Leipzig University
Institute of Computer Science
Degree Programme Computer Science, M.Sc.

From Contextualized to Static Word Embeddings

Master's Thesis

Hannes Hansen
Born Jan 26, 1996 in Husum

Matriculation Number 3741756

1. Referee: Juniorprof. Dr. Martin Potthast
2. Referee: Dr. Martin Hebart

Submission date: September 22, 2022

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Leipzig, September 22, 2022

. . . .
Hannes Hansen

Abstract

Word embeddings play an important role in natural language processing and cognitive research. Words are projected into a dense representational space where the embeddings encode the semantic meaning of words. Distributional static word embeddings are widely used examples for word embeddings. They provide fixed embeddings that can be used in different downstream tasks like token or sentence classification. Recent Transformer-based models can achieve human-level performance in different language tasks and provide contextualized word embeddings that incorporate the context where a word is used. Here we asked whether contextualized word embeddings could be used to generate improved static embeddings by averaging across a larger number of contexts. Embeddings are first extracted from several Transformer-based models using a large text corpus. Contextualized and aggregated static word embeddings are then created and further evaluated. The semantic encoding is investigated with word similarity ratings and similarity ratings of object images. The quality of these new embeddings is further investigated with downstream tasks like the prediction of superordinate categories and the prediction of dimension values. The results show that contextualized and aggregated Transformer-based embeddings can lead to improved results under intrinsic and extrinsic evaluation. Yet, classic static word embeddings can be of similar or even higher quality, depending on the datasets. The representational similarity with human-based word similarity ratings can be increased with Transformer-based embeddings while the representational similarity with human-based similarities of object images can decrease with Transformer-based embeddings.

Contents

Chapter 1

Introduction

1.1 Motivation

Word embeddings play an important part in the field of natural language processing as well as other research fields like cognitive science. They are numerical representations of words that reflect the meaning and relationship to other words. These representations are usually based on co-occurrences of words in a text. For example, the words *dog* and *animal* may occur often together and thus would yield similar representations. They can be used in different downstream tasks, for example, to train a Named Entity Recognition classifier (Sien, 2015), a tweet classifier (Khatua et al., 2019) or to build representations of clinical text (Khattak et al., 2019).

Beside the usage in downstream computational tasks, word embeddings are used in the study of human conceptual knowledge (Grand et al., 2022; Rubinstein et al., 2015; Lucy & Gauthier, 2017). For this, semantic feature norms (McRae et al., 2005) or similarity ratings for words (Hill et al., 2015) are usually used with static word embeddings. A new behavioral ground truth for similarity ratings for images of objects was introduced by Hebart et al. (2020) which shall be used in this thesis.

There are different ways of calculating word embeddings. Count-based methods like pointwise mutual information or predictive methods like Word2Vec and GloVe to name only a few. The simplest way is an one hot encoding where the length of the embedding is the number of words in the vocabulary. A word would then be represented as an one at a specific position. As this method does not reflect any word meaning, other methods emerged to represent meaning and similarity. One key idea is that similar words occur in similar contexts. Count-based methods build upon this idea and represent words as counts in context. Context can be defined as documents where words occur. In this case, words can be represented as frequencies across documents. Through dimensionality reduction, dense word embeddings can then be created through Latent Semantic Analysis (Deerwester et al., 1990). Context can also be defined as ranges in the text where multiple words occur together.

A more recent approach is the creation of word embeddings using predictive models like Word2Vec from Mikolov et al. (2013). The idea is to learn static word embeddings by predicting the context the word is used in. The model is trained using self-supervised methods on a large corpus of text. These predictive models had a high impact on the natural language processing area like text classification (Jang et al., 2019; Maas et al., 2011). During training, multiple contexts are used per word but only a short window of previous and next words is used. Therefore, the context is quite small. Static word embeddings like Word2Vec also reflect only one word sense of a word. For example, the word *bank* has the same word embedding for the meaning of a financial bank and the meaning of a river bank.

Recent advances in natural language processing led to large language models like BERT from Devlin et al. (2019) and GPT-2 from Brown et al. (2020). These so-called Transformer models (Vaswani et al., 2017) can reach human-level performance on many natural language understanding tasks (A. Wang et al., 2019; C. Wang et al., 2020). Using these models, it is possible to create word embeddings by extracting the hidden states. The models produce dynamic word embeddings that change with every context. For example, the word *bank* gets different embeddings in different sentences. Therefore, the model can create contextualized word embeddings respecting the meaning of a word in a given context. The embedding for the word *bank* with the meaning of financial bank is then different than the embedding for the word *bank* with the meaning of a river bank. Thus, Transformer-based models can create word sense embeddings. Yet, embeddings for words with the same meaning can be different, too. But for the creation of a word sense embedding set, either an annotated text corpus or a word sense prediction method is needed to map words in sentences to word senses. Models like BERT and GPT-2 have been self-supervised trained on large corpora to predict the next sentences or missing words and are available through pretrained models. They encode words in a bidirectional or unidirectional way using the whole input sentence and can use a longer context.

As Transformer-based models are the state of the art for natural language processing tasks, it can be assumed that their produced embeddings should better capture the meaning of words than previous word embeddings like Word2Vec. One problem with contextualized word embeddings is their computationally expensive calculation. A trained language model is needed as well as the processing and extraction of word embeddings from text input. Therefore, they are not very useful for other research areas like cognitive science where static representations are used to investigate the similarity of objects. Further, it is not possible to use classic interpretability methods that were developed for static word embeddings (Bommasani et al., 2020). The embeddings are also highly context-sensitive which on the one hand can lead to word sense embeddings where embeddings for different meanings of *bank* will be different. On the other hand, they will be also different when the meaning is the same but other details in the text are different. This can be seen in the sentences *I look at the river bank and feel sad* and *This bank on the Thames looks really nice* where the contextualized word embeddings of *bank* will be different even though they use the same meaning of *bank*.

