Evaluating Neural Sequence Models for Splitting (Swiss) German Compounds

Don Tuggener
Zurich University of Applied Sciences
don.tuggener@zhaw.ch

Abstract

This paper evaluates unsupervised and supervised neural sequence models for the task of splitting (Swiss) German compound words. The models are compared to a state-of-the-art approach based on character ngrams and a simple heuristic that accesses a dictionary. We find that the neural models do not outperform the baselines on the German data, but excel when applied to out-of-domain data, i.e. splitting Swiss German compounds. We release our code and data[1], namely the first annotated data set of Swiss German compounds.

1 Introduction

Splitting compound words is an important task when setting up pipelines for Natural Language Processing of the (Swiss[2]) German language. (Swiss) German features a long tail regarding word frequencies due to the phenomenon that compound words are not orthographically separated by whitespaces as in e.g. English (e.g. *Autobahnraststätte* vs. *highway service area*). Thus, when mapping words to lexical resources such as word nets or embeddings, it is likely that there are compounds which are not represented in the resource.

Neural sequence models capture characteristics of word or character sequences (ngrams) in a latent representation and are thus hypothetically well-suited for compound splitting. In this paper, we evaluate several neural sequence models on the task of (Swiss) German compound splitting. The models are compared to an unsupervised character ngram-based approach and a baseline that uses a dictionary. Furthermore, we present the first gold standard for splitting Swiss German compounds and evaluate the models on this newly available resource. Swiss German dialects feature no official spelling, and they are closely related to Standard German, but differ regarding some language changes throughout history, and we use the Swiss German compounds as a sort of out-of-domain test set for the models.

1.1 Related Work

Several methods for automatic splitting of word compounds exist. One approach is to use a dictionary to perform a full morphological analysis (Schmid et al., 2004). Others apply corpora statistics (Koehn and Knight, 2003) and combine them with linguistic heuristics (Weller-Di Marco, 2017). There are both supervised (Alfonseca et al., 2008; Henrich and Hinrichs, 2011) and unsupervised approaches (Macherey et al., 2011; Tuggener, 2016; Ziering and van der Plas, 2016) to the task. Riedl and Biemann (2016); Ziering et al. (2016) explore distributional semantics on the premise that the constituents of a compound are semantically similar to the compound. Other work researches the semantic relation between the constituents of the compounds (Schulte im Walde et al., 2016). While most approaches focus on compounds in a single language, there exists work that explores the task in a multilingual context (Alfonseca et al., 2008; Macherey et al., 2011).

While there exists work on morphological segmentation (Kann et al., 2016), to the best of our knowledge, there is no prior work on using neural sequence models to identify head constituents of compounds.

In: Mark Cieliebak, Don Tuggener and Fernando Benites (eds.): Proceedings of the 3rd Swiss Text Analytics Conference (SwissText 2018), Winterthur, Switzerland, June 2018

[1]https://github.engineering.zhaw.ch/tuge/neural_compound_splitter

[2]Swiss German subsumes the Alemannic dialects spoken in Switzerland.

2 Neural sequence models for compound splitting

We first introduce the neural sequence models that we apply in the experiments and how we adapt them to fit the task of (Swiss) German compound splitting.

2.1 Unsupervised Recurrent Neural Network

The first model we explore is a character-based language model using a recurrent neural network (RNN). Language models aim to predict the next token in a sequence given the sequence history. Different from ngram-based language models (Heafield et al., 2013, e.g.), RNN-based models feature a hidden state that is updated after consuming a token (a character or a word). Based on the hidden state, a probability distribution over the token vocabulary is calculated using the softmax function, and the most likely token is selected when generating sequences. During training, the difference between the predicted probability and the given next token constitutes the loss that is backpropagated to update the model parameters (i.e. weights). Such RNN-based language models have been shown to outperform ngram-based models in terms of perplexity on general language modelling tasks (Mikolov et al., 2010, inter alia).

To employ RNNs for unsupervised compound splitting, we exploit an implementation detail that is commonly used when working with such approaches: Before training an RNN on a given token sequence, a special END token is appended to the sequence. This token is inserted into the vocabulary of the model. When using the trained RNN to generate a sequence, the END token is used as a stopping criteria, i.e. if the END token is the most likely next token, generation is terminated and the sequence considered complete.

