# Automatic Dictionary Generation: Could Brothers Grimm Create a Dictionary with BERT?

**Hendryk T. Weiland**
Heinrich Heine University Düsseldorf
hendryk.weiland@hhu.de

**Maike Behrendt**
Heinrich Heine University Düsseldorf
maike.behrendt@hhu.de

**Stefan Harmeling**
TU Dortmund University
stefan.harmeling@tu-dortmund.de

## Abstract

The creation of the most famous German dictionary, also referred to as "Deutsches Wörterbuch" or in English "The German Dictionary", by the two brothers Jacob and Wilhelm Grimm, took more than a lifetime to be finished (1838–1961). In our work we pose the question, if it would be possible for them to create a dictionary using present technology, i.e., language models such as BERT. Starting with the definition of the task of Automatic Dictionary Generation, we propose a method based on contextualized word embeddings and hierarchical clustering to create a dictionary given unannotated text corpora. We justify our design choices by running variants of our method on English texts, where ground truth dictionaries are available. Finally, we apply our approach to Shakespeare's work and automatically generate a dictionary tailored to Shakespearean vocabulary and contexts without human intervention.

## 1 Introduction

In 1838, the brothers Jacob and Wilhelm Grimm started to create the Deutsches Wörterbuch (Grimm and Grimm, 1854), a comprehensive German dictionary with references for each entry. A *dictionary* is a resource that assigns meanings or translations to words. Words are usually displayed in alphabetical order in their canonical form, called *lemma*, and an explanation of the meaning, called a *gloss*. The first volumes of the famous German dictionary were published in 1852. Brothers Grimm could not finish their work within their lifetime, but different scholars and institutions later succeeded in 1961. The creation took 123 years in total. If the brothers Grimm started their project nowadays, they would likely use state-of-the-art technology like the internet and a pretrained language model like the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) to speed up their work. In our work, we want to examine if the automated generation of a dictionary is possible with state-of-the-art technology and if today's language models could do the extensive work.

In Natural Language Processing (NLP), we have to split sentences into smaller units which we refer to as *tokens*. Tokens are not only written words but also punctuation marks and numbers. To automatically build a dictionary from plain reference texts, we need to find all occurrences of each word and distinguish their sense only from their context, which aligns with Wittgenstein's dictum "the meaning of a word is its use in the language" (Wittgenstein, 1953). Given a fixed set of senses, choosing the correct word sense from that set is defined as the task of *word sense disambiguation* (WSD). As the number of senses for each word appearing in a given text is unknown, we need to separate the senses for each word without any prior knowledge, which is called *word sense induction* (WSI). The class of tokens that have the same meaning is referred to as type. Words with only one meaning are called *monosemous*, while ambiguous words are referred to as *polysemous*.

It has been shown that BERT's contextualized word embeddings hold syntactic and semantic knowledge (Rogers et al., 2021). In addition, they form separable clusters for polysemous words (Wiedemann et al., 2019). We want to utilize these characteristics of contextualized vector representations produced by language models such as BERT and perform word sense induction using a hierarchical clustering method to tackle the task of *automatic dictionary generation* (ADG) from raw text without any further annotations.

**Contributions.**

1. We define the task of automatic dictionary generation (ADG).

2. We discuss how to evaluate automatically generated dictionaries.



*Proceedings of the 19th Conference on Natural Language Processing (KONVENS 2023)*, pages 102–120
September 18–22, 2023. 

3. We present a simple approach for ADG using a pretrained CharacterBERT (El Boukkouri et al., 2020) model and agglomerative hierarchical clustering (AHC), and apply it to the work of William Shakespeare to create a Shakespearean dictionary.

The remaining paper is structured as follows: in the upcoming section, we discuss related work. We then define the task of ADG and afterward explain our approach to ADG called Grimm's BERT and discuss how we evaluate the model. Section 4 presents our experiments on the task of ADG. To demonstrate the applicability of our ADG pipeline to an interesting real-world text corpus, we apply it to the works of William Shakespeare in order to create a Shakespearean dictionary in Section 5 before giving a conclusion of our work in Section 6. All of our experiments are available on GitHub under: `https://github.com/Weilando/grimm_bert`.

## 2 Related Work

Among the many possibilities, we choose to use CharacterBERT (El Boukkouri et al., 2020) embeddings to calculate contextualized vector representations of each word in a given text and apply hierarchical clustering to distinguish word types used in a text to create dictionary entries. In the following section we firstly discuss previous approaches to generate dictionaries and secondly look at methods to disambiguate the meaning of words.

**Dictionary Generation.** The generation of lexical resources such as dictionaries has interested researchers for a long time (Chang et al., 1995). Past work on dictionary generation, also referred to as dictionary construction can be divided into two categories. There have been methods to construct (i) bilingual (Kaji et al., 2008) and (ii) monolingual (Tavast et al., 2020) dictionaries which are either of a general nature or focus on domain-specific terms (Ren et al., 2022). Bilingual dictionaries have been either created by translation (Varga and Yokoyama, 2009), through the use of parallel corpora (McEwan et al., 2002) or by combining two existing dictionaries (Kaji et al., 2008). One challenge all these methods face is the ambiguity of words. To solve it, additional knowledge has been necessary in form of thesauri, WordNet (Nicolas et al., 2021) or statistics given raw text in both languages (Kaji et al., 2008).

**Word Sense Induction And Disambiguation.** To tackle the task of WSD, there are knowledge-based approaches that utilize linguistic resources like thesauri and supervised (and semi-supervised) approaches that train a classifier on manually labeled training data and possibly unlabeled corpora in addition (Wiedemann et al., 2019). In contrast, we want to solve the task without the use of any annotations or further knowledge as a sub-task of the automatic dictionary generation. For WSD there already have been approaches based on word embeddings. The context-group-discrimination (Schütze, 1998) algorithm, for example, combines context independent word vectors with context vectors that capture information from second-order co-occurrences and clusters. Wiedemann et al. (Wiedemann et al., 2019) investigate the application of the contextualized word embeddings of Flair (Akbik et al., 2018), ELMo (Peters et al., 2018) and an uncased $\text{BERT}_{\text{Large}}$ (Devlin et al., 2019) for WSD. BERT was the only evaluated contextualized embedding that allowed distinguishable clusters and therefore outperformed its competitors (Wiedemann et al., 2019). For the task of WSI many methods apply clustering of some kind of word representation to discriminate the senses of each word in context. A simple clustering approach is k-means, which usually requires to know the number of clusters beforehand (Giulianelli et al., 2020). Other approaches are affinity propagation (Martinc et al., 2020) and agglomerative clustering (Arefyev et al., 2019). For our ADG pipeline we perform WSI with agglomerative clustering and contextualized CharacterBERT embeddings.

## 3 Automatic Dictionary Generation

Informally, automatic dictionary generation (ADG) is the process of creating a dictionary from raw text, containing a list of senses with reference sentences for each type. While this description appears obvious for common languages like English, there are a couple of choices to make, which we detail next.

More formally, a *text* is given as a sequence of characters, i.e., as a string. The first step is to split the string into a sequence of *tokens*. A trivial choice is to split at whitespace characters. However, many so-called tokenizers split words even further into stem and ending. Punctuation marks and numbers are most often tokens themselves. The second step is to split the sequence of tokens into subsequences called *sentences*. Sentences give context to a token

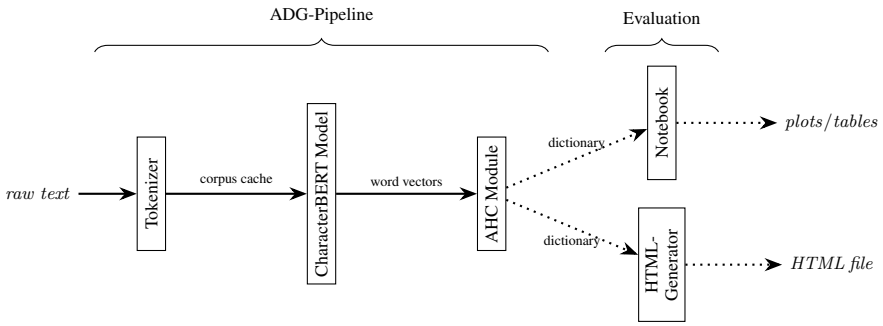



Figure 1: Grimm's BERT: our ADG pipeline implementation. *Solid* and *dotted* arrows indicate *obligatory* and *optional* connections, respectively.

and should be characteristic of the token's sense. Based on these choices, a *dictionary* is a set of one dictionary entry per unique word. Each entry consists of a list of senses, where each sense has a list of reference sentences assigned. Some tokens are possibly excluded from the dictionary, e.g., word endings.

