NoCs: A Non-Compound-Stable Splitter for German Compounds

Carmen Schacht
Ruhr-University Bochum, Germany
Faculty of Philology
Department of Linguistics
carmen.schacht@rub.de

Abstract

Compounding—the creation of highly complex lexical items through the combination of existing lexemes—can be considered one of the most efficient communication phenomenons, though the automatic processing of compound structures—especially of multi-constituent compounds—poses significant challenges for natural language processing. Existing tools like *compound-split* (Tuggener, 2016) perform well on compound head detection but are limited in handling long compounds and distinguishing compounds from non-compounds. This paper introduces NoCs (non-compound-stable splitter), a novel Python-based tool that extends the functionality of *compound-split* by incorporating recursive splitting, non-compound detection, and integration with state-of-the-art linguistic resources. NoCs employs a custom stack-and-buffer mechanism to traverse and decompose compounds robustly, even in cases involving multiple constituents. A large-scale evaluation using adapted GermaNet data shows that NoCs substantially outperforms *compound-split* in both non-compound identification and the recursive splitting of three- to five-constituent compounds, demonstrating its utility as a reliable resource for compound analysis in German.

1 Introduction

Compounding constitutes a core word formation process found in many languages of the world and is considered a highly efficient strategy for speakers to convey complex information by only employing little amount of linguistic signal. It is a process of combining pre-existing lexemes into new ones, thus creating informationally more compact structures as compared to their syntactically more embedded phrasal counterparts (Biber and Gray, 2010). For example, the compound *lexeme combination process* encodes the same information as *the process of the combination of lexemes*, but in a condensed linguistic unit, resulting in reduced signal transmission time and thus more efficient communication. This paper follows the definition of a compound of Jenkins et al. (2023) as "single orthographic words which are composed of two or more constituents". Lexicalized compounds such as *Tischbein* (Engl. 'Table-leg') will be included while opaque compounds like *Himbeere* (Engl. 'Raspberry') are not, where one of the constituents is considered an opaque morpheme (i.e. "Modifiers whose meaning is not transparent any more without considering the etymology of the word" according to Henrich and Hinrichs (2011); in this case, *Him-* is opaque). In addition, words that can not be split further into independent constituents are considered non-compounds.

German, the language of the empirical focus of this paper, is a particularly well suited candidate for the investigation of compounding processes, as it offers an high amount of observable compounding. As one of the most frequently encountered word formation processes in German, compounding has to be considered not only as an efficient mechanism to transmit complex information but also as a highly generative process for ad hoc vocabulary. German compounds exhibit a theoretically almost unrestricted length and composition, vividly demonstrated by the well known compound in Example 1.

(1) *Rindfleischetikettierungsüberwachungsaufgabenübertragungsgesetz*
(Engl. 'Beef Labeling Monitoring Task Transfer Act')

To investigate compounds computationally—especially long compounds like the one above—they often have to be split into their respective parts to examine the processes influencing their production and processing. Those

Proceedings of the Student Research Workshop associated with RANLP-2025,
pages 44–53, Varna, Bulgaria, Sep 8-10, 2025.
https://doi.org/10.26615/issn.2603-2821.2025_006

parts are called modifier and head, where the modifier modifies the head. Both can themselves be compounds embedded in the first level of a compound. German compounds are structured head-last, i.e. the right element of a split being modified by the left element. According to Henrich and Hinrichs (2011), who published an extensive WordNet-style semantic network including semantically annotated compounds called GermaNet, the meaning of the entire compound highly depends on the meaning of its parts. Günther et al. (2020), who investigate the semantic transparency of compounds based on the relatedness of their constituents also show, that it is crucial to not only consider compounds as a whole unit, but to also analyze their respective parts and underlying structure as compound interpretation is more than the sum of the compound's parts. The difficulty in splitting a compound into those parts lies in the possibility of various possible splits especially in multi-constituent compounds according to Hätty et al. (2019), who evaluate several tools for the automatization of this task. This is due to a variable internal structure called branching structure in compounds with at least three constituents. Following the examples of Kösling and Plag (2009), a three-constituent compound can either be left-branching like in Example 2, where the first split is made between the second and third constituent, or right-branching like in Example 3, where the compound is first split between the first and second constituent.

