illustrate the idea with an example: *Schweine-schnitzel* is a pork cutlet, while *Kinder-schnitzel* means a cutlet for children. Despite the same structure, the meanings are strictly different. To avoid terminological ambiguities, the authors look for periphrases in the source text. Given that periphrases contain different prepositions (*Schnitzel vom Schwein* and *Schnitzel für Kinder*), the authors include the prepositions into splits and thus facilitate understanding. This approach however goes beyond the scope of our paper.

One more alternative approach is described in [2]. The authors consider German words too, but in contrast to the previous paper the results are not simply transferred to other languages. Instead, corpora tagged with part-of-speech (POS-tagged) are used to supervise learning. Moreover, the aims of the task differ from the previous paper: the research addresses the question not of discovering of compounds, but inflectional phenomena of the German language. To perform the task, prefixes and suffixes are also treated similarly to [2]. Afterwords the results are matched with POS-tagged corpora.

2.2 Annotated Suffix Tree

Moving on to the annotated suffix trees, the paper [4] defines an annotated suffix tree (AST) as a rooted tree where each non-root node corresponds to a character and the path from the root to the leave encodes a suffix. In addition to the character label, each node is associated with a number, that denotes the frequency of the fragment. The authors specify an important property of the frequencies: the node frequency always equals the sum of its child node frequencies. The parent node corresponds to a prefix of several suffixes, and its frequency is the sum of the frequencies of these suffixes. Therefore, it is observed that the frequency of the parent node equals the sum of the frequencies of the leaves it covers. Furthermore, the authors give an exact algorithm for constructing annotated suffix trees based on the sequential enumeration of all the suffixes from the analyzed string's collection.
The paper [1] deals with the use of the AST to calculate the relevance measure between given string and text. The authors compare the results of applying popular three measures: cosine similarity, probabilistic BM25 and the new proposed measure. The measure proposed is based on a conditional probabilities of symbols in AST-induced text fragments. As confirmed in [1], the results obtained with AST-based characteristics are superior to the other more popular measures. The authors also emphasize that AST is applicable to other tasks, such as categorization.

3 Our Method

To avoid terminological ambiguities, we use the term "annotated prefix tree" (APT) to denote an annotated suffix tree constructed from the inverted words. For instance, the prefixes of *Zeit* (*time*) are represented in an annotated tree as the suffixes of a dummy word *tieZ*, namely, *Z*, *eZ*, *ieZ*, *tieZ*. The examples of both AST and APT created from word *Freizeit* (*free time*) are illustrated on Fig.1 (left and right respectively).
Now we describe our algorithm in details. At the current stage, it aims to split the simplest compounds formed by a direct connection of two words. Thus, we consider such compounds as *Kraftwerk* (*power station*), which is formed by *Kraft* (*power*) and *Werk* (*factory*), but avoid the more complex ones: *Kleidungsstück* (*item of clothing*) which includes not only *Kleidung* (*clothing*) and *Stück* (*item*)

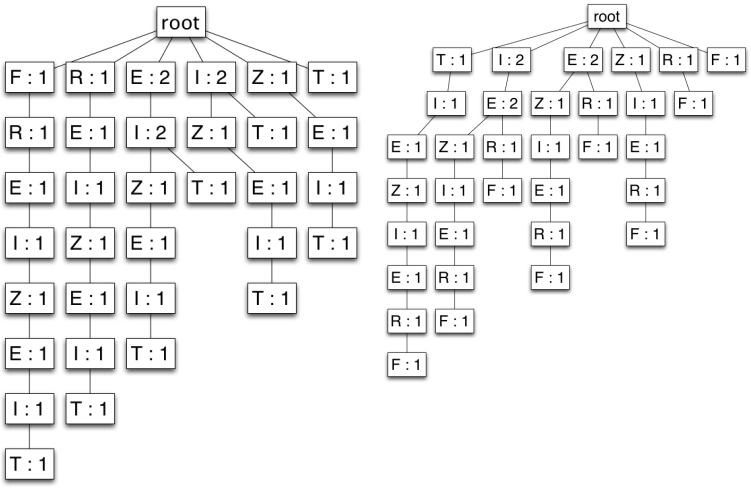

Fig. 1. AST (left) and APT (right) created from word *Freizeit* (*free time*)

but also a connecting letter *s*, or *Kirchturm* (*church steeple*), which omits the letter *e* from the word *Kirche* (*church*). Further development of the algorithm should address the difficulties mentioned above as well as splitting the words formed from more than two parts.

Therefore, the algorithm is as follows:

Step 1. Create an AST and an APT from a large raw list of (compound) words.

Step 2. For each suffix (prefix) of each word to be split, determine the frequency from the AST (APT). The frequency of a suffix (prefix) is defined by the frequency of the corresponding leaf, that is, the last element of the suffix (prefix) in the AST (APT). Then pick the longest suffix (prefix) with frequency above a certain cutoff value. For our evaluation set, the optimal cutoff frequencies are calculated as the geometric mean of the suffix and prefix frequencies divided by the multiplication of the suffix and prefix lengths. The next step can be done in two ways.

Step 3a. The compound is split using either the AST or the APT. The suffix obtained at the previous step is proposed as the second part of the compound, and the remainder becomes the first one. The case of prefix splitting (using APT) is handled similarly.