Thus to implement ADG, we have to specify how we define tokens and sentences, since these choices determine what the generated dictionary will contain. For most common languages, the dictionary entries contain an additional human-understandable description called gloss, which we exclude from our pipeline for now. Note that in contrast to WSD, the ADG task does not assume any knowledge about the number of senses a word occurring in a text corpus has. Consequently a step in the ADG pipeline is to perform word sense induction for each word in a text. To solve the task of ADG, we employ contextualized representations of each token that capture semantic and syntactic features. Several studies show that BERT embeddings capture syntactic information (Rogers et al., 2021). Wiedemann et al. (Wiedemann et al., 2019) also compared different contextualized word embeddings for WSD and found that uncased BERT embeddings perform best for this task. Motivated by these results, we use BERT embeddings to tackle the task of ADG.

**Algorithm 1:** ADG Pipeline

1 Tokenize the input.
2 Generate one contextualized word vector per token.
3 Perform token-wise sense induction clustering the contextualized word vectors.

### 3.1 Grimm's BERT for ADG.

Next, we propose a method for the ADG task, which we call Grimm's BERT. Figure 1 shows the complete pipeline of our approach. The general steps that are performed for ADG are also written down as Algorithm 1. To give further explanation we next discuss each step separately:

**1. Tokenization.** $\text{BERT}_{\text{Large}}$ solves the task of WSD better than other contextualized word embeddings (Wiedemann et al., 2019). However, the used WordPiece tokenizer (Wu et al., 2016) cuts words into so-called word pieces. As our dictionary should contain only human-readable words, we decided to use a BERT model that is pretrained using a word level tokenizer. We choose CharacterBERT (El Boukkouri et al., 2020), more precisely $\text{CharacterBERT}_{\text{General}}$ which is a pretrained variant of the uncased $\text{BERT}_{\text{Base}}$ model that uses ELMo's (Peters et al., 2018) word level CharacterCNN module instead of WordPiece embeddings.

**2. Generate Contextualized Word Embeddings.** We calculate one contextualized word vector per token with a pretrained $\text{CharacterBERT}_{\text{General}}$ model, forward the tokenized input and extract the 768 dimensional output from the model's last hidden layer.

**3. Token-wise Sense Induction.** Each occurring word has at least one word sense. We perform agglomerative hierarchical clustering (AHC) to related word vectors to detect and discriminate different word senses for polysemous words. AHC is a bottom-up method that starts with single objects and successively merges the closest objects to build a binary merge tree. A *linkage criterion* determines the relevant distance between clusters for the process of merging. Average linkage clustering uses the average distance between all pairs of objects in

two clusters, complete linkage takes the maximum distances between all objects of two clusters and single linkage uses the minimum distance between all objects of the two sets.

There are several ways to cut the binary merge tree into subtrees to get different clusters. One option is a fixed distance threshold that determines connections to cut and leaves the resulting number of clusters open. Another option is a fixed cluster count that maximizes the linkage criterion but ignores the absolute value of the cut connections. Grimm's BERT builds one dendrogram per unique word using the average linkage criterion with the Euclidean distance as linkage distance. It applies a fixed linkage distance threshold, which is a hyperparameter, to cut the dendrograms into subtrees representing different groups of senses. For our choices of the linkage and cut criterion, we performed extensive experiments presented in the Appendix.

### 3.2 Evaluation.

While WSD is a classification task, ADG is a clustering problem. The following section discusses how to evaluate an automatically created dictionary for the case where we have ground truth information about the number of word senses. We choose the Adjusted Rand Index (ARI) (Hubert and Arabie, 1985) as an objective evaluation metric to quantify the quality of the resulting clusters. Classical metrics for WSD like the accuracy or the $\mathrm{F}_1$ score are not applicable, as forming clusters of senses can rather be seen as a separation of unnamed senses than a selection from a fixed dictionary.

Let $X = \{X_1, \ldots, X_N\}$ be a set of objects, $Y = \{Y_1, \ldots, Y_K\}$ be a partitioning of $X$ in $K \in 1, \ldots, N$ disjoint sets and $m$ denote a clustering method which describes how to obtain $Y$ from a given $X$. Then $(X, Y, m)$ describes a clustering problem, and we call $Y$ a clustering. Rand (1971) defines the Rand Index (RI) via

$$\mathrm{RI}(Y, Y') = \frac{\sum_{i<j}^{n} \gamma_{ij}}{\binom{n}{2}} \in [0, \ldots, 1] \qquad (1)$$

where $Y'$ is another clustering and

$$\gamma_{ij} = \begin{cases} 1 & \text{if there exist } k, k' \text{ s.t. both } X_i, X_j \text{ are in both } Y_k \text{ and } Y'_{k'} \\ 1 & \text{if there exist } k, k' \text{ s.t. } X_i \text{ is in both } Y_k \text{ and } Y'_{k'} \text{ while } X_j \text{ is neither in } Y_k \text{ or } Y'_{k'} \\ 0 & otherwise \end{cases}$$

with $i, j \in \{1, \ldots, N\}$ and $k, k' \in \{1, \ldots, K\}$. Intuitively, $\gamma_{ij}$ is true if two objects are together or separate in both clusterings. $\mathrm{RI}(Y, Y') = 1$ indicates identical clusterings, whereas $\mathrm{RI}(Y, Y') = 0$ for two clusterings without any similarities.

Rand (Rand, 1971) defined the Rand Index (RI) to measure the similarity of two clusterings by calculating the agreement between two different partitions. The index considers every pair of the given data points in the obtained and correct clustering and counts how many pairs are in the same clusters and how many are in different clusters. The ARI (Hubert and Arabie, 1985) is the Rand Index, corrected for chance using the hypergeometric distribution. We obtain the ARI with

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathbb{E}(\mathrm{RI})}{\max(\mathrm{RI}) - \mathbb{E}(\mathrm{RI})} \in [-1, \ldots, 1] \qquad (2)$$

where $\mathbb{E}(\mathrm{RI})$ is the expected value of RI. Please note that $\mathrm{ARI} = 1$ indicates perfectly matched clusterings, $\mathrm{ARI} = 0$ indicates random clusterings regarding the hypergeometric distribution, and $\mathrm{ARI} < 0$ does not have an intuitive interpretation. ARI is symmetric, so $\mathrm{ARI}(Y, Y') = \mathrm{ARI}(Y', Y)$. Only the assignment of objects to the same or different clusters matters, as the score is invariant under the permutation of label names.

As the ARI compares a clustering with some ground truth, we cannot use it to evaluate a dictionary for corpora without any semantic annotations. In that case, we measure the density and separation of clusters using the Silhouette Coefficient (Rousseeuw, 1987) to find a sensible cluster count $k$ for a set of $n$ objects. In our context, objects are tokens and clusters are senses.

The Silhouette Coefficient $s(i)$ for each object $i$ is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \in [-1, 1], \qquad (3)$$

where $a(i)$ is the average distance inside the cluster and $b(i)$ is the average distance to the closest cluster (Rousseeuw, 1987).