(2) *seat belt law*
Interpretation: A law concerning seat belts.
Branching structure: [[NN]N]

(3) *team locker room*
Interpretation: A locker room for the team.
Branching structure: [N[NN]]

This impacts the internal structure of head and modifier, as in Example 2 the modifier itself is a compound and in Example 3 the head is a compound, each being able to be split into a head and modifier themselves. Annotating and curating compounds manually is thus a costly endeavor, both in terms of time and personnel, thereby motivating the development of automatic annotation methodologies. Multiple tools and resources have been developed and evaluated to date, addressing various aspects of automatic processing of compounds. The tool presented in this paper contributes to this line of research by offering a novel approach to automatic compound analysis by building on the *compound-split* tool by Tuggener (2016). It introduces a functionality to detect non-compounds as well as improving the handling of compounds longer than three constituents. Both tools—NoCs and *compound-split*—are then tested against each other on compounds of various lengths as well as on non-compounds. To promote open-access resources, NoCs will be made available at Gitlab under a CC BY 4.0 license.[1]

2 Related work

The production and processing of compounds have been of particular interest for psycho- and computational linguistics, as their processing is highly dependent on the audience group and their respective prior knowledge (Halliday, 1988/2004; Kendeou and van den Broek, 2007) as well as on linguistic (Meßmer et al., 2021) and communicative context (Gamboa et al., 2024, 2025). Psycholinguistic approaches to compounding behavior include the investigation of seriality of their processing (Andrews et al., 2004), differences between novelty and lexicalization in compounds (Hyönä et al., 2020), structural properties like the branching direction in multi-constituent compounds (Kösling and Plag, 2009) as well as the semantic relations between the separate constituents of compounds (Benjamin and Schmidtke, 2023). Ormerod et al. (2024) argue that Large Language Models (LLM) are able to distinguish between compounds sharing the same relation and compounds with different relations, incorporating both psycho- and computational linguistic approaches.

Especially when written or transcribed language is under investigation and has to be processed, curated and analyzed, pre-processing compounding and its underlying processes become a central task for natural language processing (NLP) and are therefore highly relevant to computational linguistics, as the computational processing of compound structures is not at all trivial. Several tasks of NLP highly depend on successfully processing compound structures such as the identification of the compound head for coreference resolution (Tuggener, 2016) or the analysis of the different dimensions of information status (Riester and Baumann, 2017).

[1] Gitlab: https://gitlab.ruhr-uni-bochum.de/schaccmr/nocs.git.

Various tools and resources tackling those tasks exist to date, each addressing specific aspects relevant to the individual task and in part building upon one another: see for example Hätty and Schulte im Walde (2018) for termhood prediction, Henrich and Hinrichs (2011) for the compound extension of GermaNet including constituent properties such as affixoid or opaque morpheme, Krotova et al. (2020) for the classification of compound idiomaticity, Tuggener (2016) for the identification of compound heads in the context of coreference resolution, Svoboda and Sevcikova (2024) for parent retrieval or Weller and Heid (2012) for compositional alignment of (compounded) terms in translation tasks. Simultaneously, those tools and resources do, however, exhibit various shortcomings, such as handling only compounds comprised of a maximum of three constituents or not being able to differentiate between compounds and non-compounds in the first place. The current paper will be addressing those aspects by presenting NoCs for German compounds based on the *compound-split* library; the Python implementation of the probabilistic n-gram based compound head detection algorithm presented in Tuggener (2016). The algorithm predicts the most probable split point within a word and returns the scores for various possible positions of a split in this word. It thus follows a machine learning approach trained on approximately one million German nouns from Wikipedia. It is a freely available NLP tool for the processing of German compounds, offering extensive documentation and achieving 95% accuracy for the detection of compound heads on the test data from GermaNet (Henrich and Hinrichs, 2011). It was selected for extension in this project due to its licensing terms, usability, and support for both nominal and adjectival compounds as it is a well-documented, license-free and straight-forward to use Python-library. Due to employing a machine learning approach it performs robustly without being computationally costly and data hungry like LLMs. It offers functions called *char_split*, which is splitting the head from its modifier and *maximal_split*, splitting the entire compound maximally not regarding the branching structure of a compound.

