Radboud Universiteit

Word sense visualization based on word embedding: A tool for linguistic analysis

Thesis MSc Computing Science (Data Science)

Reza Shokrzad (s1056369)

Faculty of Computing Science (Data Science)

Supervisors

Dr. Nelleke Oostdijk

Dr. Hans van Halteren

Second-reader

Prof. Dr. Martha Larson

Contents

1 Introduction **5**

2 Related Work **8**
- 2.1 Word Sense Disambiguation . . . 8
- 2.2 Word Embeddings . . . 10
  - 2.2.1 Statistical methods . . . 10
  - 2.2.2 NN-based methods . . . 11
  - 2.2.3 Transformer-based methods . . . 13
- 2.3 Word Embedding Visualization . . . 20
- 2.4 Clustering . . . 21

3 Data **24**
- 3.1 WordNet . . . 24
- 3.2 Corpora . . . 25
  - 3.2.1 Brown Corpus . . . 25
  - 3.2.2 SemCor Corpus . . . 26
  - 3.2.3 Manually Annotated Adjectives . . . 27
- 3.3 Hotel Reviews Dataset . . . 28

4 Methodology **30**
- 4.1 Word Selection . . . 30
- 4.2 Preprocessing . . . 31
  - 4.2.1 SemCor Preprocessing . . . 31
  - 4.2.2 Hotel Reviews preprocessing . . . 31
- 4.3 Embedding . . . 31
  - 4.3.1 Layer Number Analysis . . . 32
- 4.4 Visualization . . . 33
- 4.5 Clustering . . . 33
- 4.6 Evaluation . . . 34

5 Results and Discussion **37**
- 5.1 Examination of the data . . . 37
  - 5.1.1 Examination of SemCor . . . 37
  - 5.1.2 Examination of Hotel Reviews . . . 40
- 5.2 ContextLens Interface . . . 40
- 5.3 Evaluation of the Visualization . . . 41

5.3.1 Ambiguous Adjectives in SemCor . . . 42
5.3.2 Similar Adjectives in Hotel Reviews . . . 47
5.4 Clustering . . . 50
5.4.1 Layer Analysis . . . 50
5.4.2 Qualitative evaluation of clustering . . . 51
5.4.3 Quantitative evaluation of clustering with layer 9 . . . 52
5.5 Evaluation of Alternative Techniques . . . 54
5.5.1 Alternative Embedding Techniques . . . 54
5.5.2 Alternative Dimensionality Reduction Techniques . . . 56
5.5.3 Alternative Clustering Techniques . . . 56

6 Conclusion and Future work **57**
6.1 Answer to Research Questions . . . 57
6.2 Challenges and Limitations . . . 58
6.3 Future Work . . . 59
6.3.1 ContextLens Possible Application . . . 59
6.3.2 Potential Extensions . . . 59

Abstract

This thesis introduces ContextLens, an interactive tool that allows users to visualize, cluster, and explore high-dimensional contextualized word embeddings. With ContextLens, linguistic investigations based on word embeddings no longer require explicit model training. Given either a single desired word to explore its senses or multiple distinct words, users can explore possible patterns in the meanings of different words based on the context they appear. The tool lets users upload up to 200 sentences containing the target word(s) in either ".csv" or ".xlsx" format and introduces a novel approach to combining multiple clustering methods using majority voting to produce more robust and accurate annotations automatically. This work utilizes two different datasets for experimentation. The first is SemCor, a corpus that includes sense tags for a word to explore in the senses of a single word. The second is Hotel Reviews, which was scraped from Booking.com and is used to investigate differences in word embeddings when comparing words. ContextLens was evaluated using qualitative and quantitative methods, demonstrating its ability to provide valuable insights into the structure of word embeddings and identify distinct word senses. Through experimentation, combining dimensionality reduction and clustering methods led to a powerful way to understand the structure of word embeddings and identify different word senses. The tool's capabilities in visualizing embeddings, clustering, and providing user-friendly interfaces present significant opportunities for further research and applications in linguistics.

Chapter 1

Introduction

With their semantic properties, words are the primary means of conveying meaning in language [1]. Some words are *ambiguous*, i.e., have multiple meanings (senses), sometimes confusing. For instance, the word "mouse" can refer to either a small rodent or a computer device. Another example is the word "bank," which can refer to a financial institution or river's edge. Such words are also called **polysemous** [2].

On the other hand, there is synonymy. Different words can represent the same meaning. For example, "big," "large," and "giant" all refer to the same concept of size. Synonyms can have subtle differences in connotation or usage, but they often convey a similar meaning. Notably, words are synonymous in only one sense: "long" and "extended" are interchangeable when utilized in the context of extended time, but "long" cannot be used with "extended family."

Polysemy and synonymy are fundamental concepts relating to the structure and meaning of language. They can help us better understand how language works and is used in communication. Linguists study polysemy to learn about the mechanisms of meaning in language and how words convey complex concepts. Similarly, they examine synonymy to learn about word meanings' nature and how they evolve over time. **Context** influences the meaning of words, particularly in cases of polysemy and synonymy, by resolving ambiguity in the case of polysemy or determining the preferred word choice in the case of synonymy. In this study, we intend to develop a tool to help linguists in such investigations.

In ambiguous human language use, where a word may have different meanings in various contexts, a sense is a discrete representation of one of the word meanings [3]. The intended word sense is interpretable from context or any other external knowledge sources[1] [4]. Word Sense Disambiguation (WSD) is the task of automatically determining which word sense is intended on the basis of its use in a particular context [4]. As WSD is a long-lasting challenge in Natural Language Processing (NLP), many approaches, generally categorized into knowledge-based, supervised, unsupervised, and neural-based groups, have already been applied to tackle it. Some of these approaches provide easily interpretable representations, which are needed if linguists are to work with them. However, some other techniques, such as word embeddings, lack interpretability for linguistic analysis, which can be a limitation for linguists who require easily interpretable representations.

One of these representations is the *word embedding*, a neural modeling technique that rep-

[1] For example, either emotion, voice, and facial expression of the speaker or thesauri, glossaries, ontologies.

resents words as dense vectors in a high-dimensional vector space. These embeddings are learned from large amounts of text data and capture the relationships between words and their meaning in a way that is both efficient and effective. Word embeddings are also used to represent words in an efficient and effective way [3]. Although the performance of the most recently developed embedding models, which are even context-aware, is promising, they lack interpretability for linguistic analysis. To address this limitation, we introduce ContextLens, a visualization tool that aims to make high-dimensional context-based embeddings more accessible and interpretable for linguists. Descriptive linguists have traditionally sought to investigate the meaning of words in context through manual annotation. The advent of word embeddings has the potential to facilitate or even replace this process.

This study explores the viability of linking embeddings to the relevant meanings (or similar constructs) linguists understand, enabling linguistic labeling without requiring extensive manual annotation. The study is conducted for two possible types of investigations, considering either various senses of the same adjective or different but semantically related adjectives. Figure 1.1 clarifies the mentioned desire, that is, a visualization of high-dimensional embedding vectors after a dimensionality reduction technique such as UMAP [5] to be seen in 2D and 3D view.

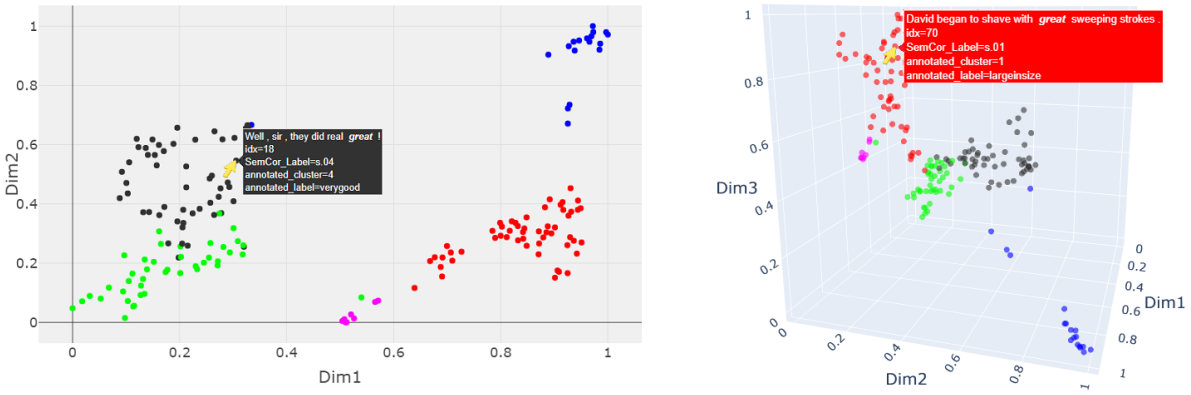


Figure 1.1: An overview of a desired 2d & 3d visualization tool. The word **great**, which has normally five distinct senses in its adjective role is illustrated in the view. By hovering on each data point, related text appears.

We choose visualization to communicate between data scientists and linguists, as it is a generic understandable language for both. We investigate if any comprehensible illustration which makes sense for linguists can be provided to connect embeddings to the related meanings. Consequently, this method offers a kind of linguistic annotation without the need for serious time investment in annotating. In summary, we attempt to answer the following research question in this thesis:

RQ. *Can we provide tools for linguistic investigations on word use, based on word embeddings, where the inquiry is to be performed by scholars with linguistic rather than deep learning training?*

To address this question in a more practical way, we subdivide it into the subquestions listed below:

- **SQ1.** Can we visualize embeddings in such a way that different "senses" are separated in

the visualization, and similar senses are grouped?

- **SQ2.** Can we cluster embeddings in such a way that the clustering corresponds to traditional "sense" groupings?
- **SQ3.** Can the tools be given a user interface that lets the scholars use the tool without further intervention from data scientists?

There has been work on diverse methods and attempts to improve contextualized embedding by fine-tuning on related domain data [6], [7], [8], to interpret the embedding vectors [9] and ways to evaluate them. However, to the best of our knowledge, there is a lack of study in automated embedding visualization for further linguistic research. In this study, we propose a visualization tool to depict different embedding algorithms color-grouped by diverse labeling systems. Figure 1.2 shows the operation of our research.

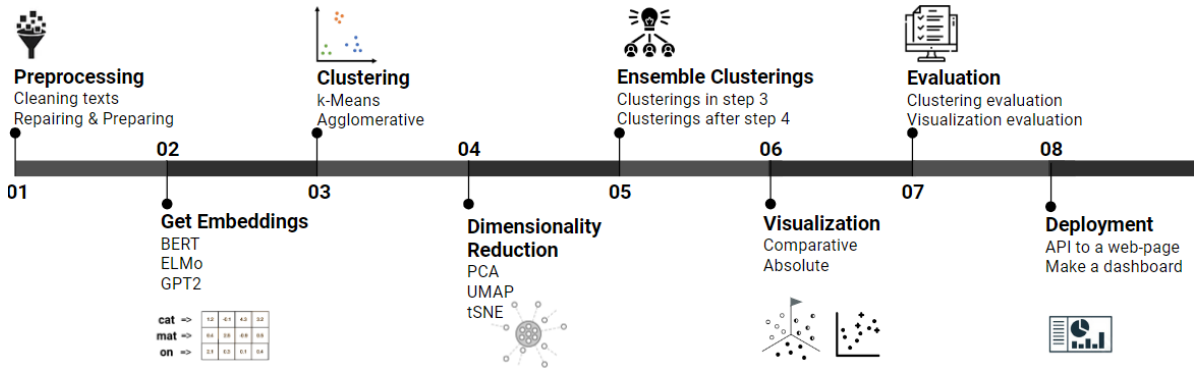


Figure 1.2: A Big Picture of Our Investigation Methodology

In the clustering step, we aim to enable linguists to explore the semantic relationships by clustering embeddings for multiple word forms specified by the linguist or, instead, senses of a single word. The resulting clusters can reveal synonyms, hypernyms, antonyms, and other semantic associations between the words, providing a deeper understanding of the meanings of individual words and their relationships. To ensure the usefulness and practicality of ContextLens, we incorporate some linguists through our experiments to gain insight into what interests them and use their feedback to improve the tool.

As there are far too many possible investigations that would be possible with such a tool, we focus on one specific use case inspired by the current work of the first supervisor. This objective concerns the use of adjectives in hotel reviews. However, it should be clear that the tool is usable for a much broader range of studies in future research.

The remainder of this thesis is organized as follows: Chapter 2, which presents a review of the related work for this thesis, sets the background for the research presented in this study. In Chapter 3, we describe the data used in detail. Then, in Chapter 4, we formalize the problem and provide implementation details for the proposed visualization tool, clustering setups, and evaluation methods. Chapter 5 presents the results of our work, including details on the experimental setup and analysis of the outcome. Following this, in Chapter 6, we conclude our findings and propose ideas for future work that can build upon the research presented in this thesis.

Chapter 2

Related Work

This chapter provides an overview of the background information relevant to our research. Specifically, we will delve into the problem of WSD in greater depth, and continue by discussing the concept of word embeddings and reviewing various embedding methods. Finally, we will also briefly review clustering methods.

2.1 Word Sense Disambiguation

This section provides an overview of the WSD development timeline (see Figure 2.1) since the primary attempt to make it an automated process. The endeavor started with the knowledge-based methods that employ external resources, such as machine-readable dictionaries, thesauri, or ontologies [4].

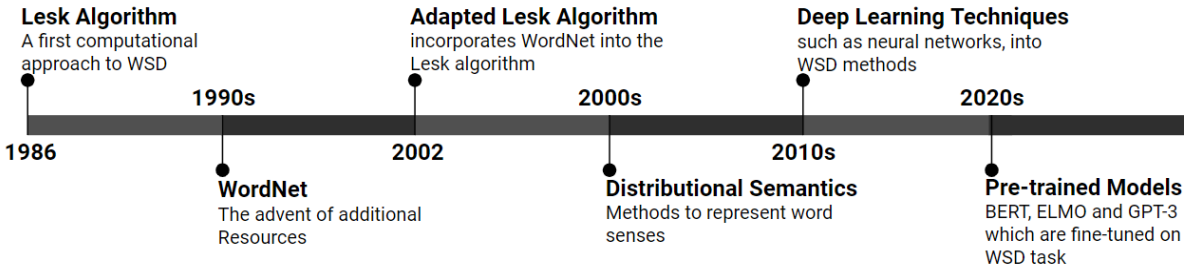


Figure 2.1: A brief timeline of the evolution of Word Sense Disambiguation (WSD) techniques, starting with the introduction of the Lesk algorithm in 1986 and progressing through the incorporation of lexical knowledge resources, the Adapted Lesk algorithm, distributional semantics methods, deep learning techniques, and the recent use of pre-trained models like BERT, ELMO, and GPT-3 for improved performance.

One of the very first endeavors to automate sense disambiguation was suggested by Lesk [10]. The idea behind Lesk's algorithm was that there is a tendency to share a common topic between a word and its given neighborhood which is a part of the text, i.e., a local context of the target word. The algorithm compares the dictionary definition of the ambiguous word with its neighbor terms. The algorithm starts with counting the number of words in both its vicinity and its senses in the dictionary. Then it selects the meaning with the most significant number of this

count. The method has some drawbacks, such as its reliance on the availability and quality of external resources, as well as its limited ability to account for context-specific senses of a word. Additionally, the algorithm may not be able to distinguish between different word senses if they share similar definitions or if the context is unclear. Since the algorithm considers the overlap merely among the glosses of the senses and dictionary glosses tend to be pretty short, there is insufficient vocabulary to relate fine-grained sense differences.

In the 1990s, researchers began integrating additional lexical resources into WSD methods. Miller et al. [11] offered a knowledge-rich approach relying on their former-proposed lexical resource, WordNet [12]. WordNet is positioned beyond typical Machine-readable dictionaries (MRDs), such as the Longman Dictionary of Contemporary English (LDOCE), because WordNet encodes a comprehensive semantic network of concepts [4]. WordNet groups English words into sets of synonyms called *synsets* and provides definitions and relationships between them. By incorporating WordNet into WSD methods, researchers aimed to improve the accuracy of WSD by providing more information about the senses of a word, such as its definition, synonyms, and antonyms. Other semantic resources, such as thesauri, ontologies, and machine-readable dictionaries, were also used to provide more information about word senses. However, these resources have limitations, including vulnerability to human errors and inconsistency, incompleteness and not being up-to-date, and being language-specific. Additionally, many WSD methods incorporating WordNet or other resources are based on manual sense annotation and may not generalize well to unseen words or new domains.

There have been many studies suggesting modifications for the Lesk WSD method based on the lexical knowledge gloss [13], [14], [15]. In 2002, Banerjee et al., [14] proposed the Adapted Lesk algorithm, which builds on the original Lesk algorithm by incorporating the WordNet lexical database into the method. The Adapted Lesk algorithm compares the definitions, synonyms, and antonyms of the ambiguous word to its local context and senses in WordNet to determine the most likely sense of the word. This approach aims to improve the accuracy of WSD by providing more information about the meanings of a word and addressing some of the limitations of the original Lesk algorithm. Although it is an improvement over the original Lesk algorithm, it still relies on external resources for its functionality. It may struggle to differentiate between similar word senses or in unclear contexts. Furthermore, its manual sense annotation leads to poor performance when applied to new words or in different domains.

Distributional semantics methods in WSD were the most dominant methods applied in the 2000s. The methods focused on using the distributional properties of words in a large corpus of text to determine their meaning. These methods rely on the idea that words that occur in similar contexts tend to have similar meanings. The most likely sense of the word was determined by analyzing the context in which a word appears. Distributional semantics methods, such as Latent Semantic Analysis (LSA) [16] and Latent Dirichlet Allocation (LDA) [17], are effective in WSD and have been used to improve the accuracy of other WSD methods, including the Lesk algorithm and the Adapted Lesk algorithm. However, distributional semantics methods have limitations, such as reliance on a large corpus and difficulty distinguishing similar word senses or unclear contexts.

In recent years, deep learning methods have been applied to WSD tasks. These methods, mostly RNN-based models such as LSTM [18], are effective in WSD by leveraging the large amounts of data available in the form of text corpora. Additionally, these methods are capable of learning distributed representations of words, which capture the meaning of a word in a

high-dimensional vector space. The ability allows them to handle polysemy and handle words in context. Furthermore, these methods can also be pre-trained on large amounts of text, which then are fine-tuned on smaller, domain-specific datasets. However, these methods require large amounts of labeled data and computational resources to train, which can be a limitation. Besides, these methods may struggle to generalize to out-of-vocabulary words or words not seen during training. Despite these limitations, deep learning methods were state-of-the-art in WSD tasks in the 2010s.

In recent years, pre-trained models have become a prominent tool in natural language processing tasks such as WSD. These models, such as ELMo [1], GPT-2 [19], and BERT [20], are pre-trained on a large corpus of text and fine-tuned on a smaller task-specific dataset. This practical idea enables the models to achieve promising performance on a wide range of NLP tasks by leveraging the knowledge learned from the large corpus of text, making them highly effective in handling complex language understanding tasks such as WSD. Additionally, these pre-trained models often incorporate word embeddings, allowing the model to capture the meaning and context of words in a more sophisticated manner, leading to improved performance in tasks such as WSD.

2.2 Word Embeddings

To make a word suitable for mathematical computations, vector embedding methods are used. In NLP, word embeddings represent words or phrases as numerical vectors in a high-dimensional space. These vectors encode the meaning and context of the words, enabling them to be used as input to machine learning algorithms. Word embeddings can be learned from large text datasets and have proven useful for a variety of language-related tasks, such as text classification, translation, and generation. Subsections 2.2.1, 2.2.2, and 2.2.3 provide an overview of various embedding methods.

2.2.1 Statistical methods

Vector Space Models (VSMs)

Vectors were a popular mathematical representation for linguistic units even before the emergence of machine learning and were first used to represent natural language by Salton and others [21]. The original use of vectors represented documents as a set of vectors based on word frequency. VSMs convert linguistic units into numerical representations or vectors that capture their meaning and context. This process involves tokenizing the text into individual words, creating a vocabulary of unique words, representing each unit as a vector containing counts of the words related to the unit. The *Term-Document Matrix* [22], and the *Word-Context Matrix* [23] are two common vector space models used to represent text data.

A Document-Term Matrix (DTM) is a mathematical representation that describes the frequency of terms in a collection of documents. It is a type of document-feature matrix where "features" can refer to various document properties apart from terms. The matrix has rows corresponding to the documents and columns representing the terms. The transpose of this matrix is known as the Term-Document Matrix, where terms are the rows and documents are

the columns. This matrix is widely used in natural language processing and computational text analysis. The value of each cell in the matrix is usually the raw count of a given term, but there are various weighting schemes, such as row normalization and tf-idf [24]. The terms in the matrix are often unigrams, which are single words separated by whitespace or punctuation. This representation is also known as a "bag of words" representation because it retains the count of individual words but not their order in the document.

Word-Context Matrices are a type of matrix representation used to encode the meaning of words in a document. Words in a document and their context are represented as rows and columns in the matrix [23]. The value at each cell corresponds to the frequency with which a context word appears in the context of the target word, i.e., this idea is similar to the statistical semantics hypothesis, which asserts that documents have a matching distribution of vocabulary usage. In this case, the assumption is that words have a similar distribution of vocabulary usage in their proximity [25].

In word-context matrices, the context window determines the breadth of the context used to define the meaning of the word and therefore affects the accuracy of the matrix representation of the meaning. Models like latent semantic analysis (LSA) [16] and latent semantic indexing (LSI) [26] are based on the idea that words that emerge in similar contexts have the same meanings. These models use the word-context matrix representation and analyze the relationships between the rows and columns to identify the underlying latent semantic structure of the document collection.