**Implementation Details.** We transform corpora into lists of sentences, where each sentence is a list of tokens. We save these lists into an archive file to avoid repeated preprocessing steps like tokenization and lower casing. For the actual WSI per token, we apply the implementation of AHC from the machine learning library scikit-learn[1]. We

[1] https://scikit-learn.org/stable/

choose the Euclidean distance between word vectors as affinity and the average linkage criterion (see Table 4 in the Appendix for a comparison of different linkage criteria and Cosine vs Euclidean distance).

## 4 Experiments

We conduct several experiments to evaluate how well our pipeline works. To numerically measure the performance of our approach, we perform ADG on different annotated corpora and measure the ARI of our obtained clusters.

### 4.1 Datasets

For the evaluation of our model, we use textual corpora with token-level sense annotations to evaluate the performance of semantic tasks. Please note that many corpora do not contain sense tags for every token, as semantic tagging by hand is a tedious and costly process. So-called all-words corpora contain tags for every token with certain part-of-speech (POS) tags but usually omit closed-class words (Snyder and Palmer, 2004; Moro and Navigli, 2015). Table 1 lists all datasets together with the number of documents, sentences and tokens per corpus and indicates the kind of POS for tokens with semantic tags. Senseval and SemEval are part of WSDEval (Raganato et al., 2017) which is a unified evaluation framework that offers several annotated corpora in the same XML format with sense annotations from WordNet (Miller et al., 1990) version 3.0. Raganato et al. (Raganato et al., 2017) applied the XML schema of the SemEval2013 all-words WSD task (Navigli et al., 2013), removed annotations for auxiliary verbs, semi-automatically updated WordNet senses to version 3.0, lemmatized and POS tagged all tokens to standardize the corpora. Some datasets like Senseval2 and Senseval3 do not contain semantic tags for all words of a POS. Sometimes, multiple sense tags exist in the case of ambiguity or if no suitable WordNet sense was available (Navigli et al., 2013).

### 4.2 Performance Evaluation of Our Approach

We compare the performance of our approach with two different baselines (see Table 2) to assess its value in practice. The first baseline assigns a distinct sense to each token, called "No Cluster"-baseline. The second baseline assigns all occurrences of the same word to a single sense, called "Single Cluster"-baseline. We perform our ADG pipeline with average linkage and the Euclidean distance (see Table 4 in the Appendix for other options). Table 2 presents our results with linkage distance thresholds, which we optimized within a range of $8 - 16$ (see Appendix A.4). Please note that all ARI scores are slightly better than our baselines (see Table 2), except for the SemEval2013 task for which our best result is equal to the "Single Cluster"-baseline. As a proof of concept, we were able to improve over the baselines with some parameter tuning of the distance threshold.

To study how the ARI scores depend on the distance threshold, we show the distribution of ARI scores in Figure 2 for every dataset for varying distance thresholds. For large distance thresholds, the ARI score matches the "Single Cluster"-baseline, since all occurring tokens end up in a single cluster. Unfortunately, the scores converge to the baseline. However, this might be due to selective annotations in the corpora. Note that for SemEval2007 we can see a peak before reaching the ARI for the "Single Cluster"-baseline, which shows that for that particular dataset, the token embeddings can give meaningful clusterings.

To further analyze the SemEval2007 dataset we show the distribution of the ARI for different linkage criteria in the left panel, together with the "Single Cluster"-baseline and the ARI for different sense counts in the right panel in Figure 3. We see that the exact choice of the linkage criterion is not critical and that for the SemEval2007 corpus the clustering works well for tokens with a single and more than three unique senses. For the other datasets the corresponding plots are in the Figure 7 in the Appendix.

## 5 ADG for Shakespeare's Works

So far, we defined the ADG task and proposed a simple pipeline to solve it. In the experiments above, we generate contextualized word vectors with a general CharacterBERT model pretrained on the English Wikipedia and the OpenWebText corpus.[2] However, ADG is much more interesting and becomes more complicated if raw text is the only available training resource for the particular language to create a dictionary. In that case we have to pretrain a language model on the exact text that is the input for the complete pipeline. Note that such an approach is also applicable to unannotated

[2] https://skylion007.github.io/OpenWebTextCorpus/

| Corpora | Docs. | Sents. | Tokens | Annotations per POS |
|---|---|---|---|---|
| Senseval2 (Edmonds and Cotton, 2001) | 3 | 242 | 5,766 | ADJ, ADV, NOUN, VERB |
| Senseval3 (Snyder and Palmer, 2004) | 3 | 352 | 5,541 | ADJ, ADV, NOUN, VERB |
| SemEval2007 (Pradhan et al., 2007) | 3 | 135 | 3,201 | NOUN, VERB |
| SemEval2013 (Navigli et al., 2013) | 13 | 306 | 8,391 | NOUN |
| SemEval2015 (Moro and Navigli, 2015) | 4 | 138 | 2,604 | ADJ, ADV, NOUN, VERB |
| SemCor (Miller et al., 1993) | 352 | 37,176 | 802,443 | ADJ, ADV, NOUN, VERB |

Table 1: Overview of WSDEval Corpora. POS tags are adjectives (ADJ), adverbs (ADV), nouns (NOUN) and verbs (VERB).

| Corpus | Dist. Threshold | Unique Senses | No Cluster | Single Cluster | ADG (ours) |
|---|---|---|---|---|---|
| SemCor | 14.50 | 54,806 | 0.0000 | 0.6521 | 0.6522 |
| Senseval2 | 14.00 | 1,626 | 0.0000 | 0.9136 | 0.9137 |
| Senseval3 | 10.30 | 2,144 | 0.0000 | 0.8395 | 0.8671 |
| SemEval2007 | 9.60 | 1,685 | 0.0000 | 0.7109 | 0.8632 |
| SemEval2013 | 15.00 | 2,376 | 0.0000 | 0.9377 | 0.9377 |
| SemEval2015 | 10.60 | 876 | 0.0000 | 0.9464 | 0.9509 |

Table 2: ARI scores (last three columns) for two baselines "No Cluster" and "Single Cluster" vs our results "ADG (ours)" using Grimm's BERT.

low resource languages.

To investigate this scenario, we apply our method to generate a dictionary for all works of the famous English poet William Shakespeare (1564 – 1616). The vocabulary and grammar from Early Modern English (used in late $15^{th}$ to mid-to-late $17^{th}$ century) is different from today's Modern English (used since mid-to-late $17^{th}$ century). Nevertheless, his works have been widely studied and understood and are readable without too much effort. As the manual creation of appropriate dictionaries is time-consuming and computationally expensive, the results of our automated pipeline (see Algorithm 1) could be a useful starting point for generating such a dictionary.

**Training Data.** We use an open corpus with sonnets and plays from Shakespeare.[3] For preprocessing, we remove stage directions beginning with "<". We delete all lines that contain only a number, e.g., years of publication or enumerations of sonnets. Additionally, we remove repeated line breaks. The resulting corpus consists of 112,521 sentences with 1,152,400 tokens and 23,547 unique words. It is small compared to typical datasets used to train CharacterBERT, but larger than SemCor (802,443 total tokens, see Table 1).

**CharacterBERT Model for Shakespearean English.** We train a CharacterBERT model with the original pretraining code[4] and our Shakespeare corpus. The used hyperparameters for pretraining can be found in Table 8 in the Appendix. We also use the LAMB optimizer (You et al., 2019), a layer-wise adaptive large batch optimization technique that works well with attention models like CharacterBERT. Please note that the training process includes two phases. The optimizer works with a higher learning rate and shorter input sequence lengths during the first phase to achieve broadly reasonable weights. The second phase requires fewer update steps and improves the weights with a lower initial learning rate and longer input sequences. Different sequence lengths require adaptions regarding the number of accumulation steps and batch size, as the target batch size and the CharacterBERT model need to fit into the GPU's memory. We perform our ADG pipeline with average linkage and the Euclidean distance, as this setup worked best for most corpora, particularly for the large SemCor. (see Table 4). We run the pipeline with threshold 8.0, 9.0, and 10.0. Table 3 shows that different numbers of senses are found as expected. Since we have no ground truth we arbitrarily choose the threshold 9.0 for the following examples.

