AUTOMATIC DICTIONARY GENERATION: COULD BROTHERS GRIMM CREATE A DICTIONARY WITH BERT?

Anonymous authors
Paper under double-blind review

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 of 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 & Grimm, 1852–1971), 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 of creating a dictionary.

In 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 (Hagiwara, 2021). To automatically build a dictionary from plain reference texts, we need to find all occurrences of each token 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, 1958). Words with only one meaning are called *monosemous*, while ambiguous words are referred to as *polysemous*.

Word sense disambiguation (WSD) is the task of choosing the correct sense word from a fixed set of senses. The task we want to propose in this paper differs from WSD as senses and also the number of senses for each word appearing in a given text is not given. We use a hierarchical clustering method to automatically generate a dictionary from raw text without any further annotations.

It has been shown that BERT's contextualized word embeddings hold syntactic and semantic knowledge (Jawahar et al., 2019; Lin et al., 2019; Rogers et al., 2020). In addition, they form separable clusters for polysemous words (Yenicelik et al., 2020; Wiedemann et al., 2019). We want to utilize these characteristics of contextualized vector representations produced by language models such as BERT to tackle the task of *automatic dictionary generation* (ADG).

Contributions.

1. We propose the task of automatic dictionary generation (ADG).
2. We discuss how to evaluate automatically generated dictionaries.
3. We present a simple approach for ADG using a pretrained CharacterBERT (El Boukkouri et al., 2020) model and agglomerative hierarchical clustering (AHC) (Nielsen, 2016), 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 under: *link will be available after the blind review process.*

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 different meanings of the words 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 (Ivanov et al., 2022; Kaji et al., 2008) and (ii) monolingual dictionaries (Tavast et al., 2021) which are either of a general nature or focus on domain-specific terms (Ren et al., 2022; Das et al., 2022). Bilingual dictionaries have been either created by translation (Varga & 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, Wordnets (Ivanov et al., 2022; Nicolas et al., 2021) or statistics given raw text in both languages (Kaji et al., 2008).

For now, to the best of our knowledge no method exists that does not depend on additional resources to deal with word ambiguities. Our goal is to generate dictionary entries, independent from any additional knowledge, only given raw text for one specific language. Recently there have been approaches to automatically generate glosses (Barba et al., 2021; He & Yiu, 2022) which we will leave to future work for now.

Word Sense Disambiguation. The task of assigning a sense from a fixed set of senses to a word occurring in a sentence or piece of text, is referred to as word sense disambiguation (WSD). There have been different approaches to perform the task of WSD which are described in the following.

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 have been already 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-occurrence and clusters. 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).

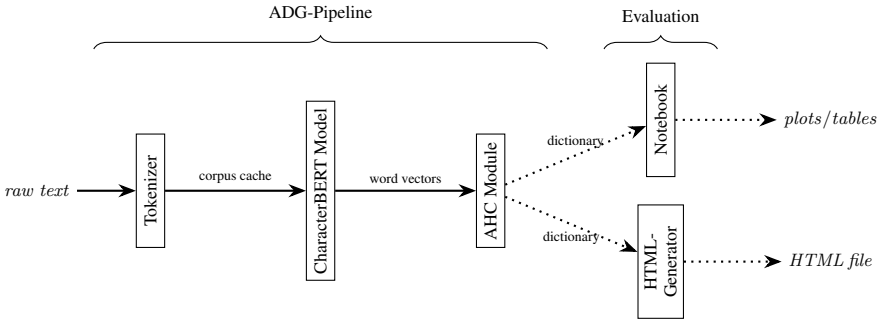


Figure 1: Grimm's BERT: our ADG pipeline implementation. *Solid and dotted arrows indicate obligatory and optional connections, respectively.*

In contrast to the WSD task, we do not assume that we have a set of known senses for every given word. Instead we attempt to derive the number of senses by building clusters. To the best of our knowledge there are currently no approaches to automatically build a dictionary from plain text without further annotations.