Although the tool performs robustly in compound head detection—its primary purpose—and achieves high accuracy values, its *maximal_split* function can only process compounds with no more than three constituents, while the *char_split* function lacks the ability to differentiate between compounds and non-compounds, resulting in erroneous segmentation like in Example 4.

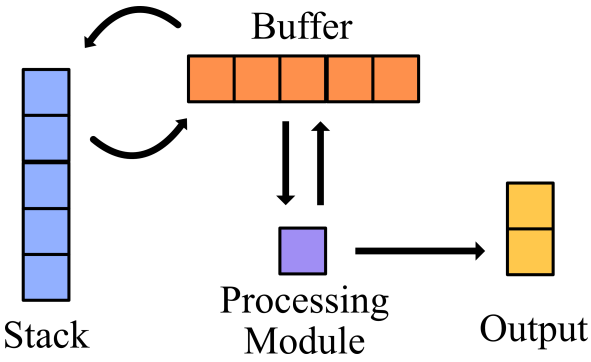

Figure 1: Architecture of the stack and buffer system implemented by NoCs.

(4) *Anma-Ssung instead of Anmassung*
(Engl. 'presumption')
Erroneous split of the non-compound
Anmassung by the *char_split* function.

These constraints render it unsuitable for recursive decomposition of longer compounds into their respective heads and modifiers using only those two functions, as this would not reliably stop the splitting process at compound constituents, that can not be split any further.

3 Architecture of NoCs

To overcome these issues of the *Compound-split* tool, the newly developed NoCs tool introduced in this paper is addressing these issues building on and extending *compound-split* by leveraging the (correct and incorrect) feedback the base functions give to implement a custom stack and buffer system for recursive compound traversal and appropriate segmentation. NoCs is also implemented in Python (Van Rossum and Drake, 2009), an open-source programming language, to promote open-access resources and accessible research.

In addition to *compound-split* it also incorporates the state-of-the-art NLP *stanza* library (Qi et al., 2020) for part-of-speech and morphological feature look-ups during compound processing. It also utilizes a minimally adapted version of the *Free German Dictionary* (Schreiber, 2021) which is—like *compound-split*— based on Wikipedia crawls for dictionary look-ups and manually created lists of German prefixes and suffixes.

The tool's architecture is schematically presented in Figure 1. Details on the stack and buffer implementation are provided in Section 3.1 and 3.2, followed by a description of the test dataset in Section 4 and an evaluation of the results of both tools in Section 5.

3.1 Stack and buffer system

While the *char_split* function only detects the head of a compound and the *maximal_split* function can not properly handle compounds with more than three constituents, reliable recursive decomposition of multi-constituent compounds and non-compound identification are challenging when relying solely on the base functionalities of *compound-split*. Specifically, *char_split* exhibits unstable behavior on non-compounds, frequently producing erroneous outputs such as *Anm-Assung* from *Anmassung* ('presumption'). The position of the erroneous splits appears to be arbitrary, rendering it impossible to detect an erroneous split based on the split position alone. However, the length of the output lists returned when further splitting erroneous constituents displays predictable patterns that NoCs leverages for non-compound detection and recursive splitting.

Example 5 shows the output of *char_split* for the correct split of the compound *Testbeispiel* (Engl. 'test example'). The function returns a ranked list of candidate split positions with associated scores.

(5) *[(0.9571421456504741, 'Test', 'Beispiel'),*
(-0.7465882530347583, 'Testbei', 'Spiel'),
(-0.9921253246209264, 'Tes', 'Tbeispiel'),
(-1.5950942705473183, 'Testbeis', 'Piel'),
(-2.2783109404990403, 'Testb', 'Eispiel'),
(-2.2790028763183123, 'Testbe', 'Ispiel'),
(-2.660451197053407, 'Testbeisp', 'Iel')]
(Engl.: 'test example.')

The length and structure of these lists, however, is arbitrary and cannot reliably serve as indicators of compoundhood as the tool returns lists of variable lengths. Spurious splits for non-compounds may also result in singleton (see 6) or arbitrarily long lists (see Example 7 and 8).