We adapt this idea and monitor the probability of emitting the END token when consuming a compound word character-wise. That is, in a character sequence $\mathbf{x}$, we consider the position x_i with the highest probability of emitting the END token as the split position:

$$\operatorname*{argmax}_i p(\text{END}|x_0 \dots x_i) \tag{1}$$

Additionally, we are interested in positions in the sequence where the probability of generating the next given character is low, based on the hypothesis that that such positions are indicators for a suitable split position:

$$\operatorname*{argmin}_i p(x_{i+1}|x_0 \dots x_i) \tag{2}$$

To combine the two features and determine the best split position, we sum the END token probability and the inverse of the probability of the next character at each position in the sequence $\mathbf{x}$ of length n and take the position with the highest score:

$$\operatorname*{argmax}_i \big(p(\text{END}|x_i) + (1 - p(x_{i+1}|x_0 \dots x_i))\big) \tag{3}$$

During initial experiments, we found that this approach did indeed yield correct boundaries of free morphems in German compounds, but struggled to find the correct boundary for splitting when compounds consist of more then two such free morphems. For example, for the compound *Autobahnraststätte (highway service area)* with four free morphems (*Auto, Bahn, Rast, Stätte*), the approach identified *Autobahnrast* as the body and *Stätte* as the head, instead of *Autobahn* and *Raststätte*. We hypothesized that one flaw of the approach is that when determining the best split position i in a sequence x, only the character sequence up to position i, i.e. $x_0 \dots x_i$, is considered, and the remaining string, $x_{i+1} \dots x_n$, is neglected. Therefore, we introduced a backward-looking RNN that consumes the sequence $\mathbf{x}$ in reverse order, which constitutes a bi-directional RNN (biRNN). We calculate the same two features as for the forward-looking RNN for the backward-looking RNN and sum the scores of the forward and backward-looking RNN for each position i in the sequence to determine the best position for a split.

Furthermore, we noted that the approach fails at placing boundaries correctly if the compounds contain so called *Fugen* elements (linking elements), as in e.g. *Installation-s-Anweisung (installation instruction)*, where a *Fugen-S* glues together the free morphemes *Installation* and *Anweisung*. The approach places the split after the first free morpheme (*Installation*), and thus attaches the Fugen-S to the head (*sanweisung*), which yields an incorrect split in the result. Therefore, we applied a regular expression (capturing bound morphemes that often occur before compound boundaries, e.g. *-ions, -täts, -keits*) that aims to remove Fugen-S heuristically before identifying a splitting position (e.g. *Installationsanweisung* $\rightarrow$ *Installationanweisung*).

For this approach, we only need a collection of German words to train the character-based RNNs, as it is unsupervised regarding the compound splitting task.

2.2 Supervised Recurrent Neural Network

A natural extension to the neural character-based language model is to add supervision. To determine the split in the unsupervised model, we summed probabilities that we deemed relevant but did not train the RNN itself on the task of compound splitting. In the supervised approach, we use the hidden states of the trained character-based language model biRNNs when consuming a German compound word character-wise as features to train a binary classifier regarding the splitting decision. That is, at each position in the sequence, we concatenate the hidden states of the forward and backward RNN to create a feature vector. This vector is then fed to a fully connected layer, as shown in Figure 1. For determining the split, we take the position in the sequence that has the highest probability according to the binary classifier. During training, the split position is known and used to calculate and backpropagate the loss.