Bommasani et al. (2020) described a method to aggregate these contextualized word embeddings where they extracted contextualized word embeddings from different Transformer-based models and aggregated them to static decontextualized word embeddings. They evaluated the produced embeddings on word similarity tasks. Furthermore, they did not take into account the presence of homonyms like *bank* and only produced word embeddings, not word sense embeddings. This is unsuitable when a different embedding is needed for multiple word senses, e.g. for calculating the similarity between *bank* as a financial bank and *bank* as a river bank. The results show that aggregated embeddings can perform better than static embeddings like Word2Vec. The evaluation of word embeddings as well as new methods to create and aggregate word embeddings and word sense embeddings shall be explored in this thesis.

1.2 Research Questions

This thesis shall try to answer the following research questions.

First, it shall be investigated, how well human similarity ratings are reflected by different types of word embeddings. Embeddings with high representational similarity with humans can be used as models of human conceptual knowledge and might be of higher quality for other downstream tasks. This will be done by an intrinsic evaluation using different similarity datasets. Datasets that use similarity between words and a dataset that uses the similarity between object images shall be used. It shall be further answered, how the different similarity datasets are characterized.

The next research question is, how different word embedding types perform compared to each other. This includes classic static word and word sense embeddings and Transformer-based embeddings, either contextualized or decontextualized. This shall bring insights into why and when a specific word embedding should be used. The experiments to that question are based on intrinsic and extrinsic evaluations. The results from the previous research question will be used. The embeddings shall be further used to predict superordinate categories and object dimension values.

To evaluate embeddings from Transformer-based models, a method to extract and create static embeddings will be created. Therefore, this thesis shall explore what are the best methods to create static embeddings and how different hyperparameters influence the quality of embeddings.

1.3 Goals

The goals of this thesis are the evaluation of different pretrained word and word sense embeddings on how well they represent human knowledge about word and object similarity and the creation of static Transformer-based embeddings. This will lead to more insights into what information is encoded in word embeddings and how large language models work. Further, new Transformer-based methods shall be explored to create new word and word sense embeddings that better capture meaning and similarity. To achieve this, several results shall be produced.

Current methods for the creation of static word and word sense embeddings and Transformer-based word embeddings shall be reviewed. Then, the evaluation results from recent work shall be reviewed. This includes intrinsic and extrinsic evaluation.

Next, current static word embeddings shall be created as well as contextualized Transformer-based word embeddings. As decontextualized word embeddings are not publicly available, a prototype shall be created to extract and decontextualize word embeddings from large language models over multiple contexts. The prototype shall be able to create word embeddings as well as word sense embeddings.

After the generation of static, contextualized, and decontextualized Transformer-based word and word sense embeddings, several evaluations shall be performed. First, the representational similarity of embeddings with humans, based on word similarity and similarity of object images, shall be investigated. Next, the embeddings shall be used in downstream tasks to predict dimension values and superordinate categories. Finally, a new method to create embeddings with higher representational similarity shall be applied. It will further be investigated whether the creation of word sense embeddings can be beneficial with respect to the representational similarity with humans.

This work will present a prototype that could be used to create a new dataset of decontextualized word and word sense embeddings. Further, new text-based embeddings shall be presented that better capture human conceptual knowledge.

Chapter 2

Background

The following chapter will define the background of this thesis and define the needed terms. First, it will be defined what words, word senses, and synsets are and how they relate to word embeddings. Second, the similarity between words and objects will be explained and different similarity measures will be presented. Next, it will be explained how Transformer-based neural network models work as they will be used to create word embeddings. Last, different evaluation methods as well as correlation and distance metrics will be defined.

2.1 Words and Word Senses

Words can be ambiguous. The same word can have multiple meanings. For example, a *mouse* can be an animal but also a cursor controller. Their meaning gets settled by the surrounding context of the word. This contextual variation can then be defined as word senses where "a sense (or word sense) is a discrete representation of one aspect of the meaning of a word." (Jurafsky & Martin, 2009, p. 646). The word *mouse* has multiple word senses, e.g *mouse*^{1} for the animal and *mouse*^{2} for the controller. The relationship between senses can then be further defined. If two senses of two different words are the same, the senses are synonyms of each other, like {*car*^{1}, *automobile*^{1}} (Jurafsky & Martin, 2009, p. 649). This rule is commonly applied to words, too. For example, {*car*, *automobile*} or {*couch*, *sofa*} can be seen as synonyms. If two senses of a word have no particular relation, they are defined as homonyms like {*mouse*^{1}, *mouse*^{1}} (Jurafsky & Martin, 2009, p. 646). If two senses define the opposite of a scale, they are defined as antonyms like {*long*^{1}, *short*^{1}} (Jurafsky & Martin, 2009, p. 650).

WordNet is a thesaurus for sense and word relations (Kilgarriff & Fellbaum, 2000). The database consists of around 120.000 nouns, 12.000 verbs, and 23.000 adjectives. In WordNet, a synonym set (synset) describes a concept,

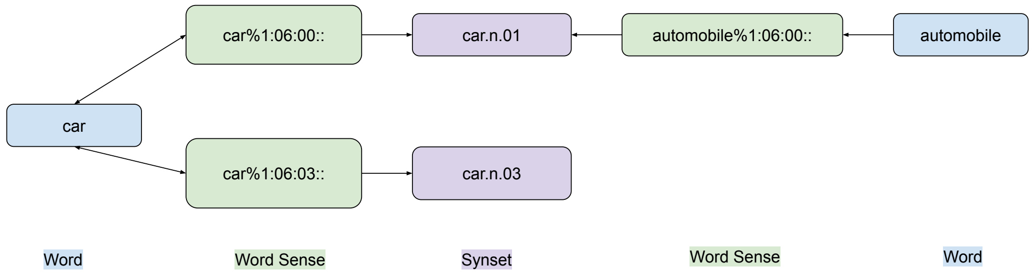


Figure 2.1: Overview of the relationship between words, word senses, and synsets in WordNet

for example, the concept of a couch as furniture. It is defined as the group of word senses that are synonyms, e.g. $\{couch^1, sofa^1\}$ (Jurafsky & Martin, 2009, p. 652).