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 token. 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 and should be characteristic of the token's sense. Based on these choices, a *dictionary* is a set of one dictionary entry per unique token. 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 token occurring in a text corpus has. Consequently, ADG is different from WSD since we not only assign occurrences of tokens to predefined word senses, but we also have to find those senses for each token.

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., 2020). They contain a kind of a syntactic tree structure in addition to the word order information (Lin et al., 2019). Wiedemann et al. (2019) also compared different contextualized word embeddings for WSD and found that BERT uncased embeddings perform best for this task. Motivated by these results, we use BERT embeddings to tackle the task of ADG.

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:

Algorithm 1: ADG Pipeline

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

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 Character-CNN 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 Disambiguation. Each token 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 (Nielsen, 2016). 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 (Sokal & Michener, 1958), complete linkage takes the maximum distances between all objects of two clusters (Defays, 1977) and single linkage uses the minimum distance between all objects of the two sets (Sibson, 1973).

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 token 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 & Arabie, 1985) as an objective evaluation metric to quantify the quality of the resulting clusters. Classical metrics for WSD like the accuracy or the 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.

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 & Arabie, 1985) is the Rand Index, corrected for chance using the hypergeometric function.

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.

3.3 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 WSD per token, we apply the implementation of AHC from the machine learning library scikit-learn[1]. We 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

Corpora	Docs.	Sents.	Tokens	Annotations per POS
Senseval2 (Edmonds & Cotton, 2001)	3	242	5,766	ADJ, ADV, NOUN, VERB
Senseval3 (Snyder & 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 & 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) (adapted from Table 1 in Miller et al. (1994)).

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 & Palmer, 2004; Moro & Navigli, 2015). All datasets used for evaluation are shortly described in the following.

- **Senseval2** (Edmonds & Cotton, 2001): Data from the Second International Workshop on Evaluating Word Sense Disambiguation Systems English all-words task.
- **Senseval3** task 1 (Snyder & Palmer, 2004): Data from the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text English all-words task.
- **SemEval2007** task 17 (Pradhan et al., 2007): Data from the Fourth International Workshop on Semantic Evaluations WSD English all-words task.
- **SemEval2013** task 12 (Navigli et al., 2013): Data from the English WSD task of the Seventh International Workshop on Semantic Evaluation.
- **SemEval2015** task 13 (Moro & Navigli, 2015): Data from the Ninth International Workshop on Semantic Evaluation English WSD task.
- **Semcor** Miller et al. (1993): The semantic concordance (SemCor) corpus, is the combination of a textual dataset and lexicographical information. Semcor includes 103 passages from the Brown Corpus (Kucera & Francis, 1967) with semantic information from the lexical database WordNet (Miller et al., 1990).

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. (2017) applied the XML schema

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

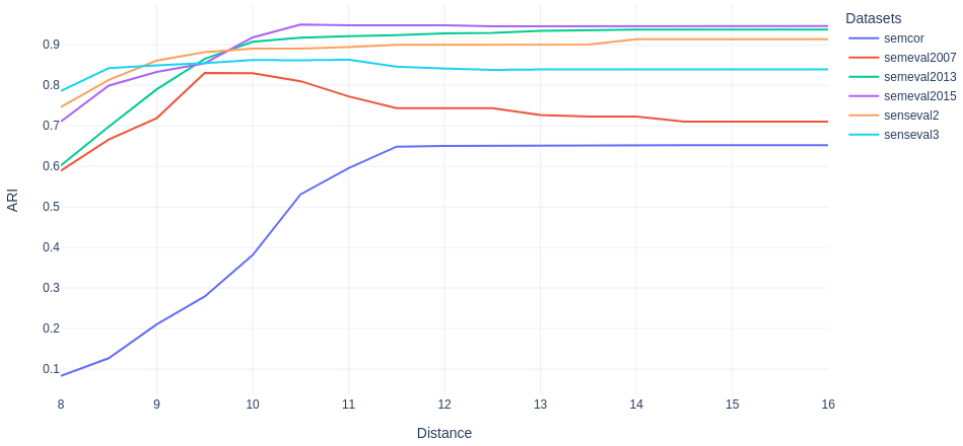


Figure 2: ARI scores for varying linkage distance thresholds.