(6) *[(-1.2889316935842348, 'Flü', 'Gel')]*
(Engl.: 'wing')

(7) *[(-1.3002238718039354, 'Bea', 'Mter'),*
(-1.5774219936893772, 'Beam', 'Ter')]
(Engl. 'administrative officer')

(8) *[(-1.137630662020906, 'Anma', 'Ssung'),*
(-1.3213892018267495, 'Anmaß', 'Ung'),
(-1.4081508515815087, 'Anm', 'Aßung')]
(Engl. 'presumption')

None of these splits are correct, but the tool still assigns a variable number of scores. In some cases however, the tool does seem to detect a non-compound and returns a score of 0 together with the unsplit token like in Example 9.

(9) *[(0, 'Käfig', 'Käfig')]]*
(Engl. 'cage')

NoCs evaluates these outputs recursively to identify failed segmentation, leveraging the output patterns of the base function and classify the input as a non-compound head.

It implements a stack-buffer architecture wherein partially segmented constituents are recursively pushed and popped from the stack and buffer through the processing module and onto the output. The individual elements are thus repeatedly examined for compoundhood and passed on through the stack-and-buffer system after successful segmentation and labeling. This architecture integrates the base function *char_split* of *compound-split* and evaluates the individual outputs through a series of tests for correct or erroneous splits.

The functionality of the basic stack-buffer architecture is demonstrated in the pseudo-code of Listing 1.

Each compound is pushed to the stack first, then recursively split using the *char_split* function and checked for compoundhood. The right elements of each split get pushed further to the buffer. The buffer-loop represents the inner processing containing the precessing module, which applies several tests to verify the compoundhood of an element. If the rightmost element is found it gets pushed to output and the next element on the buffer is processed until all elements are found and processed accordingly. Since initial *char_split* decisions and outputs are sensitive to compound length, NoCs avoids hardcoded assumptions and dynamically assigns left splits back to the stack while buffering right splits for continued processing in the processing module. This recursive traversal mechanism facilitates full decomposition, with backtracking when required (see Figure 1). To verify compoundhood, NoCs applies several criteria in this processing module to circumvent false segmentations by the base function.

NoCs_2	CS_2	CS_lemma_2	MS_2	MS_lemma_2	NoCs_3	MS_3	MS_lemma_3
2550	2351	3310	2049	2873	1530	473	907
0.51	0.47	**0.66**	0.41	**0.58**	**0.31**	0.1	0.18

Table 1: Absolute count and percentages of correct splits from the two and three constituent datasets. CS encodes the *char_split* function, MS the *maximal_split* function.

NoCs_4	MS_4	NoCs_5	MS_5	NoCs_noun	CS_noun	NoCs_adj	CS_adj
206	9	5	0	4099	935	537	187
0.21	0.01	0.21	0.0	**0.82**	0.19	**0.88**	0.31

Table 2: Absolute count and percentages of correct splits from the four and five constituent and non-compound datasets.

Listing 1: Stack-Buffer Architecture

```
procedure SplitCompounds(list)
 for compound in list do:
  push compound to stack
  while stack not empty do:
   split rightmost element in stack
       using char-split
   if split is correct:
    push right element of split to
        buffer
    push left element back to stack

    # bufferLoop:
    while buffer not empty do:
     split rightmost element in buffer
         using char-split
     if split is correct:
      push left element back to buffer

      # processingModule:
      while not done do:
       process rightmost element of
           split in buffer
      if done:
       push processed element to output

 prepare output format
 output
end procedure
```

3.2 Verification of the split

As valid splits may be associated with negative score values (see Example 10), polarity alone is insufficient as a determinative diagnostic feature.

(10) *[(-0.6059544658493871, 'Warmwasseraufbereitungsanlagen', 'Rohr')]*
(Engl.: 'Pipe in the facility for the purification of warm water.')

As previously mentioned however, in the event of an erroneous split of a non-compound NoCs can leverage the patterns of incorrect return-values as it returns either an arbitrarily long output list with the invalid split decision or a list of only one score. In the first case, the next split of the (already invalid) split non-compound's right element will always return a list of one. For Example if the split of *Beamter* in example 7 returns two scores, the next split of its (incorrect) head *-Mter* would only return one score (see Example 11), indicating that no further splitting is possible.