The senses and synsets in WordNet are further identified by a unique string (see Figure 2.1). For example, the synset identification *car.n.01* refers to a driving car which can be represented by the words *car* and *automobile*. The word senses for these specific word occurrences can be further identified by their identification, e.g *car%1:06:00:* for the word *car* in the context of a driving car and *automobile%1:06:00:* for the word *automobile* in the context of a driving car.

Word sense disambiguation is the task of predicting the word sense for a word in a given context, e.g. a sentence. Several supervised and unsupervised methods exist to solve this task. To build sense embeddings, it is necessary to have an annotated text corpus where each word occurrence is labeled with a word sense. Models to solve the word sense disambiguation task are evaluated intrinsically by measuring their classification performance on a labeled test set.

2.2 Word Embeddings

Word embeddings are numerical representations of words. They capture word meaning and similarity in a vector space and can be used in downstream tasks like word or sentence classification. Values in word embeddings can be discrete or continuous. Furthermore, word embedding vectors can be sparse or dense.

Word embeddings from predictive models like Word2Vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014) shall be referred to as static word embeddings as they do not depend on the context where the word occurs. Non aggregated word embeddings from Transformer-based models, generated with one text input as context, shall be referred to as contextualized word embeddings. Aggregated word embeddings from Transformer-based models shall be referred to as decontextualized word embeddings.

Furthermore, the granularity of the embedding has to be specified. Embeddings can be created for words, word senses per part-of-speech tag, WordNet senses, and synsets. Most embedding models produce word embeddings leading to the same embeddings for homonyms. Word sense embeddings can be based on the part-of-speech tag. For example, a word sense embedding can be produced for the word *bank* as a noun. More precise is the usage of word senses and synsets from WordNet (Figure 2.1). Then, embeddings are not only created for words but also word senses and synsets.

As seen in Figure 2.1, a word embedding for the word *car* refers to an embedding that was created using texts where the word *car* occurred. Similarly, a word embedding for the word *automobile* is based on texts where the word *automobile* occurred. Further synonyms can be incorporated to create word embeddings. For example, for a main word *car*, texts with *car* and *automobile* occurrences can be used to create an embedding for the word *car*.

Furthermore, a word sense embedding for the sense *car%1:06:00:* is only based on text where the word *car* occurs and the word sense is predicted as a driving car. A synset embedding for the synset *car.n.01* is based on text where the word *car* or *automobile* occurred and the word sense was predicted as either *car%1:06:00:* or *automobile%1:06:00:*.

2.3 Similarity

The term similarity is often used with different definitions. Similarity can be defined in a strict local way where synonyms like *keep* and *possess* are highly similar. Similarity can be also defined in a global conceptual way where words of the same concept or category are similar like *cat* and *dog* because they are both animals. The term similarity is sometimes used together with relatedness where words are related when they are used together like *money* and *bank* or *clothes* and *closet*. Both words are not synonyms of each other and also don't belong to the same concept.

A method to find out how humans define word similarity is to run a rating task where humans judge the similarity between a pair of words on a scale from dissimilar to similar. This is done using different word pairs, e.g. nouns, adjectives, and verbs, and different scales, e.g. from 0 (dissimilar) to 10 (similar). Several studies were performed in the past that led to these word similarity datasets. For example, the Wordsim-353 (Agirre et al., 2009), Simlex-999 (Hill et al., 2015) or Rare-Words (Luong et al., 2013) datasets.

This is a simple process to gather human similarity judgements but also suffers from multiple problems. First, it is hard to sample enough responses for all pair combinations. Therefore, most word similarity datasets do only contain similarity judgements for a subset of word pair combinations. Second, this approach does not take into account that two words can be similar and dissimilar, depending on the properties of the concept behind the word. For example, the words *dog* and *cat* are highly similar as they both refer to animals but can be also considered as dissimilar as both animals reflect different types of animals, e.g. a dog is seen as a more loyal animal compared to a cat.

Real-world objects can be categorized by different properties, e.g. colorful, round-shaped or animal-related. Based on their values in these dimensions, it is possible to access the similarity between objects. As there is no definition of dimensions, it is possible to define a vast number of dimensions, which could then be used to calculate similarity. Therefore, the selection of dimensions is a crucial part of investigating object similarity. Some dimensions can be more important than others. For example, shape dimensions, e.g. *round-shaped*, may be more used in human conceptual knowledge than more niche dimensions, e.g. *able-to-fly*. This leads to the idea that there might be a core set of dimensions that is mainly used by humans to describe and differ real-world objects.