**Qualitative Evaluation.** The raw text from Shakespeare does not provide semantic annotations. So we can not use metrics like the ARI for quantitative evaluation. Instead, we pick examples

[3] https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt

[4] https://github.com/helboukkouri/character-bert-pretraining

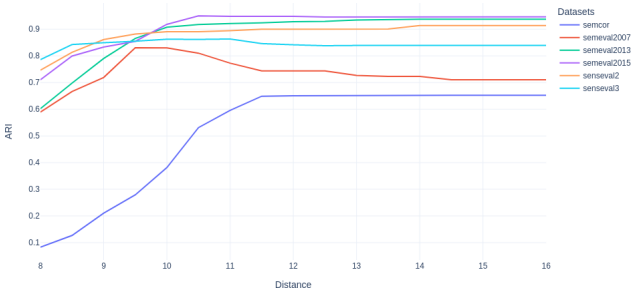



Figure 2: ARI scores for varying linkage distance thresholds.

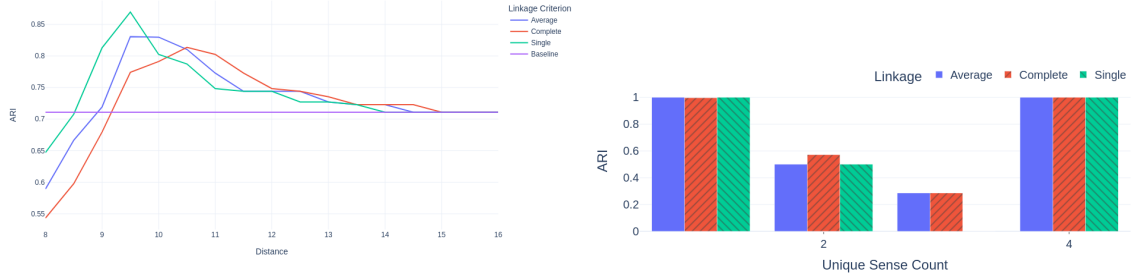



Figure 3: Results only for the SemEval2007 dataset. ARI score for different thresholds and linkage criteria and the "Single Cluster"-baseline (left) and ARI score per sense count and different linkage criteria (right).

| Linkage Distance Threshold | 8.0 | 9.0 | 10.0 |
|---|---|---|---|
| Reference Sentences | | 112,521 | |
| Number Tokens | | 1,152,400 | |
| Unique Words | | 23,547 | |
| Unique Senses | 52,797 | 51,070 | 49,272 |

Table 3: Number of unique senses for different thresholds in our Shakespeare Dictionary.

from Shakespeare's Words[5], an online glossary, thesaurus, and collection of Shakespeare's works, and compare them with our findings manually. Please note that we present all tokens lowercase and separated with single spaces. The enumerator styles indicate the assigned senses. Our created dictionary offers two different senses for the word *eyed*. Looking at the sentence examples, the first example is an adjective but the second is a verb.

- *it is the green - eyed monster , which doth mock*
- ◇ *for as you were when first your eye i eyed ,*

Our dictionary correctly lists only one sense for both occurrences of *writer*.

- *i ' ll haste the writer , and withal*
- *drive some of them to a non - come . only get the learned writer to*

However, for *wrongful*, we incorrectly get two different senses.

- *that i despise thee for thy wrongful suit ,*
- ◇ *in wrongful quarrel you have slain your son .*

[5] https://www.shakespeareswords.com

Curiously, words with more reference sentences tend to have outliers. For example, the word *englishman* only has one meaning, however our dictionary assigns eight reference sentences to the same sense but assigns two occurrences to another.

- *a soul so easy as that englishman ' s . '*
- *king henry . an englishman ?*
- *thinking this voice an armed englishman -*
- *for that my grandsire was an englishman -*
- *a box of the ear of the englishman , and swore he would pay him*
- *caius . by gar , then i have as much mockvater as de englishman .*
- *cassio . is your englishman so expert in his drinking ?*
- *i do not know that englishman alive*
- ◇ *that any englishman dare give me counsel ?*
- ◇ *where ever englishman durst set his foot .*

The word *major* is an interesting dictionary entry. The first two references form a sense, while the other two occurrences belong to a second cluster. At first glance, the division appears correct since the first sense is a noun, and the second sense is an adjective. However, *major* means "matter" in the first example, but refers to a constellation in the second one. A correct distinction might require background knowledge and logical reasoning. Nevertheless, the entry is almost correct.

- *fal . i deny your major . if you will deny the sheriff , so ; if not ,*
- *nativity was under ursa major , so that it follows i am rough and*
- ◇ *the major part of your syllables ; and though i must be content to*

◇ *my major vow lies here , this i ' ll obey .*

In this experiment, the most difficult challenges are the corpus size, which is small for training a language model and large for clustering methods. Contexts in our Shakespeare corpus are often short and incomplete, since we defined a sentence to be limited to a single line, but many sentences extend over several lines. While our generated dictionary tends to list too many senses per word, it also contains valuable groupings and correct entries.

## 6 Conclusion

In this paper, we examine whether the brothers Grimm could create a dictionary using language models like BERT. To achieve this, we define the ADG task and a first simple approach to automatically generate a dictionary from raw text using a language model and AHC.

At its core, ADG is a clustering problem, and it is possible to evaluate it with ARI scores if sense annotations are available. Thus, (partially) labeled corpora for WSD are suitable for comparing different ADG approaches. Other metrics like the Silhouette Coefficient (see Appendix A.3) measure the cluster quality without any ground truth but usually have strong assumptions and miss some crucial edge cases. In addition, we consider a scenario with texts from Shakespeare's work. We train a CharacterBERT model on it and use our pipeline to generate a customized dictionary. Many dictionary entries are reasonable but sometimes list too many senses per word.

While our first simple approach to ADG does not give perfect results yet, we see great potential for this task and believe that our contribution is a starting point that could be used by linguists who want to create new dictionaries. It might be reasonably assumed that the quality of the resulting clusters of our pipeline will further increase with the continuous improvement of state-of-the-art language models. We assume that with today's technologies, the brothers Grimm would likely have witnessed the completion of their German dictionary during their lifetime.

## Limitations

In this work, we defined the task of ADG and proposed one method to solve it. Nevertheless, there are many open questions emphasizing the key challenges and proposing new ideas beyond our experiments.

1. **How can we train language models even for low resource languages?** Our ADG pipeline can be used for low resource languages to build a preliminary dictionary, but requires to pretrain a language model from scratch. As we have seen in our experiments with the works of William Shakespeare, our approach generates reasonable outcomes, but learning language models on small corpora is challenging.
2. **Is there a better way to evaluate automatically generated dictionaries?** The evaluation of dictionaries without any ground truth remains partially open, mainly because the Silhouette coefficient is not applicable to situations where only one cluster exists. Other metrics and techniques to analyze high-dimensional clusters might be useful.
3. **How can we determine the correct number of senses for a word?** We analyze the search range for the linkage thresholds in Appendix A.4. Our experiments show that the optimal threshold is different for every dataset. It is still unclear how the optimal cut criterion can be determined in an unsupervised manner.
4. **How can we find relations between words?** We discuss the detection of relations like synonyms in Appendix A.5 but do not deliver a concrete implementation. The detection of relations like synonyms might be possible using by clustering the centroids and could lead to a reasonable extension of our pipeline.
5. **Can we automatically generate descriptions for the word types?** Generating glosses (aka descriptions) for each extracted sense is a challenging task. Currently we are only able to assign senses to each word in a text, together with references to sentences. In the future it will be interesting to automatically generate short descriptions for each word and each sense respectively and find a meaningful way to evaluate automatically generated glosses.

Our work has shown the potential of ADG, yet some aspects of the approach remain unsolved and for future work. Nevertheless, we believe that ADG can lead to powerful practically useful tools for dictionary generation which will profit from new and more powerful language models and additional input created by the deep learning community.