(11) *[(0, 'Mter', 'Mter')]]*

NoCs identifies those single-score outputs or consecutive singleton results as indicative of failed splits and labels the original term as a head, as it can thus assume there are no more correct splits to follow. Additional verification is then conducted by the processing module. It applies prefix/suffix disambiguation to avoid prefix/suffix conflicts with identical sequence slices of words like *-Gel* (Engl.: 'gel') or suffixes like *-haft*, which has the identical surface form as the noun *Haft* (Engl.: 'detention') by using dictionary and suffix/prefix-list look-ups and morphological features as well as POS tagging parsed via *stanza*. For example, it avoids incorrect parses such as *Flü-Gel* (Engl.'wing') or *Anma-Ssung* (Engl. 'presumption') by cross-referencing gender and affixes. In the case of *Flü-Gel*, which is erroneously split into *Flü-* and *-Gel*, the head could theoretically function as the noun *Gel* (Engl.: 'gel'). To verify this, the tool first looks up the morphological features like grammatical gender of the input token, which is masculine for *Flü-Gel*. As the split would only be correct if the grammatical gender of the input token was neutral corresponding to the grammatical gender of its head *Gel* (Engl.: 'gel') it then decides that this can not be a correct split and returns the initial token as head. In the case of *Anma-Ssung* it detects the erroneous split by a look-up from the suffix list for the suffix *-Ung* in combination with a look-up from the dictionary

NoCs_2	CS_2	CS_lemma_2	MS_2	MS_lemma_2	NoCs_3	MS_3	MS_lemma_3
2751	2801	3579	2402	3092	1871	756	1113
0.55	0.56	**0.72**	0.48	**0.62**	**0.37**	0.15	0.22

Table 3: Absolute count and percentages of correct splits from the two and three constituent datasets in the lower-case test.

NoCs_4	MS_4	NoCs_5	MS_5	NoCs_noun	CS_noun	NoCs_adj	CS_adj
268	17	8	0	4099	935	537	187
0.28	0.018	0.33	0.0	**0.82**	0.19	**0.88**	0.31

Table 4: Absolute count and percentages of correct splits from the four and five constituent and non-compound datasets in the lower-case test.

list for its constituents to verify that this split is incorrect.

After deciding not to split any further it also runs a dictionary check on the current token, to check if the current non-compound is a valid word, before labeling it and pushing it to output. For example if it gets *(-1.3002238718039354, 'Bea', 'Mter')* and the next split of *-Mter* returns a score of 0 like in Example 11, it applies a dictionary look-up to verify the existence of the initial token *Beamter*. Thus, it collects all constituents and returns a Python dictionary structure like in Example 12, where it collects all constituents in the first index of the tuple, labels them with either 'head' or 'modifier' and also saves the respective modifier in the third index.

(12) {*'Testbeispiel': [('Test', 'modifier', '-'), ('Beispiel', 'head', 'Test')]*}
(Engl.: 'Test example.')

In a last step before returning the output, NoCs performs lemmatization and removes linking morphemes such as '-s-', resulting in linguistically plausible compound constituents.

4 Test-data and analysis

The current release of the GermaNet compound collection (Seminar für Sprachwissenschaft, University of Tübingen, 2024) was selected as a basis for the creation of test data. As NoCs primarily targets the detection of non-compounds and recursive split of multi-constituent compounds rather than compound head detection, the data set from GermaNet had to be slightly adapted to the task before applying to the two tools. The original dataset is comprised of three columns containing the compounds of lengths of two to six constituents and the respective modifiers and heads, see Example 13.

(13) *Abendbrot Abend Brot*
Abendbrottisch Abendbrot Tisch
(Engl.: 'Dinner' and 'Dinner table')

As this task needed not only the compound head in the case of multi-constituent compounds but the maximally split version, the respective modifiers were automatically searched and collected from the dataset until all heads were found, forming the individual constituents of the original compound. Thus all compounds were collected and categorized by number of constituents. From all the individual compound heads two datasets of nominal and adjective non-compounds were extracted by running automatic stanza parses on the non-compounds to determine nouns and adjectives. Where two possible constituents were listed, the nominal constituent was chosen, as they match the output of the tools more closely. Compounds containing numbers and hyphens were excluded.