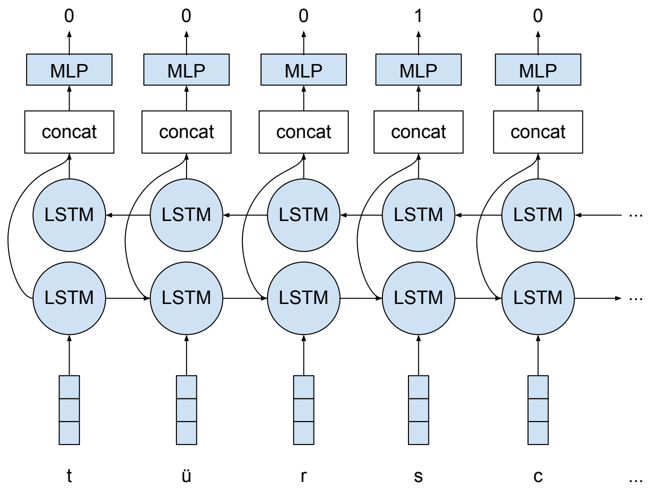

Figure 1: Supervised RNN-MLP architecture, shown for the compound *Türschloss (door lock)*, where the split position is at the character *s*.

2.3 Sequence-to-sequence with attention

An alternative supervised neural model is the Sequence-to-sequence model (Sutskever et al., 2014). The model is applied to transform sequences into other sequences in e.g. Machine Translation (Cho et al., 2014). It consists of an encoder component that encodes the input sequence into a latent representation, and a decoder that generates the desired output sequence based on the encoded input.

The initial version of the seq2seq model encodes the whole input sequence into one vector (i.e. the last hidden state of the encoder) on which the generation of the output is based. To address this limitation, Bahdanau et al. (2014) introduced an attention mechanism that lets the decoder peak at all hidden states in the input and apply a weight to each which represents its respective importance while generating the correct output token at each step during generation.

We apply this model to the compound splitting task by using the compounds as input and the head noun as the desired output.[3] We hypothesize that the attention mechanism is helpful in our case, because not all characters in the input sequence are equally important when identifying the split boundary. The attention mechanism enables the decoder to focus on different character groups, which, ideally, represent (groups of) relevant free morphemes. We thus apply the seq2seq with attention model.

3 Experiments

Having outlined our models, we now describe our data, the baselines, followed by evaluation results.

3.1 Data

We use the dataset discussed in Henrich and Hinrichs (2011), i.e. a list of 75 000 German compounds and their head nouns extracted form GermaNet (Henrich and Hinrichs, 2010), a German wordnet.[4] Henrich and Hinrichs (2011) used this resource to evaluate several approaches to compound splitting. They also included non-compounds in their evaluation, however, these are not provided in the resource. Therefore, we only compare the ability of our approaches to determine the correct split positions in known compounds (corresponding to the task in section 7.2 in Henrich

[3]We also experimented with generating both the body and the head constituents, but obtained slightly better results with generating the head only.

[4]`http://www.sfs.uni-tuebingen.de/lsd/compounds.shtml`, we use version v12.0 (2017)

and Hinrichs (2011)).[5] Furthermore, we remove compounds from the list that contain a hyphen, since splitting them at the hyphen is straight-forward. We randomly split the remaining 73 1333 compounds into 80% train and 20% test data.

Additionally, we created the first gold standard of Swiss German compounds and their head nouns. We extracted the 150 longest Swiss German words in the SB-CH corpus (Grubenmann et al., 2018) which consists of Swiss German texts gathered from different sources (e.g. social media messages, business reports). We then manually annotated their (recursive) head nouns. For example, for the compound *Wältuntergangskatastrophefilm (apocalypse catastrophe movie)*, we extract the heads *Katastrophefilm* and *Film* and consider both as a correct head in evaluation. We use this data as a kind of out-of-domain test set in the evaluation.

3.2 Baselines

We compare the neural models to two baselines:

Dictionary-based: The first baseline uses a dictionary and matches its words to the end of the compounds in the test set. If a word from the dictionary is found to be a substring at the end of a compound in the test set, it is taken as the head noun of that compound. Clearly, the order in which the dictionary is traversed matters, because the method stops after finding the first match. We experimented with sorting the dictionary by longest to shortest words and vice versa. Also, we included a subroutine to check if the found body (the remaining word after removing the head noun from the compound; potentially removing the Fugen-S) is also in the dictionary and favored those splits which have both body and head in the dictionary. The algorithm is outlined in Algorithm 1.