of the SemEval2013 all-words WSD task (Navigli et al., 2013), remove annotations for auxiliary verbs, semi-automatically update WordNet senses to version 3.0, lemmatize and POS tag all tokens to standardize the corpora. Table 1 lists the number of documents, sentences and tokens per corpus and indicates the kind of POS for tokens with semantic tags. 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

			ARI		
Corpus	Distance 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 for two baselines “No Cluster” and “Single Cluster” vs our results “ADG (ours)” using Grimm’s BERT. Overall the improvements are small.

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 token 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.

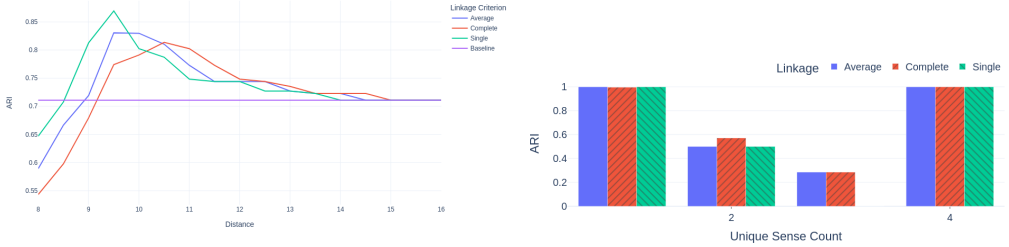


Figure 3: Results only for the SemEval2007 dataset. ARI score for different thresholds and linkage criteria and the "Single Cluster"-baseline (left) and ARI score per sense count and different linkage criteria (right).

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 problem 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 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 15th to mid-to-late 17th century) is different from today's Modern English (used since mid-to-late 17th 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 result 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 constists of 112,521 sentences with 1,152,400 total and 23,547 unique tokens. 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.

Shakespearean Dictionary from Grimm's BERT. We perform our ADG pipeline with average linkage and the Euclidean distance, as this setup worked best for most corpora, particularly for the

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

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

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

Linkage Distance Threshold	8.0	9.0	10.0
Unique Senses	52,797	51,070	49,272

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

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. Please note that we present all tokens lowercase and separated with single spaces. The enumerator styles indicate the assigned senses.

Our created dictionary offers for the token *eyed* two different senses. 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 .*

Curiously, tokens 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.

The token *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 token, it also contains valuable groupings and correct entries.

6 Conclusion

In this paper, we examine whether the brothers Grimm could create a dictionary using a language models like BERT. To achieve this, we propose 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) 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 works. 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 token.

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 further 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.

Open Questions and Key Challenges

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 on the available data. As we have seen in our experiments with the works of William Shakespeare, our approach generates reasonable outcomes, but the small corpora are challenging for learning language models.
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 token?** 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. Maybe this is possible using by clustering the centroids and could lead to a reasonable extension of our pipeline.
5. **What is the best way to visualize the learned dictionaries?** Our generated dictionaries contain fixed groups of senses. Therefore, we can save them in data frames or render static but clickable HTML files. A more interactive approach could directly access the binary merge trees from AHC. The user might inspect one hierarchy of contextualized word vectors per token and set individual criteria to find clusters.
6. **Can we automatically generate descriptions for the tokens?** Generating glosses (aka descriptions) for each extracted sense is a challenging task. Currently we are only able to assign senses to each token 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.

BIBLIOGRAPHY

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 1638–1649, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1139.

Edoardo Barba, Luigi Procopio, Caterina Lacerra, Tommaso Pasini, and Roberto Navigli. Exemplification modeling: Can you give me an example, please? In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 3779–3785. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/520. URL https://doi.org/10.24963/ijcai.2021/520. Main Track.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, 7 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3302. URL https://aclanthology.org/W14-3302.