From the two and three constituent compound datasets as well as the nominal non-compounds a random sample of five thousand was drawn. The four constituent compound dataset contains 965 compounds, the five constituent compound dataset contains 24 compounds and the adjective non-compound dataset contains 610 non-compounds (see Table 5). As there was only one single six constituent compound and it contained hyphens it was excluded from the evaluation.

Both tools processed all of the datasets. For *compound-split* the *maximal_split* function was used on the two to five constituent datasets. The *char_split* function was only tested on the two constituent dataset and the two non-compound datasets, as it only splits the head. The outputs of the *maximal_split* function on the two and three constituent datasets were lemmatized using stanza parsing at the constituent level, in addition to being retained

Dataset Type	Description	Sample Size
2-constituents	Random sample	5,000
3-constituents	Random sample	5,000
4-constituents	Full dataset	965
5-constituents	Full dataset	24
Nom. noc	Random sample	5,000
Adj. noc	Full dataset	610

Table 5: Dataset sample sizes by compound type.

in their original form, to evaluate whether lemmatization enhances alignment with the GermaNet gold standard, given that NoCs outputs are also lemmatized. To keep computation time minimal this was only conducted on the two and three constituent test-sets, as those offer the full five thousand samples as opposed to the four and five constituent test-sets and were therefore judged more representative for potential effects of the condition. In addition to these tests a second iteration of all the outputs was tested, where all constituents (test data and output) were set to lower case before comparing them, to account for possible spelling divergence when the token would technically be correct (see Table 6 for more details).

To calculate correct splits the outputs for each individual test-set were first compared in length and then tested for string-matches. If all constituents matched exactly the output was considered correct. The percentage of correct splits was then calculated. As all lists were tested separately, this was not considered a real classification task and thus the conventional evaluation metrics of precision and recall were not deemed appropriate for this evaluation.

Calculations and handling of data were carried out with the random and Pandas library (pandas development team, 2020).

5 Results

The comparative evaluation of both tools focused primarily on the handling of non-compounds and the accuracy in splitting multi-constituent compounds. With respect to compound segmentation, NoCs consistently outperforms the base *compound-split* functions across all unlemmatized datasets and under standard evaluation conditions, as shown in Tables 1 and 2, even though *char_split* surpasses *maximal_split* function in the two constituent dataset. A particularly prominent divergence in performance can be observed in the three to five constituent datasets, as the base function hardly captures any splits correctly (see NoCs-values in boldface). However, in the two-constituent dataset, the lemmatized condition significantly boosts performance, clearly surpassing NoCs. This indicates that lemmatization plays a substantial role in improving segmentation accuracy in the output of *compound-split* for simpler compounds.

As illustrated in Tables 2 and 4, the new NoCs demonstrates a clear advantage over *compound-split* in the domain of non-compound handling, as *compound-split* is only able to detect 19 percent of nominal non-compounds and 31 percent of adjective non-compounds. In this regard performance does not profit from the lower-case test, as NoCs still handles 82 percent of nominal and 88 percent of adjective non-compounds. Performance does, however, substantially benefit from the lower-case testing in the case of the two and three constituent datasets across all conditions as presented in table 3, accumulating to 72 percent correctly identified compounds in the lemmatized two constituent lower case condition.

6 Conclusion

Compounding representing an informationally compact and highly efficient linguistic phenomenon for encoding communicated information is particularly interesting within various linguistic fields and frameworks, including computational linguistics. In order to process those complex structures automatically highly specialized tools are necessary, especially in languages like German, where compounding is a highly productive process to (spontaneously) expand the language's vocabulary. Given the virtually unlimited number of potential constituents in German compounds, developing tools capable of reliably decomposing multi-constituent compounds into their component parts is of significant value for downstream applications. As the *compound-split* tool only offers a robust compound head detection and a considerable less robust maximal split approach, expanding the functionality by non-compound detection and a more stable multi-constituent compound split was the aim of the newly introduced NoCs tool.