## Ethics Statement

In this paper we propose an approach to automatically generate a dictionary from plain text. Using technology for communication is a great advantage of today's world. Having sentences and whole documents translated in the blink of an eye is beneficial for the communication between humans of all kinds of languages and cultures. The aim of this area of research is to use machines to study languages, potentially also low-resource languages in the context of written text. It is to say that this kind of technology should always have a supporting role and should not be used to make final decisions. Machine learning models always hold the risk of producing biased and incorrect predictions. Our work relies on the use of large language models such as BERT and CharacterBERT. These models are trained on large amounts of data and encode various parts of it. There is a risk that they contain sensitive data, generate false information or are actively misused.

## Acknowledgments

Computational infrastructure and support were provided by the Centre for Information and Media Technology at Heinrich Heine University Düsseldorf.[6]

## References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proc. of the 27th Int. Conf. Comput. Ling.*, pages 1638–1649.

Nikolay Arefyev, Boris Sheludko, and Tatiana Aleksashina. 2019. Combining neural language models for word sense induction. In *Analysis of Images, Social Networks and Texts*, pages 105–121, Cham. Springer International Publishing.

Jing-Shin Chang, Yi-Chung Lin, and Keh-Yih Su. 1995. Automatic construction of a Chinese electronic dictionary. In *Third Workshop on Very Large Corpora*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. pages 276–286.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of the 2019 Conference of the North American Chapter of the Assoc. Comput. Linguist.*, pages 4171–4186.

Philip Edmonds and Scott Cotton. 2001. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France. Association for Computational Linguistics.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. Analysing lexical semantic change with contextualised word representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3960–3973, Online. Association for Computational Linguistics.

Jacob Grimm and Wilhelm Grimm. 1854. *Deutsches Wörterbuch*, volume 2. S. Hirzel.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proc. of the 57th Annual Meeting of the Assoc. Comput. Linguist.*, pages 3651–3657.

Hiroyuki Kaji, Shin'ichi Tamamura, and Dashtseren Erdenebat. 2008. Automatic construction of a Japanese-Chinese dictionary via English. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Matej Martinc, Syrielle Montariol, Elaine Zosa, and Lidia Pivovarova. 2020. Capturing evolution in word usage: Just add more clusters? In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 343–349, New York, NY, USA. Association for Computing Machinery.

Craig J. A. McEwan, Iadh Ounis, and Ian Ruthven. 2002. Building bilingual dictionaries from parallel web documents. In *Proceedings of the 24th BCS-IRSG European Colloquium on IR Research: Advances in Information Retrieval*, page 303–323, Berlin, Heidelberg. Springer-Verlag.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–244.

[6] https://wiki.hhu.de/display/HPC/Wissenschaftliches+Hochleistungs-Rechnen+am+ZIM

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, page 303–308, USA. Association for Computational Linguistics.

Andrea Moro and Roberto Navigli. 2015. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.

Gandalf Nicolas, Xuechunzi Bai, and Susan T Fiske. 2021. Comprehensive stereotype content dictionaries using a semi-automated method. *European journal of social psychology*, 51(1):178–196.

Karl Pearson. 1901. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic. Association for Computational Linguistics.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.

William M. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Wei Ren, Hengwei Zhang, and Ming Chen. 2022. A method of domain dictionary construction for electric vehicles disassembly. *Entropy*, 24(3).

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Peter J. Rousseeuw. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Benjamin Snyder and Martha Palmer. 2004. The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain. Association for Computational Linguistics.

Arvi Tavast, Kristina Koppel, Margit Langemets, and Jelena Kallas. 2020. Towards the Superdictionary: Layers, Tools and Unidirectional Meaning Relations. pages 215–223.

István Varga and Shoichi Yokoyama. 2009. Bilingual dictionary generation for low-resourced language pairs. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 862–870, Singapore. Association for Computational Linguistics.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*, pages 161–170, Erlangen, Germany. German Society for Computational Linguistics & Language Technology.

Ludwig Wittgenstein. 1953. *Philosophical Investigations*. Wiley-Blackwell, New York, NY, USA.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144v2*.

David Yenicelik, Florian Schmidt, and Yannic Kilcher. 2020. How does BERT capture semantics? a closer look at polysemous words. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 156–162, Online. Association for Computational Linguistics.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962v5*.

# A Appendix

Our design choices for the ADG pipeline are based on extensive experiments that we conducted. These are described in the following sections. Namely, we compare different linkage criteria and metrics for clustering.

## A.1 ARI for Clusterings with Different Affinities and Linkage Criteria

As the geometry of clusters in the embedding space is not trivial, we empirically search for the best linkage criterion with the WSDEval corpora. Therefore, we perform AHC with average linkage, complete linkage and single linkage and aid it with the true number of clusters it should find. More precisely, we cluster the word vectors for each distinct, sense annotated token based on all corresponding word vectors and the total, unique number of its annotated senses. We compute the ARI to measure the quality of the clustering and omit generated senses.

While the cosine distance measures angles between vectors, the Euclidean distance compares their lengths. Even if we usually use the cosine distance for NLP tasks, we also set both affinities side by side.

Table 4 presents the ARIs for sense clusterings with different affinities and linkage criteria and underline indicates the best performance for each corpus. All runs but for SemCor with complete linkage outperform our baselines from Table 4, indicating that our pipeline extracts meaningful word senses. Average linkage works best for SemCor, Senseval2, Senseval3 and SemEval2013. Complete linkage yields the highest ARI for SemEval2007 and SemEval2015. Often, the three criteria perform similarly well and SemCor is the only corpus for which complete linkage works significantly worse than the other two criteria. SemCor contains not only far more tokens and sense annotations but also some words with a higher disambiguity with up to 57 unique senses, whereas the other corpora only hold words with at most 5 unique senses.

We expect marginally better results for the Euclidean distance $\mathsf{D}_{\text{Euc}}(A, B)$ with $A, B \in \mathbb{R}^d$ and $d \in \mathbb{N}_{>0}$, because the cosine distance $\mathsf{D}_{\cos}(A, B)$ is equivalent to the Euclidean distance of normalized vectors. By expansion, it holds

$$\begin{aligned}\mathsf{D}^2_{\text{Euc}}(A - B) &= (A - B) \cdot (A - B) \\ &= \mathsf{D}^2_{\text{Euc}}(A) + \mathsf{D}^2_{\text{Euc}}(B) - 2(A \cdot B).\end{aligned}$$

With normalized vectors $\mathsf{D}^2_{\text{Euc}}(A) = \mathsf{D}^2_{\text{Euc}}(B) = 1$, this term is equal to $2(1 - \cos(A, B))$ and therefore $\mathsf{D}_{\cos}(A, B) = \frac{\mathsf{D}^2_{\text{Euc}}(A,B)}{2}$.

However, the Euclidean distance usually produces slightly stronger results, but yields the same ARI as the cosine distance for SemEval2013 and SemEval2015 with average linkage, Senseval3 and SemEval2013 with complete linkage and SemEval2013 with single linkage. Senseval2 with single linkage is the only setup for which the cosine distance moderately outperforms the Euclidean distance. This experiment suggests a setup with the Euclidean distance and average linkage. Possible explanations for the results are rounding errors and word vectors that are not exactly normalized to length one.

Please note that Yenicelik et al. (2020) investigate the organization of BERT's word vectors for polysemous words. More precisely, they use the semantic annotations in SemCor (Miller et al., 1993) to analyze the separability and clusterability of the 768 dimensional output of BERT's last layer.

They perform a dimensionality reduction via principal component analysis (PCA) (Pearson, 1901) and predict a semantic class per token with a linear classifier (Yenicelik et al., 2020). They interpret its accuracy as a measure of linear separability. Results for frequently occurring words show that individual semantic classes are reasonably linearly separable and contextual word embeddings form closed semantic regions (Yenicelik et al., 2020).