Jing-Shin Chang, Yi-Chung Lin, and Keh-Yih Su. Automatic construction of a Chinese electronic dictionary. In *Third Workshop on Very Large Corpora*, 1995. URL https://aclanthology.org/W95-0109.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does BERT look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

Sanjiv R. Das, Michele Donini, Muhammad Bilal Zafar, John He, and Krishnaram Kenthapadi. Finlex: An effective use of word embeddings for financial lexicon generation. *The Journal of Finance and Data Science*, 8:1–11, 2022. ISSN 2405-9188. doi: https://doi.org/10.1016/j.jfds.2021.10.001. URL https://www.sciencedirect.com/science/article/pii/S2405918821000131.

D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 01 1977. ISSN 0010-4620. doi: 10.1093/comjnl/20.4.364. URL https://doi.org/10.1093/comjnl/20.4.364.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Philip Edmonds and Scott Cotton. SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pp. 1–5, Toulouse, France, July 2001. Association for Computational Linguistics. URL https://aclanthology.org/S01-1001.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6903–6915, Barcelona, Spain (Online), 12 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.609. URL https://www.aclweb.org/anthology/2020.coling-main.609.

Jacob Ludwig Karl Grimm and Wilhelm Karl Grimm. *Deutsches Wörterbuch*. 1852–1971.

Masato Hagiwara. *Real-World Natural Language Processing: Practical Applications with Deep Learning*. Simon and Schuster, 2021.

Xingwei He and Siu Ming Yiu. Controllable dictionary example generation: Generating example sentences for specific targeted audiences. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 610–627, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.46. URL https://aclanthology.org/2022.acl-long.46.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985. doi: https://doi.org/10.1007/BF01908075.

Vladimir Ivanov, Valery Solovyev, David Pinto, Beatriz Beltrán, and Vivek Singh. Automatic generation of a large dictionary with concreteness/abstractness ratings based on a small human dictionary. *J. Intell. Fuzzy Syst.*, 42(5):4513–4521, jan 2022. ISSN 1064-1246. doi: 10.3233/JIFS-219240. URL https://doi.org/10.3233/JIFS-219240.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, 7 2019. URL https://hal.inria.fr/hal-02131630.

Hiroyuki Kaji, Shin'ichi Tamamura, and Dashtseren Erdenebat. 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, May 2008. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/175_paper.pdf.

H. Kucera and W. N. Francis. Computational analysis of present-day american english. pp. –, 1967.

Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 241–253, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4825. URL https://aclanthology.org/W19-4825.

Craig J. A. McEwan, Iadh Ounis, and Ian Ruthven. Building bilingual dictionaries from parallel web documents. In Fabio Crestani, Mark Girolami, and Cornelis Joost van Rijsbergen (eds.), *Advances in Information Retrieval*, pp. 303–323, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45886-9. doi: https://doi.org/10.1007/3-540-45886-7_20.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: An On-line Lexical Database*. *International Journal of Lexicography*, 3 (4):235–244, 12 1990. ISSN 0950-3846. doi: 10.1093/ijl/3.4.235. URL https://doi.org/10.1093/ijl/3.4.235.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pp. 303–308, USA, 1993. Association for Computational Linguistics. ISBN 1558603247. doi: 10.3115/1075671.1075742. URL https://doi.org/10.3115/1075671.1075742.

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994. URL https://aclanthology.org/H94-1046.pdf.

Andrea Moro and Roberto Navigli. SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 288–297, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-2049. URL https://aclanthology.org/S15-2049.

Roberto Navigli, David Jurgens, and Daniele Vannella. 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)*, pp. 222–231, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/S13-2040.

Gandalf Nicolas, Xuechunzi Bai, and Susan T Fiske. Comprehensive stereotype content dictionaries using a semi-automated method. *European Journal of Social Psychology*, 51(1):178–196, 2021. doi: https://doi.org/10.1002/ejsp.2724.