2.2.2 NN-based methods

Word2Vec

Among the global embedding methods, word2vec [27] is one of the most widespread. It estimates the meaning of words based on their occurrences in the text. More specifically, Continuous Bag Of Words (CBOW) and Skip-grams are flavors of the Word2Vec algorithm. In this approach, given a corpus, the model slides over all words of each sentence to either predict the neighbors by using the current token (Skip-gram) or guess this word by the knowledge of the context (CBOW). Although Word2Vec and other similar global embedding methods consider the meaning of a word, these models are context-free. Thus, a context-dependent model is needed to have different vectors for possibly multiple senses of a word.

GloVe

GloVe (Global Vectors) [28] is a Stanford University word embedding method used to denote semantic relationships between words in a language, generating dense vector representations of words. GloVe works by training an extensive neural network on a dataset of co-occurrence counts derived from a corpus of text. The neural network is trained to predict the co-occurrence counts of word pairs in the corpus, given the vectors of the individual words. This process results in word vectors that capture the semantic relationships between words in the corpus. GloVe combines two successful approaches: global matrix factorization and the prediction of context words by neural network approaches. Consequently, it learns word vectors across the entire corpus of text. The global perspective allows GloVe to capture the broader context and

meaning of words, which makes it particularly useful for tasks such as document classification and language translation. The Continuous Bag-Of-Words (CBOW) in the Word2Vec model achieved an accuracy of 68.4% on various word analogy tasks, but GloVe improved upon that score with 75.9% accuracy. The output of this model is two equivalent vector representation spaces for words, but these spaces are not the same due to the random initialization of neural networks. The researchers also found that summing these two vector spaces results in a vector space that is less prone to overfitting and noise [28].

FastText

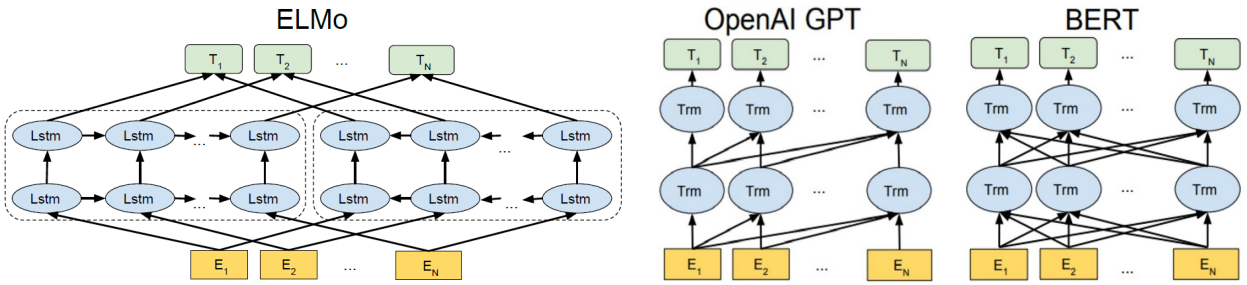
To address the issue of out-of-vocabulary words in word vectors, Bojanowski et al. present a method that improves the representation of rare or unseen words in word vector models by incorporating subword information. The FastText architecture allows accounting for subword information by using character n-grams [29].

In FastText, each word is represented not only by itself but also by a set of n-grams (sequences of n consecutive characters) that make up the word, with special boundary symbols < and > added to the beginning and end of each word. For instance, with n = 3, the word 'where' would be represented by the sequence <where> as well as the following character n-grams: <wh, whe, her, ere, re> In this process, skip-gram embedding is learned for each n-gram that makes up the word. The final representation of the word is obtained by adding together the embeddings of all of its constituent n-grams. For example, the term 'where' would be represented by the sum of the embeddings of its constituent n-grams [3].

ELMo

Embeddings from Language Models (ELMo) is a deep learning method for NLP tasks that researchers at the Allen Institute for Artificial Intelligence introduced in 2018 [1]. It is a type of word representation learned from a large text dataset and then fine-tuned for specific NLP tasks. Unlike traditional word embeddings, which are typically trained using a fixed vocabulary and are not context-dependent, ELMo embeddings are learned from the context in which words appear. This feature allows ELMo to capture more nuanced and task-specific meanings for words, which can improve the performance of NLP models on tasks such as language translation, text classification, and question answering. ELMo uses a deep bidirectional language model (BLM) to learn the embeddings, which enable it to capture contextual information from both the left and right contexts of each word that is shown in Figure 2.2a. The BLM is trained on a large dataset of text and can be fine-tuned for specific NLP tasks by adding task-specific layers on top of the pre-trained model. The word representations are generated by foremost representing the word as a sequence of character n-grams and then passing this representation through multiple layers of a deep neural network. The final representation of the word is generated by concatenating the outputs of all the layers and adding task-specific weight matrices to produce context-dependent word embeddings.

ELMo was a cutting-edge NLP model at the time of its development and has been widely utilized in both industry and academia. It has also inspired the development of other context-aware word representation methods [1]. The two-phase training process in ELMo and transformers-based models are comparable, in that they are first trained on word prediction given a large cor-



(a) The architecture of ELMo (image taken from [1]). **(b)** Working principle of GPT. **(c)** Working principle of BERT.

Figure 2.2: An overview of context-aware language models

pus of text and then fine-tuned on specific tasks. However, the way they represent words and process language makes them distinct. ELMO and transformer-based models use different approaches to represent language. ELMO uses a combination of character n-grams and context-dependent embeddings to capture the meaning of words in context. In contrast, transformer-based models use self-attention mechanisms to model long-range dependencies and learn contextual representations directly from the input. This difference is because each word is represented as a linear combination of all the words in the input sequence, with weights that are learned based on the relationships between the words in the self-attention mechanism. This allows transformer-based models to capture long-range dependencies and relationships in the input language, which can result in more powerful and nuanced word representations. In the following subsection we give a detailed description of transformers.

2.2.3 Transformer-based methods

Transformers are a type of neural network architecture that have proven to be more effective than LSTM or any other Language Models [30]. Two prevalent Transformer-based Pre-trained Language Models, BERT and GPT, have achieved state-of-the-art results in various NLP tasks, which are elaborated on in the following parts. Figure 2.3 illustrates the transformers' architecture. The attention mechanism is a key component of transformer-based models that allows the model to selectively focus on certain parts of the input sequence when making predictions, enabling the capture of long-range dependencies and contextual information in NLP tasks. This objective is achieved by allowing the model to focus selectively on certain input parts and weight them differently. By doing so, the attention mechanism enables the transformer to dynamically adjust its processing to understand the input and context better. This practical idea is why attention is often referred to as the key of transformers, as it allowed the model to achieve promising performance when it was proposed on a wide range of NLP tasks. Transformers consist of encoders and decoders, which work together to process input sequences and generate output sequences. The encoder processes the input sequence and generates a set of hidden states, which capture the contextual information from the input. The decoder then takes these hidden states and generates the output sequence, which can be a translation, a summary, or any other desired output in an autoregressive way. BERT and GPT are both transformer-based language models, but BERT uses the encoder component of transformers, and GPT employs the decoder segment.

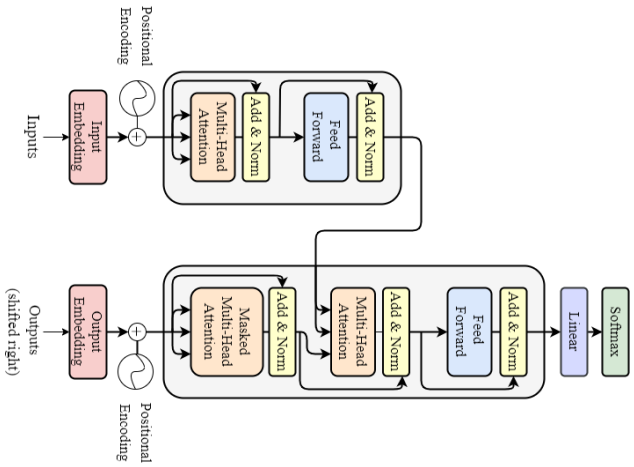


Figure 2.3: The transformers' architecture. The encoder, including a multi-headed attention block, is the top component, and the decoder, an autoregressive part with two attention blocks, is on the bottom.

In the field of context-oriented embedding techniques, there are two widely adopted strategies for leveraging pre-trained language models in downstream tasks: *feature-based* and *fine-tuning*. In the feature-based approach, words are embedded context-specific by considering additional features, such as part-of-speech (POS) tags, entity types, or attention scores, in addition to their word form. These features generate a separate representation for each word in each context, allowing the model to capture the subtle variations in meaning that arise from different contextual uses. On the other hand, fine-tuning is adapting a pre-trained language model to a specific task, such as sentiment analysis or named entity recognition, by continuing training on a smaller task-specific dataset. In fine-tuning, the model's parameters are updated to improve its performance on the specific task. Fine-tuning aims to leverage the knowledge learned from a sizeable pre-trained model and transfer it to the new task, which can be accomplished with relatively small amounts of training data.

Embeddings from Language Models (ELMo) [1] are categorized in feature-based approaches, and GPT [31] and BERT [20] are fine-tuning-based techniques.

GPT

The Generative Pre-Training (GPT) model [31] is a transformer-based language model developed by OpenAI that is capable of manipulating the semantics of words in context. GPT has been trained on an extensive collection of unstructured text data and is able to perform well on various downstream tasks. One of the advantageous features of GPT is its use of the transformer decoder, which enables it to model language as an auto-regressive model that generates the next word based on its previous context. Figure 2.2b illustrates the overall architecture of GPT.

The three models, GPT [31], GPT-2 [19], and GPT-3 [32], depict the evolution of the model and demonstrate how sophisticated these models have become. Figure 2.4 illustrates a chronological development of the model architecture. All three models use the Transformer architecture and share standard components in their structure, including multi-layer self-attention

mechanisms, fully connected feedforward layers, and pre-training on massive amounts of text data. These common elements allow the models to process sequential data effectively and generate human-like text. Each newer GPT model has introduced new capabilities and improvements over the previous model, including larger model sizes, increased training data, and new abilities for performing NLP tasks.

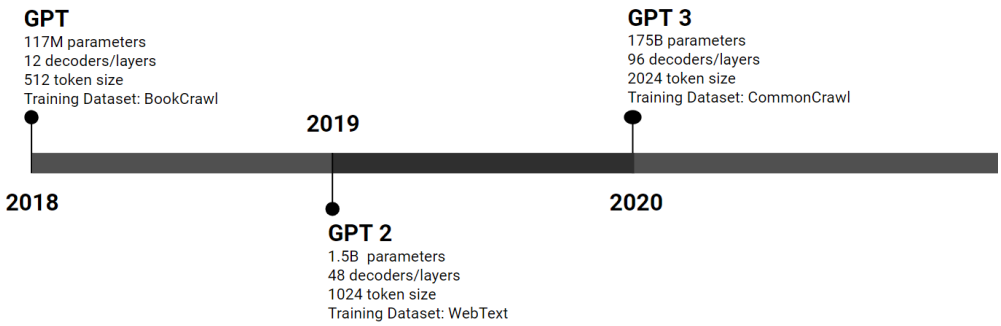


Figure 2.4: The Evolution of GPT: From GPT-1 to GPT-3. The performance of newer models in most NLP tasks, from chatbots and virtual assistants to more complex tasks such as question answering and text summarization, outweigh prior ones.

GPT-3.5[1] is the latest and improved version of the GPT-3 language model developed by OpenAI. It was launched in April 2022 and offers enhanced performance and efficiency thanks to its more extensive and diverse training dataset, advanced architecture, and training process. Meanwhile, Codex is a fine-tuned model of GPT-3 released in July 2021, which was trained on both text and code to generate programming code from natural language inputs. However, it is not part of the GPT-3.5 family of language models.

ChatGPT is a language model explicitly designed for generating responses to user queries in a conversational setting. It was released in November 2022 and is based on the GPT-3.5 architecture. Unlike GPT-3.5, ChatGPT has been fine-tuned on conversational data, which allows it to excel at developing coherent and relevant responses to a wide range of user inputs. It includes features such as context tracking and persona-based generation to enhance its performance in this specific application. Figure 2.5 illustrates the development of ChatGPT and its origins back to the initial GPT-3 model and examines the evolution of the GPT-3.5 family of language models.

The word embedding process in GPT models includes learning continuous vector representations of words during pre-training on a massive corpus of text data. The model learns to predict the next word in a sequence based on the preceding words. The learned word embeddings are then retained constant during fine-tuning on a smaller task-specific dataset, while the remaining model parameters are changed to execute a specific NLP job like sentiment analysis or machine translation. Word embeddings are critical to the model's capacity to comprehend and create human-like text.

Biases limit the GPT models in the output due to the training data, a large amount of high-quality training data requirements, difficulty interpreting their outcomes, high computational requirements, and problems in fine-tuning for specific NLP tasks. Besides, the GPT models

[1] `https://platform.openai.com/docs/model-index-for-researchers/models-referred-to-as-gpt-3-5`

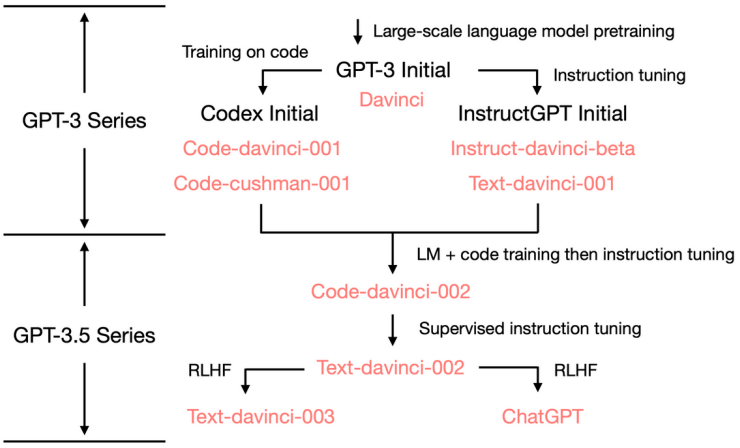


Figure 2.5: The diagram shows the GPT-3 series and the evolution of the GPT-3.5 series, which includes several variants such as Davinci, Codex, and InstructGPT. These models were trained on massive amounts of text and code data and were fine-tuned using supervised and reinforcement learning techniques. The ChatGPT model is the latest member of this family and was designed explicitly for generating responses to user queries in a conversational setting.

are left-to-right unidirectional, meaning they generate predictions based solely on the previous tokens in a sequence and do not consider future tokens that might be generated.

BERT

BERT (Bidirectional Encoder Representations from Transformers) [20] is a transformer-based model applicable to NLP tasks that researchers introduced at Google in 2018. It is designed to pre-train deep bidirectional representations from large amounts of unlabeled text data, which can then be fine-tuned for specific NLP tasks such as text classification, question answering, and language translation. One of the marked innovations of BERT is its use of self-attention mechanisms [30], which allow the model to consider the context of each input word concerning all the other words in the input sequence. The method enables BERT to capture long-range dependencies in the input text, which is essential for most NLP tasks. BERT also utilizes a multi-headed attention mechanism, allowing it to attend to multiple input tokens simultaneously. BERT has achieved state-of-the-art performance on a wide range of NLP benchmarks and has been widely adopted in industry and academia. It has also inspired the development of many other transformer-based models for NLP tasks.

There is a wide range of NLP applications that BERT performs state-of-the-art. BERT is widely used in areas such as Information Retrieval, Information Extraction (e.g., name entity recognition, event detection, and relation extraction), Text Classification, Text Generation (e.g., poetry, joke, and story), Text Summarization, Question Answering, and Machine Translation [33].

BERT-base and BERT-large are two versions of the BERT. The former has 12 layers and 110 million parameters and is trained on a dataset of approximately 3.3 billion words. The latter, with 24 layers and 340 million parameters, is trained on a dataset of roughly 16 billion tokens. Both models use a transformer-based architecture and utilize word embeddings and

self-attention mechanisms. Still, BERT-large has additional layers and a more significant parameter size, allowing it to capture the complexities of language better and leading to improved performance on many tasks.

During training, BERT represents each word as a vector in a high-dimensional space (768 in BERT-base and 1024 in BERT-large). One prominent feature of BERT is that it is *bidirectional*, meaning that it considers the context both to the left and right of each word in the input sentence. This capability allows BERT to better understand the context of words in a sentence (see Figure 2.2c). Rather than relying on the traditional left-to-right approach for predicting the next word in a sequence, BERT uses a bidirectional algorithm called Masked Language Model (MLM) that randomly masks tokens of a text. This merit allows the model to see a word itself implicitly and to predict its id based on the context, leading to a more accurate understanding of the text. BERT's input representation combines token, segmentation, and position embeddings to form the final input embeddings. The token embeddings represent each word in the input text as a vector and capture its meaning and context. The segmentation embeddings differentiate between different segments or sentences in the input, and the position embeddings encode the position of each word and help the model understand its context. These embeddings provide BERT with a rich representation of the input text, allowing it to effectively process and understand the input to make predictions or perform other natural language processing tasks.

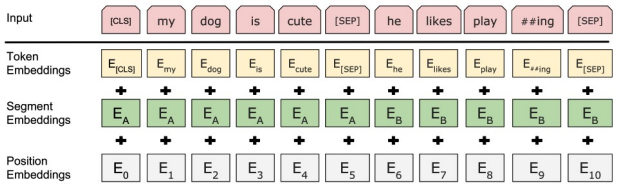


Figure 2.6: BERT embedding provides a rich representation of words and their contexts. It allows the model to effectively process and understand the sentence and use this information to make predictions or perform other NLP tasks[20].

The training process comprises two phases: an unsupervised phase that is used for pre-training and a supervised one that is used for fine-tuning [20]. Figure 2.7 shows the structure of the pre-training and fine-tuning processes in BERT, which are similar but serve different purposes. Pre-training involves training the model on a large dataset of unannotated text to predict the next word in a sequence. In contrast, fine-tuning consists of adapting the model to a specific task or dataset by training on a smaller, labeled dataset. Pre-training helps the model learn the structure and context of language, while fine-tuning further refines the model's abilities to perform specific tasks such as NER[2] and answering questions on the SQuAD[3], a popular benchmark dataset for testing and evaluating machine reading comprehension systems.

The relevant BERT variants

Pre-trained language models like BERT have shown promising performance across various NLP tasks, including WSD. However, classic BERT models do not link each embedding to a specific

[2]Name Entity Recognition
[3]Stanford Question Answering Dataset

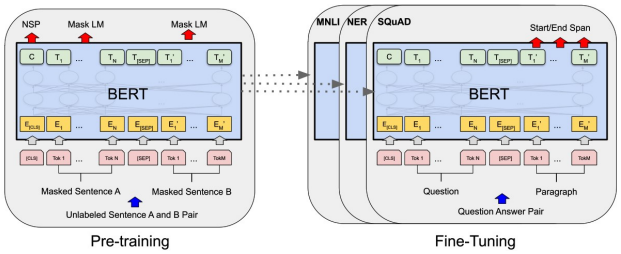


Figure 2.7: The pre-training and fine-tuning procedures for BERT involve using the same model structure, except for the output layers, and initializing the model with pre-trained parameters for downstream tasks. All parameters are fine-tuned during fine-tuning. The input to the model includes a special symbol, [CLS], at the beginning of each example, and a separator token, [SEP], to separate segments in the input. These symbols help the model process and understand the input text [20].

sense of the target word, which might limit their performance on sense-level tasks. In response, several variants of BERT have been proposed for sense-level embedding extraction.

One of these variants is SenseBERT [6], which pre-trains using a sense-tagged corpus to incorporate sense-level information, considerably improving its WSD performance. The proposed model predicts masked words and their WordNet super-senses, demonstrating promising performance in lexical-semantic understanding. The output of SenseBERT consists of the predicted word and its corresponding sense, which the model jointly predicts.

Similarly, GlossBERT [8] incorporates glosses from WordNet to enhance word representations in context. Specifically, GlossBERT pre-trains on a large unlabeled corpus and uses glosses from WordNet to augment the input representation of each word. This additional information allows GlossBERT to better capture the various senses of a word, improving its performance on sense-level tasks such as WSD. In addition, GlossBERT also employs a novel task called gloss prediction, which involves predicting the glosses of a masked word using the glosses of other words in the same sentence. This task encourages the model to learn the relationships between words and their corresponding glosses, further improving its ability to handle sense-level tasks.

Blevins et al. [7] propose a bi-encoder (context and gloss encoder) model that separately encodes the target word with its surrounding context and the dictionary definition of each sense to address the uneven distribution of word senses in WSD. The encoders are jointly optimized, allowing sense disambiguation by finding the nearest sense embedding for each target word embedding. Their system outperforms previous models, achieving a 31.1% error reduction on less frequent senses, demonstrating that rare senses can be more effectively disambiguated by modeling their definitions. The authors accomplished these results by jointly fine-tuning multiple pre-trained encoders on WSD and using a bi-encoder model built on top of BERT to improve performance on rare and zero-shot senses. Overall, their approach significantly reduces the performance gap between frequent and rare senses in WSD.

BERT geometry

Reif et al. [9] investigated the geometry of BERT to gain insights into how the model internally represents linguistic information. They found that BERT exhibits a fine-grained geometric representation of word senses and that linguistic features appear separated in semantic and syn-

tactic subspaces. Their work also presents empirical descriptions of syntactic representations in attention matrices and individual word embeddings, providing a deeper understanding of how BERT processes language. In addition, they proposed two types of interpretable visualizations for word embeddings, PCA-based and UMAP-based, that allow for a better understanding of how BERT models represent language. Their findings provide meaningful insights into the inner workings of BERT and can be leveraged to improve its performance on various NLP tasks, including WSD.