Step 3b. AST and APT results are handled jointly, producing a combined split. Since in Step 3a prefix splitting shows better results, it is applied in most cases. However, if the frequency of the prefix obtained from prefix splitting is lower than that of *every* part of suffix splitting, the latter is selected.

The Step 3b is justified by the experiment presented in the following section. However, even without particular experiments the superior results of prefix splitting can be explained from a linguistic point of view. German has a fixed set of common word endings. Let us proceed with an example: if two words which

share an ending (e.g. *Sommer* and *Winter*) are the prefixes of some compounds with the same second part (*Sommerwetter* and *Winterwetter*), the frequency of the erroneous suffix (*erwetter*) increases. That may lead to wrong suffix splitting, but does not affect the prefix frequencies.
In the next section, we present the results produced by an implementation of the three options (prefix, suffix and combined splitters) and evaluate these by precision and recall.

4 Experiments

To build AST and APT we use a list from Wiktionary, Category: German compound words. It consists of 7144 words, but we use 6740 of them with well-described etymology. An example of well-described etymology is, for *Apfelblüte* (*apple blossom*), *Apfel* + *Blüte*. Availability of etymology has no effect on splitting quality, but allows checking its correctness.
Since the algorithm only handles two-part compounds with no connecting letters, the experiment is conducted on an appropriate 4619-word subset of the original list. The split may consist of one part, which makes it unsuccessful. However, in presence of non-compounds this might be the correct answer; such cases should be taken into account when calculating precision. Precision is measured as the ratio of number of correct determined splits to number of all words which splits were found. The numerator of recall is the same while the denominator is the number of all compounds in the evaluating set.
The splitting examples of two different words are shown in Table 1.

Table 1. Splitting of *Bierglas* (*glass of beer*) and *Apfelbaum* (*apple tree*)

word	suffixes (freq.)	suff. split	prefixes (freq.)	pref. split	combined split
Bierglas	s (5186) as (455) las (71) glas (16) **rglas (4)** erglas (3) ierglas (1) Bierglas (1)	Bie+rglas (false)	B (423) iB (42) eiB (10) **reiB (7)** greiB (2) lgreiB (1) algreiB (1) salgreiB (1)	Bier+glas (true)	**Bier+glas** **(true)** because Bier (**7**), Bie (10), rglas (4), 4 < **7** <10
Apfelbaum	m (2311) um (622) aum (66) **baum (21)** lbaum (3) elbaum (3) felbaum (1) pfelbaum (1) Apfelbaum (1)	Apfel+baum (true)	A (352) pA (13) fpA (9) efpA (9) lefpA (9) **blefpA (3)** ablefpA (1) uablefpA (1) muablefpA (1)	Apfelb + aum (false)	**Apfel+baum** **(true)** because Apfelb (**3**), Apfel (9), baum (21), **3** < 9 < 21

Using AST provides precision and recall of about 48%, while using APT improves these to 57%. The combined method further improves precision and recall up to 63%. It is also necessary to point out that in 78% of the cases at least one split (obtained either by AST or APT) is correct. That means that if the algorithm contains a manual post-processing component, we would have easily reached such value.

5 Conclusion

In this paper, we have proposed an algorithm for compound splitting based on the annotated suffix tree method. Precision and recall of more than 60% provide a good starting point. With manual post-processing precision can reach 78%, which suggests a prospective direction of further work if automating of the heuristics is applied.

At the current stage, the algorithm only splits simple compounds, which account for two-thirds of the total Wiktionary list. A minor modification of the algorithm should allow it to handle connecting letters or an independent third (middle) stem. However, such modifications have not been implemented yet. Additionally, the extensions may consider the more advanced cases, such as the removal of certain ending letters from the parts in compounds.

Finally, it is also necessary to carry out the experiments on other word lists, containing non-compounds, both at tree-construction and splitting stages.

References

1. *Chernyak E. L.* An approach to the problem of annotation of research publications, in: Proceedings of The Eighth International Conference on Web Search and Data Mining. NY, United States of America : Ch. 58. P. 429-434. ACM, 2014
2. *Finkler, W., Neumann, G.* Morphix: A fast realization of a classification-based approach to morphology. In: 4. Österreichische Artificial-Intelligence-Tagung. Wiener Workshop - Wissensbasierte Sprachverarbeitung, 1998
3. *Holz, F., Biemann, C.* Unsupervised and knowledge-free learning of compound splits and periphrases. CICLing'08 Proceedings of the 9th international conference on Computational linguistics and intelligent text processing, P. 117-127. Springer, Heidelberg, 2008
4. *Mirkin B. G., Chernyak E. L.* An AST method for scoring string-to-text similiarity in semantic text analysis, in: Clusters, orders, trees: methods and applications. In Honor of Boris Mirkin's 70th Birthday / Sci. Ed.: F. T. Aleskerov, B. I. Goldengorin, P. M. Pardalos. Vol. 92. Berlin : Springer, 2014
5. *Monz, C., de Rijke, M.* Shallow morphological analysis in monolingual information retrieval for dutch, german, and italian. In: Peters, C., Braschler, M., Gonzalo, J., Kluck, M. (eds.) CLEF 2001. LNCS, vol. 2406, P. 262–277. Springer, Heidelberg, 2002