Towards Unsupervised and Language-independent Compound Splitting using Inflectional Morphological Transformations

Patrick Ziering
Institute for Natural Language Processing
University of Stuttgart, Germany
Patrick.Ziering@ims.uni-stuttgart.de

Lonneke van der Plas
Institute of Linguistics
University of Malta, Malta
Lonneke.vanderPlas@um.edu.mt

Abstract

In this paper, we address the task of language-independent, knowledge-lean and unsupervised compound splitting, which is an essential component for many natural language processing tasks such as machine translation. Previous methods on statistical compound splitting either include language-specific knowledge (e.g., linking elements) or rely on parallel data, which results in limited applicability. We aim to overcome these limitations by learning compounding morphology from inflectional information derived from lemmatized monolingual corpora. In experiments for Germanic languages, we show that our approach significantly outperforms language-dependent state-of-the-art methods in finding the correct split point and that word inflection is a good approximation for compounding morphology.

1 Introduction

Compounding represents one of the most productive word formation types in many languages. In particular, Germanic languages (e.g., German or Dutch) show high productivity in closed compounding, i.e., in creating one-word compounds such as the German *Armutsbekämpfungsprogramm* 'poverty elimination program'. Previous studies on German corpora reveal that almost half of the corpus types are compounds, whereas individual compounds are very infrequent (Baroni et al., 2002). Therefore, an automatic compound analysis is indispensable and represents an essential component in many natural language processing (NLP) tasks such as machine translation (MT) or information retrieval (IR).

Besides determining the concatenated constituent forms, i.e., the correct split points (e.g., *Armuts* | *bekämpfungs* | *programm*), a compound splitter needs to normalize each part (e.g., *Armut* + *Bekämpfung* + *Programm*), because down-stream applications such as MT systems expect lemmatized words as input. However, normalization of constituent forms is non-trivial and usually requires language-specific knowledge (e.g., linking elements). State-of-the-art lemmatizers, designed for regular word inflection, would fail, because constituent forms often contain linking elements leading to a non-paradigmatic word form of the corresponding lexeme (e.g., *Armuts* 'poverty + *s*' never occurs as an isolated token in German corpora, since the *s*-suffix, often used for genitive or pluralization, is not used with *Armut*). Moreover, morphological operations during compounding vary a lot across languages and lexemes: we find cases that start from the lemma and have additions (e.g., linking elements), truncations (e.g., reductions to a verbal stem), word-internal operations (e.g., Umlautung) and combinations thereof (e.g., the first constituent of the German *Weihnachts|baum* 'Christmas tree', *Weihnachten*, undergoes both the *en*-truncation and the *s*-suffixation).

In this paper, we present a language-independent, unsupervised compound splitter that normalizes constituent forms by tolerantly retrieving candidate lemmas using an Ngram index and weighting string differences with inflectional information derived from lemmatized corpora.

Most previous work on compound splitting includes language-specific knowledge such as large

Proceedings of NAACL-HLT 2016, pages 644–653,
San Diego, California, June 12-17, 2016.

lexicons and morphological analyzers (Fritzinger and Fraser, 2010) or hand-crafted lists of linking elements and rules for modeling morphological transitions (Koehn and Knight, 2003; Stymne, 2008; Weller and Heid, 2012), which makes the approaches language-dependent. Macherey et al., (2011) were the first to overcome this limitation by learning morphological compounding operations automatically by retrieving compounds and their constituents from parallel corpora including English as support language.

We would like to take this one step further by avoiding the usage of parallel data, which are known to be sparse and frequently domain-specific, while Bretschneider and Zillner (2015) showed that compounding morphology varies between different domains. Instead, we exploit lemmatized corpora and use word inflection as an approximation to compounding morphology. This way, we are able to process compounds of any type of domain.

Our contributions are as follows. Firstly, we develop a language-independent and unsupervised compound splitter that does not rely on parallel data. As we will show, our system significantly outperforms language-dependent, knowledge-rich state-of-the-art methods in predicting the best split point. Secondly, in a controlled experiment, we show that compound splitting based on inflectional morphology performs similarly to splitting based on an extensive hand-crafted set of rules for compounding morphology. Thirdly, we perform a comprehensive, intrinsic evaluation of compound splitting, which is often missing in previous work that focuses on task-based evaluation (e.g., MT), and thus evaluates performance only indirectly. We compare splitting performance for several languages for two disciplines: (1) prediction of the correct split points and (2) normalization of the constituent forms. To the best of our knowledge, we are the first to evaluate these disciplines separately.