For clusterability, they apply several clustering algorithms on word vectors for sampled words from SemCor and the news.2007.corpus[7] and measure the quality of the resulting clusters with the $ARI$ (Rand, 1971; Hubert and Arabie, 1985) (Yenicelik et al., 2020). No clustering method was able to distinguish between multiple semantic classes on a satisfying level (Yenicelik et al., 2020). For several words, resulting clusters differ not only in meanings but also in other linguistic properties like sentiment (Yenicelik et al., 2020). BERT's word embeddings form closed but overlapping semantic regions (Yenicelik et al., 2020).

We perform the analysis on the whole SemCor corpus with AHC and without PCA. In contrast, Yenicelik et al. (2020) use more complex clustering methods and sample polysemous words.

[7] https://www.statmt.org/wmt14/training-monolingual-news-crawl/

| Corpus | Average Linkage | | Complete Linkage | | Single Linkage | |
|---|---|---|---|---|---|---|
| | Cosine | Euclidean | Cosine | Euclidean | Cosine | Euclidean |
| Semcor | 0.6615 | 0.6627 | 0.4899 | 0.4958 | 0.6561 | 0.6558 |
| Senseval2 | 0.9594 | 0.9596 | 0.9553 | 0.9558 | 0.9541 | 0.9540 |
| Senseval3 | 0.9322 | 0.9326 | 0.9280 | | 0.9261 | 0.9287 |
| SemEval2007 | 0.9457 | 0.9612 | 0.9687 | 0.9844 | 0.9457 | 0.9612 |
| SemEval2013 | 0.9700 | | 0.9632 | | 0.9694 | |
| SemEval2015 | 0.9712 | | 0.9723 | 0.9734 | 0.9711 | 0.9681 |

Table 4: ARI for Clusterings with Different Affinities and Linkage Criteria using known sense counts. Underline indicates the best result per corpus. Joined cells indicate identical ARIs for both affinities. We ignore all tokens with generated senses.

### A.2 Sense-Count-Level ARI for Clusterings with Different Linkage Criteria

While Table 4 presents the overall performance of our approach with one ARI per corpus, Figure 4 shows bar plots with one average ARI per unique sense count. We analyze the dictionaries from Table 4 and completely omit tokens with generated senses in our plots again.

The results for monosemous words are almost perfect for all corpora and linkage criteria, because we provide the true number as we generate these dictionaries (see Section A.1). For polysemous words, the average ARI is usually smaller but clearly positive, indicating clusterings that are better than random choice. Especially for larger corpora like SemCor or SemEval2013, the drop is more evident. Please note that the total number of words per bar usually decreases with higher unique sense counts.

### A.3 Sense-Count-Level Silhouette Coefficient for Clusterings with Different Linkage Criteria

Now we calculate one average Silhouette coefficient per sense count for the dictionaries from Table 4 to investigate the quality of sense clusterings. Please note that the score requires $2 \leq k \leq n-1$ with the sense count $k$ and token count $n$ (Rousseeuw, 1987). Thus, we omit all annotated tokens that do not fulfil the condition and cannot provide any measurements for $n = 1$. Similar to Figure 4, the significance of the bars decreases with higher unique sense counts. As the SemEval corpora provide very few polysemous words, their plots are less representative.

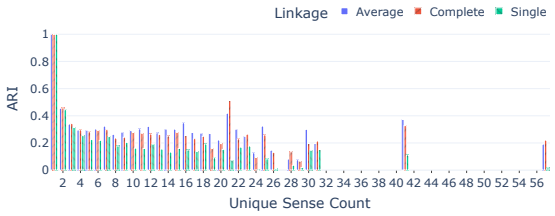
The cluster quality decreases with increasing sense counts, starting at a score of approximately $0.15$ for $n = 2$ and approaching values near $0.1$ for most configurations and corpora. Average and complete linkage usually yield similar scores, whereas single linkage often performs worse and even gets some negative scores for SemCor.

### A.4 Linkage Distances at the Cut

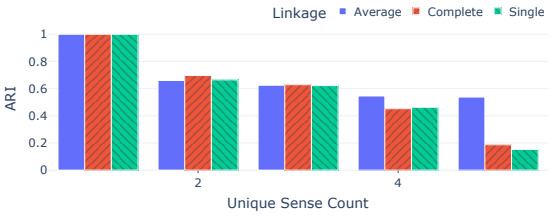
The sensible prediction of sense counts per word is problematic due to the fact that we need to evaluate multiple clusterings per word and do not have any reliable information for single senses. Choosing one linkage distance threshold above which we do not merge any clusters avoids the choice of a suitable number of senses and requires only one clustering per word. Therefore, we investigate the linkage distances at the last cuts that occur during our clusterings with known sense counts (see Table A.1).

As the linkage criterion optimizes a certain distance between clusters, there are $n-1$ distances for $n$ samples. AHC is a bottom-up approach that starts with clusters that contain only one sample and successively builds a binary merge tree. We investigate the exact linkage distance at the tree node that marks the last merge. If all linkage distances differ, we can generate the same clustering by setting the distance threshold to the exact distance at the last merge and add a small number. In cases with no available distance, for example, if we merge all samples into one cluster, we pick the closest obtainable distance in the tree. For words with a single occurrence, we do not consider any distances.

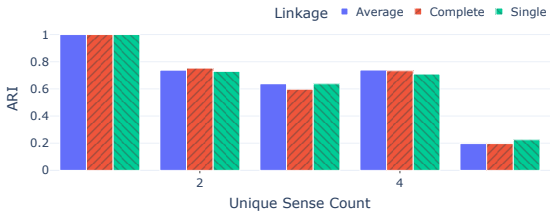
Table 5 and Table 6 show the averages and standard deviations of the Euclidean and cosine linkage distances at the last performed merge in the merge tree. Again, we analyze the dictionaries from Table A.1 and only consider tokens with known senses. The averages are fairly similar for all corpora and the standard deviations are rather small. Our results for SemEval2015 are clearly the worst due to lower averages and higher standard deviations, possibly because it contains comparatively few samples.



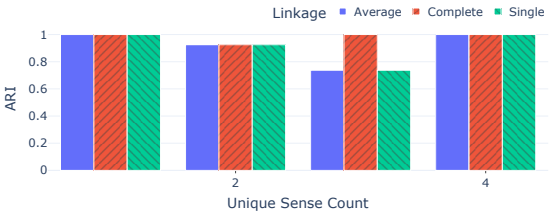

(a) SemCor



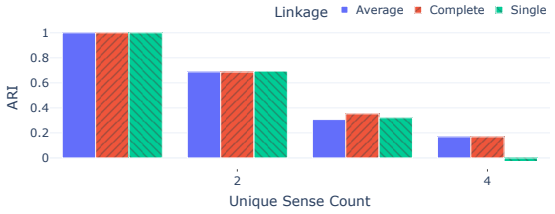

(b) Senseval2



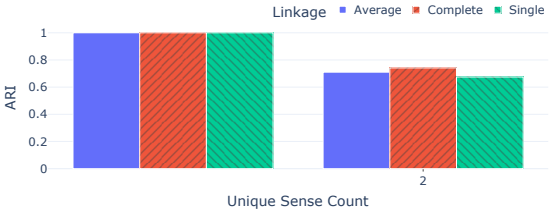

(c) Senseval3




(d) SemEval2007

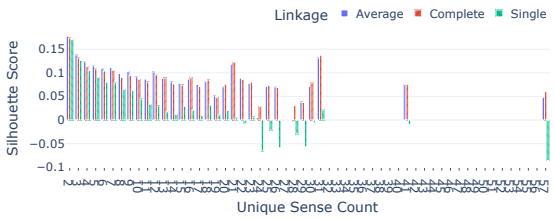



(e) SemEval2013

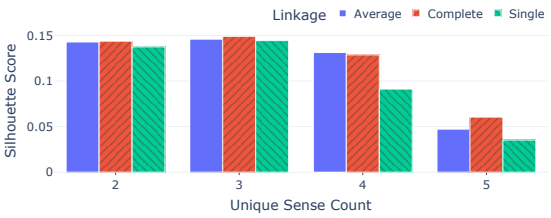



(f) SemEval2015

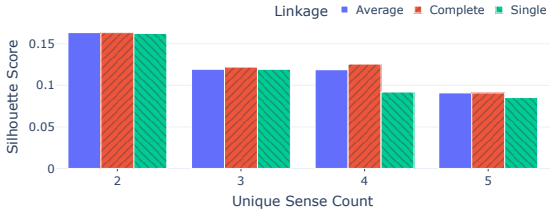
Figure 4: Sense-Count-Level ARI for Clusterings with Different Linkage Criteria using known sense counts and the Euclidean distance as affinity. Each bar plot shows the average ARI for all words that have the same number of true unique senses and no generated senses. We analyze the dictionaries from Section A.1.