Frank Nielsen. *Hierarchical Clustering*, pp. 195–211. Springer International Publishing, Cham, 2016. ISBN 978-3-319-21903-5. doi: 10.1007/978-3-319-21903-5_8. URL https://doi.org/10.1007/978-3-319-21903-5_8.

Karl Pearson. 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, 1901.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, pp. 2227–2237, New Orleans, Louisiana, 10 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pp. 87–92, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/S07-1016.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 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*, pp. 99–110, 4 2017.

William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. doi: 10.1080/01621459.1971.10482356. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

W Ren, H Zhang, and M Chen. A method of domain dictionary construction for electric vehicles disassembly. *Entropy*, 24:363, 2022.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987. ISSN 0377-0427. doi: https://doi.org/10.1016/0377-0427(87)90125-7. URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Hinrich Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998. URL https://aclanthology.org/J98-1004.

Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

Benjamin Snyder and Martha Palmer. 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*, pp. 41–43, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-0811.

Robert R Sokal and Charles D Michener. A statistical method for evaluating systematic relationships. *University of Kansas, Science Bulletin*, 38:1409–1438, 1958.

Arvi Tavast, Kristina Koppel, Margit Langemets, and Jelena Kallas. Towards the superdictionary: Layers, tools and unidirectional meaning relations. *EURALEX XIX*, 2021.

István Varga and Shoichi Yokoyama. Bilingual dictionary generation for low-resourced language pairs. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 862–870, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/D09-1090.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 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*, pp. 161–170, Erlangen, Germany, 2019. German Society for Computational Linguistics & Language Technology. URL https://konvens.org/proceedings/2019/papers/KONVENS2019_paper_43.pdf.

Ludwig Wittgenstein. *Philosophical investigations*. Basil Blackwell, 1958.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Stephan Gouws, Yoshikiyo Kato, Lukasz Kaiser, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

David Yenicelik, Florian Schmidt, and Yannic Kilcher. 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*, pp. 156–162, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.15. URL https://aclanthology.org/2020.blackboxnlp-1.15.

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.

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

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

Table 4 presents the ARIs for sense clusterings with different affinities and linkage criteria and <u>underline</u> 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 tokens with a higher disambiguity with up to 57 unique senses, whereas the other corpora only hold tokens with at most 5 unique senses.

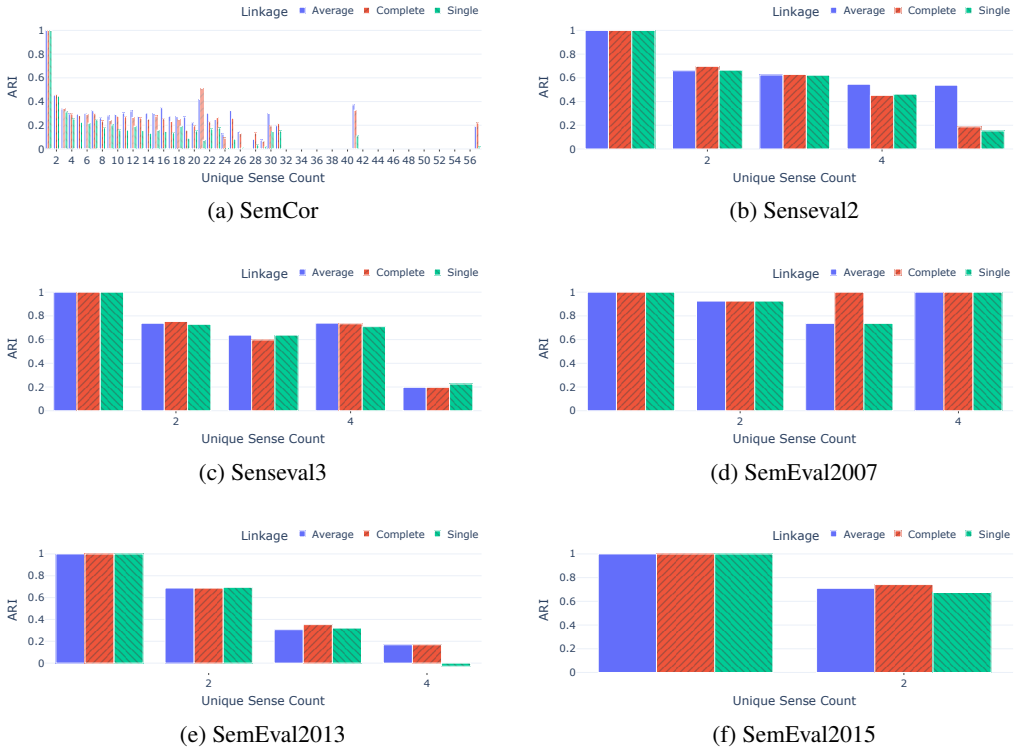
We expect marginally better results for the Euclidean distance $\mathsf{D}_{\text{Euc}}(A, B)$ with $A, B \in \mathbb{R}^d$ and $d \in \mathbb{N}_{>0}$, because the cosine distance $\mathsf{D}_{\cos}(A, B)$ is equivalent to the Euclidean distance of normalized vectors. By expansion, it holds