The paper is structured as follows. Section 2 outlines previous work on compound splitting. Section 3 discusses some theoretical assumptions on which we base our splitting method. Section 4 shows two efficient and flexible data structures used for our statistical compound splitter, which is described in Section 5. Section 6 presents some splitting experiments performed on German, Dutch and Afrikaans. Finally, Section 7 concludes and points to future work.

2 Related work

In the following discussion, we focus on splitting approaches that address morphological transformations, as these are most relevant for our work. Previous work on compound splitting can be roughly divided into two groups: (1) *statistical* approaches that are mainly based on large corpora and (2) *linguistically based* splitters, usually relying on knowledge-rich morphological analyzers or rules.

Statistical approaches generate all possible splits and rank them according to corpus statistics. Although independent of lexical resources, most methods contain morphological knowledge in terms of linking elements. The most influential statistical splitter is developed by Koehn and Knight (2003) who addressed German compound splitting by scoring splits according to the geometric mean of the potential constituents' frequencies. For normalization, they selected the two fillers ⊕s and ⊕es. Stymne (2008) performed several experiments to measure the impact of varying parameters of Koehn and Knight's (2003) algorithm for factored statistical MT. Instead of using two single fillers, she implemented the collection of the 20 most frequent morphological transformations for German compounding as presented by Langer (1998). She observed that splitting parameters should not necessarily be the same for translating in different directions. Bretschneider and Zillner (2015) compared the splitting performance between Koehn and Knight's (2003) two fillers and Langer's (1998) collection, illustrating the necessity of an exhaustive set of linking elements. Moreover, they showed that Langer's (1998) data is still not sufficient for domain-specific targets. Macherey et al., (2011) were the first to overcome the need for manual morphological input and the limitation to a fixed set of linking elements by learning morphological operations automatically from parallel corpora including a support language which creates open compounds and has only little inflection, such as English. We take this one step further by avoiding the dependence on such parallel corpora, known to be sparse, and by approximating compounding mor-

phology with word inflection learned from monolingual preprocessed data.

Linguistically based splitters are usually relying on a lexical database or a set of linguistic rules. While these splitters outperform statistical approaches (Escartín, 2014), they are designed for a specific language and thus less applicable to other languages. Nießen and Ney (2000) used the morpho-syntactic analyzer GERTWOL (Mariikka Haapalainen and Ari Majorin, 1995) for splitting compounds. Schmid (2004) developed the morphological analyzer SMOR, that enumerates linguistically motivated compound splits. Fritzinger and Fraser (2010) combined SMOR with Koehn and Knight's (2003) statistical approach and outperformed both individual methods. Weller and Heid (2012) extended the splitter of Koehn and Knight (2003) with a list of PoS-tagged lemmas and a handcrafted set of morphological transition rules. While our approach similarly exploits lemma and PoS information, we avoid the manual input of transition rules.

3 Theoretical preliminaries

The splitting architecture, data structure, features and evaluation we propose in this paper are based on a number of assumptions and considerations that we would like to discuss first.

3.1 Morphological transformation

Closed (or concatenative) compounding is the main spelling form in many languages around the world, e.g., Germanic languages such as *German*, *Dutch*, *Swedish*, *Afrikaans* or *Danish*, Uralic languages such as *Estonian* or *Finnish*, Hellenic languages such as *Modern Greek*, Slavic languages such as *Russian* and many more. Most closed-compounding languages use morphological transformations. German or Dutch often insert a linking element between the constituents while Greek reduces the first constituent to its morphological stem and adds a compound marker. In contrast to conjugations of irregular verbs (such as *to be*), there is only a minor string difference (e.g., in terms of edit distance (ED)) between constituent form and corresponding lemma (usually they differ in at most two characters). This minimal difference makes it possible to interpret constituent normalization as a kind of tolerant string retrieval (which is presented for the case of spelling correction within IR by Manning et al., (2008)). This is why we are using an Ngram index for retrieving the candidate lemmas with the highest string similarity to the constituent form.

3.2 Inflectional morphology

Relying exclusively on the highest string similarity for the normalization, would lead to candidate lemmas that result from linguistically unmotivated operations, e.g., the German *Hühner* 'chickens' would be normalized to the most string-similar lemma *Hüne* 'giant' (ED=2) and not to the correct but less string-similar lemma *Huhn* 'chicken' (ED=3). Thus, a linguistic restrictor is indispensable for finding the underlying lemma for a given constituent.