Hebart et al. (2020) defined an approach to extract these core dimensions, based on human similarity judgements and a computational model. The approach is based on the THINGS dataset (Hebart et al., 2019) which consists of 1,854 objects annotated with a word, an image, synonyms, a WordNet synset id, a category, and other features. The authors first used a triplet odd-one-out task where 5,983 participants had to decide which object in a given image triplet is the odd one. Using a large behavioral crowdsourcing task, 1.5 million samples were retrieved. As it is not feasible to get enough participant responses for all possible triplets, a computational model was further developed to capture most of the information. The model is based on a $m \times p$ representational embedding matrix with m as the number of THINGS objects and p as the number of initial dimensions. The matrix was randomly initialized with $m = 1,854$ and $p = 90$. Then the vector representations of the triplet objects are extracted and the similarity, based on the dot product, is calculated for all object pairs. The probability of choosing the odd one out is then computed using the softmax function. The cross-entropy and a L1 regularization term were used as the loss function to create sparse representational embeddings. The following equation defines the loss function and is taken from Hebart et al. (2020).

$$\sum^{n} log(\frac{exp(x_i x_j)}{exp(x_i x_j) + exp(x_i x_k) + exp(x_j x_k)}) + \lambda \sum^{m} ||x||_1 \tag{2.1}$$

with n for the number of triplets, x as the embedding of an object, and i, j, k

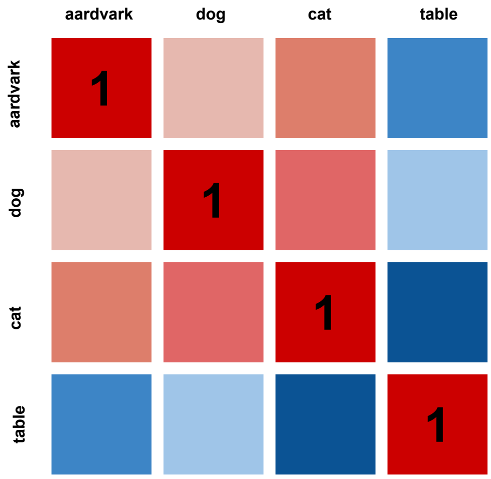


Figure 2.2: Exemplary similarity matrix between objects.

as the indices in the triplet. This model finally produced 49 interpretable and meaningful dimensions which were then manually labeled with descriptive names, for example, *animal-related*, *round-shaped*, or *colorful*. Other dimensions have been regularized to zero and were therefore removed. The term *dimension* shall refer to these values unless otherwise stated.

Further, a similarity matrix, containing a similarity value between all possible pairs of objects, was created (see Figure 2.2). This matrix can be seen as an alternative to word similarity datasets, where human-created similarity ratings are produced for given word pairs. The authors "defined object similarity in the triplet odd-one-out task as the probability $p(i, j)$ of the participants choosing objects i and j to belong together, irrespective of context." (Hebart et al., 2019). For this purpose, the authors "created all predicted choices for all possible 1.06 billion triplets and calculated the mean choice probability for each pair of objects." (Hebart et al., 2019). This similarity matrix is used in this work to study the representational similarity with humans of different word embeddings.

2.4 Transformer Models

The Transformer architecture is the basis for the large language models used in this thesis and the basis for the creation of new word and word sense embeddings. The following explanations about the Transformer model are based on Vaswani et al. (2017).

The Transformer model and architecture were introduced by Vaswani et al.

(2017). It was introduced as a deep neural network as a machine translation model. Transformer models are trained on a large amount of text which makes it possible to use them for the creation of word embeddings. It is composed of an encoder and a decoder block where the encoder block gets an input and the decoder block gets the target output and produces an output. An overview of the architecture is shown in Figure 2.3.

The encoder block takes token representations, that are produced by a specific tokenizer, as its input. An arbitrary number of encoder layers then process these token representations. An encoder layer is further composed of a self-attention sublayer, normalization, and feedforward sublayers as well as residual connections.

The self-attention layer is used by the model to learn which tokens the model has to pay attention to when it processes a specific token. It is based on the matrix multiplication between the query Q, key K, and value V matrices:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

The authors further propose to use multiple self-attention heads $head_i$ resulting in the multi-head self-attention, where the number of heads is a hyperparameter. The resulting outputs of the self-attention are then concatenated and multiplied with a weight matrix W^O to retain the layer output dimensions.

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.3}$$

After the multi-head self-attention sublayer processed the input, the results and the original input are added through a residual connection and normalized by a normalization sublayer. Afterwards, the results are processed by a feedforward layer, added to the original values, and normalized. Finally, the results are sent to the next encoder layer. Therefore, the encoder layers process token representations through attention, normalization, and non-linear transformations. The self-attention process in the encoder block has access to all tokens in the input. Finally, the output of the last encoder layer is sent to all decoder layers in the decoder block.

The decoder block gets the token representations of the target output as input and processes them through several decoder layers similar to the encoder block.

A decoder layer is built similarly to an encoder layer as it is composed of a multi-head self-attention sublayer and several normalization and feedforward sublayers. In contrast to the encoder layer, it also consists of a multi-head encoder-decoder attention sublayer. The encoder-decoder attention sublayer