Providing more context about the abovementioned work, the authors manually annotated some words, such as *fair* meanings (see Figure 2.8), to illustrate the context-dependent nature of word senses. While their annotation process resulted in distinct clusters for the different senses, it was not automated. Additionally, they presented examples of clustering and visualization without evaluating their performance since no labels were available for evaluation. Therefore, future research could focus on developing automated methods for evaluating the performance of these visualizations and their potential applications in improving the performance of BERT on various NLP tasks.

Figure 2.8: The picture from Reif et al. [9] illustrates the context-dependent nature of word senses by manually annotating the different meanings of the word *fair*. The authors found that BERT exhibits a fine-grained geometric representation of word senses, and linguistic features appear separated in semantic and syntactic subspaces.

While Reif et al.'s [9] work on BERT embeddings has provided valuable insights into the geometry of language representations, manual annotation processes can be time-consuming and subjective. More work is needed on developing automated methods for visualizing and interpreting the structure of BERT embeddings. Automated visualization methods for large-scale BERT models can reduce the reliance on deep learning expertise for linguistic scholars, bridging the gap between data science and linguistics. By enabling faster and more efficient iteration and experimentation, these methods can facilitate the discovery of new insights into the nature of language and the structure of BERT embeddings. To address this gap, our research concentrates on investigating how embeddings can be employed to differentiate between distinct word senses and cluster them into related groups as an interpretable visualization for linguists.

2.3 Word Embedding Visualization

Human visual perception is limited to a maximum of three dimensions, making it challenging to visualize higher-dimensional data or spaces. In this regard, dimensionality reduction techniques can map high-dimensional embedding vectors in lower dimensions that are more meaningful representations that we can interpret. Word embeddings can be effectively visualized and provide insights into the underlying relationships and structure in this way. Dimensionality reduction is the process of reducing the number of features (dimensions) in a dataset while retaining as much information as possible. One of the main applications of dimensionality reduction is the ability to visualize high-dimensional data in 2D or 3D space. By projecting the data onto a lower-dimensional space, it becomes possible to create scatterplots and other visualizations that can help to understand and analyze the data.

Figure 2.9 demonstrates the utilization of dimensionality reduction techniques to map three-dimensional data onto a two-dimensional plane. As depicted in the example, the process successfully represents the data points, assuming linear separability. However, in the case of a complicated manifold of data, the method may not satisfactorily discriminate data in low dimensions because some information may be lost during the mapping process when applied to high-dimensional vectors, such as the 768-dimensional word embeddings generated by BERT. Despite this, a survey by Sorzano et al. [34] discussed numerous previous studies that demonstrated the efficacy of dimensionality reduction techniques for visualizing complex data structures in 2D and 3D.

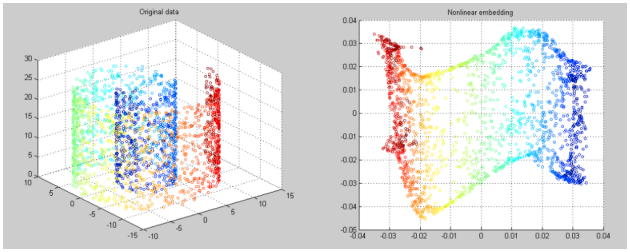


Figure 2.9: Dimensionality Reduction: Mapping 3D Data onto a 2D Plane.

This technique is especially practical when working with high-dimensional embedding such as BERT-base. Although there are numerous dimensionality reduction methods available, PCA, UMAP, and t-SNE are among the most widely used techniques in NLP research. This is because they are computationally efficient, easy to interpret, and have been demonstrated to be effective for visualizing word embeddings. As a result, many researchers in the field choose these methods to gain insight into the structure of large-scale language models such as BERT. Below is a quick review of these three methods.

- Principal Component Analysis (PCA) [35]: A technique that projects the data onto a new set of axes chosen to maximize the variance of the data on the first dimensions. To compute the principal components, PCA first standardizes the data by subtracting the mean and scaling the features to have unit variance. It then calculates the covariance matrix, which represents the correlations between the different features. Finally, it computes the eigenvectors of the covariance matrix, which are the principal components.

- Uniform Manifold Approximation and Projection (UMAP) [5]: The main idea behind UMAP is to preserve the pairwise distances between the points. To do this, UMAP first constructs a weighted graph in which the nodes represent the data points, and the edges represent the distances between the points. It then uses a technique called "stochastic gradient descent" to find a low-dimensional embedding of the data that preserves the structure of the graph as well as possible.

- t-distributed Stochastic Neighbor Embedding (t-SNE) [36]: A non-linear model that works by mapping the data points to a lower-dimensional space in a way that preserves the local structure of the data. t-SNE tries to keep nearer points in the original dimension closer together in the lower-dimensional space and creates a distance between far original points in the mapped space. t-SNE uses a Gaussian distribution to create the distance between data points in the lower-dimensional space. A conditional probability proportional to a Gaussian kernel centered on the point of interest measures the distance between two points. The variance of the Gaussian kernel is chosen so that nearby points in the high-dimensional space are still relatively close in the low-dimensional space, while distant points in the original space are farther apart in the projected space. This technique preserves the local structure of the data in the lower-dimensional space while separating distant points.

While PCA is widely used for dimensionality reduction, it assumes that the data is linearly separable and may not effectively capture non-linear relationships between data points. UMAP and t-SNE, on the other hand, are non-linear methods that are better suited for capturing complex, non-linear relationships in the data. UMAP preserves the global structure, while t-SNE is particularly good at maintaining local structure in the data. Both methods have been shown to be effective for visualizing high-dimensional data, including word embeddings. Therefore, using all three methods can provide a more comprehensive understanding of the data structure, as each method has its own strengths and limitations.

2.4 Clustering

Clustering is an unsupervised method that is supposed to group similar words together, making it easier to see patterns and relationships between words. For example, in the case of visualization of adjectives in hotel reviews, the dataset we explore in this work, we expect that clustering can help you see that words like "good," "great," and "excellent" are all grouped in the same cluster, whereas words like "bad," "terrible," and "awful" are grouped in a separate one. In the task of word embedding visualization, several clustering algorithms can be used, including k-means [37], and Agglomerative Hierarchical Clustering (AHC) [38]. While k-means is a simple and widely used algorithm for visualization purposes, we expect to find the optimal number of clusters by utilizing AHC, which employs a tree-like dendrogram to provide a visual representation of the clusters similar to how humans perceive them. DBSCAN [39] is useful when the number of clusters cannot be set manually, as it finds the number of clusters automatically.

K-means [37] is an iterative algorithm that partitions a set of n data points into k clusters in which k is specified as a hyperparameter in advance. Typically, the initial centroids are chosen randomly from the data points. The algorithm then iteratively updates the centroids by

computing the mean of all the data points assigned to each cluster and reassigning each to the category whose centroid is closest to it. This process is repeated until the centroids of the clusters stop moving, or a predetermined number of iterations is reached. K-means is popular because it is simple to understand and implement and can be used to discover patterns in large datasets quickly. It is also scalable, meaning it can handle enormous numbers of data points and clusters. Additionally, the algorithm can be used with different distance metrics, making it versatile for word embedding clustering.

Agglomerative Hierarchical Clustering (AHC) [38] is a bottom-up clustering algorithm that starts with each data point as its own cluster and iteratively merges the closest groups to form a hierarchy of clusters. The algorithm first calculates the distance between each data point pair using a metric such as Euclidean or cosine distance. Then, the closest data points are combined into a single cluster until all data points are in a single cluster or a predetermined number of clusters is reached. To represent the hierarchical structure, a dendrogram is a tree-like diagram that shows the order in which clusters were merged and the distances between linked clusters at each level. The threshold of a dendrogram is a cut-off point used to determine the number of clusters by selecting a level in the dendrogram that corresponds to a desired number of clusters. Clusters merged below this level will be considered a single cluster, while those above will be considered separate clusters. The threshold can be determined by methods like the elbow method [40] or the silhouette score [41], which helps to identify the optimal number of clusters.

In the case of using clustering algorithms for visualization, a hierarchical representation of the word relationships, with similar words grouped and displayed as branches or clusters, can be a choice. Figure 2.10 illustrates an example of this method that can be employed to (1) determine an appropriate measure of similarity for grouping the words; (2) deliver the best way to represent the hierarchy of word relationships in the visualization; (3) ensure that the algorithm accurately groups similar words rather than words that happen to be nearby in the embedding space due to other factors.

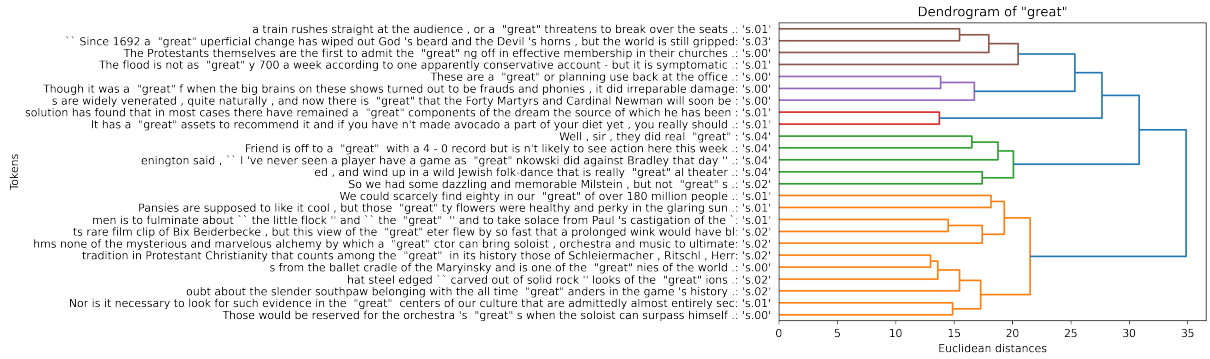


Figure 2.10: A dendrogram illustration of word 'great' for 25 senses of this word in SemCor dataset.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [39] is a density-based clustering algorithm that groups data points that are closely packed together and separate data points that are farther apart. It works by defining clusters as areas of high density separated by areas of low density. The algorithm starts by selecting an arbitrary point, called a seed point, and then finds all the data points in its neighborhood that are within a certain distance threshold, called *Eps*. These points are considered part of the same cluster. The algorithm

then proceeds to find all the points that are within *Eps* distance of any point in the cluster and adds them to the cluster as well. This process is repeated until no more points can be added to the cluster. The density-based approach of DBSCAN allows it to find clusters of any shape, it can identify noise points, and it is not sensitive to the initial conditions.

Chapter 3

Data

This chapter provides an overview of the data sources used in this thesis. The first section describes WordNet, a lexical database that provides lists of senses for words and their relations to each other. The second section introduces the Brown Corpus, a collection of texts used as the basis for SemCor, a corpus annotated with WordNet senses. The third section discusses the SemCor corpus itself, which is used for supervised sense disambiguation. The fourth section presents our manually annotated data, which is used to assess the quality of the SemCor annotations. Finally, the chapter concludes with a brief description of the Hotel Reviews dataset, utilized as a case study to evaluate our proposed approach.

3.1 WordNet

WordNet [12] is a computational lexical database[1]. It describes the semantic relations between words in more than 200 languages with brief definitions and examples. WordNet can be considered a combination of dictionary and thesaurus, which is utilizable by either web browsers for human users or in automatic text analysis and AI applications. It was created at Princeton University and focuses on linking words in terms of semantic associations, including synonyms, meronyms, and hyponyms. In WordNet, each word is assigned one or more senses, and each sense is represented as a "synset" (a set of synonyms). Each synset is a group of words that are synonyms and that are all used to express a specific concept. Therefore, concepts are encoded into synsets; for instance, each of six sense of the term *great*, in its adjective role, is categorized into six distinct synsets in WordNet. As observed in Figure 3.1, the synset denoting "major significance or importance" is composed of two constituents, *great* in its second meaning and *outstanding* in its fourth meaning.

These synsets are arranged in a graph that displays connections such as hypernym/hyponym (more or less specific), antonym, and numerous other relationships. Figure 3.1 is the definition of *great* with all label systems in the WordNet thesaurus[2]. To elaborate, the word *great* can have an adjective meaning *large in size or number*, *important*, *remarkable*, *very good*, *uppercase*, and *in an advanced stage of pregnancy*. This webpage provides extra lexical and sense information about the looked-up word via the dropdown menu "Display Options".

[1] An online thesaurus, i.e., a database that represents word senses.

[2] `http://wordnetweb.princeton.edu/perl/webwn`

The English version currently available through the Princeton website is WordNet 3.1[3] [42], holds approximately 155,000 words collected in over 117,000 synsets. The English version of WordNet comprises four databases: one for each nouns, verbs, adjectives and adverbs, with few cross-POS pointers[4]. Closed class words are not included in these databases, which do not change or add new members to a language (such as pronouns, conjunctions, and prepositions). There are a total of 117,798 nouns, 11,529 verbs, 22,479 adjectives, and 4,481 adverb, with nouns having an average of 1.23 senses, verbs having an average of 2.16 senses, adjectives 2.06 senses, and adverbs 1.59 senses.

Figure 3.1: Online WordNet thesaurus provides glosses, frequency counts, example sentences, database locations, sense keys, and sense numbers of a query.

3.2 Corpora

A corpus, or collection of text data, is an essential tool in natural language processing and computational linguistics. Corpora are used to train and evaluate models, and are used for various types of linguistic analysis [43]. Corpora such as Brown and SemCor are rich data sources for natural language processing tasks. They are essential for understanding the structure and meaning of the language. They allow researchers to analyze the frequency and distribution of words and phrases and to gain insights into the patterns and relationships that exist in language. Since this study is based on these corpora, this subsection is dedicated to their basis.

3.2.1 Brown Corpus

The Brown Corpus has been a widely-used linguistic resource consisting of a carefully selected sample of written texts representing a broad range of American English, including 15 genres. Compiled in the 1960s, the corpus has been instrumental in advancing our understanding of the structure and usage of the English language and is widely cited in linguistic research. The Brown Corpus comprises 500 text samples, each around 2,000 words in length, taken from various genres such as fiction, newspaper articles, hobbies, government, and academic writing. The corpus has a total size of 1,161,192 words. The corpus includes a total of 57,340 sentences and 56,057 unique words, or 'word types'. The number of occurrences of a particular word (type) in the corpus is known as its word (token) frequency [44]. Researchers have used the corpus to study a range of linguistic phenomena, including vocabulary, syntax, and grammatical features,

[3] WordNet 3.1. Can be accessed either online or by downloading it on a local machine.
[4] `https://wordnet.princeton.edu/`

and to develop computational models of language processing. Although with one million words extremely small compared to modern corpora, the level of annotation still makes the Brown corpus useful for specific types of research, such as ours.

3.2.2 SemCor Corpus

The sense-tagged SemCor corpus is a collection of sentences from the Brown corpus that have been annotated with semantic information. Initially this annotation was done manually using the WordNet 1.6 sense labels (for SemCor version 1.6[5]) and later automatically mapped to WordNet 3.0 (for SemCor version 3.0). Overall, SemCor is a subset of Brown corpus built on the WordNet semantic concordance [45].

The SemCor corpus contains a total of 37,176 sentences, 778,587 chunks, and 820,411 words of which more than 200,000 have been annotated for their sense. Table 3.1 presents an example of each mentioned SemCor component. An investigation of all adjectives in SemCor indicates that SemCor includes 6,031 adjective types that together occur 63,237 times in the corpus. 28,813 are not annotated with any sense tag. Thus, there are 34,424 adjectives with sense labels in the corpus.

Component	Frequency	Example
Sentences	37176	The Fulton County Grand Jury said Friday an investigation of Atlanta 's recent primary election produced " no evidence " that any irregularities took place .
Chunks	778587	['The'], ['Fulton', 'County', 'Grand', 'Jury'], ['said'], ['Friday'], ['an'], , ['investigation'], ['of'], ['Atlanta'], ["'s"], ['recent'], ['primary', 'election'], , ['produced'], ['"'], ['no'], ['evidence'], ['"'], ['that'], ['any'], ['irregularities'], ['took', 'place'], ['.']
Words	820411	'The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an' , 'investigation', 'of', 'Atlanta', "'s", 'recent', 'primary', 'election', 'produced', '"', 'no', 'evidence', '"', 'that', 'any', 'irregularities', 'took', 'place', '.'

Table 3.1: SemCor components description.

The components are either tagged with their WordNet senses, which are a representation of the synset, or may be unlabeled. Each sense label is a combination of a part-of-speech (POS) tag and a WordNet sense identifier distributed under the Princeton Wordnet License and available in the *nltk* Python package.It should be noted that the current version of the NLTK Python package is 3.7, and the WordNet version used in this thesis is 3.0. The POS tags are based on the Penn Treebank tag set [46], developed at the University of Pennsylvania and consists of approximately 4.5 million words of text. For each sentence, certain words (open-class words and multi-word expressions) and named entities (NE) are identified and labeled. Not all expressions in the sentence are labeled. It is important to note that the labeled synset groups in the text may not be consecutive (such as in the phrase "get up") and that some elements (usually multi-word expressions like "on one's feet") may not be labeled. Closed-class words like articles and prepositions are only labeled if they are part of a multi-word expression [47]. Table 3.2 provides some examples of the tagging system employed by SemCor.

[5]Developed at Princeton University

Tag System	Example
Semantic	[['The'], Tree('group.n.01', [Tree('NE', ['Fulton', 'County', 'Grand', 'Jury'])]), Tree('say.v.01', ['said']), Tree('friday.n.01', ['Friday']), ['an'], Tree('investigation.n.01', ['investigation']), ['of'], Tree('atlanta.n.01', ['Atlanta']), ["'s"], Tree('recent.s.02', ['recent']), Tree('primary$_e$*lection.n.*01', ['*primary*', '*election*']), *Tree*('*produce.v.*04', ['*produced*']), ['"'], ['*no*'], Tree('evidence.n.01', ['evidence']), ['"'], ['that'], ['any'], Tree('irregularity.n.01', ['irregularities']), Tree('take$_p$*lace.v.*01', ['*took*', '*place*']), ['.']]
POS	[Tree('DT', ['The']), Tree('NNP', ['Fulton', 'County', 'Grand', 'Jury']), Tree('VB', ['said']), Tree('NN', ['Friday']), Tree('DT', ['an']), Tree('NN', ['investigation']), Tree('IN', ['of']), Tree('NN', ['Atlanta']), Tree('POS', ["'s"]), Tree('JJ', ['recent']), Tree('NN', ['primary', 'election']), Tree('VB', ['produced']), Tree(None, ['"']), Tree('DT', ['no']), Tree('NN', ['evidence']), Tree(None, ['"']), Tree('IN', ['that']), Tree('DT', ['any']), Tree('NN', ['irregularities']), Tree('VB', ['took', 'place']), Tree(None, ['.'])]
Both	[Tree('DT', ['The']), Tree('group.n.01', [Tree('NE', [Tree('NNP', ['Fulton', 'County', 'Grand', 'Jury'])])]), Tree('say.v.01', [Tree('VB', ['said'])]), Tree('friday.n.01', [Tree('NN', ['Friday'])]), Tree('DT', ['an']), Tree('investigation.n.01', [Tree('NN', ['investigation'])]), Tree('IN', ['of']), Tree('atlanta.n.01', [Tree('NN', ['Atlanta'])]), Tree('POS', ["'s"]), Tree('recent.s.02', [Tree('JJ', ['recent'])]), Tree('primary$_e$*lection.n.*01', [*Tree*('*NN*', ['*primary*', '*election*'])]), *Tree*('*produce.v.*04', [*Tree*('*VB*', ['*produced*'])]), Tree(None, ['"']), Tree('DT', ['no']), Tree('evidence.n.01', [Tree('NN', ['evidence'])]), Tree(None, ['"']), Tree('IN', ['that']), Tree('DT', ['any']), Tree('irregularity.n.01', [Tree('NN', ['irregularities'])]), Tree('take$_p$*lace.v.*01', [*Tree*('*VB*', ['*took*', '*place*'])]), *Tree*(*None*, ['.'])]

Table 3.2: There are two types of manual tagging, namely Semantic and POS based, in SemCor. *nltk* is a Python package that provides three possible types of labels. Semantic tags consist of WordNet lemma IDs, with an additional 'NE' node if the chunk is a named entity without a specific entry in WordNet. Named entities of type 'other' do not have a lemma, while other chunks not in WordNet are not assigned a semantic tag. Punctuation tokens have a part-of-speech label 'None.'

3.2.3 Manually Annotated Adjectives

Although humans have annotated the SemCor corpus, it is known that many annotations are inaccurate, particularly for complex words with a higher entropy of sense distribution. As Bentivogli et al. [48] noted: "For example, the word 'pocket' in the sentence 'He put his hands on his pockets' can be misclassified as a pouch or enclosed space (according to the WordNet synset pouch, sac, sack, pocket), rather than the intended meaning of a small pouch in a garment, due to the lack of context information" (p.3). This example highlights the limitations of relying solely on sense inventories like WordNet for NLP tasks. Furthermore, Bentivogli et al. report that their study's subset of 4101 labels had an error rate of 2.8%, with a total of 117 errors identified. This further supports the notion that SemCor labeling may not always be accurate and reliable.