In many languages, inflectional morphology shares operations with compounding morphology, e.g., the German *Hühner* in *Hühner|suppe* 'chicken soup' is equivalent to the plural form of *Huhn*. But even for non-paradigmatic constituent forms (e.g., *Armuts* 'poverty'), we can find cases of inflection that use the transformation at hand (e.g., the genitive form of window: *Fensters*).

We thus decided to approximate compounding morphology by using inflectional morphology as derived from lemmatized corpus tokens. We realize that the inflectional approximation does not work for all closed-compounding languages but it does for a large subset that is known to have a large variety of linking elements and is therefore most in need of unsupervised morphology induction, the Germanic languages. Moreover, our flexible system can be easily supported with morphological information, which is suitable for languages like Greek, that use a special compound marker.

3.3 Compound headedness

Most closed-compounding languages usually follow the **r**ight**h**and **h**ead **r**ule (RHHR), i.e., the head of a compound is the right-most constituent and encodes the principal semantics and the PoS of the compound. As done by previous splitting approaches (Stymne, 2008; Weller and Heid, 2012), we assume the RHHR and allow only splits for which the right-hand side constituents has the same PoS as the compound.

3.4 Splitting depth

The granularity of the morphological analysis needed differs with the type of application. For MT, a compound should not be split deeper than into parts for which a translation is known, whereas for linguistic research, a deeper morphological analysis is desirable.

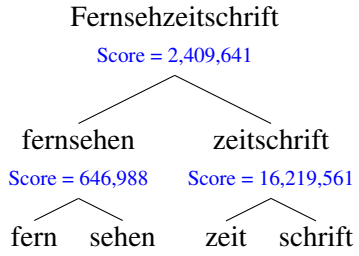

Figure 1: Linguistically motivated split

For example, while an MT system needs a binary split for the German *Fernsehzeitschrift* 'television journal', for a linguistic analysis, a split into four parts as given in Figure 1 is also valid and introduces etymological clues (e.g., how far is *Zeit* 'time' related to *Zeitschrift* 'journal'). Our flexible approach caters for all tasks[1].

3.5 Constituent length balance

While compounds can be build up from almost any semantic concept pair, we observed a bias towards constituent pairs having a similar word length.

For the German, Dutch and Afrikaans compound splitting gold standards (described in Section 6.2) comprising m split compounds, we randomly recombine all modifiers with all heads to a set of m recombinations. For both original compounds and recombined compounds, we measure the character difference in length between modifier and head form.

Compound set	German	Dutch	Afrikaans
Original	2.62	2.26	2.87
Recombination	3.43	2.74	2.92

Table 1: Average constituent length difference in characters

As shown in Table 1, all original compounds have a smaller difference than the recombinations. Therefore, we decided to promote compound splits with more balanced constituent lengths.

[1] As the splitting depth is very dependent on the task at hand and the gold standard we used is not created with a specific task in mind, we do not evaluate our system with respect to splitting depth but treat it as given. Our system cuts off subtrees with the lowest splitting score until the desired splitting depth is achieved.

4 Data preparation

4.1 Ngram index

As described in Section 3.1, we tolerantly retrieve candidate lemmas using an Ngram index, in order to limit the search space and allow for a quick candidate lookup during splitting.

Ngram	Lemma length (LL)	Lemmas
^hund	4	hund#13162
^hund	11	hundeführer#251, hundehalter#81, hundesteuer#64
^h*hn	4	hahn#2078, huhn#1839, hohn#506

Table 2: Examples from the German Ngram index

As search key, we use Ngrams of variable length ($N \leq 15$). Word-initial Ngrams are indicated by ^ and word-final Ngrams end on $. By using Ngrams, we are able to capture any kind of transformation a lemma can undergo when involved in compounding.

Sometimes, a transformation includes a character replacement within the word (e.g., Umlautung). This leads to a very small set of Ngrams a constituent has in common with its underlying lemma. For example *Hühner* and *Huhn* only have the bigrams ^h and hn in common, which is also true for many irrelevant words such as *^Haarschnitt* 'haircut'. In order to reduce noise and increase efficiency, we include the wildcard $*$ for a single character in Ngrams[2]. This way, *Hühner* and *Huhn* have the common 5gram ^H*hn. As a further cue, we consider the lemma length (assuming that there is only a minor difference to the constituent's length).

For a given lemmatized and PoS-tagged corpus, we index all content words (i.e., nouns, adjectives and verbs) by generating all Ngrams and mapping them to a list of frequency-ranked lemma-freq pairs. Table 2 shows some examples for German[3].

[2] For efficiency reasons, we add wildcard Ngrams only for $3 \leq N \leq 7$.