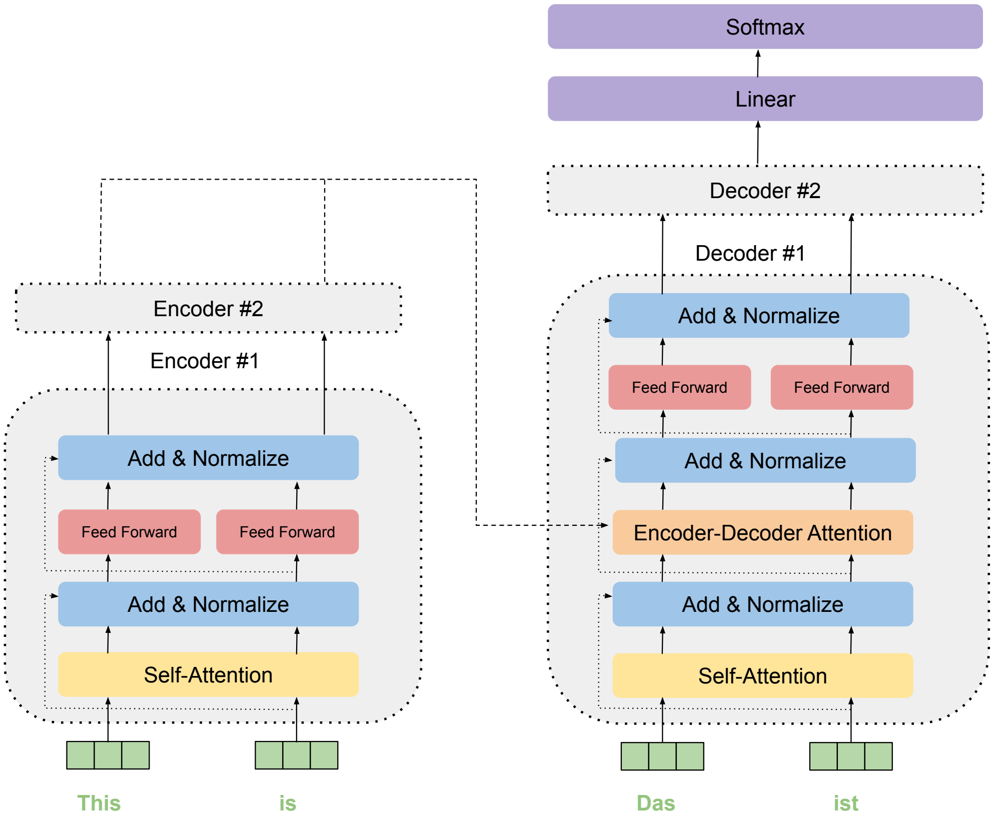


Figure 2.3: Overview of the Transformer architecture. Adapted from Vaswani et al. (2017).

takes the output of the last encoder layer and enables the decoder layer to get access to the encoded information about the original source text. The encoder-decoder attention sublayer works similarly to the self-attention sublayer as it uses the query matrix Q from the previous layer but takes the key matrix K and the value matrix V from the last encoder layer. The self-attention sublayer in the decoder block is further different from the sublayer in the encoder block as it can only use token representations from earlier positions in the text. Therefore, tokens at later positions are masked in this so-called masked multi-head self-attention.

Finally, a linear feedforward layer processes the output of the last decoder layer after which a softmax layer produces a probability for the next token.

Different models are built upon the idea of the Transformer. Encoder-based models are built using only encoder layers whereas Decoder-based models are built using decoder layers.

2.4.1 Encoder Models

Following on the work from Vaswani et al. (2017), new models and architectures emerged from the Transformer model.

The BERT (Bidirectional Encoder Representations from Transformers) model from Devlin et al. (2019) was introduced as a large language model based on the Transformer architecture. In contrast to the original Transformer model, it is only composed of the encoder block. Also, the training objectives are different. Instead of translating a sentence, the BERT model uses a masked language model objective where the model randomly masks input tokens and has to predict the correct token and a next sentence prediction objective where the model has to predict whether a sentence comes next or not. Therefore, BERT uses a semi-supervised training objective as no task-specific labels exist.

The authors further propose to first pretrain BERT using these training objectives on the BooksCorpus (Zhu et al., 2015) and English Wikipedia. Afterwards, BERT can be either finetuned using supervised methods on specific tasks like Named Entity Recognition or sentiment analysis, or features can be extracted from the model and fed into downstream tasks.

Special tokens were further introduced. The SEP token is used to separate two sentences which is needed for the next sentence prediction and sentence-based tasks like the prediction of sentence similarity. The CLS token is inserted as a classification token which can be used in classification tasks where a classification head is built on top of the CLS token representation.

Improved models, based on the BERT architecture, emerged afterwards. For example, the RoBERTa model from Liu et al. (2019) introduced more robust training methods. The DeBERTa model from He et al. (2021) introduced the separation from positional and token encoding. Further, distilled models like DistillBERT (Sanh et al., 2019) were built to create smaller and faster models while maintaining similar performance. In conclusion, BERT is a bidirectional language model which produces token representations based on the left and right context.

2.4.2 Decoder Models

Parallel to the BERT model, the decoder based large language model GPT (Generative Pretrained Transformer) was introduced by Radford et al. (2018) after which the larger successor models GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) followed.

In contrast to the BERT model, the GPT models are trained with the causal language model training objective where the model has to predict the next word in a sequence. Decoder-based models are composed of decoder layers and use masked self-attention and process tokens unidirectional.

As the GPT-3 model is not available, GPT-2 shall be focused on. It was trained on the custom WebText dataset. In conclusion, the GPT models are large language models with human-level text generation capabilities. Models from this decoder architecture family and the previous encoder architecture family will be used to create Transformer-based embeddings.

2.5 Intrinsic and Extrinsic Evaluation

Word embeddings can be intrinsically and extrinsically evaluated (Bakarov, 2018). Intrinsic evaluation measures the internal properties of the embeddings. For example, how well they capture meaning. This can be done by calculating similarities between word pairs and their comparison with human similarity judgements. This evaluation method is fast but not useful for specific application performance. In the context of this thesis, the intrinsic evaluation will give insights about how well human knowledge about semantics is encoded in different word embeddings.

Extrinsic evaluation measures the quality of word embeddings for specific downstream tasks e.g. text classification. For this evaluation, an algorithm or model has to be trained on the embeddings which makes the evaluation slow but useful if the word embeddings shall be evaluated for a specific task. In the context of this thesis, this evaluation will investigate the quality of recent embeddings and the quality of the new embeddings produced in this work.