Fellbaum et al. [49], in "Performance and Confidence in a Semantic Annotation Task," discuss the challenges of sense annotation and the limitations of relying solely on sense inventories like WordNet for NLP tasks. The authors note that even human annotators can provide false sense annotations. Fellbaum et al.'s assessment included expert and non-expert annotators to measure agreement levels. Results showed 74% agreement between experiment and expert taggers and 78.6% between experiment-experiment taggers for sense annotation. Although no explicit agreement measurement is mentioned in [49], Bentivogli et al. [48] reported that Fellbaum et al. used the Dice metric. This fact suggests there is still room for improvement in sense annotation even when humans are involved in the process. Additionally, they indicate that the WordNet sense inventory may not always include the intended sense of a word in a given context, leading to inaccurate sense annotations. These limitations highlight the need for continued research and development of better methods and resources for sense annotation.

To explore the difficulty of the annotation task to observe inconsistencies, we decided to annotate the word *great* in sentences where it appears as an adjective (according to our criteria, i.e., excluding adverbial and nominalized cases) in SemCor. We chose *great* because it presents different challenging meanings. One of the supervisors manually annotated the chosen adjective in 174 sentences from SemCor, using a tagset inspired on but not strictly adhering to WordNet and not inspecting the tag in SemCor. The task was influenced by the demand to

use only five different tags, the same number found in SemCor. After annotation, we visualize the embeddings to compare the new and original labels. Table 3.3 provides an overview of the mentioned annotated labels.

Annotated Label	Example	frequency
Intense|Extensive	Since the difficulty of drawing the net is **great**, we will merely discuss it.	69
Important|remarkable	Here in 1815 the **great** nations assembled to legislate not merely for Europe, but for the world.	47
Large in size	The **great** spire shone as if the lightning had polished it.	36
Large in count	With the return of our soldiers, it soon became apparent that the belief was not shared by the **great** majority of citizens.	15
Very good	These are a **great** aid for planning use back at the office.	7

Table 3.3: Our linguists manually annotated the adjective *great* in five categories of senses. There are 174 sentences in the SemCor corpus, including *great*, which is in the role of an adjective.

3.3 Hotel Reviews Dataset

The 515K Hotel Reviews Data in Europe is an extensive customer feedback collection on European hotels. This dataset contains more than 500,000 reviews written in English, providing a wealth of information on 1493 luxurious European hotels. Each review in the dataset includes details about the hotel, such as its location, amenities, and overall experience. The dataset has been compiled from the website *www.Booking.com* by Jason Liu in 2017, which provides information about the reviewer and the review's date. The dataset is usually used for sentiment analysis, gathering customer feedback, or performing market research on the European hotel industry. The dataset is publicly available from Kaggle[6] and is provided in a CSV file containing 17 columns describing hotels and reviews details. These columns include the name, address, geographical longitude, and latitude of hotels. Besides, the date, score, and nationality are facts regarding each reviewer. Also, negative reviews, positive reviews, word count, and tags are details recorded with each review.

In this work, we concentrate on the meanings of adjectives in reviews. So, we only keep the negative and positive reviews and disregard the rest of the columns in the dataset. Table 3.4 contains relevant figures and some examples of reviews. There is a diversity of perspectives among the guests, as evidenced by the 479,792 positive and 387,848 negative reviews. Some reviewers may leave both positive and negative comments, while others might just provide one or none.

Table 3.5 shows instances in Hotel dataset with different parts of speech and senses of the word *fair*.

[6]https://www.kaggle.com/datasets/jiashenliu/515k-hotel-reviews-data-in-europe

Connotation	Review Example	Nr. Reviews	Reviews Word Count
Negative	for the rate I paid it wouldn t be fair to complain about something	387,848	9,561,499
Positive	Pleasant staff wonderful location clean and simple and fair price	479,792	9,167,995

Table 3.4: Examples of text entered in the negative and positive comment fields.

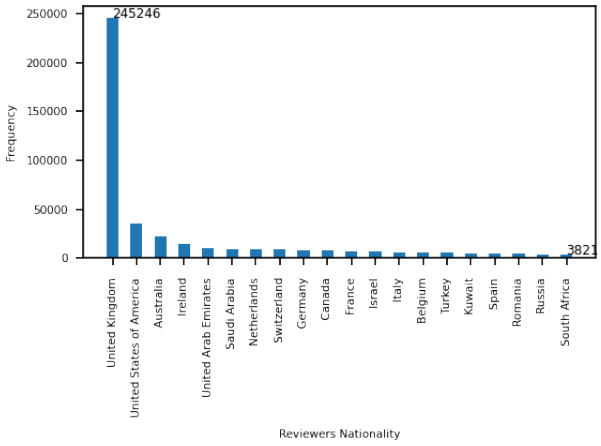


Figure 3.2: Top twenty nationalities of reviewers.

Index	Review	Connotation	POS	Sense
78,935	... The complementary minibar **fair** use was not restocked as promised ...	Negative	adj	legal
15,190	Loved the Sky bar the range of whiskys were superb the French waiter new his stuff **fair** play	Positive	adj	equitable
38,539	... The breakfast was adequate to my needs and had a **fair** share of options ...	Positive	adj	amount
297,553	The only down side was the noise from the fun **fair** even louder from our balcony in our room	Negative	noun	gathering

Table 3.5: The sentences that contain the word *fair* with different parts of speech and senses in Hotel Reviews dataset.

As our study focuses on adjectives, which require a delicate understanding of their usage, it is crucial to consult reviews written by native speakers. This reason enhances confidence in our experimental results. Thus, we examined the availability of such reviews in the hotel dataset. In this regard, Figure 3.2 illustrates the nationalities of reviewers, indicating that the United Kingdom has the most reviews, making us keep only them. This choice is particularly significant in light of the findings of Fellbaum et al.'s study [49], which suggests that accurately tagging adjectives and verbs are more challenging than tagging nouns and adverbs due to their higher degree of semantic ambiguity and context-dependent variability.

Chapter 4

Methodology

In this chapter, we dive into more details of the proposed method of automatization interpreting different intended meanings of a word to answer our research question and sub-questions. To receive feedback on the feasibility of achieving our goal and the effectiveness of our methods, we consult with linguists using plots. These plots, which are two- or three-dimensional, allow us to visually highlight the aspects of adjective use that the linguists are interested in. This initial step is crucial in determining the success of our efforts. Thus, the goal is to connect a context-based word embedding to its corresponding sense using a tailored visualization tool for linguistic studies.

In the following sections, we describe the individual steps in creating these plots. After word selection and preprocessing, we created embeddings of words using contextual networks, like ELMo, GPT, and BERT. On the basis of the embeddings we created both low-dimensional projections and clustering suggestions. These were then visualized for the linguists. All steps are automated and are intended to be - eventually - integrated into a single interactive tool.

4.1 Word Selection

To conduct the experiment on two levels, senses of a word and word types, the first step involves selecting a list of words that meet specific essential criteria. Regarding sense analysis, we have chosen words from the SemCor corpus because of the availability of sense tags. We consider these words' features, frequency, number of senses, and sense distribution to select them. Choosing words with a low frequency of both tokens and senses can result in biased visualization outcomes and poor clustering model performance. Additionally, an imbalanced distribution of senses may pose a challenge to achieving a well-performing model. Therefore, our methodology for selecting words for sense-level analysis involves considering these metrics.

To measure the spread of senses we calculate the Shannon entropy [50] of their senses in the SemCor corpus. By knowing the entropy of the sense distribution of a word, we get an indication of how the different senses are represented. Given the same number of senses, a word with a higher entropy would indicate a more uniform or evenly distributed set of senses, whereas a word with a lower entropy would mean a more concentrated or uneven distribution of senses. For example, assume a word has two senses. If one sense is hardly used at all, the entropy

would be close to zero. If the two senses are equally likely, the entropy would be one. For our experiments, equal distribution of senses would be better for visualization analysis training a clustering, so we would prefer words with higher entropy values.

Since we also explore distinct meanings for similar words, choosing a list of candidate word types exhibiting subtle nuances in meaning while remaining close in proximity is imperative. We shall delve into the Hotel review dataset to accomplish this task and filter through the abundance of words available. However, we shall not settle for just any word that crosses our path. Instead, we shall only pick words surpassing the frequency threshold of fifty, ensuring they are significant and relevant. In addition, to provide a level of ambiguity, we intentionally select sets of words that encompass both positive and negative adjectives with similar meanings.

4.2 Preprocessing

4.2.1 SemCor Preprocessing

The downloaded SemCor corpus from package nltk needs to be preprocessed. First, we converted the structure of sentences in the corpus from tagged separated tokens into standard sentences. Meanwhile, we extracted adjectives and their indices and labels in each mentioned sentence from Penn Treebank tag set.

4.2.2 Hotel Reviews preprocessing

Many reviewers are non-native English speakers (if we go by their nationality) in the Hotel dataset. The present study assumes that the reviews left by reviewers from the United Kingdom are likely to utilize words more appropriately. Using only these reviews should give a better impression of the system's functionality. Moreover, the number of available British reviews is sufficient[1] and we can afford to remove the others. We merged all UK negative and positive review comments, assuming - for now - that the senses of words are mostly independent of the review's tone.

The standard preprocessing methods, such as lowercasing, dropping stopwords and removing punctuation, may not be suitable for visualizing contextualized word embeddings since stopwords, punctuation, and capitalization can provide useful contextual information. However, the provider has already omitted the punctuation and casing in the Hotel review dataset used in this study. Despite this limitation, we have no choice but to continue with the same version.

4.3 Embedding

In general, larger BERT models have often performed better than its base model [51], [52]. However, there are exceptions to this rule; For example, BERT-base outperformed BERT-large on the subject-verb agreement [53], and sentence subject detection [54]. With this in mind, we utilized

[1] The British left approximately 250K out of 515K reviews in the Hotel dataset.

BERT-base and GPT-2 pre-trained language models for generating embeddings in this study instead of their larger and more advanced counterparts for two reasons.

Firstly, using larger models such as BERT-large or GPT-3 would require significantly higher computational resources, which is a practical factor to consider. Secondly, the use of BERT-base and GPT-2 aligns with the goals of our study, which is to develop a practical and accessible tool that researchers with limited computational resources can quickly implement. By choosing these models, we balance computational efficiency, practicality, and performance, ensuring that a wider audience can effectively use our tool.

We assess the effectiveness of the visualizations of embeddings by collaborating with our linguists and applying various metrics to measure the quality of the clustering results. In more detail, the process involves checking the visual representations with the linguists to ensure that they accurately reflect the intended information and using the metrics explained in one of the following subsections to quantify the acceptability of the clustering.

Our embedding extractor functions can take as input either preprocessed reviews from the Hotel dataset or sentences from the SemCor corpus that contain the target word. Before tokenization using the provided BERT tokenizer, the input is preprocessed and marked with the [CLS] and [SEP] tags. The tokenized input is then passed through the BERT model, which generates a set of hidden states for each word in the input. To obtain the final word embedding, we select the target word's output of a specific hidden state layer, represented as a 768-dimensional vector. ELMo and GPT-2 do not require any preprocessing steps for extracting embeddings, unlike BERT.

4.3.1 Layer Number Analysis

BERT receives its input in the form of tokens, segments, and positional embeddings at its first layer [51]. As a result, the lower layers contain the greatest amount of information about linear word order. However, as the layers progress, linear word order knowledge decreases around layer 4 in BERT-base, with increased hierarchical sentence structure simultaneously. This has been confirmed by several studies that used different tasks, datasets, and methodologies [54], [55], [56]). These findings indicate that syntactic information is most prominent in the middle layers of BERT [54] [55]. In fact, the best subject-verb agreement was found to be around layers 8-9 [53], and syntactic probing tasks showed peak performance around the middle of the model [56].

Liu et al. (2019) [51] observed that the middle layers of Transformers perform the best overall and are the most transferable across tasks. There is conflicting evidence about syntactic chunks, with Tenney et al. (2019) [57] suggesting that basic syntactic information appears earlier in the network while high-level semantic features appear in higher layers. On the other hand, Jawahar et al. (2019) [56] and Liu et al. (2019) [51] found that both POS tagging and chunking performed best at the middle layers of BERT-base and BERT-large, respectively. The final layers of BERT are concrete to the pre-training task, which explains why the middle layers are more transferable [51]. The final layers change the most during fine-tuning [58], but restoring the weights of the lower layers does not hurt model performance [59].

Tenney et al. (2019) [57] suggested that while syntactic information appears early in the model and can be localized, semantics is spread throughout the entire model. However, they found that the pattern of cumulative score gains between BERT-base and BERT-large is the

same, only more spread out in the larger model. It is important to note that Tenney et al.'s experiments focus on sentence-level semantic relations, while Cui et al. (2020) [60] found that encoding of ConceptNet semantic relations improves as the layers progress towards the top. Jawahar et al. (2019) [51] concluded that lower layers contain surface features, middle layers contain syntactic features, and higher layers contain semantic features. However, their study's only semantic task peaked at the last layer.

We intend to investigate layers of the BERT model to understand if layers 8-9 in the BERT base are the best choice for the WSD task. Our supervised evaluation metrics are explained in the following subsections. Since our study focuses on WSD, we will conduct the layer experiments on different senses of the same word in the SemCor corpus.

4.4 Visualization

As discussed in chapter 2, we cannot illustrate a high dimensional (more than three) vectors in a figure. In this case, there are various tools to reduce the vectors' dimensions into two or three. The reduction approach is such that correlation (i.e. distance) between data in reduced space and original space is preserved as much as possible.

One way to understand and interpret high-dimensional vectors is to create visualizations that show multiple data views or allow for interactive exploration. This approach could be made using scatter plots with hover-over text, parallel coordinate plots, or a combination of different plot types. Using color or other visual encodings can also help convey additional information about the vectors, such as their relationships or group membership.

We use several dimensionality reduction methods to project the high-dimensional word embedding into two dimensions. Principle Component Analysis (PCA) [61], Uniform Manifold Approximation and Projection (UMAP) [5], t-distributed stochastic neighbor embedding (t-SNE) [36] will be applied for this purpose. To assess the effectiveness of the aforementioned dimensionality reduction techniques, we visually represent selected words with diverse characteristics from the SemCor dataset.

4.5 Clustering

The idea behind applying clustering methods is that the suggested clusters can demonstrate how the embedding of various senses of the same adjective and distinct adjective types can be differentiated into separate groups. We investigate if embedding vectors generated by context-aware models can be separable into dense data groups in embedding space. Clustering can provide valuable insights into the relationships between words and the structure of word embeddings. In this work, k-means, and agglomerative hierarchical method are applied as the chosen clustering algorithms due to the promising performance and simplicity, and ability to provide a dendrogram, respectively.

Since we are using clustering as a method to identify and annotate similar senses, the steps we follow are outlined below:

- we first apply clustering on high-dimensional vectors before projection with dimensionality reduction techniques for visualization because all information is conserved in high-

dimensional embedding space, .

- we also use dimensionality reduction prior to clustering, which can help to improve the performance of the clustering algorithms and to understand the structure of the data better [62].

- finally, we add a combination clustering method in which we employ the majority voting idea among multiple clustering setup labels, including k-means and agglomerative clustering before and after projection.

The voting concept assigns each clustering method the role of an expert in annotating words with sense labels. According to the wisdom of the crowd principle, such idea leads to a more robust result [63]. Simple majority voting is used as an unsupervised ensemble method for fusing labels in the absence of ground-truth, which is common in real-world datasets such as hotel reviews. We adopt a straightforward majority voting approach to combine the labels from our clustering system in all of our experiments. Additionally, when labels are available, we use a weighted majority voting method. The mentioned weights are computed based on the alignment of the annotations with the ground-truth. The alignment process involves searching for the best label map that maximizes the matching accuracy with the actual labels. We name this function *AlignLabels* in this work.

Clustering is a valuable method for categorizing words into groups based on their resemblances. In our study, we investigate the resulting labels obtained through clustering to determine if they accurately represent different senses of the words. Particularly, in the case of the SemCor corpus, where actual senses of the words are already annotated, evaluating the effectiveness of the clustering results is possible. By comparing the subjective cluster labels to the real senses, we can assess the accuracy and usefulness of the clustering process as a practical metric. Subsection 4.4 describes the evaluation of applied metrics in detail.

4.6 Evaluation

Since our work focuses on Word Sense Disambiguation, which requires a sense-level understanding of words, and there are no downstream NLP tasks that give insight into this granularity, commonly-used extrinsic evaluation methods are not applicable to our work. Therefore, we need to apply intrinsic evaluation metrics. However, evaluating the quality of a word embedding visualization can be challenging when no labels or annotations are available, as it relies on subjective interpretation of the relationships between words in the visualization. We evaluate ContextLens using both qualitative and quantitative methods to tackle the challenge.

We analyze the structure and patterns in the visualization generated by the tool for qualitative evaluation. We pay close attention to the density, separation, overlap, and orientation of the projected words, as they are critical factors in assessing the quality of the embeddings. We examine the proximity of words expected to have a close semantic relationship and the distance between words with different meanings. We also inspect the distribution of words within the embedding space, looking for any clustering or grouping corresponding to specific semantic concepts. Additionally, we analyze the orientation of the visualization and the consistency of the word placements across multiple projections to ensure that the embeddings are stable and

reliable. Overall, the qualitative evaluation provides us with valuable insights into the quality and usefulness of both the embeddings and their visualization generated by the ContextLens tool.

Quantitative metrics will be used to evaluate the quality of clustering labels in our study. We will employ two types of metrics: supervised and unsupervised. Supervised metrics are commonly used to assess the quality of embeddings and clustering algorithms by comparing the predicted labels with the ground truth labels. However, in the case of SemCor senses visualization, the ground truth is the sense tags from SemCor, limiting the applicability of supervised metrics. Nevertheless, these metrics can be used to evaluate the quality of the embeddings extracted from different layers in one of our experiments.

On the other hand, the first metric proposed is unsupervised, meaning it does not require ground truth labels. This metric, silhouette score, measures the compactness and separation of clusters, with higher values indicating better clustering quality. Additionally, we will use the supervised metrics listed below to evaluate the quality of clustering labels. By combining quantitative and qualitative evaluation approaches, we aim to assess ContextLens' performance comprehensively.

- **Silhouette Coefficient [41]:** An unsupervised metric is often used when the number of clusters is not known a priori and needs to be chosen, as it can give insight into the structure of the data and can be used to select the optimal number of clusters. Although we preferred to set the number of clusters with the number of senses, we also attempted to find the optimum categories using the Silhouette coefficient. The metric computes the mean silhouette coefficient of all samples. This coefficient for a single point is $(b - a)/max(a, b)$, where a is the mean distance to the other samples in the same cluster and b is the mean nearest-cluster distance. The silhouette coefficient can vary between -1 and 1: a high value indicates that the sample is well-matched to its own cluster and poorly matched to neighboring groups. In contrast, a low value indicates that the point is mismatched to its own set. The silhouette coefficient is only defined for clusters with more than one sample.

- **Adjusted Rand Index (ARI) [64]:** A popular supervised metric for evaluating the similarity between two clusterings and is often used when the actual cluster assignments are known. In the case of using the SemCor sense tags as the real labels, we applied this metric. The Rand Index counts the number of pairs of data points assigned to the same or different clusters in both solutions, but it does not account for chance agreements that can occur due to random labeling. The ARI adjusts the labeling result by considering that some consensuses could be due to random chance. The method score is between -1 and 1, where a score of 1 means a perfect agreement between the two solutions, and a score of -1 indicates that they are completely different.

- **Normalized Mutual Information (NMI) [65]:** The NMI can be used as a supervised metric to compare the results of a clustering algorithm with actual labels. The NMI measures the similarity between two label sets by computing their entropy, joint entropy, and mutual information. The entropy of each label set measures its uncertainty as a random variable. The joint entropy measures the combined uncertainty of both label sets. The mutual information is calculated as the difference between the entropy of each label set and the

joint entropy. The mutual information is then normalized by dividing it by the square root of the product of the entropies of the two label sets. The resulting NMI score ranges from 0 to 1, where a score of 1 indicates a perfect agreement between the two label sets, and 0 indicates complete disagreement between them.

- **Fowlkes-Mallows index (FMI) [66]:** A metric that measures the similarity between two label sets (in our case, clustering and SemCor labels). It provides a geometric mean of precision and recall between the two sets of labels, indicating their degree of matching. If we name True Positive (TP) pairs items assigned to the same cluster in both sets of labels, False Positive (FP) pairs items assigned to the same group in the predicted set but not in the true set, and False Negative pairs items assigned to the same cluster in the actual set but not in the expected set, Precision, Recall, and FMI are calculated shown in equations (4.1) to (4.3).