[3] Note that the lemmas include compounds, because these are necessary for our binary recursive splitter, described in Section 5.

Language	MOP	Corpus frequency	Examples
German	u/ü:$/er$	117K	<Huhn, Hühner> 'chicken', <Buch, Bücher> 'book'
	um$/en$	264K	<Studium, Studien> 'study', <Medium, Medien> 'medium'
Dutch	$/en$	1.6M	<arts, artsen> 'doctor', <band, banden> 'tyre'
Afrikaans	$/se$	34K	<proses, prosesse> 'process'

Table 3: Examples of MOPs for German, Dutch and Afrikaans

4.2 Morphological operation patterns

Macherey et al., (2011) describe a representation of compounding morphology using a single character replacement at either the beginning, the middle or the end of a word. For our experiments, we adopt this format. Since it is possible that a morphological operation takes place at several positions of a word, we combine all atomic replacements into a pattern describing a series of operations. This transformation from a word Σ to a word Ω is referred to as **m**orphological **o**peration **p**attern (MOP). For compiling an MOP, we use the Levenshtein edit distance algorithm including the four operations INSERT (adding a character), DELETE (removing a character), REPLACE (exchanging a character σ_i by ω_i) and COPY (retaining a character). In a backtrace step, we determine the first set of operations that lead to a minimum edit distance. Except for COPY, we interpret all operations as replacements (insertion and deletion are replacements of or by an empty element ϵ respectively). We merge all adjacent replacements by concatenating the source and target characters. Word-initial source and target sequences start with ^ and word-final sequences end on $. Sequences of adjacent COPY operations are represented by '`:`' and separate the merged replacements. For example, in *Hühner*|*suppe*, the modifier lemma *Huhn* is transformed to *Hühner* by replacing `u` by `ü` (i.e., Umlautung) and adding the suffix `er`. The corresponding MOP is '`u/ü:$/er$`'. The second column in Table 3 shows some additional German, Dutch and Afrikaans examples of MOPs.

As discussed in Section 3.2, we try to approximate compounding MOPs using inflectional MOPs. In a lemmatized corpus, for each lemmatized word token, we determine the MOP that represents the transformation from lemma to word form. We collect all inflectional MOPs with their token-based corpus frequency. The third column in Table 3 shows the corpus frequencies for the corresponding MOPs.

5 Compound splitting method

Our compound splitter can process compounds composed of any content word type (i.e., nouns, verbs and adjectives) and of any number of constituents, and provides both the split points (e.g., *Hühner*|*suppe*) and the normalized constituents (e.g., *Huhn* + *Suppe*). The splitter is designed recursively, which allows us to represent the compound split both hierarchical (i.e., as a tree structure) and as a linear sequence. Figure 2 shows the architecture of our splitting algorithm. The recursive main method starts with the target word as a single constituent and recursively splits the constituents produced by the binary splitter (Section 5.1) until an atomic result is returned. The binary splitter has two subtasks: (1) for each potential constituent form, a set of candidate lemmas is retrieved (Section 5.2) and (2) all candidate lemma combinations are ranked and the best split is returned (Section 5.3).

5.1 Binary splitter

We first generate all possible binary splits with a minimum constituent length of 2 (e.g., for *Ölpreis* 'oil price', we generate *Öl*|*preis*, ..., *Ölpre*|*is*) and add a non-split option. For each potential constituent among the generated splits, we retrieve the M most probable lemmas as described in Section 5.2. We consider all M^2 lemma combinations of all possible splits and rank them as described in Section 5.3. The highest-ranked split is returned.

5.2 Candidate lemma retrieval

In this step, we retrieve the M most probable candidate lemmas for a given constituent. For this task, we make use of the Ngram index, described in Section 4.1. Instead of applying MOPs directly which would be the classical and more efficient way, we decided to look up candidates using the Ngram in-