2.6 Correlation and Distance Metrics

For the intrinsic evaluation, correlation with a ground truth, e.g. human similarity judgements, has to be investigated by calculating the pearson r_p and spearman r_s correlations.

$$r_p = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2(y_i - \overline{y})^2}} \tag{2.4}$$

$$r_s = 1 - \frac{6\sum d_i^2}{n(n^2 - 1)} \tag{2.5}$$

where d_i are the pairwise distances of the ranks of the variables, n as the number of samples, and x_i as a sample. A high similarity between human similarity ratings and similarity ratings that are based on word embeddings is reflected in a high correlation. The key difference between both correlation measures is that spearman correlation operates on ranks instead of regular similarities like pearson correlation. This makes a qualitative investigation possible where pairs with the highest rank difference can be extracted. Pearson correlation further assumes a ratio scale whereas spearman correlation only assumes an ordinal scale.

To calculate similarity and dissimilarity between two embeddings x and y in the vector space, different distance and similarity metrics like the dot product, cosine similarity or euclidean distance can be used. Euclidean distance is a dissimilarity measure and is defined as:

$$euclidean(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.6}$$

The dot product is a similarity measure and is defined as:

$$dot(x, y) = x \cdot y = \sum_{i=1}^{n} x_i \cdot y_i \tag{2.7}$$

Word embedding similarity, based on the dot product, can be biased by word frequency (Jurafsky & Martin, 2009). Therefore, the cosine similarity is commonly used which is defined as the normalized dot product:

$$cos(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{2.8}$$

Cosine similarity will be used in this thesis to calculate embedding similarity.

Chapter 3

Related Work

This chapter will introduce related work about the creation of word and word sense embeddings, the intrinsic and extrinsic evaluation of different embeddings as well as methods to predict the word sense and methods to predict sentence similarity.

3.1 Creation of Word Embeddings

Different methods to create word embeddings exist. One method is a simple one-hot encoding of words (see Figure 3.1). The embedding size equals the number of words in the vocabulary and each position is assigned to a word. This method allows fast creation of embeddings but is not useful as it does not reflect any meaning or similarity. Word embeddings for *dog* and *cat* are equally similar to embeddings for *table*.

Count-based Methods

The distributional hypothesis states that words are similar when they are used in similar contexts where the context has to be defined (Firth, 1957).

Count-based methods use this idea by counting the number of times a word occurs in a defined context. For example, in a co-occurrence count matrix, it

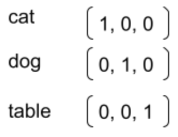
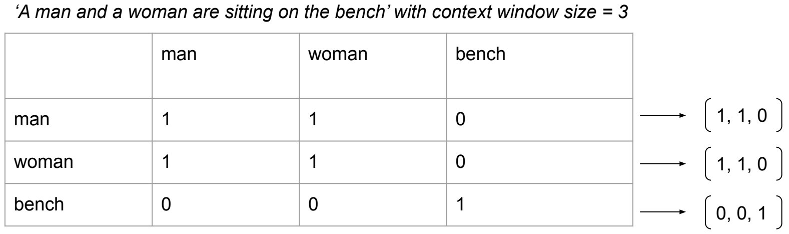


Figure 3.1: One-Hot encoding



'A man and a woman are sitting on the bench' with context window size = 3

	man	woman	bench
man	1	1	0
woman	1	1	0
bench	0	0	1

Figure 3.2: Example of co-occurrence embeddings

can be counted how many times a word occurs with other words in a fixed-sized window within a text corpus where the window represents the context a word is used (see Figure 3.2). A different method is to count how many times a word occurs in different documents where the documents represent the context. The counts can be binary, normal counts, or weighted occurrences like TF-IDF.

Pointwise Mutual Information (PMI) is a metric for co-occurrence (Jurafsky & Martin, 2009, p. 696). It is normally used to quantify the probability that two words occur together. For example, words like *Las* and *Vegas* usually co-occur. As it could be that it is just likely for both words to occur together because they are very frequent in the text, the frequency of the single words has to be taken into account for the calculation of the co-occurrence probability. PMI is then calculated by

$$PMI(x,y) = \log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{3.1}$$

where x and y are two words and $P(x,y)$ is the probability of both words occurring together, and $P(x)$ is the probability of a single occurrence of x.

This metric can then be used to calculate word embeddings by defining the context as surrounding words in an l-sized window and calculating the PMI for a word and context pair.

Latent Semantic Analysis (Deerwester et al., 1990) uses the frequencies of words across documents to create a term-document matrix. For this either counts or normalized counts like TF-IDF can be used. The matrix is then factorized by applying Singular Value Decomposition which yields dense topic vectors for words and documents.

Predictive Methods

Predictive models try to predict the context a word is used where the context is defined as the l-sized window of surrounding words within a sentence. Models like Word2Vec from Mikolov et al. (2013) achieved better results on word similarity tasks than previous methods.

As Word2Vec is one of the most used word embeddings, it shall be explained in more detail. The basic method of Word2Vec is to loop through a text corpus with a sliding window with a fixed length. Depending on the architecture, the model will either predict the surrounding words (Skip-Gram) or the word that is used within the window (CBOW). The Skip-Gram will be explained further.

The probability of the outside context word o conditional on the central word c is then calculated by a softmax function:

$$P(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w \in V} exp(u_w^T v_c)}$$

where V is the set of all words in the vocabulary, v the set of word embeddings and u the set of context embeddings. The similarity is measured as the dot product of the corresponding word vectors. The objective function for one central word c and one context word o is then the negative log-likelihood:

$$-\log P(o|c) = -\log \frac{exp(u_o^T v_c)}{\sum_{w \in V} exp(u_w^T v_c)} \tag{3.2}$$

The text corpus is iterated through a fixed-sized window. This results in the average negative log-likelihood as the final objective function with T as the number of positions in the text corpus, d as the context window length, and w_j as a word at position j:

$$-\frac{1}{T} \sum_{t=1}^{T} \sum_{-d<j<d} log(P(w_{t+j}|w_j)) \tag{3.3}$$

Common publicly available word embeddings are the Word2Vec embeddings, trained on the Google News corpus. It was shown by Levy and Goldberg (2014) that Word2Vec leads to a factorization of the word-context matrix using the pointwise mutual information which indicates a strong relationship of Word2Vec word embeddings with co-occurrences of words. One disadvantage of the architecture is the loss of positional information as all words in the context are treated equally. The original Word2Vec embeddings used a context window of five to ten words (Mikolov et al., 2013).