CharSplit: The dictionary-based method is prone to fail where a head noun of a compound is never seen in the training corpus. CharSplit, proposed in Tuggener (2016), alleviates this by basing the splits on character ngrams rather than words. CharSplit

[5]Clearly, an end-to-end system for compound splitting needs to be able to identify whether a given word constitutes a compound. Unfortunately, we are not able to evaluate our approaches in this regard here. Another resource containing non-compounds is discussed in Escartín (2014), but it does not seem to be available.

Algorithm 1 Dictionary-based compound splitting

```
1: Create dictionary $D$ from train set
2: Sort $D$ based on word length
3: procedure SPLIT(compound)
4:     for $word \in D$ do
5:         if $compound$ ends with $word$ then
6:             $head = word$
7:             if $body \in D$ then
8:                 break
```

calculates probabilities of ngrams (length 2 to 20) to occur at the beginning, middle, and end of words in an unlabeled corpus and calculates a splitting score at each character position in a (compound) word. Tuggener (2016) found that this method outperforms several other splitters in the task of identifying correct splitting boundaries on the GermaNet data, achieving 95% accuracy.

3.3 Evaluation

Next, we evaluate the models regarding their ability to identify correct splitting boundaries on the German and Swiss German data. Since we only evaluate on known compounds, we measure accuracy, i.e. the percentage of compounds for which the methods find the correct split. Results are given in Table 1.

The first striking observation is that the dictionary-based method outperforms all other approaches on the GermaNet data. The reason for the high accuracy lies in the overlap of the words in the train and test set. While there exist no direct duplicates in the sets, almost all head nouns (97%) that need to be identified in the test set are included in the train set as constituents of a compound. For example, the test set contains the instance *Fruchtnektar (fruit nectar)* $\rightarrow$ *Frucht* $\rightarrow$ *Nektar* and the train set contains *Bananennektar (banana nectar)* $\rightarrow$ *Banane* $\rightarrow$ *Nektar*. As we see from the evaluation on the SB-CH data, the approach clearly fails when this overlap diminishes.

The CharSplit baseline performs 2 accuracy points below the dictionary method on the GermaNet data, but achieves better results on the Swiss German compounds. Relying on ngram representations of the German training data leads to an advantage when moving from the German training data to the related, but not identical domain of Swiss German compounds.

For the unsupervised biLSTM, we found that training for more than 1 epoch did not improve results.

Model	Parameters	Acc.
Dictionary	check long words first	94.07
Dictionary	check short words first	58.35
Dictionary	+favor known words as body	**95.18**
CharSplit	include Fugen-S	90.65
CharSplit	remove Fugen-S	**93.26**
Unsup. biRNN	1 layer, 32 hidden size, 1 epoch	67.34
Unsup. biLSTM	1 layer, 32 hidden size, 1 epoch	78.13
Unsup. biLSTM	1 layer, 32 hidden size, 1 epoch, keep Fugen-S	71.51
Unsup. biLSTM	2 layer, 512 hidden size, 1 epoch	**87.04**
Unsup. biLSTM+MLP	MLP: 3 layers, 128-64-16 hidden size, 2 epochs	**95.10**
Seq2seq+attention	1 layer, 128 hidden size, 3 epochs	**92.40**

Table 1: Models, parameters, and splitting accuracy for different approaches. Trained and evaluated on German compounds (GermaNet). Best results per category are in bold, best overall underlined.

Model	Acc.
Dictionary	20.67
CharSplit	36.67
Unsup. biLSTM	57.33
Unsup. biLSTM+MLP	**68.67**
Seq2seq+attention	52.00

Table 2: Splitting accuracy for different approaches evaluated on Swiss German compounds (SB-CH). Using best performing models trained on GermaNet.

However, increasing the model size in terms of layers and hidden state size benefited accuracy to a certain extent, and removing the Fugen-S is vital. Also, we found that a vanilla biRNN fares poorly compared to using a biLSTM. The results show that for the German compounds, the model falls behind the baselines by a considerable margin. However, on the Swiss German compounds, it leads to a large improvement compared to the baselines (+20 accuracy points).