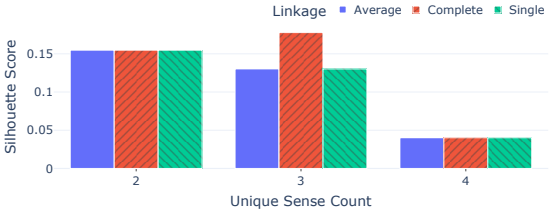

(a) SemCor



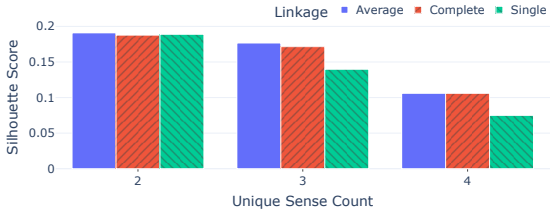

(b) Senseval2



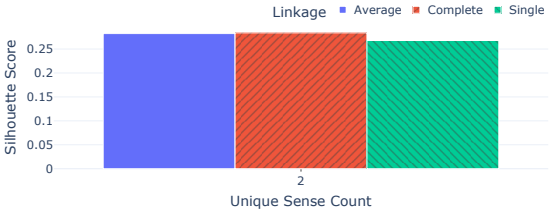

(c) Senseval3




(d) SemEval2007

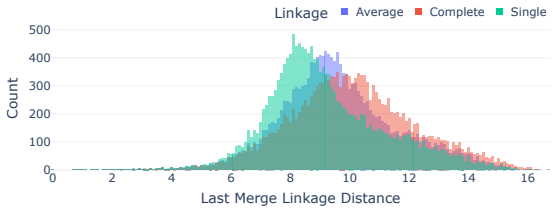



(e) SemEval2013

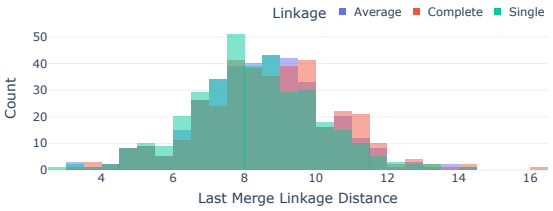



(f) SemEval2015

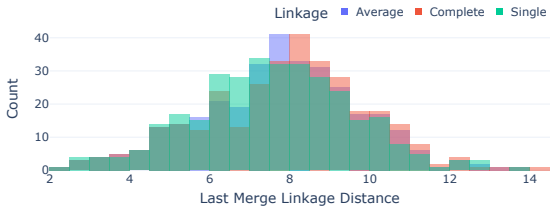
Figure 5: Sense-Count-Level Silhouette Coefficient for Clusterings with Different Linkage Criteria *using known sense counts and the Euclidean distance as affinity. Each bar plot shows the average Silhouette Score for all words that have the same number of true unique senses and no generated senses. We analyze the dictionaries from Section A.1.*



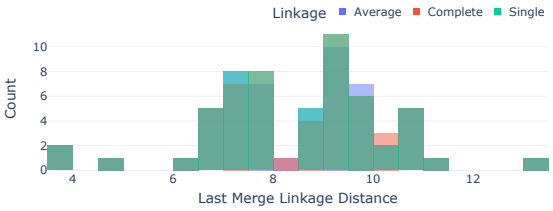

(a) SemCor



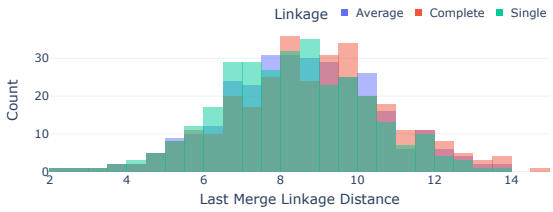

(b) Senseval2



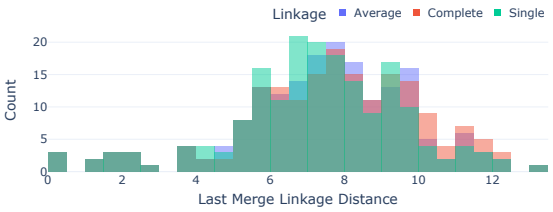

(c) Senseval3




(d) SemEval2007




(e) SemEval2013




(f) SemEval2015

Figure 6: Euclidean Linkage Distances at the Last Merge using known sense counts. Each histogram shows frequencies for all words that have no generated senses. We analyze the dictionaries from Section A.1.

| Corpus | Average Linkage Avg. | Std. | Complete Linkage Avg. | Std. | Single Linkage Avg. | Std. |
|---|---|---|---|---|---|---|
| SemCor | 9.6395 | 2.0394 | 10.0726 | 2.1631 | 9.1755 | 2.0249 |
| Senseval2 | 8.4796 | 1.8256 | 8.7032 | 1.9295 | 8.2633 | 1.7940 |
| Senseval3 | 7.7612 | 2.0223 | 7.9206 | 2.0855 | 7.5898 | 2.0015 |
| SemEval2007 | 8.5086 | 1.7495 | 8.5396 | 1.7577 | 8.4842 | 1.7464 |
| SemEval2013 | 8.5505 | 2.0034 | 8.7958 | 2.1140 | 8.3034 | 1.9379 |
| SemEval2015 | 7.4175 | 2.4254 | 7.6239 | 2.5386 | 7.2192 | 2.3627 |

Table 5: Average and Standard Deviation of Euclidean Linkage Distances at the Last Merge using known sense counts. We analyze the dictionaries from Section A.1 and ignore tokens with generated senses or only one occurrence.

| Corpus | Average Linkage Avg. | Std. | Complete Linkage Avg. | Std. | Single Linkage Avg. | Std. |
|---|---|---|---|---|---|---|
| SemCor | 0.3624 | 0.1469 | 0.3960 | 0.1610 | 0.3285 | 0.1423 |
| Senseval2 | 0.2900 | 0.1198 | 0.3060 | 0.1304 | 0.2753 | 0.1156 |
| Senseval3 | 0.2464 | 0.1183 | 0.2566 | 0.1234 | 0.2358 | 0.1165 |
| SemEval2007 | 0.2928 | 0.1156 | 0.2947 | 0.1161 | 0.2912 | 0.1152 |
| SemEval2013 | 0.2946 | 0.1297 | 0.3125 | 0.1410 | 0.2771 | 0.1224 |
| SemEval2015 | 0.2357 | 0.1306 | 0.2497 | 0.1404 | 0.2229 | 0.1257 |

Table 6: Average and Standard Deviation of Cosine Linkage Distances at the Last Merge using known sense counts. We analyze the dictionaries from Section A.1 and consider all words with at least two occurrences and no generated senses.

Figure 6 exhibits histograms for Euclidean linkage distances at the cut corresponding to Table 5. Considering the averages and standard deviations of linkage distances in combination with their distributions from the histograms, we propose that most last merges occur near a Euclidean linkage distance of about $8.5 - 9.0$ with a standard deviation of about $1.7$. This observation holds for most examined corpora and even accross different linkage criteria. Due to the sample size of SemCor, the related results are most representative and suggest a bell curve with said parameters. Therefore, a linkage distance threshold slightly above the maximum of the bell curves should yield a reasonable dictionary. The similarities in our experiments suggest that $8.0 - 9.5$ is a reasonable initial search space in hyperparameter optimization.