Most predictive models have the disadvantage of encoding all word meanings into a single embedding (E. H. Huang et al., 2012; Pilehvar & Collier, 2016). For example, the embedding of *bank* from Word2Vec encodes the meaning of a river bank and a financial bank. Sense2Vec from Trask et al. (2015) mitigated this problem by building word sense embeddings rather than word embeddings. The architecture is based on Word2Vec. It differs from Word2Vec as it uses a sense annotated corpus and defines context as surrounding word senses. This approach allows the creation of word sense embeddings which are distinguished by their part-of-speech tag. For example, the word sense embedding for *work* can be distinguished as *work* as a noun or as a verb. Still, this approach does not work for homonyms that have the same part-of-speech tag like *bank* as a river bank or a financial bank.

Pilehvar and Collier (2016) created word sense embeddings by leveraging the semantic network of WordNet and static word embeddings. The goal is to place word senses in a given semantic space, e.g. from Word2Vec so that the word sense embeddings are close to words that are semantically related to the word sense. For example, for a given synset, e.g *digit* as an anatomical part of the body, the word sense embedding would be placed close to the word embeddings from *dactyl*, *finger* and *toe*. Therefore, the words of the synset, e.g *digit* or *finger*, are retrieved from WordNet. As the synset is defined as the group of synonymous words, the word sense embedding can then be constructed based on the word embeddings of these synonyms. The list of related words is further enlarged by using a graph algorithm to retrieve other synsets and the respective words that are related to the target synset. Finally, the new word sense embeddings are created by de-conflating the original word embeddings. This is done by minimizing the distance between the new word sense embedding and the word embeddings, obtained from the list of related words from the previous step. These embeddings shall be referred to as Deconflated embeddings.

The GloVe embedding model from Pennington et al. (2014) can be seen as a combination of count-based and predictive models. It uses co-occurrence counts to optimize a dense word embedding matrix.

Knowledge-based Embeddings

Saedi et al. (2018) used the WordNet knowledge base to create distributional word embeddings, called Wnet2Vec. Instead of text and the resulting co-occurrence between words, they used the semantic network of WordNet to define word semantics. The similarity is then defined by the number and length of paths between two word nodes in the network.

Transformer-based Embeddings

Bommasani et al. (2020) investigated the possibility to create decontextualized embeddings from contextualized embeddings and specified two important methods. The first method describes the subword pooling and the second method describes the combination of contexts.

Subword pooling determines the way how representations of subwords of a word in a fixed context are aggregated to generate one representation for the word. For example, the mean, maximum, minimum, or the last of representations can be used. If a word w in a given context c is not present in the tokenizer vocabulary, w will be split into k subwords, yielding multiple representations $\{w_c^1, ..., w_c^k\}$. The subword pooling function f defines how the k subword representations are then combined to a single word representation w_c. The following formula specifies this process and is taken from Bommasani et al. (2020).

$$w_c = f(w_c^1, ..., w_c^k); f \in min, max, mean, last \tag{3.4}$$

The second method defines how contexts are used. The authors defined the strategy *Decontextualized* where only the word itself is used and no context. Thus, a static decontextualized word embedding can be created. The strategy *Aggregated* is defined as the usage of n contexts $\{c_1, ..., c_n\}$ per word, e.g. multiple sentences containing the word. This leads to n word representations $\{w_{c1}, ..., w_{cn}\}$. It shall be noted, that the term *decontextualized* in this thesis refers to the process of aggregation rather than the removal of context. As contextualized embeddings change with their respecting context, an aggregation over multiple contexts is needed to create static embeddings. Then the word embeddings are aggregated, for example by taking the mean, maximum, or minimum of all embeddings of a specific word. The following formula specifies this process and is taken from Bommasani et al. (2020).

$$w = g(w_{c1}, ..., w_{cn}); g \in min, max, mean \tag{3.5}$$

The authors evaluated their approaches by using intrinsic word similarity tasks and showed higher correlations when mean aggregation across contexts and subword representations were used ($f = mean$ and $g = mean$). The authors compared the performance of static, contextualized, and decontextualized word embeddings by an intrinsic evaluation of word similarity. The datasets Wordsim-353, Simlex-999, RG-65, and SimVerb-3500 were used.

Word2Vec and GloVe were used as examples for static embeddings. Contextualized and decontextualized word embeddings were extracted from the language models BERT, GPT-2, RoBERTa, DistilBERT, and XLNet. They further showed that decontextualized word embeddings can outperform classic static word embeddings. Furthermore, the results indicate that the usage of more contexts for aggregating contextualized word embeddings as well as larger models results in better performance. The best results are achieved by a BERT model with 24 layers and one million total contexts. Embeddings from early layers perform better than later layers. As the analyses are only based on the total number of contexts, a new analysis shall be performed in this work, to find out, what number of contexts per word shall be used.

Akbik et al. (2019) also investigated the process of aggregating word embeddings. The authors evaluated their embeddings on the downstream task Named Entity Recognition. They only used contextualized word embeddings and did not take the comparison to classic static embeddings into account. One contextualized embedding per word was used as the baseline. This means that only one context was used. The results show a slightly better performance for the aggregated embeddings.