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

$$FMI = \sqrt{Precision \times Recall} \tag{4.3}$$

- **Dice Coefficient [67]:** also known as the Sørensen–Dice index is another similarity measure commonly used in evaluating the performance of parallel text alignment systems. The Dice coefficient calculates the ratio of the intersection size of the two label sets to the sum of their lengths. It is expressed in Equation (4.4), where S and T are two sets of labels, and $D(S,T) = 1$ if $|S \cup T| = 0$. The Dice coefficient ranges from 0 to 1, where a score of 1 indicates a perfect match between the two label sets, and a score of 0 means no overlap between the two sets of labels. The Dice coefficient is sensitive to the size of the clusters.

$$J(S,T) = \frac{2 \times |S \cap T|}{|S| + |T|} \tag{4.4}$$

- **Accuracy:** A commonly used metric for classification that can be also used in comparing two sets of labels which are matched in terms of class names. In our case we define a label aligning function searches all permutations of clustering labels. For each permutation we calculate the accuracy metric to evaluate every single label permutation. eventually, the best permutation with highest accuracy is selected for re-indexing and aligning clustering labels. Equation (4.5) is the formula for the accuracy score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.5}$$

Chapter 5

Results and Discussion

This chapter extensively discusses the experimental findings and offers an in-depth analysis of each result. The first section centers around the comprehensive report of the data exploration, which led to identifying the fundamentals that influenced the research. Subsequently, the ContextLens view and a brief primer on its implementation are introduced. The third section evaluates ContextLens across varying levels, assessing its efficacy in handling complex data. Afterward, the chapter delves into the clustering task, which involves an analysis of the BERT's layers, experimenting with different techniques, and thoroughly evaluating the resulting clustering quality. The chapter concludes with a discussion of the results of alternative methods employed in the study, providing critical insights into their efficacy compared to the proposed approach.

5.1 Examination of the data

Upon conducting preliminary preprocessing measures, the subsequent section presents the outcomes of the study on the SemCor corpus and Hotel Reviews dataset, divided into two subsections.

5.1.1 Examination of SemCor

In this study, we utilize the latest version (3.7) of the Natural Language Toolkit (*nltk*)[1] library and make use of the SemCor corpus, which can be downloaded by executing the command `nltk.download('semcor')`. The current version of SemCor available in the *nltk* library is 3.0. Because we believe that punctuation, capital letters, and stopwords can convey important contextualized information, we decided to retain them in the SemCor corpus. As it is listed in the chapter 3, SemCor contains 6,031 adjective types with 63,237 occurrences from which 34,424 are annotated with sense labels in the corpus.

After an initial examination, we excluded multi-word adjectives (adjectives made up of multiple words separated by blank space) and hyphenated adjectives from our analysis. These adjectives can be challenging to handle at this stage of our study because they require special attention during the embedding process. Moreover, the meaning of these adjectives can be

[1] `https://www.nltk.org/news.html`

highly dependent on the specific combination of words, making it challenging to create general embeddings that can capture their full range of meanings. By excluding multi-word adjectives and focusing on (non-hyphenated) single-word adjectives that standard embedding methods can more easily handle, we aimed to simplify our analysis for the main objective of the research. Some examples of multi-word adjectives containing a blank space are "in use," "in order," "out of bounds," "long range," or "at work." Examples of hyphenated adjectives are long-term, upper-level, well-known, well-defined, cell-free, and counter-balanced. There are 331 multi-word types with a blank space and 363 with a hyphen. They account for 1,027 instances and 563 instances, respectively. There are also 79 numerals, including 16 types, e.g., "1" and "1000", that are eliminated from the list.

Consequently, 4,952 adjective types with 6,801 distinct senses and 32,755 types remained in the list of SemCor adjectives. Figure 5.1 shows an overview of these numbers. Furthermore, there are a few sentences in SemCor including more than one desired word that we decided to drop to facilitate the process.

SemCor Description	
#Words	820,411
#Chunks	778,587
#Sentence	37,176
#Adj types	6,031
#Adj tokens	63,237
#Removed Types	710
#Romoved Tokens	1,669
#Adj types after removal	4,952
#Adj tokens with sense after removal	32,755
#Distinct senses after removals	6,801

Adjective tokens Description	
#Adj tokens with sense	34,424
#Adj tokens without sense	28,813

Removals	#types	#tokens
Adj with blank space	331	1,027
Adj with hyphen	363	563
Numerical adj	16	79
Total	710	1,669

Figure 5.1: The number of items in SemCor corpus in detail.

There are 26 different annotated senses in SemCor labeled in the system from "a.00" to "a.12" and "s.00" to "s.14". The bar chart in Figure 5.2.b shows the distribution of the mentioned senses.

Figure 5.3a displays a bar chart of the top twenty adjectives in SemCor with the most distinct senses, meaning that each of these adjectives is associated with a high number of different meanings or senses in the corpus. For example, the word "open" has 15 separate adjective senses in SemCor. Moreover, Figure 5.3b displays a bar chart of the most frequent adjectives in SemCor, meaning that these adjectives occur most often in the corpus. To clarify, the bar chart indicates that for example the adjective *new* occurs 307 times in the corpus.

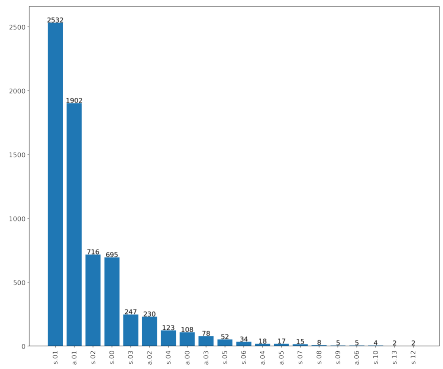
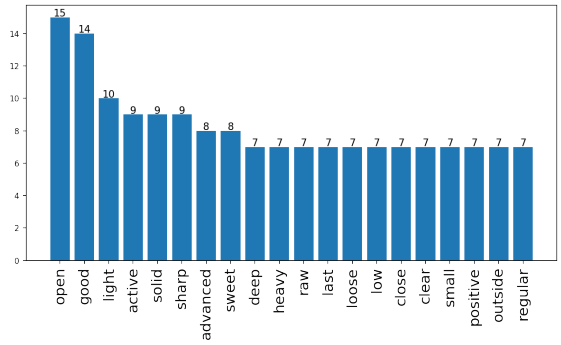
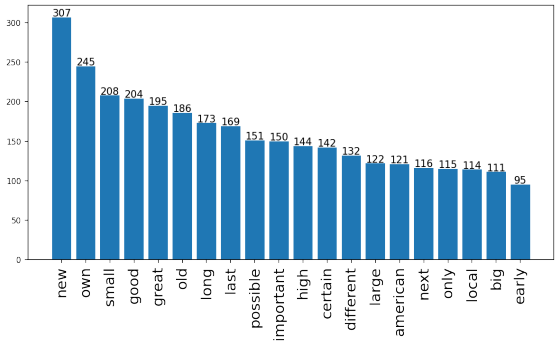


Figure 5.2: Top-twenty sense labels that are used to label adjectives in the SemCor corpus.

Detailed information about the adjectives we selected from the SemCor corpus based on the criteria explained in Section 4.1 is shown in Table 5.1. We chose adjectives with higher entropy, at least two senses, sufficiently frequent, and sense interoperability. These adjectives were used as pilots in our experiments. We present the results of some of these experiments in the following sections of this chapter.



(a) Top twenty adjectives with the highest number of distinct senses in SemCor.



(b) Top twenty most frequent adjective types in SemCor.

Figure 5.3: An initial investigation of SemCor adjectives.

Adj type	Frequency	Entropy of senses	Nr. adj senses	SemCor sense label	Frequency in corpus
domestic	30	0.87	3	a.01	19
				a.02	8
				a.03	3
cold	39	0.55	2	a.01	29
				a.02	9
				s.01	1
human	80	0.97	3	a.01	46
				a.02	19
				a.03	15
present	94	0.68	2	a.01	55
				a.02	39
high	144	0.86	6	a.00	1
				a.01	100
				a.02	34
				a.04	5
				s.03	2
				s.05	2
great	174	1.26	5	s.00	69
				s.01	61
				s.02	33
				s.03	6
				s.04	5

Table 5.1: A detailed information of pilot adjectives from SemCor corpus for the experimental analysis of visualization and clustering in this study (ascending sorted by frequency).

Accessing a sense-labeled dataset was a challenge in the investigation. SemCor was the one of the limited options available, despite being deemed not completely reliable in its tags. Although Fellbaum et al. [49] claim that the annotation of SemCor is highly reliable, they also acknowledge that the reliability of the annotation may be influenced by the semantic properties of the words being annotated. They suggest that a more fine-grained analysis of the confidence ratings of the taggers is necessary to determine which types of words posed the most difficulties for taggers and resulted in decreased certainty.

Sense tagging is a complex and labor-intensive task that requires experts. In our study, we could only deliver alternative labels for the adjective *great* due to resource constraints. To compare our labels with those provided by SemCor, we used the Dice metric, which was also uti-

lized by Fellbaum et al. [49] in their work. The Dice metric reported an agreement level of 39.66% between the two sets of labels. This result indicates that consistently assigning sense tags to intricate words remains a significant challenge that extends beyond the scope of our study. However, it is essential to note that the low score must be attributed to different choices in exactly which five sense groupings to use.

5.1.2 Examination of Hotel Reviews

To ensure an appropriate distribution of the data, we withdrew duplicate reviews from our dataset, removing 16,450 rows out of 515,738, comprising 6,159 (25,485 words) for negative reviews and 14,755 (55,462 words) for positive reviews. Most of the duplicated comment texts consisted of neutral terms such as *nothing*, *No negative*, *No positive*, and *everything*, as well as single words, such as *bed*, *staff*, *pool*, *Wifi*, and *like* which reflect hotels' advantages or drawbacks. Following withdrawing duplicates, our dataset contained 998,576 reviews, both positive and negative. We then extracted sentences for each experimental adjective in our study. Notably, some sentences had more than one desired adjective, but we opted to exclude them from our analysis, as addressing this issue fell outside the scope of our study.

Following the guidelines described in Section 4.1, one of the supervisors selected two sets of adjective types. The first set consists of four adjectives that are closely related in meaning to "cleanliness" - *clean*, *unclean*, *dirty*, and *filthy*. The second set contains five adjectives related to the "cost" sense - *cheap*, *inexpensive*, *expensive*, *costly*, and *pricey*. To examine the usage patterns of these adjectives, we collected 50 sentences for each adjective, aiming to keep the entropy of their distribution high.

5.2 ContextLens Interface

The word embedding visualization tool, ContextLens, we have developed is a comprehensive dashboard that offers users a variety of features for exploring and interpreting word embeddings. ContextLens tool is accessible with the pointer `https://contextlens.cls.ru.nl`, and additional resources, such as the codes, as well as more documentation, can be found on GitHub repository[2]. We tried to design ContextLens user-friendly mostly for linguists, interactive, and dynamic, with the ability to visualize embeddings in both 2D and 3D scatter plots.

One dashboard feature is its menu, which lets users select from different data frames and dimensionality reduction methods. The user can also choose one out of five low-dimensional embeddings for each axis and select a labeling system, which enables the user to label the data points in the plot. Hovering over each data point shows further detail, including the exact sentence the word is in, the index of the sentence in the data frame, and the label of the word. This characteristic allows users to quickly and easily identify the context in which a particular word appears and to gain a deeper understanding of the relationship between words and their surrounding context.

The word embedding visualization tool's dashboard, shown in Figure 5.4, allows users to upload a ".csv," or ".xlsx" file, including up to 200 sentences[3]. Users can specify their *desired*

[2] `https://github.com/rezashokrzad/ContextLens.git`

[3] To reduce the waiting time for embedding generation, we have imposed a limit of 200 sentences for input.

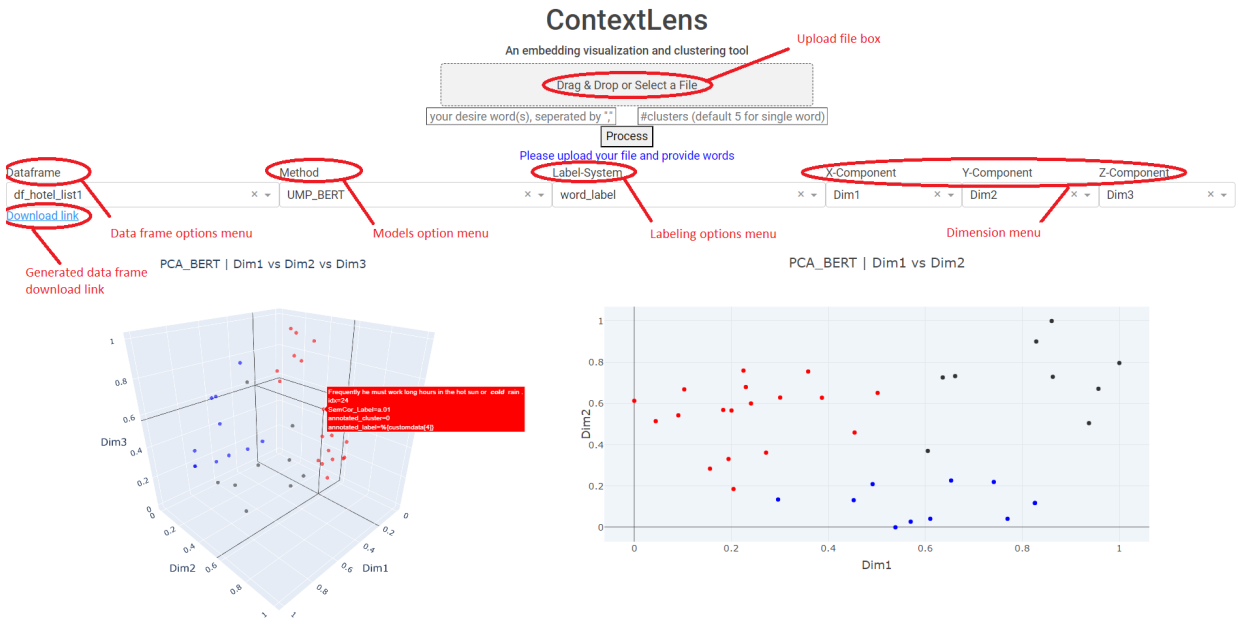


Figure 5.4: The Dashboard View of the Word Embedding Visualization Tool. The view is for the word *cold* in the SemCor corpus in which PCA and majority-voting clustering are used for the visualization method and labeling system.

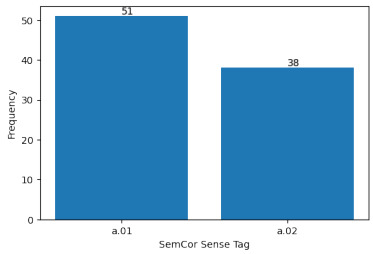
word and *the number of clusters* in two text boxes below the upload box. After clicking the "Process" button, BERT-base embeddings, two types of projections (PCA and UMAP), and combined clustering labels (simple and weighted) are generated. The dashboard's six drop-down menus provide users with a wide range of options for exploring and analyzing word embeddings. Users can select a default or the uploaded data frame, choose dimensionality reduction methods from the second left menu, and select a labeling system. The labeling system allows users to choose between their own annotations, extracted from the list of words they entered in case of a set of words or voted labels in case of sense level. The dashboard's three rightmost menus enable users to select their preferred dimensions for either PCA or UMAP embeddings, both for the 2D and 3D display, for each axis of the resulting visualization.

The user can obtain a file from the hyperlink below the data frame menu containing the columns: sentences, dimensions after reduction procedures, and annotated labels. The above features make the dashboard easy to navigate, allowing users to navigate their data and generate insightful visualizations quickly.

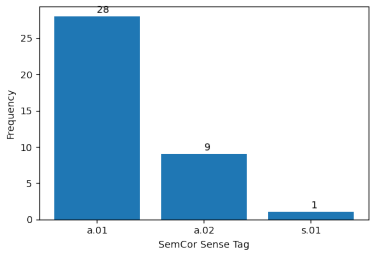
In the next sections, we present the results of tools' analysis based on both qualitative and quantitative evaluation metrics, providing insights on the structure and patterns in the visualizations, as well as the quality of the clustering labels derived from both supervised and unsupervised evaluation metrics.

5.3 Evaluation of the Visualization

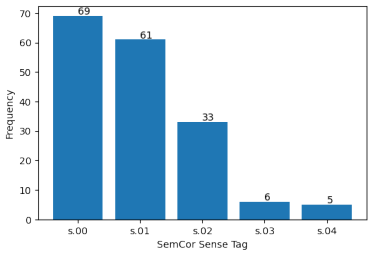
Next, we will report the analysis outcomes of the two visualization aspects in the following two subsections, based on the subjective qualitative evaluation of the tool's visualizations. These



(a) Senses distribution of the adjective *present* with senses: a.01 and a.02, with frequencies 53 and 38, respectively.



(b) Senses distribution of the adjective *cold* with senses:: a.01, a.02, and s.01, with frequencies 28, 9, and 1, respectively.



(c) Senses distribution of *great* with senses: s.00, s.01, s.02, s.03, and s.04, with frequencies 69, 61, 33, 6, and 5, respectively.

Figure 5.5: The distribution of senses for the selected words from the SemCor corpus at the sense level analysis.

Adjectives	**Meaning**	**Frequency in SemCor**	**SemCor example**
present	time	51	This is especially in evidence among the **present** generation of the suburban middle class.
	state/existence	38	In his concept there could be no one else **present**.

Table 5.2: Two distinct meanings of the word *present* as an adjective. Detailed information is from the SemCor corpus. This word with these two adjective senses has a 0.68 entropy score, which is why we chose it for some experiments.

subsections are organized into two distinct perspectives: one for exploring the senses of a single adjective and the other for investigating various adjective types.

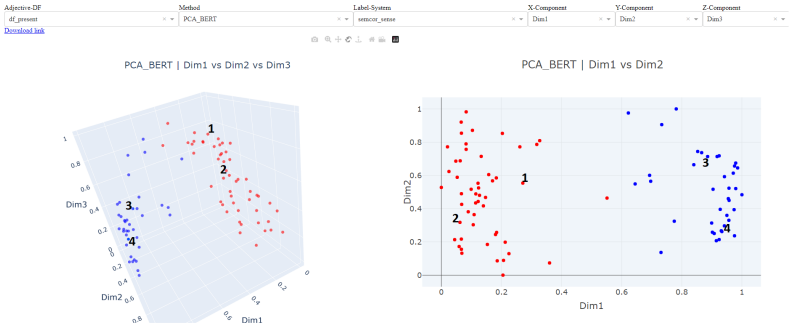
5.3.1 Ambiguous Adjectives in SemCor

For the analysis at the sense level, we selected three words, namely *present*, *cold*, and *great*, that exhibit varying numbers of senses and distributions. The distribution of senses for these three words is depicted in Figure 5.5.

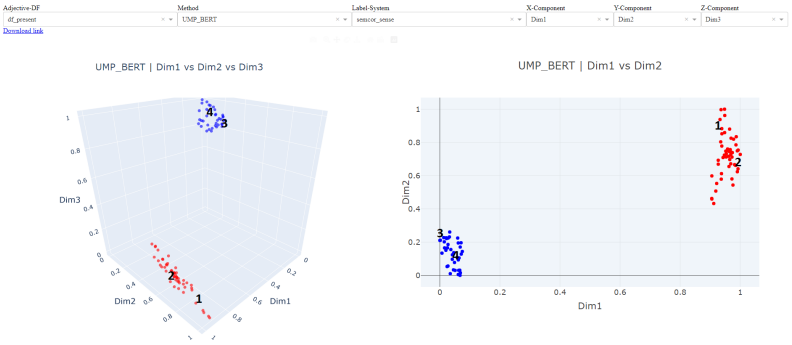
To evaluate the ability of the embedding model to distinguish between the different senses of a word, we began our experiments with the word *present*, which, when classified as an adjective, has two separate and distinct senses concerning "time" and "state/existence." For instance, Table 5.2 describes *present* in its adjective POS with distribution entropy 0.68 in the SemCor corpus.

Illustrating the word *present* with two distinct meanings in its adjective roles: related to time and related to existence. Interestingly, the SemCor labels associated with this word are sufficiently accurate, as our linguist assessed most of the sentences. After applying UMAP and PCA, both meanings of the adjective *present* were still distinguished accurately, indicating that embedding models are proficient in extracting embeddings based on context. Figure 5.6 demonstrates that the embedding model successfully separates these senses.

However, it remains unclear whether the model captures the syntactic difference between the two senses or the semantic difference, as the distinction between the two senses of *present* is not only semantic but also syntactic. Therefore, it is difficult to determine the extent to which



(a) Visualization of *present* projected by PCA



(b) Visualization of *present* projection by UMAP

Figure 5.6: The different senses of the word *present* in its adjective roles. The first sense relates to time and is exemplified by phrases such as "present technology" and "present scientific view," while the second sense relates to existence and is exemplified by phrases such as "Neither of them had been present" and "what is normally present in thyroid." The embedding model successfully distinguishes between these two senses, even when using SemCor labels. The SemCor label of the red dots are a.01 (time) and the blue ones are a.02 (state).

the model fully captures the different aspects of the word's meaning.

To elaborate, the four samples marked in the diagrams are extracted from SemCor sentences and listed in Table 5.3. In examples 1 and 2, with the time sense, *present* is used attributively, while in example 3 and 4, with the state sense, it is used predicatively.

SemCor tag	Sample Number	Marked sample
a.01	1	There have been, indeed, many important and valuable gains from the development of our *present* scientific view of the world for which we may be rightly grateful.
	2	Just as *present* technology had to await the explanations of physics, so one might expect that social invention will follow growing sociological understanding.
a.02	3	This iodoprotein does not appear to be the same as what is normally *present* in the thyroid, and there is no evidence so far that thyroglobulin can be iodinated in vitro by cell-free systems.
	4	Neither of them, I understood, had been *present* at the filming session earlier.

Table 5.3: Corresponding sentences of samples marked in the Figure 5.6 for the adjective *present*.