Table 7 offers the averages and standard deviations of the Euclidean linkage distances at the successor of the last merge in the tree. We need to cut the tree between the last performed merge and its successor to obtain the same clustering. The latter distances are significantly higher than those from Table 7 and the standard deviations indicate minor overlaps of both distributions. These results indicate clear gaps and further suggest the existence of a reasonable linkage distance threshold.

### A.5 Relation Detection

Often, terms and names consist of more than one token, for example, the “White House”. We could use syntactic knowledge to find related words in the sentences. For instance, the contextualized word embedding BERT encodes some syntactic rules (Clark et al., 2019; Jawahar et al., 2019). In contrast, there are syntactic correlations between different words, e.g., for the combination of an auxiliary verb and its participle like “has finished”. Some approaches mitigate such problems with semantic knowledge about existing entities and phrases. Inflections help determine syntax and context in a sentence. Usually, only one entry per infinitive exists in resources like dictionaries. Mapping inflected forms to their infinitives is challenging and may require prior knowledge. We need to pick a distance criterion to separate clusters of word vectors. The criterion could be a fixed threshold or a relative factor for distances. Some methods might depend on more sophisticated geometric criteria or estimate the number of clusters or objects. Its choice might depend on the given corpus.

Similar contexts yield word vectors close in the embedding space, so similar word vectors for different words might indicate synonyms. In contrast, word vectors that point in the opposite direction might reveal antonyms.

However, performing clustering methods on all

| Corpus | Average Linkage Avg. | Std. | Complete Linkage Avg. | Std. | Single Linkage Avg. | Std. |
|---|---|---|---|---|---|---|
| SemCor | 10.1048 | 2.0594 | 10.6219 | 2.1876 | 9.5731 | 2.0454 |
| Senseval2 | 8.7927 | 1.7905 | 9.0681 | 1.9032 | 8.5312 | 1.7627 |
| Senseval3 | 8.0880 | 2.1119 | 8.2969 | 2.2062 | 7.8749 | 2.0615 |
| SemEval2007 | 8.8758 | 2.0199 | 8.9492 | 2.0532 | 8.7969 | 1.9871 |
| SemEval2013 | 8.7354 | 2.0456 | 9.0253 | 2.1949 | 8.4543 | 1.9713 |
| SemEval2015 | 7.7537 | 2.3858 | 8.0112 | 2.5202 | 7.5166 | 2.3156 |

Table 7: Average and Standard Deviation of Euclidean Linkage Distances after the Last Merge using known sense counts. We analyze the dictionaries from Section A.1 and ignore tokens with generated senses or only one occurrence.

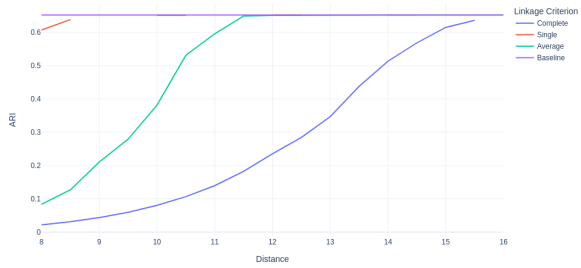
words is more computationally expensive. Clustering the centroids of sense clusters might also reveal related words. In this setting, the definition of negative concepts like antonyms is less obvious. We could measure the strength of the relation between two clusters via the distance between their centroids. The closer any two centroids are, the stronger their relation is and vice versa. We could choose thresholds or ranges to define certain concepts like synonyms and antonyms.

### A.6 ARI scores for different linkage distance thresholds

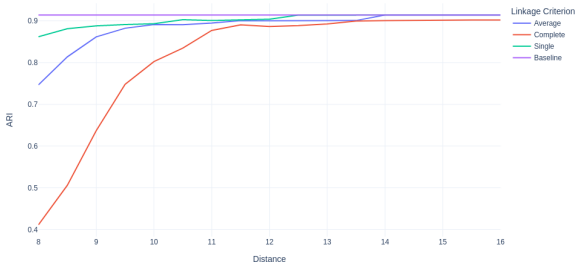
In Figure 7 we show the details for every dataset on the distribution of the ARI score next to the "Single Cluster"-baseline. As described before, for large thresholds the ARI converges to the ARI of our "Single Cluster"-baseline. For both the Senseval3 and the SemEval2007 corpus a clear peak before converging to the baseline. To gain further insights why our method works better for some corpora than for others, an analysis of the corpora and the tagged annotations is necessary.

### A.7 Details on Creating Shakespearean Dictionary

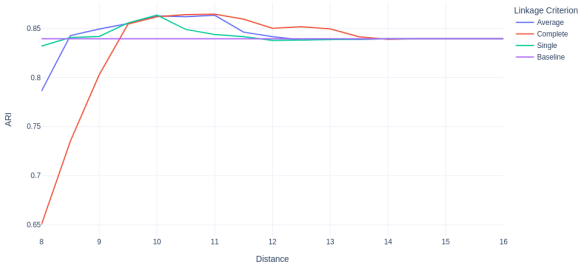
Table 8 shows all hyperparameters we used to pretrain CharacterBERT for creating a Shakespearean dictionary.



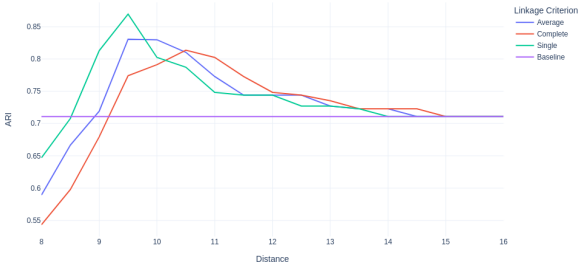

(a) SemCor



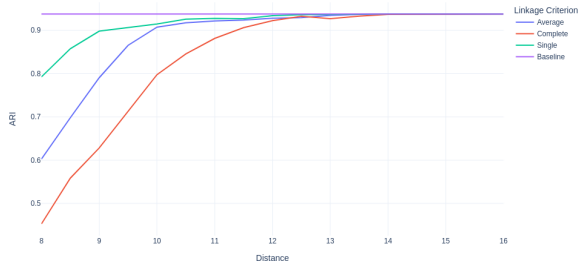

(b) Senseval2



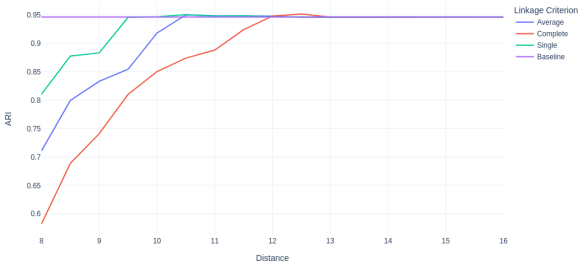

(c) Senseval3




(d) SemEval2007




(e) SemEval2013




(f) SemEval2015

Figure 7: Linkage distance thresholds per dataset

| Hyperparameter | Phase 1 | Phase 2 |
|---|---|---|
| Learning Rate | $6 \times 10^{-3}$ | $4 \times 10^{-3}$ |
| Warm-Up Proportion | 0.2843 | 0.128 |
| Warm-Up Rate | 0.01 | |
| Weight Decay | 0.01 | |
| Target Batch Size | $2,048$ | |
| Accumulation Steps | 256 | $1,024$ |
| Total Batch Size | 8 | 2 |
| Update Steps | $1,800$ | 800 |
| Max. Input Sequence Size | 128 | 512 |
| Max. Masked Tokens per Input | 20 | 80 |

Table 8: Hyper-Parameters for Training CharacterBERT on Shakespeare's Works, based on the hyper-parameters for the general CharacterBERT model (El Boukkouri et al., 2020).