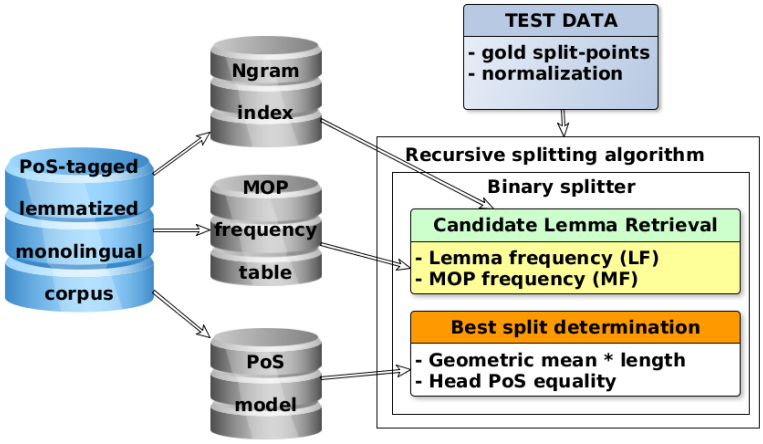

Figure 2: Architecture of our splitting algorithm

dex first, thereby following the assumption that there is only a minor string difference between lemma and constituent form (cf. Section 3.1). In a second step, the inflectional MOPs are used to rank the candidate lemmas. While following this order, we put less weight on our approximation and thereby avoid false lemmas due to irrelevant inflectional MOPs. The pseudocode for the candidate lemma retrieval is given in Algorithm 1.

Algorithm 1 Candidate lemma retrieval

```
1:  Constituent c
2:  LLs ← lemma lengths, ±Δ around len(c)
3:  CLs ← [ ]                          ▷ the candidate lemmas
4:  for L ← len(c) to 1 do
5:      LGs ← generate all Lgrams of c
6:      for Lg in LGs do
7:          CLs ← CLs + topλ(IDX[Lg][LLs])
8:      if len(CLs) > 1 then
9:          break        ▷ otherwise, L is decremented
10: score(CLs)           ▷ according to a lemma model
11: rank(CLs)            ▷ according to the scores
12: return topM(CLs)
```

For a given constituent c, we search for lemmas with a minimum lemma length (LL) of 2 which ranges between $\pm\Delta$ around the length of c (lines 1-2). All retrieved candidate lemmas are stored in the list CLs (line 3). Starting with the L = len(c), we inspect all Lgrams of c (lines 4-5). For a given Lgram, we retrieve the top λ most frequent lemmas that have a length $\pm\Delta$ around the length of c (lines 6-7). If there are no lemmas retrieved, we decrement L (lines 8-9)[4]. All retrieved candidate lemmas are scored (line 10) according to our lemma model, for which we present two lemma features.

$$LP(l_i) = cf(l_i) \cdot count(l_i, \texttt{CLs}) \quad (1)$$

The first feature is based on the lemma prominence (LP) as given in (1), i.e., we multiply the corpus frequency (cf) of a lemma l_i (as given in the Ngram index) with the token number of l_i in CLs (i.e., with the prominence of l_i among all inspected Lgrams).

The second feature estimates the suitability of the MOP (MS) transforming the candidate lemma l_i to the constituent form at hand, c, (represented as '$MOP[l_i, c]$'), as given in (2). As the first component, we use the corpus frequency extracted with the inflectional MOPs as described in Section 4.2. We rescale the MOP frequency with the resulting edit distance between the candidate lemma l_i and the constituent form at hand, c, (represented as $ED(l_i, c)$)[5]. As motivated in Section 3.1, we expect MOPs having a small edit distance to be more prominent in compounding. Such MOPs are not necessarily most frequent in inflection, e.g., the frequent irregular Afrikaans verb *wees* (*to be*) leads to MOPs like MOP[*wees*,*is*] = `^wee/^i`, which has an ED of 3.

$$MS(l_i) = \frac{cf(MOP[l_i, c])}{ED(l_i, c) + 1} \quad (2)$$

[4]For noise reduction due to lemmatizer errors, we can predefine a minimum number of L decrements.

[5]For avoiding a division by zero, we add 1.

All candidate lemmas are finally scored as product of lemma prominence and MOP suitability, as given in (3).

$$score(l_i) = LP(l_i) \cdot MS(l_i) \quad (3)$$

We rank all candidate lemmas and return the top M candidates (lines 11-12).

5.3 Best split determination

In the final step, we determine the best split among all split combinations (i.e., pairs of retrieved candidate lemmas for modifier (l_m) and head (l_h), and corresponding split point) and the non-split option. For this task, we use a combination model, which considers the interaction between l_m and l_h. Inspired by Koehn and Knight (2003), as a first feature, we take the geometric mean of the products of lemma score multiplied by the length of the corresponding constituent form, as given in (4). The length factor promotes splits with more balanced lengths (as motivated in Section 3.5), which mitigates the impact of short and high-frequent words on the overall score. For binary splits, we use the constituent set $con = \{l_m, l_h\}$ and for the non-split option, we use $con = \{l_h\}$.

$$geoLen(con) = \sqrt[|con|]{\prod_{l_i \in con} score(l_i) \cdot len(c_{l_i})} \quad (4)$$