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

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.



(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 tokens that have the same number of true unique senses and no generated senses. We analyze the dictionaries from Section A.1.

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).

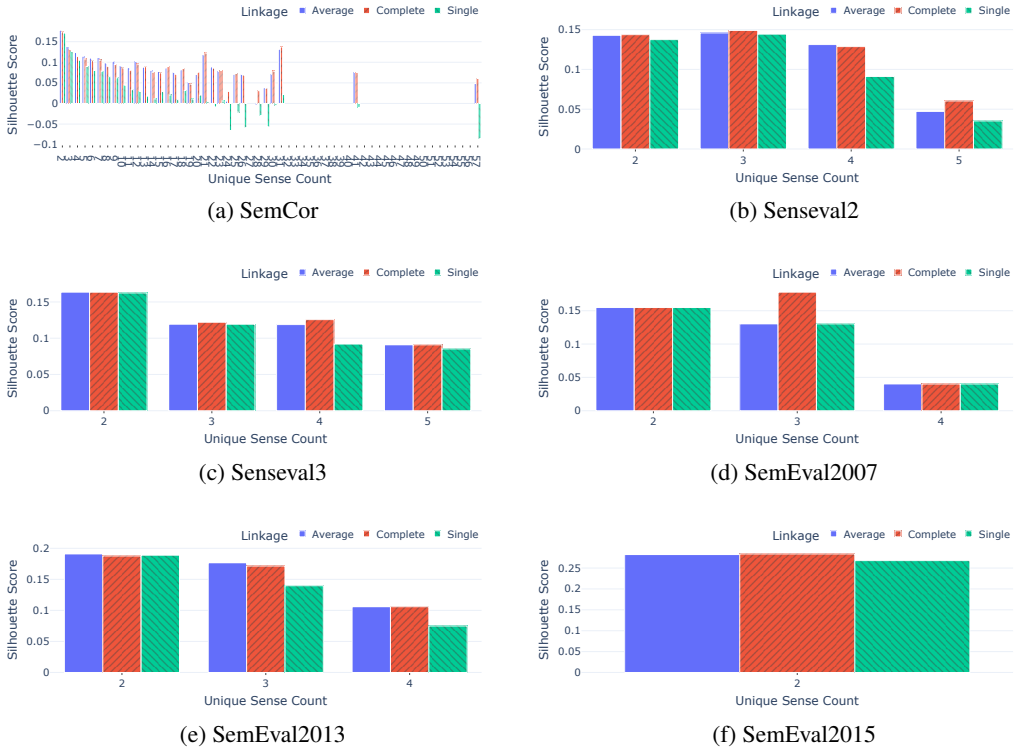
For clusterability, they apply several clustering algorithms on word vectors for sampled words from SemCor and the news.2007.corpus (Bojar et al., 2014) and measure the quality of the resulting clusters with the ARI (Rand, 1971; Hubert & Arabie, 1985) (Yenicelik et al., 2020). No clustering method was 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.