For the combination of the unsupervised biLSTM coupled with the supervised MLP, we took the best performing biLSTM model (2 layers, 512 hidden size) to create the inputs for the MLP. We experimented with different numbers of layers, hidden state sizes, and epochs for the MLP and report results of the best performing configuration. On the German data, this approach is on par with the dictionary baseline. It also improves performance on the Swiss German compound by +11 accuracy points compared to only using the unsupervised biLSTM, similar to the gains on the German data.

The seq2seq model was trained independently from the other models. It outperforms the unsupervised biLSTM on the German data, but not on the out-of-domain Swiss German test set. It falls behind the other supervised approach on both test sets, but features some other interesting properties discussed in the next section.

3.4 Output analysis

In this section, we qualitatively compare the outputs and other properties of the different approaches and discuss (dis)advantages of each.

In general, we hardly ever encountered system outputs that put splits at seemingly random positions within the compounds. The two main errors we observed are related to the Fugen-S and errors in choosing the correct split position for compounds consisting of more than two free morphemes.

Dictionary-lookup: As this method relies on a predefined list of know words, it is only able split compounds that have a head noun which is contained in the dictionary. Hence, a main error cause are unknown head nouns, which especially affects performance on the Swiss German data. The method is robust against Fugen-S, since it looks for known words at the compound end and hence never attaches a Fugen-S to a found head noun. However, it is the only approach that does not provide a way to distinguish compounds from non-compounds (i.e. it provides no measure for the confidence of a found split). That is, it does not provide any means to detect non-compounds. The word *Fahrzeug (vehicle)* e.g. includes the head noun *Zeug (thing*, but it is not a hypernym of the word *Fahrzeug* and hence cannot be split without straying away from its meaning, and the dictionary approach would perform this split. Since we only evaluate the splitting methods on known compounds, this disadvantage does not affect the accuracy of the approach in evaluation. In conclusion, this methods works well when train and test data feature a largely overlapping vocabulary and the approach is coupled with a method to distinguish compounds from non-compounds.

CharSplit: This approach often fails when encountering the Fugen-S, i.e. it frequently attaches it to the head noun if the regular expression is not able to remove it before the split. However, this model improves performance on out-of-domain data compared to the dictionary approach, as it relies on an ngram representation of the training data. This enables it to better handle vocabulary differences between training and testing domain compared to the dictionary baseline. As an unsupervised approach, it only relies on an unannotated corpus to calculate the ngram probability distributions.

Unsupervised biLSTM: Similar to CharSplit, the approach also often fails attaching Fugen-S to the body instead of to the head if removing the Fugen-S using the regular expression fails. In that regard, the unsupervised biLSTM output is similar to CharSplit. However, the latent representation of the character sequences (compared to the ngram representation in CharSplit that directly relies on the surface forms) allows it to better generalize to the out-of-domain data than CharSplit, yielding better splitting accuracy.

Unsupervised biLSTM + MLP: The combination of the unsupervised biLSTM and the supervised MLP yieled best overall results in our experiments. As one of the supervised approaches, it does not seem to struggle with occurrences of the Fugen-S and most errors stem from splitting compounds with multiple free morphemes at the wrong free morpheme boundary.

seq2seq: As the second supervised model, this model also does not struggle with removing Fugen elements. The main error cause is thus splitting compounds with multiple free morphems at the incorrect free morpheme boundary, e.g. for the compound *Parkleitsystem (parking guiding system)*, it generates the head *System* instead of *Leitsystem*.
A unique error source for this model is that it often generates spelling errors in the (otherwise) correctly identified heads, e.g. for *Neukonstruktion (new construction)*, it produces *konstr**o**ktion* as the head. Clearly, generating the head noun instead of just finding its starting character seems to overcomplicate the task in our setting and has an unnecessary impact on performance.
When moving to the out-of-domain data, the model shows a bigger performance impact than the biLSTM coupled with the MLP. One possible reason could be that the model does more heavily rely on full words during testing, i.e. the input representation is constructed over the full compound and the attention mechanism does not focus strongly enough on relevant character sequences. A seq2seq model that replaces the input representation with convolution operations is presented in Gehring et al. (2017). An interesting experiment would be to evaluate if character convolutions are the better option for representing important character sequences than the biLSTM.
Finally, we noted that when training the model to generate both the body and head constituents given the compound, it often produces the correct lemmatization of the body by removing Fugen elements from it (*-es, -en, -s, -n*, e.g. *bundesforschungsminister* $\rightarrow$ *bund, forschungsminister* or *landesverwaltung* $\rightarrow$ *land, verwaltung*). It thus seems to be the appropriate candidate for a neural model if identifying the lemmatized body of a compound is a goal of splitting compounds.