The second feature is based on the assumption that the PoS of a compound word Ψ usually equals the PoS of its head l_h, as discussed in Section 3.3. Since our splitter works out of context, we try to subsume all possible PoS tags by representing them as a distribution over the PoS probabilities $p(\text{PoS}|\text{word}) = \frac{freq(PoS \cap word)}{freq(word)}$ acquired from the monolingual PoS-tagged corpus. The value of the head-PoS-equality (hEQ) feature is defined as the the cosine similarity between the PoS probability distributions of compound word Ψ and head l_h, $hEQ(\Psi, l_h)$. If the PoS tag of the compound is unknown, we take 1.0 as default value.

$$split(con) = geoLen(con) \cdot hEQ(\Psi, l_h) \quad (5)$$

Finally, all candidate lemma combinations (including the non-split option) are ranked according to the splitting score given in (5). The highest-scored split is returned as output of the binary splitter, being subject to the recursive process. Figure 3 shows an example of the recursive splitter output for the German compound *Studienbescheinigungsablaufdatum* 'enrollment certification expiration date' with the related MOPs.

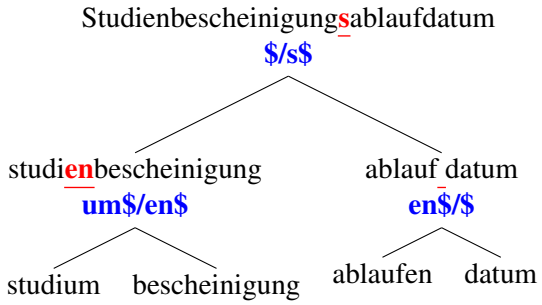

Figure 3: Example of a split tree structure with related MOPs

6 Experiments

In our experiments, we focus on German, Dutch and Afrikaans, but expect to see similar performance for other Germanic languages.

6.1 Data

We use the German and Dutch version of Wikipedia[6] and the Afrikaans Taalkommissie corpus[7]. For tokenizing, PoS-tagging and lemmatizing Wikipedia, we use Treetagger (Schmid, 1995).

Corpus language	# tokens	# types		
	words	words	lemmas	MOPs
German	665M	9.0M	8.8M	1201
Dutch	114M	2.0M	1.9M	920
Afrikaans	57M	748K	696K	459

Table 4: Corpus statistics

We tokenize the Taalkommissie corpus using the approach of Augustinus and Dirix (2013). We PoS-tag the corpus using the tool described in Eiselen and Puttkammer (2014) and use the lemmatizer of Peter Dirix, the second author of the previous paper.

Table 4 shows some statistics of the three preprocessed corpora. Since the Afrikaans corpus is one order of magnitude smaller than the German corpus, we expect a lower performance for the Afrikaans splitter.

[6] {de,nl}.wikipedia.org

[7] Taalkommissie van die Suid-Afrikaanse Akademie vir Wetenskap en Kuns (2011)

System	SPAcc			NormAcc		
	@1	@2	@3	@1	@2	@3
(A) LP.MS$_{\text{infl}}$	**95.2%**^{B,C}	**98.9%**^{B,C}	**99.4%**^{B,C}	86.6%	**94.6%**^{B,C}	**96.5%**^{B,C}
(B) WH2012	93.3%	95.6%	95.7%	81.0%	85.9%	86.4%
(C) FF2010	91.4%	92.3%	93.0%	**88.4%**^{A,B}	89.7%	90.2%
(D) LP.$\varnothing$	54.1%	70.5%	78.4%	28.4%	42.6%	50.8%
(E) LP.MS$_{\text{Langer}}$	94.5%	98.7%	99.1%	87.1%	94.2%	95.4%
(F) LP.MS$_{\text{GS}}$	95.4%	99.0%	99.4%	87.8%	95.6%	97.2%

Table 5: German results for binary compound splitting, scores δ^{Φ} outperform the system Φ significantly

6.2 Gold standard

For evaluating our splitting method on **German**, we use the binary split compound set developed for GermaNet[8] by Henrich and Hinrichs (2011). After removing hyphenated compounds[9], it comprises **51,230** binary split samples. For **Dutch** and **Afrikaans**, we use the split point gold standards developed by Verhoeven et al., (2014), which comprise **21,941** samples for Dutch and **17,369** for Afrikaans.