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 tokens 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 tokens, the average ARI is usually smaller but clearly positive, indicating clusterings that are better than



(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 tokens that have the same number of true unique senses and no generated senses. We analyze the dictionaries from Section A.1.*

random choice. Especially for larger corpora like SemCor or SemEval2013, the drop is more evident. Please note that the total number of tokens 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 tokens, 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 token is problematic due to the fact that we need to evaluate multiple clusterings per token 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 token. 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 tokens with a single occurrence, we do not consider any distances.

	Average Linkage		Complete Linkage		Single Linkage	
Corpus	Avg.	Std.	Avg.	Std.	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

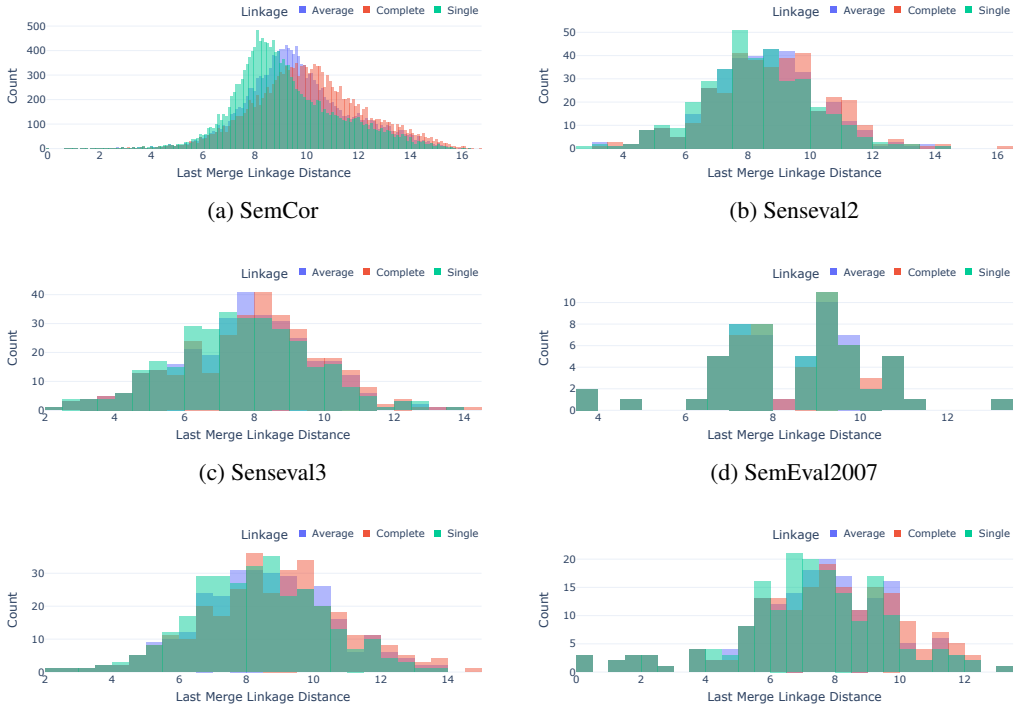
Table 5: Average and Standard Deviation of Euclidean Linkage Distances at the Last Merge using known sense counts. We analyze the dictionaries from Section A.1 and ignore tokens with generated senses or only one occurrence.

	Average Linkage		Complete Linkage		Single Linkage	
Corpus	Avg.	Std.	Avg.	Std.	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 tokens with at least two occurrences and no generated senses.

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.

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.



(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 tokens that have no generated senses. We analyze the dictionaries from Section A.1.

Corpus	Average Linkage Avg.	Average Linkage Std.	Complete Linkage Avg.	Complete Linkage Std.	Single Linkage Avg.	Single Linkage 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.