Output combination: To gain insight on how complementary the outputs of the different models are, we calculated the percentage of compounds that are split identically by all models, which is 76.63%. This suggests that there is a substantial difference in the outputs.
We also calculated the upper bound splitting performance for ensembling the models. To do so, we regarded a compound as split correctly if at least one of the outputs contained the correct split. This upper bound achieves an accuracy of 99.56% on the GermaNet data, which indicates that the model

outputs are sufficiently different to be combined in an ensemble system.

4 Conclusion

We evaluated three common neural sequence models for the task of splitting (Swiss) German compounds and compared them to ngram- and dictionary-based baselines. We found that for in-domain data (German compounds), the neural sequence models were not able to outperform the baselines, but that for out-of-domain data (Swiss German), they achieved vastly better accuracy in identifying splitting positions. We hypothesize that the latent representations in the neural models of character sequences (in the form of vectors) allows them to process similar character sequences in a similar way. Thus, when applying models trained on German data to Swiss German data, the models are able to resolve the differences between the languages because slightly modified but similar and corresponding ngram sequences lead to similar hidden representations, which in turn yield similar outputs (i.e. splitting positions), where ngram- or dictionary approaches, which directly rely on surface forms, fail. Future work in the direction of Alfonseca et al. (2008) will have to determine if the approaches based on neural sequence models is applicable to other language groups with similar spelling.

References

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008. Decompounding query keywords from compounding languages. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*. Association for Computational Linguistics, pages 253–256.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Kyunghyun Cho, Bart van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1724–1734.

Carla Parra Escartín. 2014. Chasing the perfect splitter: A comparison of different compound splitting tools. In *LREC*. pages 3340–3347.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *ArXiv e-prints* .

Ralf Grubenmann, Don Tuggener, Pius von Dniken, Jan Deriu, and Mark Cieliebak. 2018. SB-CH: A Swiss German corpus with sentiment annotations. In *Proceedings of the 11th Language Resources and Evaluation Conference (LREC), 2018*. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 690–696.

Verena Henrich and Erhard Hinrichs. 2010. GernEdiT-the GermaNet editing tool. In *Proceedings of the ACL 2010 System Demonstrations*. pages 19–24.

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.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural morphological analysis: Encoding-decoding canonical segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. pages 961–967.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, pages 187–193.

Klaus Macherey, Andrew M Dai, David Talbot, Ashok C Popat, and Franz Och. 2011. Language-independent compound splitting with morphological operations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 1395–1404.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Martin Riedl and Chris Biemann. 2016. Unsupervised compound splitting with distributional semantics rivals supervised methods. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 617–622.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. 2004. SMOR: A German computational morphology covering derivation, composition and inflection. In *LREC*. Lisbon, pages 1–263.

Sabine Schulte im Walde, Anna Hätty, and Stefan Bott. 2016. The role of modifier and head properties in predicting the compositionality of english and german noun-noun compounds: A vector-space perspective. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*. pages 148–158.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Don Tuggener. 2016. *Incremental Coreference Resolution for German*. Ph.D. thesis, University of Zurich.

Marion Weller-Di Marco. 2017. Simple compound splitting for German. In *Proceedings of the 13th Workshop on Multiword Expressions (MWE 2017)*. pages 161–166.

Patrick Ziering, Stefan Müller, and Lonneke van der Plas. 2016. Top a splitter: Using distributional semantics for improving compound splitting. In *Proceedings of the 12th Workshop on Multiword Expressions*. pages 50–55.

Patrick Ziering and Lonneke van der Plas. 2016. Towards unsupervised and language-independent compound splitting using inflectional morphological transformations. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 644–653.