6.3 Evaluation measures

We evaluate the splitting quality with respect to two disciplines: (1) determination of the correct split points and (2) normalization of the resulting modifier constituents[10]. For both disciplines, we use the accuracy measure as described in Koehn and Knight (2003). The split point accuracy (**SPAcc**) refers to the correctness of the split points (on word level) and the normalization accuracy (**NormAcc**) measures the amount of both correct split points and modifier lemmas. All systems presented in this paper provide a ranked list of splits. This allows for a more fine-grained ranking evaluation of the binary splitting decisions with respect to the first n positions. Accuracy@n refers to the amount of correct splits among the top n splits. We stop at $n = 3$, because we do not expect to see a crucial difference in the performance gap for higher values of n.

[8] sfs.uni-tuebingen.de/GermaNet

[9] We consider hyphenated compounds as trivial cases of splitting that can be disregarded for our purpose.

[10] Since for Germanic languages compounding morphology is exclusively found on the modifier, we disregard the head.

6.4 Parameter setting and models in comparison

There are three parameters presented in the candidate lemma retrieval. For efficiency reasons, we set the number of lemmas retrieved per Lgram and lemma length (λ) to 20 and the final number of retrieved candidate lemmas (M) to 3. For the maximum difference in length between lemma and constituent form, we observed that $\Delta = 2$ covers all compounding operations for Germanic languages.

For German, we compare our system based on inflectional MOPs (LP.MS$_{\text{infl}}$) against the LP baseline (LP.$\varnothing$), which lacks a linguistic restrictor after candidate lemma lookup from the Ngram index, against an upper bound (LP.MS$_{\text{GS}}$), which uses the MOPs and frequencies derived from the normalizations in the gold standard and against a version that uses a hand-crafted set of MOPs and frequencies derived from Langer's (1998) set of fillers (LP.MS$_{\text{Langer}}$). In addition, we compare our system against previous work: the splitting methods of Fritzinger and Fraser (2010) and of Weller and Heid (2012)[11]. For Dutch and Afrikaans, we compare with the SPAcc numbers of Verhoeven et al., (2014).

6.5 Results and discussion

Table 5 shows the **German** results for the binary compound splitting. We present the split point accuracy (SPAcc) and the normalization accuracy (NormAcc) for the splits ranked @1-3. We first compare LP.MS$_{\text{infl}}$ against the previous work of FF2010 and WH2012. Our system significantly[12] outperforms both systems with respect to SPAcc and reaches

[11] We use an updated version of Weller and Heid (2012) developed and provided to us by Marion Di Marco.

[12] Approximate randomization test (Yeh, 2000), $p < 0.05$

99.4% for SPAcc@3. While for NormAcc@1 our splitter's performance is less than 2 percentage point lower than the system of FF2010, which heavily relies on language-dependent and knowledge-rich resources, we significantly outperform both systems in comparison for NormAcc@2 and NormAcc@3. This proves that one can attain state-of-the-art performance on compound splitting by using language-independent and unsupervised methods, and in particular by means of inflectional information.

In an error analysis, it turned out that FF2010 (i.e., SMOR) cannot process 2% of the gold samples. However on a common processable test set, we still find our system to outperform FF2010 significantly, which indicates that the difference in performance is not just a matter of coverage. WH2012 leaves several compounds unsplit, for which our splitter provides the correct analysis. This is partly due to the hand-crafted transition rules of WH2012, which cannot capture all morphological operations, such as in *Hilfs|bereitschaft* 'cooperativeness', for which the MOP `e$/s$` (i.e., the combination of *e*-truncation and *s*-suffixation) is not even covered by Langer's (1998) published collection.

To evaluate whether our assumption about the usability of inflectional MOPs holds, we run some controlled experiments with two variants of our system, shown in the last two lines of Table 5. The exhaustive set of inflectional MOPs ($\text{LP.MS}_{\text{infl}}$) shows competitive performance in SPAcc with the hand-crafted set of Langer (1998) ($\text{LP.MS}_{\text{Langer}}$) and with the upper bound (LP.MS_{GS}). The latter outperforms $\text{LP.MS}_{\text{infl}}$ by only 1 percentage point in NormAcc.

Our separate evaluation of SPAcc and NormAcc reveals a lower performance for the normalization across all systems, as this is a much harder discipline. In addition, we can conclude that normalization requires more linguistic knowledge: while the LP-baseline (LP.∅) underperforms heavily, both FF2010 and LP.MS_{Langer}, systems with a lot of lexical and morphological information, outperform all systems in comparison at NormAcc@1.