To gain more clarity regarding semantic differences, we moved on to the word *cold*, which exhibits more semantic variation compared to the adjective *present* and is also less bound to the syntactic construction. The results after PCA projection, shown in Figure 5.7, indicate that the sense clusters of *cold* are adjusted with its sense tags in SemCor, providing evidence that

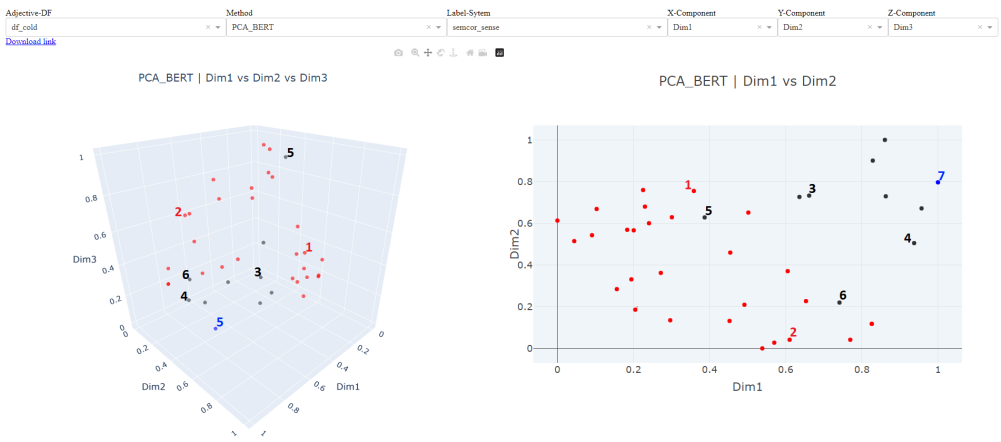


Figure 5.7: The two- and three-dimensional representations of BERT embeddings for the word *cold* obtained through PCA projection, revealing how the model separates different senses of the word while minimizing overlaps.

BERT considers semantic as well as syntactic differences when generating embeddings based on context.

The well-separated clusters in the Figure 5.7 indicates that ContextLens could distinguish the semantic senses without being distracted by the syntactic differences. The highlighted examples on the diagram are provided in Table 5.4. Samples 1 and 2 have the "temperature" sense and 3 and 4 have the "psychological" sense. The fact that 1 and 3 are "attributive" and 2 and 4 are "predicative" does not seem to be in the way. Example 7, however, is again both semantically and syntactically different, as the adjective is used adverbially. We do not have sufficient examples to judge ContextLens' behaviour in this case.

Among red dots with sense "temperature" a syntactic difference pattern can also be detected. The dot points placed tend to the bottom and right are mostly at their "predicative" meaning of cold. On the other side, top left dot points are representing cold in its "attributive" role. Regarding samples 5 and 6, we can obviously observe that sample 5 with the meaning of "temperature" is in a right-place after reducing dimension, but wrong SemCor annotation. Oppositely, sample 6 is clearly in the meaning of "psychological" that is labeled right, but placed wrong. Ignoring the first half of the sentence in sample 6 is likely to result in an ambiguous interpretation, which could potentially tend to a opposite cluster.

SemCor tag	Sample Number	Marked sample
a.01	1	He came home overheated, ran straight to the ice-chest, and gulped shivery *cold* water.
	2	The night was *cold* but the crowd kept one warm.
a.02	3	I looked unceasingly With my *cold* mind and with my burning heart.
	4	Brenner continued to smile, but his eyes were *cold*.
	5	Underneath the big one, in the silent moonlight, lay a dead pigeon, and on the smaller bell, the Clemence, two gray and white birds slept huddled together in the *cold* winter air.
	6	I told him who I was and he was quite *cold*.
s.01	**7**	Hearing his voice ring raucously up from the road, Kate would await him anxiously and watch perplexed as he walked into the house, *cold* sober.

Table 5.4: Corresponding sentences of samples marked in the Figure 5.6 for the adjective *cold*.

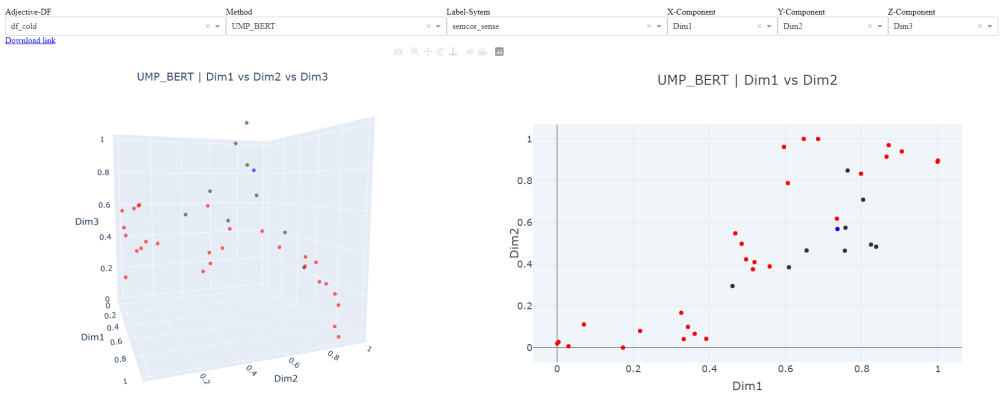


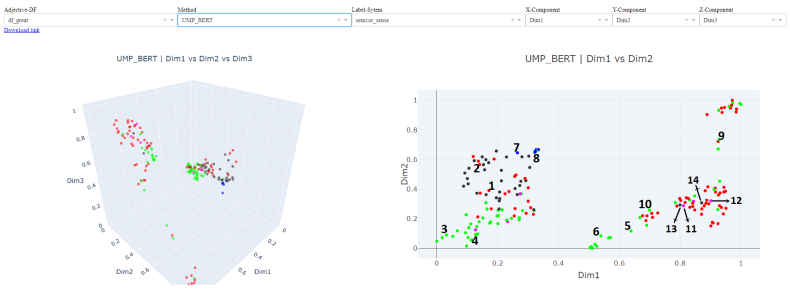
Figure 5.8: The two- and three-dimensional representations of BERT embeddings for the word *cold* after UMAP projection.

We also utilized UMAP to visualize BERT embeddings for the adjective *cold* to gain deeper insights into the relationship between context and word meaning. Figure 5.8 displays the UMAP projection of the embeddings. While UMAP may not be as effective as PCA in capturing the first two dimensions, switching the 3D view can significantly diminish this limitation as the black cluster is oriented in higher values of the z-axis. Since UMAP outperformed PCA in most cases in our experiments, we present the results of UMAP for subsequent discussions.

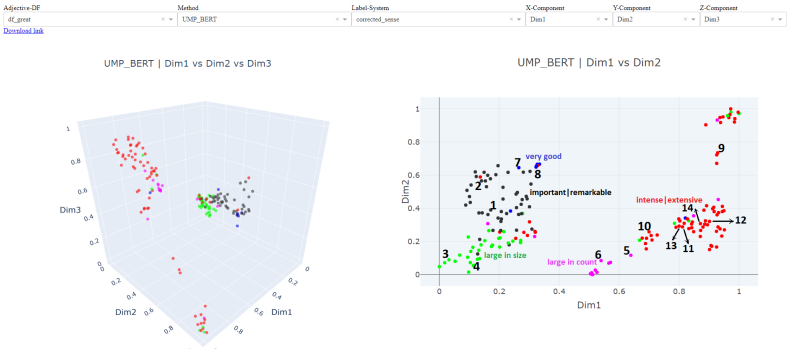
During our SemCor visualization experiments, we encountered difficulty with erroneous labeling. We noticed that adjectives with a higher number of senses, especially those with closely related senses in terms of their meanings, suffered more from the wrong annotations, which diminished their quality. Fellbaum et al.[49] also examined the dependency of lower confidence of annotation where the distribution of sense is skewed. After visualizing the annotated data using UMAP, we found that our annotations agreed with the clustering more than those provided by SemCor did. Figure 5.9a displays a 2D and 3D view of the adjective *great*, colored by SemCor labels, while Figure 5.9b is colored based on our ground truth. Apart from some overlap in the borders of clusters, the results suggest that our annotations were more informative in interpreting what the clustering appeared to do.

In case of disagreements, we attempted to investigate most of the sentences to compare the two existing annotations. Providing a part of our analysis, the sentences, including 14 samples marked in the diagrams, are listed in Table 5.5, demonstrating a difference of opinion on the tagging scheme. Admittedly, on top of the different opinions, both groups of annotators run the risk of making errors. By making this comparison, we were able to evaluate the potential of utilizing a combination of clusterings as an automated labeling process, provided that their suggested labels are better aligned with our annotations. We discuss this later in section 5.4.3.

The usage of the word *great* in samples 3 to 6 and 9 to 10 is consistently labeled as "a.01" in SemCor, while our annotator has identified distinct categories of meaning for these instances. Contrarily, SemCor proposes distinct categories for samples 9 to 14, which our annotator put under one label. These discrepancies between the two labeling systems are a major factor in the low agreement ratio observed. Case 9 furthermore demonstrated the difficulty of the task, as it has elements of both largeness and intensity.



(a) colored by SemCor labels.



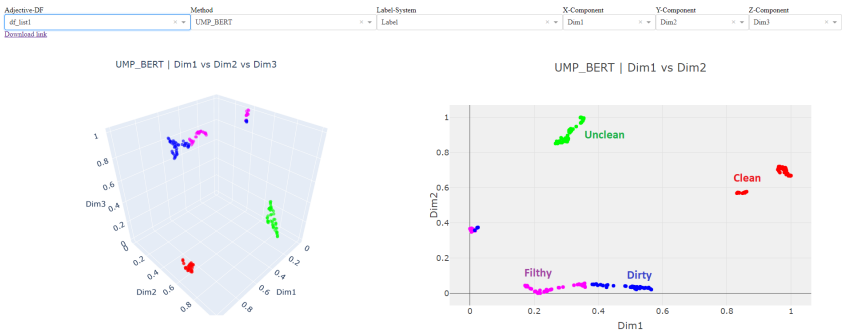
(b) colored by our own annotations.

Figure 5.9: Embedding visualizations of the adjective *great* projected by UMAP. Comparing SemCor and our labeling indicates that SemCor annotations are less reliable, particularly for adjectives with more total number and more sophisticated senses. This finding enables us to assess the annotations from the subjective clustering better.

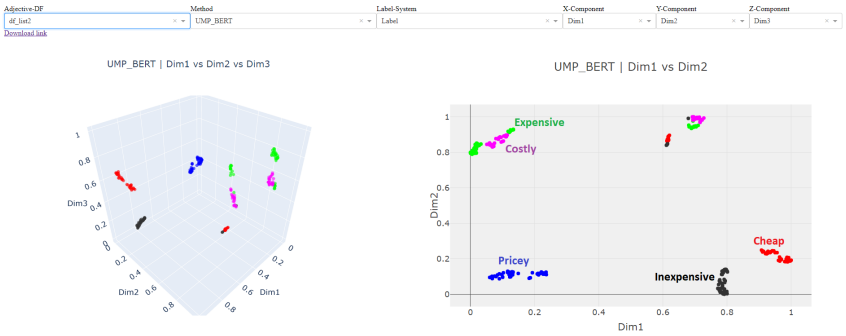
SemCor tag	Our Annotation	Sample Number	Marked sample
s.00	**Important|remarkable**	1	Thus such *great* American documentaries as The River and The Plow That Broke the Plains were composed as visual stories rather than as illustrated lectures.
		2	He could think of nothing else to tell them: no assurances, no hopeful hints at *great* discoveries that day.
s.01	**Large in size**	3	He thought of the old man laughing under the glitter of the *great* chandelier.
		4	The clouds bulged downward and burst suddenly into a *great* black funnel.
s.01	**Large in count**	5	There are a *great* number and variety of private commercial schools, trade schools and technical schools.
		6	With the return of our soldiers, it soon became apparent that the belief was not shared by the *great* majority of citizens.
s.04	**Very good**	7	LaGuardia's multi lingual rallies, when he is running for Congress, are well staged, and wind up in a wild Jewish folk-dance that is really *great* musical theater.
		8	Friend is off to a *great* start with a 4-0 record but isn't likely to see action here this week.
s.01	**Intense|extensive**	**9**	But to go from here to the belief that those more sensitive to metaphor and language will also be more sensitive to personal differences is too *great* an inferential leap.
		10	The downtown store continues to offer the *great* inducement of variety, both within its gates and across the street, where other department stores are immediately convenient for the shopper who wants to see what is available before making up her mind.
s.03	**Intense|extensive**	11	We have already witnessed *great* changes through mergers and acquisitions in the food industry at both the manufacturing and retail ends.
		12	A few of his examples are of very *great* interest, and the whole discussion of some importance for theory.
s.02	**Intense|extensive**	13	In addition, the neocortical hypothalamic relations play a *great* role in primates, as Mirsky's interesting experiment on the "communication of affect" demonstrates.
		14	No one will deny that such broad developments and transitions are of *great* intrinsic interest and the study of ideas in literature would be woefully incomplete without frequent reference to them.

Table 5.5: Corresponding sentences of samples marked in the Figure 5.12 for the adjective *great*.

Furthermore, Figure 5.9b demonstrates that the embedding model is able to effectively sep-



(a) 2D and 3D views of the words *clean*, *unclean*, *dirty*, and *filthy* extracted from the hotel reviews dataset.



(b) 2D and 3D views of the words *cheap*, *expensive*, *inexpensive*, *costly*, and *pricey* extracted from the hotel reviews dataset.

Figure 5.10: The visualizations resulting from an experiment on similar adjectives in Hotel Reviews dataset using two lists of related words, as demonstrated by the clear visual separation of related words in both 2D and 3D views.

arate the five distinct senses of the adjective *great*. However, some degree of overlap between the classes can be observed. Yet, we noticed that the "very good" cluster appears to be located in close proximity to the "important|remarkable" group, potentially due to the limited number of samples available for that class. This finding would suggest that the embedding model may not have had sufficient contextual information to distinguish between these two senses entirely or that the distinction is not as straightforward as the annotator thought.

5.3.2 Similar Adjectives in Hotel Reviews

In Figure 5.10, we present the results of our experiment on visualizing the senses of various words which are somehow related. This experiment was conducted on hotel reviews, using two lists of related words extracted from the dataset. The first list included the tokens *clean*, *unclean*, *dirty*, and *filthy*, while the second list included *cheap*, *expensive*, *inexpensive*, *costly*, and *pricey*. The resulting visualizations are presented in Figures 5.10a and 5.10b, respectively. The experiment showed that the embedding model effectively separated the different words, while also showing sense difference within each word. The results indicate that the model could differentiate between words with similar meanings, thus demonstrating its ability to capture the nuances of language. This ability to differentiate between subtle differences in meaning supports the finding that the embedding model is effective in capturing the complexities of language and its use.

It is evident that not only are the word clusters dense and away from other clusters but similar words regarding meaning or connotation are also arranged near each other. For example, in

Figure 5.10a, the cluster of *clean* is far from other words with a connotation of not being clean. Interestingly, *clean*'s cluster is placed approximately at coordinates $(1, 0.5, 0)$ in the 3D plot for the first three dimensions, indicating that this cluster is located in a totally different region than the other words.

Given their distinct meanings, we anticipated the other three words would be absolutely differentiated in the clustering process, despite their shared negative connotation of uncleanliness. To clarify the differences between them, we have provided their definitions from the Cambridge Dictionary[4] below. Please note that the samples used in Figure 5.10a visualization correspond to their first meanings of the below list.

- **Clean**
  1. free from any dirty marks, pollution, bacteria, etc.
  2. honest or fair, or showing that you have not done anything illegal.
  3. morally acceptable.
- **Unclean**
  1. not *clean* and therefore likely to cause disease.
  2. not *clean* or pure, or morally bad, as described by the rules of a religion.
- **Dirty**
  1. marked with dirt, mud, etc., or containing something such as pollution or bacteria.
  2. unfair, dishonest, or unkind.
- **Filthy**
  1. extremely or unpleasantly *dirty*.
  2. containing sexually offensive words or pictures.
  3. In sport, a filthy move or action is one that is very difficult for an opponent to defend against.

Additionally, the visualization reveals that British reviewers, who are native speakers, use these words in the specific context, as *filthy* is entirely separate from *dirty* and both are far apart from *clean*. Although their variations are oriented only in the first and most informative dimension, that is adequate to discern the distinction in their usage patterns. Particularly interesting, the distinction between *filthy*, as an extreme adjective, and *clean* is accentuated when observing the first dimension. In fact, *filthy* is positioned farther away from *clean* than *dirty* is. Like the word *clean*, the term *unclean* is distinct from the other three words in all three dimensions. Apart from the semantic usage in the context that is obvious in their definitions, this separation may be due to its morphological difference in structure as it includes the negative morpheme the prefix "un-." The method by which BERT generates tokens for "un-" and "clean" separately may potentially impact the difference in embedding results. The third potential reason for such a difference may be related to the way the word *unclean* appears in a context, which will be explained in the following paragraph.

After scrutinizing the negative words in the sentences of the given list, we have observed distinct semantic and syntactic differences among them. The term *unclean* is primarily used

[4] `https://dictionary.cambridge.org/`

to describe rooms and other spaces and is seldom paired with amplifiers or downtoners such as "very" or "somewhat." Conversely, *dirty* is more widely applicable and commonly describes objects such as cups, plates, carpets, as well as rooms and spaces. This term is often paired with amplifiers and downtoners. Finally, the word *filthy* carries an element of disgust and is typically used in contexts where human senses, such as smell and touch, are involved.

Figure 5.10b illustrates the projected and clustered embeddings of words in the second list, which consists of adjectives related to "cost", with the exception of a few samples from different clusters in the top right corner of the 2D plot, causing some overlap and ambiguity at first glance. This disorder is alleviated by switching to the 3D view where the words *cheap* and *inexpensive* are positioned at a lower value of the z-axis, representing the third dimension of UMAP. This observation highlights that the tool's availability of 2D-3D view options enhances its flexibility in facilitating linguistic interpretation. Even in the case of the two other words, *expensive* and *costly*, the former is positioned at a higher level than the latter, with the same pattern of their relationship observed in their main defined clusters.

When it comes to defining high prices, the terms *expensive*, *costly*, and *pricey* have contextual differences that distinguish them into separate clusters. Likewise, the words *cheap* and *inexpensive* have similar meanings and are placed near each other in the bottom right corner of the diagram, but they remain distinct from each other. This outcome can be attributed to their respective definitions from Cambridge Dictionary, provided below.

- **Expensive**
    1. costing a lot of money.
- **Costly**
    1. *expensive*, especially too *expensive*. (is a more formal term, often used in business or economic contexts)
    2. involving a lot of loss or damage.
- **Pricey**
    1. expensive. (is a more informal and colloquial term that also describes something high in price)
- **Cheap**
    1. costing little money or less than is usual or expected.
    2. used to describe goods that are both low in quality and low in price.
    3. unwilling to spend money.
- **Inexpensive**
    1. not costing a lot of money.

Among 50 sentences chosen from the Hotel reviews dataset for each word, four[5] and seven[6] sentences including *inexpensive* and *cheap*, respectively, are far away from their corresponding clusters and approximate each other. We scrutinized these sentences and saw that they all mention the disappointments despite the price being cheap. For example, one of the sentences

[5]The indices 121, 125, 136, 145 in list02 on the GitHub repository.
[6]The indices 13, 25, 31, 32, 36, 41, 44 in list02 on the GitHub repository.

with *inexpensive* is "*Although the hotel is inexpensive, it is not worth the price.*" Similarly, one of the sentences with *cheap* is "*The room was cheap, but it was also dirty and poorly maintained.*" Therefore, it seems that these exceptions are related to negative experiences despite the low price.

Similarly, the reverse situation is present for the outlying seven[7] and eight[8] cases of the words *expensive* and *costly*, respectively. Here, the price is high, but the room is worth it. For example, one of the sentences containing the word *expensive* is "*The hotel room was expensive, but it was worth it for the amazing view of the ocean.*" Here, the word *expensive* refers to the high price of the hotel room, but the amazing view of the ocean justifies it. Similarly, one of the sentences containing the word *costly* is "*The product is costly upfront, but it will save you money in the long run.*" In this case, the word *costly* refers to the high upfront cost of the product, but it is justified by the long-term cost savings. This finding supports the importance of considering the context and overall sentiment of the text in the interpretation of word embeddings.

Again, dimension 1 shows the main difference, high or low price. On dimension 2, *pricey* stands out as it is arranged far apart from other words. This separation might be due to the informal nature of the term *pricey*, which is less commonly used than other words in the list. It is also possible that the method used to extract the embedding of *pricey* by separately extracting the embedding of "price" and "#y" could have affected its position in the embedding space. We can conclude that ContextLens succeeds to visualize and cluster similar words given their distinct senses.