As presented in section 5, the tool particularly excels in the domain of non-compound detection. While NoCs outperforms the base splitter on the

Dataset Type	NoCs	CharSplit	MaxSplit	CS_lemma	MS_lemma	regular	lower
2-constituents	x	x	x	x	x	x	x
3-constituents	x	-	x	-	x	x	x
4-constituents	x	-	x	-	-	x	x
5-constituents	x	-	x	-	-	x	x
Nom. noc	x	x	-	-	-	x	x
Adj. noc	x	x	-	-	-	x	x

Table 6: Testconditions across tools and datasets.

multi-constituent datasets, these improvements remain moderate, with approximately one-third of correctly split compounds in the three to five constituent datasets. A first preliminary and non-conclusive evaluation of the output data suggests that two key challenges persist: (1) as NoCs builds upon *compound-split*, it inherits certain limitations in the splitting of longer compounds, particularly due to the constraints of the original head detection mechanism, which performs more robustly in conditions with shorter compounds; and (2) longer compounds inherently increase the risk of incorrect splits, and the strict evaluation criterion of testing only perfect string matches on all constituents likely results in the reported performance as a conservative estimate and would probably benefit from a more fine-grained analysis including partial correctness.

Furthermore, both the lemmatization of output constituents and the bulk of the decision-making processes in NoCs rely heavily on stanza parses, increasing the risk to propagate early parsing errors through the entire system. Incorporating a more reliable lemmatizer might improve performance further. To improve overall performance on the actual compound splits, the tool might also greatly benefit from a more flexible handling of potential constituents within the split decision process. Though NoCs still leaves plenty of room for improvement on the split of multi-constituent compounds it provides a promising and practical solution for non-compound detection and contributes valuable functionality to the repertoire of NLP tools available for compound processing.

Limitations

Even though this new extension of the *compound-split* splitter addresses some of the shortcomings of the base splitter and expands the repertoire of it by several functionalities, it still exhibits various limitations and leaves room for improvements. First, it still is not able to confidently disambiguate two theoretically correct but possibly context-inappropriate splits due to its context-independent design. Integrating a language model with contextual understanding might contribute to solving this problem. Second, as NoCs relies on a dictionary for the decision on valid words, it also struggles with abbreviations contained in the dictionary, which might collide with non-abbreviations as sequence slices of words, causing the tool to falsely split. In this regard NoCs might benefit from a separate dictionary of abbreviations in the future.

Additionally, the aforementioned long compounds of five or more constituents still challenge the tool as well as the dependency on a lemmatizer, as this dependency introduces an increased likelihood of cascading errors through the process. A test employing edit distance metrics could allow insights on how many incorrect splits could be captured by a more precise lemmatizer. This online parsing during the processing of compounds also increases the runtime of the tool, rendering it more suitable for applications on smaller datasets. Furthermore, as not all compounds in the original GermaNet dataset were maximally split, this possibly caused some splits to be considered incorrect. A manually curated test set might alleviate this problem. In regard of the test data it also needs to be considered, that the GermaNet data is not 'new' in the context of the base tool, as it was used to test compound head detection accuracy for *compound-split*. To authentically simulate out-of-vocabulary testing a new test set would be desirable. To test on unseen ad hoc compounds, synthetic data could be generated and used for testing of new compounds of variable length in the future. For now, those limitations suggest a combination of both tools across the different conditions for the time being as an improved iteration for the task compared to the base functions of the *compound-split* splitter on its own.

Acknowledgments

I am grateful to the anonymous reviewers for their helpful comments and Stefanie Dipper and Ronja Laarmann-Quante from Ruhr-University Bochum for their valuable feedback. This research is funded by *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102.

References

Sally Andrews, Brett Miller, and Keith Rayner and. 2004. Eye movements and morphological segmentation of compound words: There is a mouse in mousetrap. *European Journal of Cognitive Psychology*, 16(1-2):285–311.

Shaina Benjamin and Daniel Schmidtke. 2023. Conceptual combination during novel and existing compound word reading in context: A self-paced reading study. *Mem. Cognit.*, 51(5):1170–1197.

Douglas Biber and Bethany Gray. 2010. Challenging stereotypes about academic writing: Complexity, elaboration, explicitness. *Journal of English for Academic Purposes*, 9(1):2–20.

John Gamboa, Kristina Braun, Juhani Järvikivi, and Shanley E. M. Allen. 2025. The distributional properties of long nominal compounds in scientific articles: an investigation based on the uniform information density hypothesis. *Corpus Linguistics and Linguistic Theory*, 21(1):137–171.