For illustrating the multilingual applicability of our splitter, we perform an experiment on Dutch and Afrikaans. Table 6 shows the SPAcc[13] results for N-ary splits. While Verhoeven et al., (2014) use a supervised approach for predicting the correct split points, our unsupervised splitter outperforms their **Dutch** results significantly[14]. Although Dutch and Afrikaans are similar languages, the SPAcc achieved by our system for **Afrikaans** is 3.6% worse than the method presented by Verhoeven et al., (2014). This result is partly due to the crucial corpus size differences presented in Table 4.

System	Dutch	Afrikaans
$\text{LP.MS}_{\text{infl}}$	**93.4%**	84.7%
Verh.et.al.2014	91.5%	**88.3%**

Table 6: SPAcc results for N-ary splits in Dutch and Afrikaans

7 Conclusion

We presented a language-independent, unsupervised compound splitter based on inflectional morphology that significantly outperforms state-of-the-art methods in finding the correct split points, relying only on monolingual PoS-tagged and lemmatized corpora. We provided a comprehensive, intrinsic evaluation of several systems in comparison for several languages on two separate disciplines: split point determination and constituent normalization. As a result, we draw the conclusions that inflectional morphology is a practical approximation for compounding in Germanic languages and overcomes the necessity of manual input, because both hand-crafted sets of compounding operations and operations derived from the gold standard lead to small differences in performance only. In future work, we plan to adapt our methods for learning compounding morphology for languages such as Greek, that have a special compound marker.

Acknowledgments

We thank the anonymous reviewers for their helpful feedback. We also thank our colleague Stefan Müller for the discussions and feedback, and Marion Di Marco and Fabienne Cap for providing their splitting methods. This research was funded by the German Research Foundation (Collaborative Research Centre 732, Project D11).

[13]The Dutch and Afrikaans gold standard only provides split points and no reliable normalized constituents.

[14]z-test for proportions; $p < 0.05$

References

Liesbeth Augustinus and Peter Dirix. 2013. The IPP effect in Afrikaans: A Corpus Analysis. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 213–225.

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Predicting the Components of German Nominal Compounds. In *ECAI*, pages 470–474. IOS Press.

Claudia Bretschneider and Sonja Zillner. 2015. Semantic Splitting of German Medical Compounds. In *Text, Speech, and Dialogue*. Springer International Publishing.

Roald Eiselen and Martin J. Puttkammer. 2014. Developing Text Resources for Ten South African Languages. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 3698–3703.

Carla Parra Escartín. 2014. Chasing the Perfect Splitter: A Comparison of Different Compound Splitting Tools. In *LREC 2014*.

Fabienne Fritzinger and Alexander Fraser. 2010. How to Avoid Burning Ducks: Combining Linguistic Analysis and Corpus Statistics for German Compound Processing. In *Proceedings of the ACL 2010 Joint 5th Workshop on Statistical Machine Translation and Metrics MATR*, pages 224–234.

Verena Henrich and Erhard W. Hinrichs. 2011. Determining Immediate Constituents of Compounds in GermaNet. In *RANLP 2011*, pages 420–426.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *EACL*.

Stefan Langer. 1998. Zur Morphologie und Semantik von Nominalkomposita. In *KONVENS*.

Klaus Macherey, Andrew M. Dai, David Talbot, Ashok C. Popat, and Franz Och. 2011. Language-independent Compound Splitting with Morphological Operations. In *ACL HLT 2011*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Mariikka Haapalainen and Ari Majorin. 1995. GERTWOL und Morphologische Disambiguierung für das Deutsche. Technical report.

Sonja Nießen and Hermann Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *COLING 2000*, pages 1081–1085.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German Computational Morphology Covering Derivation, Composition, and Inflection. In *LREC 2004*, pages 1263–1266.

Helmut Schmid. 1995. Improvements in Part-of-Speech Tagging with an Application to German. In *ACL SIGDAT-Workshop*.

Sara Stymne. 2008. German Compounds in Factored Statistical Machine Translation. In *GoTAL*.

Taalkommissie van die Suid-Afrikaanse Akademie vir Wetenskap en Kuns. 2011. Taalkommissiekorpus 1.1. Technical report, CTexT, North West University, Potchefstroom.

Ben Verhoeven, Menno van Zaanen, Walter Daelemans, and Gerhard B van Huyssteen. 2014. Automatic Compound Processing: Compound Splitting and Semantic Analysis for Afrikaans and Dutch. In *ComAComA 2014*, pages 20–30.

Marion Weller and Ulrich Heid. 2012. Analyzing and Aligning German compound nouns. In *LREC 2012*.

Alexander Yeh. 2000. More Accurate Tests for the Statistical Significance of Result Differences. In *COLING 2000*.