5.4 Clustering

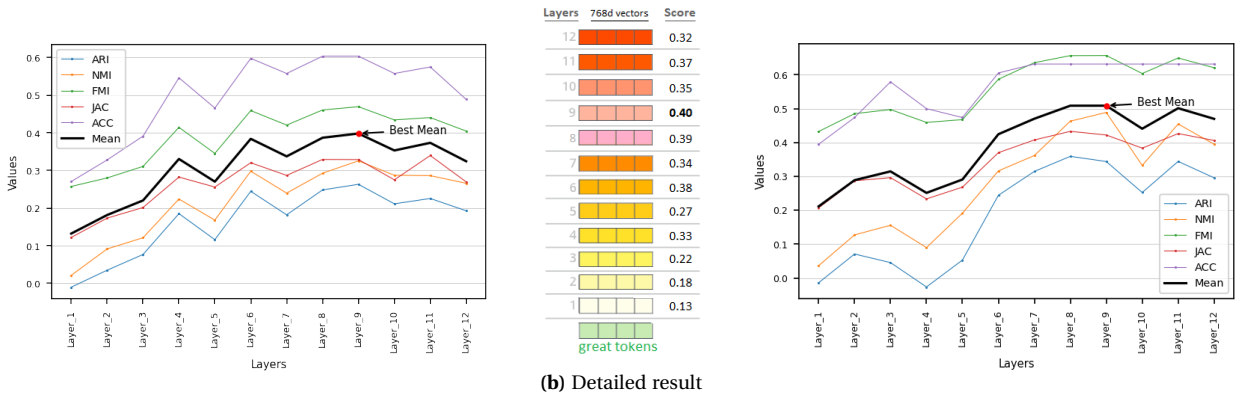
5.4.1 Layer Analysis

BERT is a transformer-based NLP architecture that uses multiple self-attention layers to process input text and generate a high-dimensional representation of the input. The dimensions of the embeddings from the middle or initial layers of BERT are set to 768, which is the number of hidden units in BERT's architecture. Therefore, we could evaluate what is discussed in section 4.3.1 by experiments on SemCor adjectives *great* (see Figures 5.11a and 5.11b) and *cold* (see Figure 5.11c) for embeddings extracted from all layers. To facilitate the process, we consider k-means as the clustering method instead of using majority voting as we assume that the result for voting and each expertise should be the same in relative terms based on our other findings.

We first extracted embedding vectors from all layers and labeled each word using a clustering method. Next, we measured the similarity between the SemCor and clustering method labels using five supervised evaluation metrics described in section 4.6. Based on the comprehensive analysis of the evaluation metrics and experimental results conducted in this study, we can conclude that layer 9 of the BERT architecture outperforms all other layers in effectively preserving the semantic meaning of the input text.

[7] The indices 53, 79, 86, 88, 93, 96, 98 in list02 on the GitHub repository.

[8] The indices 150, 153, 163, 168, 173, 175, 194, 199 in list02 on the GitHub repository.

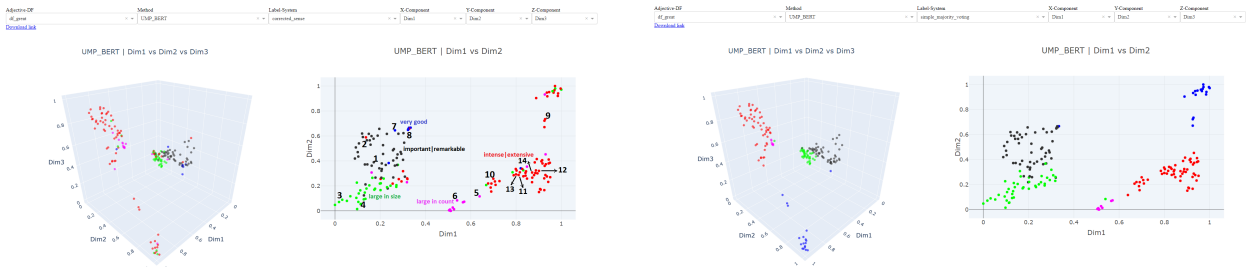


(a) use-case: the word "great"

(b) Detailed result for "great"

(c) use-case: the word "cold"

Figure 5.11: Comparison of sense labels in SemCor and k-means on embedding shows the performance of each layer in the BERT base model. The mean of five metrics, explained in section 4.6, indicates that layer nine outperforms the other layers.



(a) The color of the clusters corresponds to our annotation. This plot is the same view as Figure 5.9b provided for easier comparison.

(b) cThe color of the clusters corresponds to our majority voting method.

Figure 5.12: The majority voting method was used to generate annotations shown in this diagram for the adjective *great*.

5.4.2 Qualitative evaluation of clustering

A comparison between the labeling systems depicted in Figures 5.12a and 5.12b suggests that the majority voting labeling method may serve as a reliable alternative to actual annotation, as it corresponds well with the - at least our - manual annotation. However, some inaccuracies in the clustering of the *blue* cluster are noticed, indicating that the clustering model may not have performed as efficiently in this area, likely due to the constraint of forcing to group the data into exactly five clusters.

The challenge of forcing the model to cluster data into an inappropriate number of categories is also evident in the *cold* visualization shown in Figure 5.13. To clarify, the word *cold* has three meanings, but one of them has a single example in SemCor, resulting in a bimodal distribution. Despite two main categories and only one sample, the clustering models attempt to generate three balanced. An interesting finding is that the clustering models separate the red cluster in Figure 5.7 (with SemCor tag "a.01") into two parts, "predicative" (blue cluster) and "attributive" (red cluster), explained in more detail in Section 5.3.1. Therefore, the clustering is meaningful but partly at the syntactic and not the semantic level. If a linguist uses the tool, ContextLens, can recognize this issue and combines the relevant clusters, then the resulting clustering will be very similar to the one obtained using SemCor.

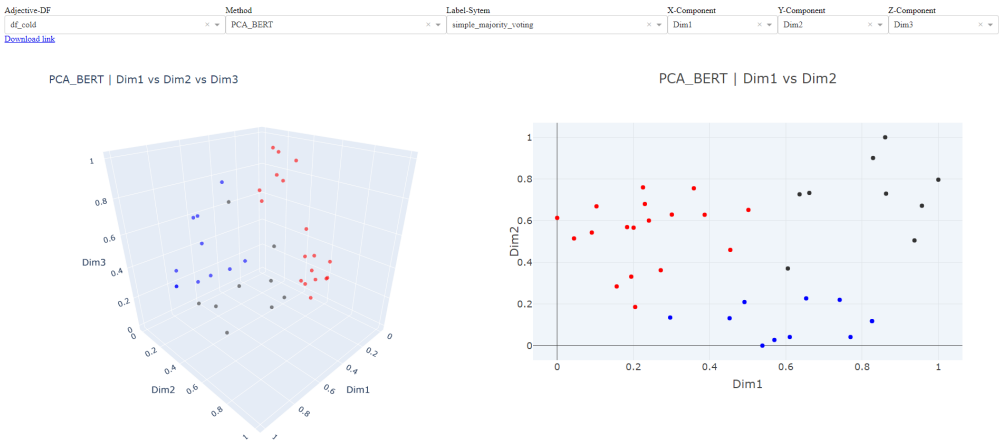


Figure 5.13: A challenge in majority voting idea for *cold* with three senses, but bimodal distribution.

5.4.3 Quantitative evaluation of clustering with layer 9

From the visual inspection of manual and majority voting labels for *great*, we suggested that the automatic annotation was viable. In this section, we investigate this suggestion quantitatively. We evaluated this in two stages. First, we used the unsupervised Silhouette metric to assess the separation of clusters. Second, we applied the five supervised metrics listed in section 4.6 to evaluate the correspondence between the annotations provided by majority voting clustering to the annotations in SemCor and our own. To evaluate the model's performance for ambiguous words, we chose adjectives *great* and *present*. Additionally, two lists of words, discussed in subsection 5.3, were used to evaluate the model's performance for different but related words.

The mean Silhouette scores obtained from ten runs of different stochastic methods, as shown in Table 5.6, consistently exhibit low values, with only minor variations. In cases where the clusters are imbalanced in size or the embedding features have high dimensionality, it is not uncommon to observe low Silhouette scores even if the clusters appear well-separated visually. Considering that the Silhouette scores do not offer conclusive evidence, we have decided to withhold judgment until we obtain results from the supervised measurements. For the details of the word lists used in the evaluation process, see the Appendix.

Clustering Model	great	present	cold	list1	list2
K-means	$0.105_{\pm 0.005}$	$0.260_{\pm 0.007}$	$0.102_{\pm 0.009}$	$0.240_{\pm 0.002}$	$0.205_{\pm 0.002}$
Agglomerative	$0.109_{\pm 0.008}$	$0.260_{\pm 0.009}$	$0.098_{\pm 0.003}$	$0.240_{\pm 0.001}$	$0.189_{\pm 0.003}$
Simple Voting	$0.108_{\pm 0.003}$	$0.260_{\pm 0.001}$	$0.102_{\pm 0.005}$	$0.218_{\pm 0.004}$	$0.204_{\pm 0.004}$
Weighted Voting	$0.108_{\pm 0.004}$	$0.260_{\pm 0.007}$	$0.104_{\pm 0.005}$	$0.218_{\pm 0.004}$	$0.204_{\pm 0.005}$

Table 5.6: This table presents the Silhouette clustering results to investigate the quality of clustering methods (See the Appendix for a detailed list of the words used in the evaluation).

As the voting approach is proposed as a substitute for sense tags, we need data where manually placed labels are presently available. Accordingly, we selected the adjective *great*, which has two existing label sets, SemCor tags, and our annotations, in addition to the adjectives *present* and *cold*, for which we have labels in SemCor. The results of the experiments are presented in

Table 5.7, where all evaluation metrics, introduced in subsection 4.6, are supervised due to the presence of labels. The result of supervised metrics is after running ten times of all clustering methods with a negligible variation.

Sense Labels	Clustering Model	ARI	NMI	FMI	DIC	ACC	mean
present-SemCor-labeled	K-means	0.955	0.921	0.978	0.988	0.989	0.966
	Agglomerative	1.000	1.000	1.000	1.000	1.000	**1.000**
	Simple Voting	1.000	1.000	1.000	1.000	1.000	**1.000**
	Weighted Voting	1.000	1.000	1.000	1.000	1.000	**1.000**
cold-SemCor-labeled	K-means	0.322	0.420	0.593	0.638	0.657	0.526
	Agglomerative	0.334	0.366	0.599	0.670	0.697	0.533
	Simple Voting	0.380	0.463	0.602	0.667	0.694	**0.561**
	Weighted Voting	0.372	0.460	0.599	0.659	0.686	0.555
great-SemCor-labeled	K-means	0.231	0.294	0.451	0.305	0.574	0.371
	Agglomerative	0.172	0.253	0.417	0.268	0.553	0.333
	Simple Voting	0.249	0.339	0.464	0.569	0.596	**0.443**
	Weighted Voting	0.249	0.339	0.464	0.569	0.596	**0.443**
great-own-labeled	K-means	0.493	0.502	0.625	0.730	0.742	0.618
	Agglomerative	0.442	0.454	0.602	0.712	0.714	0.585
	Simple Voting	0.519	0.506	0.633	0.678	0.739	0.615
	Weighted Voting	0.538	0.526	0.639	0.718	0.748	**0.634**

Table 5.7: This table presents the results of four experiments evaluating the proposed voting approach as a substitute for sense tags. All evaluation metrics used in the table are supervised due to the availability of labels.

Accordingly, it is unfortunate that the current evaluation methodology for SemCor, which relies solely on an overall Dice score, proposed by Fellbaum [49], is limited in its ability to provide a detailed assessment of the performance of individual words or word classes. In particular, the outcome for the adjective *present* is expected to receive a perfect score, just like human perception. However, for the adjective *cold*, which has three distinct senses, its score is anticipated to be somewhat lower than the overall score of 78.6%. While a score of 0.659 may initially appear too low, it is still promising. Furthermore, the adjective *great*, which possesses five partially related senses, is expected to receive an even lower score. However, a score of 0.569, compared with the Fellbaum scores, is nonetheless a promising outcome.

In light of Kilgariff's demand [68] for raw accuracy in the range of $80-90\%$, it is evident that current evaluation scores for SemCor fall well below this threshold. It remains unclear, however, whether Kilgariff's demand pertains to all words within a given text or solely to the ambiguous ones. Our evaluation has focused explicitly on challenging cases, which may partially account for the relatively low scores achieved.

We also evaluated the voting method's performance by measuring how well the clustering separated the different words from the Hotel Reviews. We used the same five supervised metrics for two lists of words itemized in the appendix. Because the labels for word lists are specific, contrary to SemCor's, the evaluation can be more accurate. Table 5.8 provides details on the results, in which the mean column indicates that using weighted majority voting is more promising than using each method alone. However, the weighted voting method and its simple version perform less efficiently than the actual labels, i.e., the words themselves, as we consider the 78.6% agreement rate proposed by Felbeum et al. [49]. However, the low value can be

explained by the fact that we are differentiating between senses rather than words.

Word Lists	Clustering Model	ARI	NMI	FMI	DIC	ACC	mean
List1	K-means	0.606	0.707	0.712	0.679	0.740	0.689
	Agglomerative	0.606	0.707	0.712	0.629	0.790	0.689
	Simple Voting	0.693	0.789	0.778	0.794	0.795	**0.770**
	Weighted Voting	0.693	0.789	0.778	0.794	0.795	**0.770**
List2	K-means	0.628	0.732	0.702	0.737	0.772	0.714
	Agglomerative	0.450	0.579	0.561	0.557	0.624	0.555
	Simple Voting	0.722	0.749	0.727	0.790	0.788	**0.756**
	Weighted Voting	0.722	0.749	0.727	0.790	0.788	**0.756**

Table 5.8: Table showing the results of evaluating the voting method's performance.

5.5 Evaluation of Alternative Techniques

This study also explored other methods of contextualized embedding, clustering, and dimensionality reduction, as detailed in the Methodology section.

5.5.1 Alternative Embedding Techniques

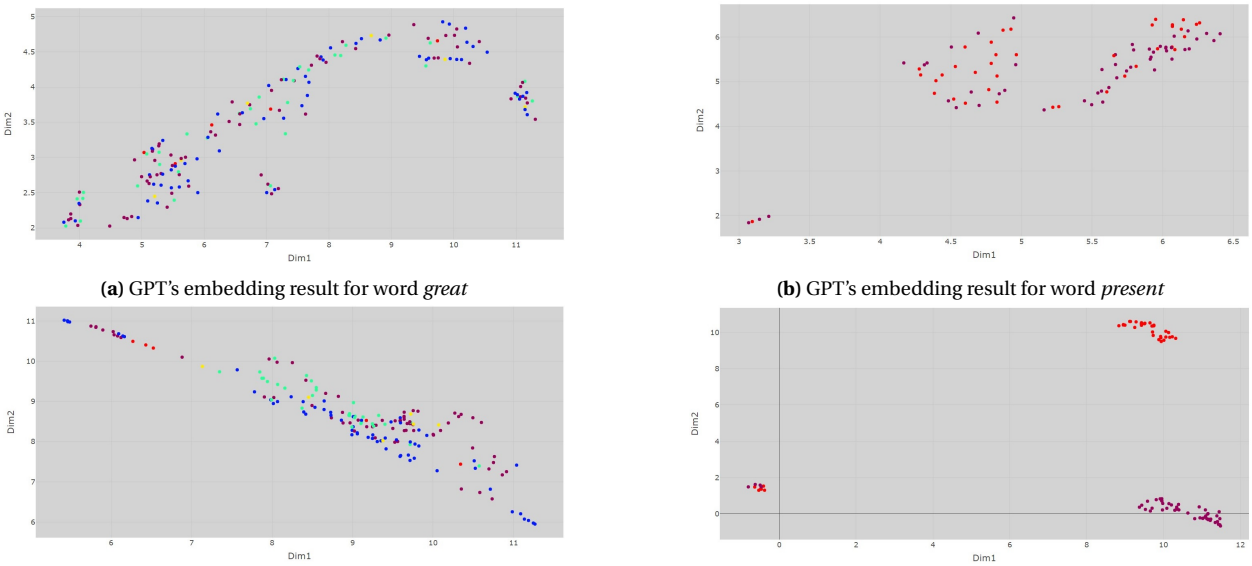
The other two embedding methods, ELMo and GPT, did not yield satisfactory results compared to BERT. From various experiments we conducted, Figure 5.14 shows two examples for each model. The outcome indicates that these two methods generate embeddings that are not suitable for visualization, implying that they cannot differentiate words at the same level as BERT. Their performance was not satisfactory even for a simple test word such as *present*.

GPT performance

Unexpectedly, we could not observe any comparable result for GPT output in the same use cases as we saw in BERT. We estimate that unidirectionality can be seen as a limitation in WSD. Since WSD is the process of determining the correct sense of a word in context, it often requires considering the previous words in a sentence and the future words. Since GPT models are unidirectional, they may be limited in their capacity to do efficient WSD as they do not have access to the entire context of a phrase.

ELMo performance

In comparison to BERT, ELMo's performance in similar use cases was not comparable. A potential reason can be that ELMo does not model word order explicitly, which can affect its ability to capture semantic information. Word order can change the meaning of a sentence, even if the words used are the same. ELMo may struggle to distinguish between sentences with similar words in different orders. For example, consider the sentences "Elmo loves cookies" and "Cookies love Elmo." These sentences have different meanings and implications, but ELMo may assign similar embeddings to "Elmo" and "cookies" regardless of their position or role in



(a) GPT's embedding result for word *great*

(b) GPT's embedding result for word *present*

(c) ELMo's embedding result for word *great*

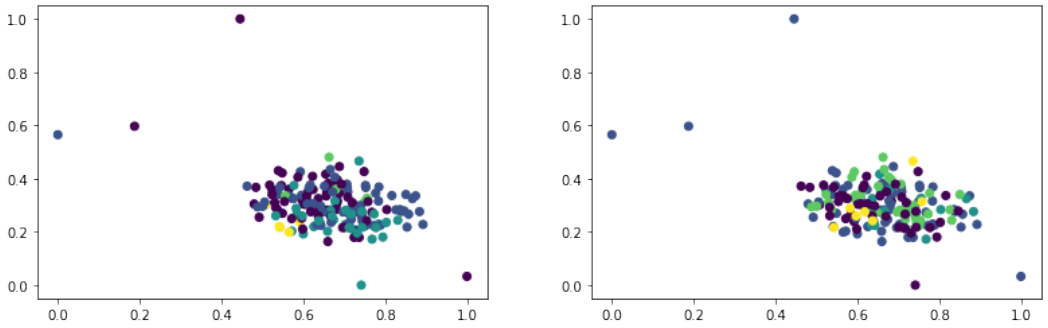
(d) ELMo's embedding result for word *present*

Figure 5.14: Figure 5.14: Embedding results for the words *great* and *present* using different methods. (a) and (b) show the embedding results for the GPT model, while (c) and (d) show the results for the ELMo model. All results were obtained by implementing UMAP as the dimensionality reduction technique, and the labels were derived from SemCor tags.

the sentence. Besides, Peters et al. [1] discuss the influence of syntax on ELMo's comprehension of semantics. Their research showed that models incorporating syntax outperform those without it when both use ELMo, especially on out-of-domain data.

The other possible reason is that, unlike BERT, ELMo does not account for WSD, meaning it cannot distinguish between senses of the same word based on context. According to the findings of Wiedemann et al., [69], BERT has a more robust capability to capture word sense information than ELMo, as evidenced by its outperformance of ELMo in WSD across various languages and domains. This limitation may lead to confusion when dealing with polysemous words with multiple meanings in different contexts. For instance, ELMo may assign similar embeddings to "bank" regardless of whether it refers to a financial institution or a river shore, negatively impacting its performance on downstream tasks that require semantic understanding [70].

Furthermore, we may find the reason for ELMo's limitation by comparing it to BERT due to its architecture. While ELMo concatenates the forward and backward models, BERT is purely bidirectional. This difference means that BERT can take advantage of both contexts simultaneously, while ELMo has limited ability. By both contexts, we mean the words that come before and after a given the word in a sentence. These words can provide crucial information about the meaning and nuance of the word, especially when it has multiple possible interpretations. Additionally, BERT's contextualized representations may be more effective in capturing the meaning of a word in different contexts compared to ELMo's multi-layer representations. Second, BERT is pre-trained on a larger corpus of text, including Wikipedia and BooksCorpus, which may provide it with a better understanding of word senses compared to ELMo, which was trained on a smaller corpus. Finally, BERT's transformer-based architecture is considered to be more powerful for NLP tasks compared to ELMo's bi-LSTM-based architecture.



(a) SemCor annotation.

(b) Our own annotation.

Figure 5.15: tSNE visualization of word embeddings showing poor performance on adjective *great* in case of applying both available annotations.

5.5.2 Alternative Dimensionality Reduction Techniques

tSNE

During our research, we experimented with tSNE, a method commonly used to reduce the dimensionality of high-dimensional data, such as the BERT embeddings, into lower dimensions. However, we found that tSNE's performance was often unsatisfactory in visualizing the data, as shown by the poor visualization of the word *great* in Figure 5.15. One potential explanation for this poor performance is the curse of dimensionality, which occurs when the high-dimensional data is spread out across many dimensions, making it difficult for tSNE to accurately capture the underlying structure of the data in lower dimensions. In such cases, tSNE may be unable to identify significant patterns or relationships within the data due to the many dimensions involved [36]. Besides, tSNE is known to be prone to get stuck in local optima, which could lead to suboptimal embeddings. On the other hand, the issue of overcrowding could also be a factor, as tSNE may struggle to differentiate between points that are too close together in the lower-dimensional space [71].

5.5.3 Alternative Clustering Techniques

DBSCAN

If senses of ambiguous words are absent, DBSCAN may be a suitable clustering choice since it does not require the user to specify the number of clusters. Despite our efforts to fine-tune the hyperparameters, we could not train this model effectively in our study. Although the reason behind that is still unrevealed, we decided to focus on the main research path rather than spend more time debugging what only an alternative option was.