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

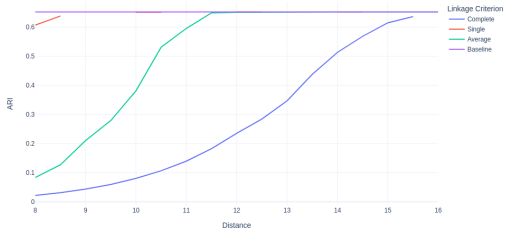
Often, terms and names consist of more than one token, for example, the "White House". We could use syntactic knowledge to find related words in the sentences. For instance, the contextualized word embedding BERT encodes some syntactic rules (Clark et al., 2019; Jawahar et al., 2019). In contrast, there 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 tokens might indicate synonyms. In contrast, word vectors that point in the opposite direction might reveal antonyms.

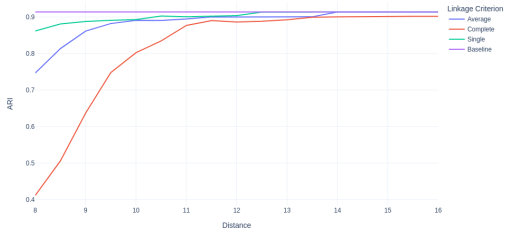
However, performing clustering methods on all tokens is more computationally expensive. Clustering the centroids of sense clusters might also reveal related tokens. In this setting, the definition of negative concepts like antonyms is less obvious. We could measure the strength of the relation between 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

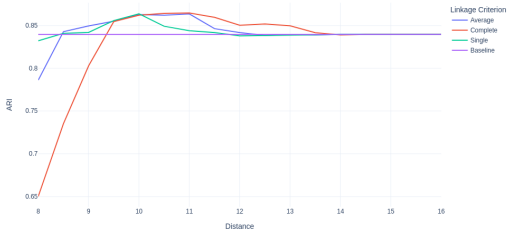
In Figure 7 we show the details for every dataset on the distribution of the ARI score next to the "Single Cluster"-baseline. As described before, for large thresholds the ARI converges to the ARI of our "Single Cluster"-baseline. For both the Senseval3 and the SemEval2007 corpus a clear peak before 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.



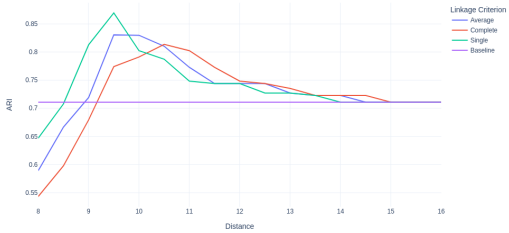
(a) SemCor



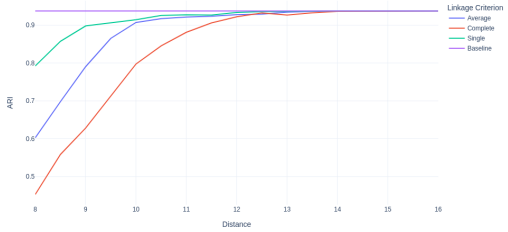
(b) Senseval2



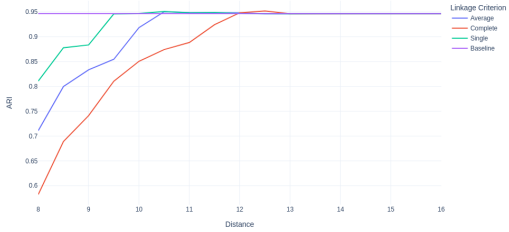
(c) Senseval3



(d) SemEval2007



(e) SemEval2013



(f) SemEval2015

Figure 7: Linkage distance thresholds per dataset

A.7 DETAILS ON CREATING SHAKESPEAREAN DICTIONARY

Hyperparameter	Phase 1	Phase 2
Learning Rate	6×10^{-3}	4×10^{-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).

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