John C. B. Gamboa, Leigh B. Fernandez, and Shanley E. M. Allen. 2024. Investigating the uniform information density hypothesis with complex nominal compounds. *Applied Psycholinguistics*, 45(2):322–367.

Fritz Günther, Marco Marelli, and Jens Bölte. 2020. Semantic transparency effects in german compounds: A large dataset and multiple-task investigation. *Behavior Research Methods*, 52(3):1208–1224.

M. A. K. Halliday. 1988/2004. On the language of physical science. In Jonathan J. Webster, editor, *The Collected Works of M. A. K. Halliday (Vol. 5)*, pages 140–158. Continuum, London and New York.

Anna Hätty and Sabine Schulte im Walde. 2018. Fine-grained termhood prediction for German compound terms using neural networks. In *Proceedings of the Joint Workshop on Linguistic Annotation, Multiword Expressions and Constructions (LAW-MWE-CxG-2018)*, pages 62–73, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Jukka Hyönä, Alexander Pollatsek, Minna Koski, and Henri Olkoniemi. 2020. An eye-tracking study of reading long and short novel and lexicalized compound words. *Journal of Eye Movement Research*, 13(4).

Anna Hätty, Ulrich Heid, Anna Moskvina, Julia Bettinger, Michael Dorna, and Sabine Schulte im Walde. 2019. Akkubohrhammer vs. akkubohrhammer: Experiments towards the evaluation of compound splitting tools for general language and specific domains. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*, pages 59–67, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Chris Jenkins, Filip Miletic, and Sabine Schulte im Walde. 2023. To split or not to split: Composing compounds in contextual vector spaces. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16131–16136, Singapore. Association for Computational Linguistics.

Panayiota Kendeou and Paul van den Broek. 2007. The effects of prior knowledge and text structure on comprehension processes during reading of scientific texts. *Memory & Cognition*, 35(7):1567–1577.

Irina Krotova, Sergey Aksenov, and Ekaterina Artemova. 2020. A joint approach to compound splitting and idiomatic compound detection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4410–4417, Marseille, France. European Language Resources Association.

Kristina Kösling and Ingo Plag. 2009. Does branching direction determine prominence assignment? an empirical investigation of triconstituent compounds in english. *Corpus Linguistics and Linguistic Theory*, 5(2):201–239.

Julia A. Meßmer, Regine Bader, and Axel Mecklinger. 2021. The more you know: Schema-congruency supports associative encoding of novel compound words. evidence from event-related potentials. *Brain and Cognition*, 155:105813.

Mark Ormerod, Jesús Martínez del Rincón, and Barry Devereux. 2024. How is a "kitchen chair" like a "farm horse"? exploring the representation of noun-noun compound semantics in transformer-based language models. *Computational Linguistics*, 50(1):49–81.

The pandas development team. 2020. pandas-dev/pandas: Pandas.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Arndt Riester and Stefan Baumann. 2017. *The RefLex Scheme — Annotation Guidelines*, volume 14 of *SinSpeC — Working Papers of the SFB 732 "Incremental Specification in Context"*. OPUS, Stuttgart.

Schreiber. 2021. Free German Dictionary. https://sourceforge.net/projects/germandict/files/. Accessed 04-07-2025.

Seminar für Sprachwissenschaft, University of Tübingen. 2024. GermaNet v19.0. https://uni-tuebingen.de/en/faculties/faculty-of-humanities/departments/modern-languages/department-of-linguistics/chairs/general-and-computational-linguistics/ressources/lexica/germanet/description/compounds/#c1081929. Accessed 04-07-2025.

Emil Svoboda and Magda Sevcikova. 2024. PaReNT (parent retrieval neural tool): A deep dive into word formation across languages. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12611–12621, Torino, Italia. ELRA and ICCL.

Don Tuggener. 2016. *Incremental Coreference Resolution for German*. Phd thesis, University of Zürich, Zürich, Switzerland.

Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Marion Weller and Ulrich Heid. 2012. Analyzing and aligning German compound nouns. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2395–2400, Istanbul, Turkey. European Language Resources Association (ELRA).