Chapter 6

Conclusion and Future work

In this work, we proposed ContextLens, a tool for visualizing, clustering and exploring high-dimensional contextualized word embeddings. ContextLens is an interactive tool that enables the user to see the embedding visualization of the senses of a specific word or a number of different words in a single view. Users can upload their ".csv" or ".xlsx" file, including up to 200 sentences, to visualize a single or multiple preferred word embedding(s). We also introduced a novel approach combining multiple clustering methods using majority voting to automatically produce more robust and accurate annotations.

We evaluated ContextLens using qualitative and quantitative methods, demonstrating its usefulness in gaining insights into the structure of word embeddings and identifying the distinct senses of words. Our evaluation metrics included both supervised and unsupervised metrics, enabling us to assess the clustering results when labels were available and when not.

Our experiments showed that combining dimensionality reduction and clustering methods provided a powerful way to understand word embeddings' structure and identify the different senses of words. The majority voting approach for combining clustering methods also proved to be an effective way to improve the accuracy and robustness of the clustering results.

In summary, ContextLens provides a robust set of tools for exploring and visualizing high-dimensional word embeddings, which can be used for various NLP tasks, such as word sense disambiguation (WSD). We think that our work can contribute to developing better NLP models and applications that can help us better understand the meaning of language.

6.1 Answer to Research Questions

The study's central research question was whether it is possible to design tools that enable linguists to investigate word use on the basis of word embedding without requiring deep learning training. The answer to this question is affirmative. Our research efforts have yielded a tool that allows linguists to interpret embeddings visually, preventing the need for specialized knowledge in neural networks.

Regarding the first subquestion, "*Can we visualize embeddings in such a way that different "senses" are separated in the visualization and similar senses are grouped?*", our tool has achieved impressive results, visualizing embeddings in a manner that separates different senses and groups similar senses together. Our approach has enabled the generation of comprehensi-

ble illustrations that connect embeddings to related meanings, offering a linguistic annotation without requiring substantial time investment.

Addressing the second subquestion, "*Can we cluster embeddings in such a way that the clustering corresponds to traditional "sense" groupings?*", We have been able to cluster embeddings based on traditional "sense" groupings primarily. Furthermore, the corresponding clusterings have enlightened us about this tool's power in helping scholars in the annotation task. Additionally, when examining clustering in dividing different word types, the results are more promising than the result of discriminating between the senses of a single word.

Regarding the third subquestion, "*Can the tools be given a user interface that lets the scholars use the tool without further intervention from data scientists?*", our tool's user interface requires no further intervention from data scientists, ensuring it is readily accessible to scholars without deep learning or data science training. As a result, we have provided an initial tool capable of visualizing different words and different senses of a word in separate clusters that linguists can easily access and utilize.

Overall, our research has demonstrated that designing tools to facilitate linguistic investigations of word use based on word embedding is a promising avenue. Our tool's capabilities in visualizing embeddings, clustering, and providing user-friendly interfaces present significant opportunities for further research and applications in the field of linguistics. Based on the results of our study, we have also identified several key findings that help to answer the research questions we posed that we address below.

6.2 Challenges and Limitations

In this study, we faced several challenges related to training and evaluating a clustering system due to labeling issues. Specifically, we found that the SemCor's sense tags were not reliable enough to assess the performance of the models accurately. The primary limitation, in this case, was the need for alternative sense-level annotated corpora to compare the results. In response to this limitation, one of the supervisors manually annotated a set of 174 sentences that included the adjective *great*, which can be considered notorious for its intricate sense variations. This alternative approach significantly improved our clustering evaluations' reliability and helped ensure more precise results.

In addition to the labeling issues, we also encountered the skewed distribution of word senses. As we forced the clustering model to categorize data into clusters based on the number of sense classes, the model sometimes struggled to find groups with few samples. However, we found a valuable insight in this study: our clustering model was able to ignore rare clusters and instead proposed meaningful sub-clusters by analyzing the case of the adjective *cold*. This finding suggests that our clustering approach could effectively identify subtle variations in word sense usage, even in cases where the sense distribution is highly skewed.

It is important to note that the scope of this study was limited, and we could only examine a relatively small number of adjectives. While we tried to include a range of adjectives with varying degrees of complexity, further investigation is necessary to ensure robust evaluation across a broader range of use-cases. In particular, future research should extend the analysis to larger sample size and explore additional factors that may influence the performance of the clustering model.

Another challenge we encountered was deciding between PCA and UMAP. While UMAP outperformed PCA in most cases, there were a few instances where PCA performed better. We observed no consistent pattern indicating one technique consistently beat the other. Therefore, we included both options in the tool so that users can choose the most appropriate technique for their data patterns.

Lastly, the challenge of determining the appropriate number of clusters for the clustering labels still persists. Since contextualized embedding models such as BERT effectively separate sense clusters, there is still room for improvement through the automation of this process. One potential approach is to leverage models like DBSCAN, which can automatically determine the number of clusters without the need for manual intervention. This vision would enable a more efficient and streamlined annotation process, reducing the burden on human annotators and facilitating broader applications of this approach in NLP research.

6.3 Future Work

In the following paragraphs, the areas of potential related research that could be pursued in the future are proposed. We describe each topic and explain why it would be of valuable. These research possibilities emerged during this thesis, either because we could not fully address a particular question or because we had an idea that went beyond the scope of the current work.

6.3.1 ContextLens Possible Application

Several potential directions could be pursued in future work to apply our research. One possible avenue for exploration in ContextLens would be to split the reviews in the Hotel Reviews dataset into two distinct primary groups, such as negative/positive reviews or native/non-native authors, and investigate the differences between these groups. This investigation could provide insights into how these additional factors influence the occurrence of linguistic features in review comments. Additionally, exploring these ideas could help to identify more specific patterns and trends within data. This probe represents essential areas that could yield valuable insights into the nature of review comments and their role in shaping online discourse.

6.3.2 Potential Extensions

Pre-trained language models like BERT have achieved impressive results in NLP tasks, including WSD. However, standard BERT models do not provide sense-level embeddings. SenseBERT, GlossBERT, and bi-encoder are variants of BERT that focus on sense-level information. Integrating these sense-level models into the tool can potentially enhance its performance on sense-level tasks. Future work includes exploring the integration of sense-level models and supporting fine-tuning these models on custom datasets. Incorporating sense-level models into the tool can benefit from the richer embeddings extracted by these models, improving its performance on sense-level tasks.

Future work could also involve conducting interviews with further linguists, also with other interests than adjectives, to evaluate the effectiveness of the ContextLens tool in assisting linguistic scholars in deriving meaningful insights from the visualizations. Understanding how

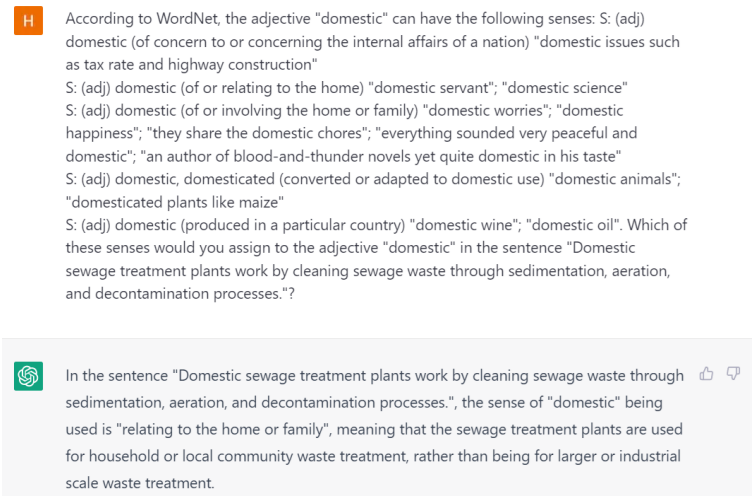


Figure 6.1: Screenshot of a chat session with ChatGPT for word sense labeling in ContextLens.

linguists use visualizations is crucial for enhancing the tool's utility in this domain. Additionally, exploring the inclusion of additional lists of words relevant to linguists' interests can further strengthen communication and the tool's usefulness. Accordingly, gathering feedback from linguists and incorporating their insights can lead to a more effective and tailored tool for the linguistic community.

A language model like ChatGPT might provide a more efficient and consistent method for annotating word sense labels. One approach could be to provide the model with a word's definition in WordNet and its corresponding sense tags and then asking the chatbot to label the word based on its contextual usage. Figure 6.1 illustrates how ChatGPT could label word senses based on contextual usage through a chat screenshot. This approach may leverage the power of machine learning and has the potential to offer a more robust and reliable way of annotating word senses.

We suggest further research to enhance the tool's capabilities by leveraging up-to-date and more extensive context-aware models, such as the recently released and upcoming variants of GPT. Notably, we merely employed GPT-2 due to the different aims of this study. Hence, exploring more latest and gigantic models may improve the quality of the embeddings. Moreover, automating the clustering process without requiring the user to specify the number of word senses by training models like DBSCAN can be a profitable avenue for future research.

Acknowledgement

I am deeply grateful to my supervisors, Dr. Nelleke Oostdijk and Dr. Hans van Halteren, for their exceptional guidance and unwavering support throughout my thesis. Their extensive knowledge and expertise in linguistics were instrumental in shaping my research and maintaining focus on my study objectives. I also thank Dr. Martha Larson for her significant contributions to my research. Her insightful feedback was invaluable in helping me navigate the challenges I encountered along the way. Furthermore, I appreciate Dr. Saeed Abbasi's constructive role in motivating me throughout the research process. His support and encouragement were critical in keeping me on track toward achieving my research goals.

Bibliography

[1] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: 1802.05365. URL: http://arxiv.org/abs/1802.05365.

[2] *Polysemy*. URL: https://www.oxfordbibliographies.com/view/document/obo-9780199772810/obo-9780199772810-0259.xml.

[3] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2020.

[4] R. Navigli. "Word sense disambiguation: A survey". In: *ACM computing surveys (CSUR)* 41.2 (2009), pp. 1–69.

[5] L. McInnes, J. Healy, and J. Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

[6] Y. Levine, B. Lenz, O. Dagan, O. Ram, D. Padnos, O. Sharir, S. Shalev-Shwartz, A. Shashua, and Y. Shoham. "Sensebert: Driving some sense into bert". In: *arXiv preprint arXiv:1908.05646* (2019).

[7] T. Blevins and L. Zettlemoyer. "Moving down the long tail of word sense disambiguation with gloss-informed biencoders". In: *arXiv preprint arXiv:2005.02590* (2020).

[8] L. Huang, C. Sun, X. Qiu, and X. Huang. "GlossBERT: BERT for word sense disambiguation with gloss knowledge". In: *arXiv preprint arXiv:1908.07245* (2019).

[9] E. Reif, A. Yuan, M. Wattenberg, F. B. Viegas, A. Coenen, A. Pearce, and B. Kim. "Visualizing and measuring the geometry of BERT". In: *Advances in Neural Information Processing Systems* 32 (2019).

[10] M. Lesk. "Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone". In: *Proceedings of the 5th annual international conference on Systems documentation*. 1986, pp. 24–26.

[11] G. A. Miller. "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[12] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. "Introduction to WordNet: An on-line lexical database". In: *International journal of lexicography* 3.4 (1990), pp. 235–244.

[13] J. Cowie, J. Guthrie, and L. Guthrie. "Lexical disambiguation using simulated annealing". In: *COLING 1992 Volume 1: The 14th International Conference on Computational Linguistics*. 1992.

[14] S. Banerjee and T. Pedersen. “An adapted Lesk algorithm for word sense disambiguation using WordNet”. In: *International conference on intelligent text processing and computational linguistics*. Springer. 2002, pp. 136–145.

[15] P. Basile, A. Caputo, and G. Semeraro. “An enhanced lesk word sense disambiguation algorithm through a distributional semantic model”. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 2014, pp. 1591–1600.

[16] T. K. Landauer, P. W. Foltz, and D. Laham. “An introduction to latent semantic analysis”. In: *Discourse processes* 25.2-3 (1998), pp. 259–284.

[17] D. M. Blei, A. Y. Ng, and M. I. Jordan. “Latent dirichlet allocation”. In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[18] M. Le, M. Postma, J. Urbani, and P. Vossen. “A Deep Dive into Word Sense Disambiguation with LSTM”. In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 354–365. URL: https://aclanthology.org/C18-1030.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[21] G. Salton, A. Wong, and C.-S. Yang. “A vector space model for automatic indexing”. In: *Communications of the ACM* 18.11 (1975), pp. 613–620.

[22] P. D. Turney and P. Pantel. “From frequency to meaning: Vector space models of semantics”. In: *Journal of artificial intelligence research* 37 (2010), pp. 141–188.

[23] J. Camacho-Collados and M. T. Pilehvar. “From word to sense embeddings: A survey on vector representations of meaning”. In: *Journal of Artificial Intelligence Research* 63 (2018), pp. 743–788.

[24] K. Sparck Jones. “A statistical interpretation of term specificity and its application in retrieval”. In: *Journal of documentation* 28.1 (1972), pp. 11–21.

[25] K. Lund and C. Burgess. “Producing high-dimensional semantic spaces from lexical co-occurrence”. In: *Behavior research methods, instruments, & computers* 28.2 (1996), pp. 203–208.

[26] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. “Using latent semantic analysis to improve access to textual information”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1988, pp. 281–285.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[28] J. Pennington, R. Socher, and C. D. Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[29] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. “Enriching word vectors with subword information”. In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. “Attention is all you need. 2017”. In: *arXiv preprint arXiv:1706.03762* (2017).

[31] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. “Improving language understanding by generative pre-training”. In: (2018).

[32] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[33] D. W. Otter, J. R. Medina, and J. K. Kalita. “A survey of the usages of deep learning for natural language processing”. In: *IEEE transactions on neural networks and learning systems* 32.2 (2020), pp. 604–624.

[34] C. O. S. Sorzano, J. Vargas, and A. P. Montano. “A survey of dimensionality reduction techniques”. In: *arXiv preprint arXiv:1403.2877* (2014).

[35] K. Pearson. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2.11 (1901), pp. 559–572.

[36] L. Van der Maaten and G. Hinton. “Visualizing data using t-SNE.” In: *Journal of machine learning research* 9.11 (2008).

[37] J MacQueen. “Classification and analysis of multivariate observations”. In: *5th Berkeley Symp. Math. Statist. Probability*. 1967, pp. 281–297.

[38] J. H. Ward Jr. “Hierarchical grouping to optimize an objective function”. In: *Journal of the American statistical association* 58.301 (1963), pp. 236–244.

[39] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *kdd*. Vol. 96. 34. 1996, pp. 226–231.

[40] R. Thorndike. “Who belongs in the family?” In: *Psychometrika* 18.4 (1953), pp. 267–276.

[41] P. J. Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[42] J. P. McCrae, A. Rademaker, E. Rudnicka, and F. Bond. “English WordNet 2020: Improving and extending a WordNet for English using an open-source methodology”. In: *proceedings of the LREC 2020 workshop on multimodal WordNets (MMW2020)*. 2020, pp. 14–19.

[43] M. Baker. “Corpus Linguistics and Translation Studies*: Implications and applications”. In: *Researching Translation in the Age of Technology and Global Conflict*. Routledge, 2019, pp. 9–24.

[44] H. Kucera, H. Kučera, and W. N. Francis. *Computational analysis of present-day American English*. Brown university press, 1967.

[45] S. Landes, C. Leacock, and R. I. Tengi. “Building semantic concordances”. In: *WordNet: An electronic lexical database* 199.216 (1998), pp. 199–216.

[46] M. A. Marcinkiewicz. “Building a large annotated corpus of English: The Penn Treebank”. In: *Using Large Corpora* 273 (1994).

[47] T. Petrolito and F. Bond. “A survey of wordnet annotated corpora”. In: *Proceedings of the Seventh Global WordNet Conference*. 2014, pp. 236–245.

[48] L. Bentivogli and E. Pianta. “Exploiting parallel texts in the creation of multilingual semantically annotated resources: the MultiSemCor Corpus”. In: *Natural Language Engineering* 11.3 (2005), pp. 247–261.

[49] C. Fellbaum. “Performance and Confidence in a Semantic Annotation Task Christiane Fellbaum, Joachim Grabowski, and Shari Landes”. In: *WordNet: An electronic lexical database* (1998), p. 217.

[50] C. E. Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.

[51] N. F. Liu, M. Gardner, Y. Belinkov, M. E. Peters, and N. A. Smith. “Linguistic knowledge and transferability of contextual representations”. In: *arXiv preprint arXiv:1903.08855* (2019).

[52] A. Roberts, C. Raffel, and N. Shazeer. “How much knowledge can you pack into the parameters of a language model?” In: *arXiv preprint arXiv:2002.08910* (2020).

[53] Y. Goldberg. “Assessing BERT’s syntactic abilities”. In: *arXiv preprint arXiv:1901.05287* (2019).

[54] Y. Lin, Y. C. Tan, and R. Frank. “Open Sesame: getting inside BERT’s linguistic knowledge”. In: *arXiv preprint arXiv:1906.01698* (2019).

[55] J. Hewitt and C. D. Manning. “A structural probe for finding syntax in word representations”. 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)*. 2019, pp. 4129–4138.

[56] G. Jawahar, B. Sagot, and D. Seddah. “What does BERT learn about the structure of language?” In: *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*. 2019.

[57] I. Tenney, D. Das, and E. Pavlick. “BERT rediscovers the classical NLP pipeline”. In: *arXiv preprint arXiv:1905.05950* (2019).

[58] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. “Revealing the dark secrets of BERT”. In: *arXiv preprint arXiv:1908.08593* (2019).

[59] Y. Hao, L. Dong, F. Wei, and K. Xu. “Visualizing and understanding the effectiveness of BERT”. In: *arXiv preprint arXiv:1908.05620* (2019).

[60] L. Cui, S. Cheng, Y. Wu, and Y. Zhang. “Does bert solve commonsense task via commonsense knowledge”. In: *arXiv preprint arXiv:2008.03945* 4 (2020).

[61] A. Maćkiewicz and W. Ratajczak. “Principal components analysis (PCA)”. In: *Computers & Geosciences* 19.3 (1993), pp. 303–342.

[62] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas. “Randomized dimensionality reduction for k-means clustering”. In: *IEEE Transactions on Information Theory* 61.2 (2014), pp. 1045–1062.

[63] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. “Learning from crowds.” In: *Journal of machine learning research* 11.4 (2010).

[64] W. M. Rand. “Objective criteria for the evaluation of clustering methods”. In: *Journal of the American Statistical association* 66.336 (1971), pp. 846–850.

[65] A. Strehl and J. Ghosh. “Cluster ensembles—a knowledge reuse framework for combining multiple partitions”. In: *Journal of machine learning research* 3.Dec (2002), pp. 583–617.

[66] E. B. Fowlkes and C. L. Mallows. “A method for comparing two hierarchical clusterings”. In: *Journal of the American statistical association* 78.383 (1983), pp. 553–569.

[67] J. Véronis and P. Langlais. “Evaluation of parallel text alignment systems: The ARCADE project”. In: *Parallel text processing: Alignment and use of translation corpora* (2000), pp. 369–388.

[68] A. Kilgarriff. “Gold standard datasets for evaluating word sense disambiguation programs”. In: *Computer Speech & Language* 12.4 (1998), pp. 453–472.

[69] G. Wiedemann, S. Remus, A. Chawla, and C. Biemann. “Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings”. In: *arXiv preprint arXiv:1909.10430* (2019).

[70] A. Kutuzov and E. Kuzmenko. “To lemmatize or not to lemmatize: how word normalisation affects ELMo performance in word sense disambiguation”. In: *arXiv preprint arXiv:1909.03135* (2019).

[71] E. Jin, M. Li, X. Feng, Z. Yang, and W. Nai. “Hybrid dimension reduction method based on Isomap and t-SNE with beetle antennae search algorithm”. In: *2020 13th International Symposium on Computational Intelligence and Design (ISCID)*. IEEE. 2020, pp. 392–395.

Appendix

The Lists of words

The content of lists 1 and 2 used in the evaluation is presented in this appendix. List 1 consists of words related to cleanliness, such as "clean," "unclean," "dirty," and "filthy." List 2 contains terms related to price, such as "cheap," "expensive," "inexpensive," "costly," and "pricey." These lists were used to evaluate the performance of the subjective clustering method, and the results are presented in subsection 5.4.3.

It is worth noting that the list of adjectives we removed from SemCor based on our analysis, mentioned in the first part of section 5, is provided below:

- **cardinal numbers**: 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine', 'ten', 'eleven', 'twelve', 'thirteen', 'fourteen', 'fifteen', 'sixteen', 'seventeen', 'eighteen', 'nineteen', 'twenty'
- **ordinal words**: 'first', 'second', 'third', 'fourth', 'fifth', 'sixth', 'seventh', 'eighth', 'ninth', 'tenth', 'eleventh', 'twelfth', 'thirteenth', 'fourteenth', 'fifteenth', 'sixteenth', 'seventeenth', 'eighteenth', 'nineteenth', 'twentieth'
- **quantifiers**: 'many', 'much', 'more', 'most', 'several', 'some', 'any', 'few', 'little', 'less', 'least', 'all', 'enough', 'whole', 'no', 'none', 'every', 'each', 'either', 'neither', 'both', 'half', 'several', 'various', 'numerous', 'plenty', 'scarce', 'abundant', 'ample', 'excessive', 'adequate', 'insufficient', 'extra', 'superfluous', 'insubstantial', 'inadequate', 'deficient', 'sparse', 'limited', 'unlimited', 'infinite', 'countless', 'incalculable', 'myriad', 'umpteen', 'multitudinous', 'umpteen'
- **pronouns**: 'such', 'same', 'other', 'another'

,