Gupta and Jaggi (2021) used a CBOW-based static word embedding model that incorporates contextualized word embeddings of the context words.

It was shown by Ethayarajh (2019), Li et al. (2020) and Mu and Viswanath (2017) that static and contextualized word embeddings are anisotropic. The embedding vectors are placed in a cone in the embedding space where the overall similarity is relatively high (Li et al., 2020). Mu and Viswanath (2017) proposed a postprocessing method to create more isotropic embeddings. This is done by "(a) removing the nonzero mean vector from all word vectors, effectively reducing the energy; (b) projecting the representations away from the dominating D directions, effectively reducing the dimension." (Mu & Viswanath, 2017). The authors showed improved correlation for postprocessed Word2Vec and GloVe word embeddings on word similarity, word analogy, semantic text similarity, and category prediction tasks.

3.2 Evaluation of Word Embeddings

As one goal of this thesis is the creation of new word embeddings, an evaluation of recent embeddings is done. This includes intrinsic evaluation with different similarity rating datasets and extrinsic evaluation on different downstream tasks.

Static word embeddings have been intrinsically evaluated many times with different word similarity datasets where it was shown that word embeddings can reflect semantic meaning based on similarity and relatedness (Hill et al., 2015; Agirre et al., 2009; Bojanowski et al., 2017).

Rubinstein et al. (2015) investigated how well human semantic knowledge is represented by distributional word embeddings. The authors use a supervised task to predict semantic features of concrete nouns, taken from the semantic feature norm of McRae et al. (2005), e.g. *an animal* for the word *rabbit* or *has ears* for the word *dog*. They showed that taxonomic features like *a mammal* are harder to predict than attributive features like *is large* or *is green*.

This finding correlates with the results from Lucy and Gauthier (2017) where word embeddings from Word2Vec and GloVe were also used to predict semantic features of concrete objects, taken from the semantic feature norms of McRae et al. (2005) and Devereux et al. (2014). The authors showed that multiple embedding models encode perceptual features worse. They further show a correlation between this result and wrong cosine similarities between words.

Grand et al. (2022) built upon the idea that words can be differently similar depending on the dimension, e.g *size* or *dangerousness* and that this information might be encoded in word embeddings. The authors used word embeddings from GloVe to investigate whether word embeddings can be projected onto lines, representing these dimensions and whether these projected embeddings reflect human similarity judgements. These lines are then defined as the vector differences between antonyms like *large* and *small* for the dimension *size* (Grand et al., 2022). Using external human similarity ratings, it was shown that some dimensions can be modeled by these projections.

Chronis and Erk (2020) proposed a method to create multi-prototype embeddings based on contextualized word embeddings from the BERT-base model. Instead of creating static word embeddings with one embedding per word, the authors used K-Means clustering to calculate centroids per word usage type. According to the authors, single contextualized word embeddings are more sensitive to outliers and are not suitable to calculate similarity. The authors further state that this approach does not always lead to word sense clusters but often differentiates clusters of homonyms. Instead of the cosine similarity between word embeddings, the cosine similarity between the two closest centroids is used to determine the similarity between two words. This approach leads to higher correlations of similarity ratings with human-created similarity ratings. While this approach leverages the ability of contextualized word embeddings to reflect the context, it does create multiple embeddings per word and does not assign word sense identification to the centroids.

Hollenstein et al. (2019) compared the performance of static and contextualized word embeddings by an extrinsic evaluation of the predictiveness of brain activity. The evaluation was done across multiple modalities (EEG, fMRI, and eye tracking data) and multiple datasets per modality. The authors used Word2Vec, GloVe, WordNet2Vec, Fasttext (Bojanowski et al., 2017) as static word embeddings. Contextualized word embeddings were extracted from

BERT and ELMo (Peters et al., 2018). Embeddings from BERT, Fasttext, and ELMo achieved the lowest mean errors. This indicates higher predictiveness of contextualized word embeddings for brain data.

Ethayarajh (2019) investigated how contextual contextualized word embeddings are and whether these models create highly context-specific word embeddings or assign a word sense embedding depending on the context. To answer these questions, the author defined different measures of contextualization. Self-similarity measures the cosine similarity of word embeddings for a specific word across different contexts. A high self-similarity value means that the context does not influence the word embedding whereas a low self-similarity means a high contextualization. The results show that the self-similarity decreases with later layers, indicating a strong effect of contextualization on the word embeddings. The results further show that contextualized word embeddings are anisotropic. The average cosine similarity between random words gets higher with later layers. This effect leads to almost completely similar word embeddings in GPT-2.

Tenney et al. (2019) investigated where contextualized word embeddings improve compared to static word embeddings. The authors used different syntactic and semantic tasks and found out that contextualized word embeddings improve more on syntactic tasks rather than semantic tasks.

3.3 Word Sense Disambiguation

Simple methods like using the most frequent word sense or the Lesk algorithm only need WordNet as a resource and not a labeled training dataset (Jurafsky & Martin, 2009). As word senses are sorted by frequency in WordNet, it is an easy baseline to predict the word sense. The Lesk algorithm uses the provided word sense explanation in WordNet to choose the word sense that has the most overlap of words with the target sentence.

To create more powerful embeddings that are respecting the sense of a word, it is crucial to either have an annotated corpus or to have an algorithm that predicts the sense of a word in a given sentence. One way could be to use contextualized embeddings as they can encode word meaning (Chronis & Erk, 2020; Wiedemann et al., 2019). Therefore, Wiedemann et al. (2019) investigated whether contextualized word embeddings could be used for predicting the sense of a word in a given sentence. They found out "that CWEs in general are able to capture senses, i.e. words, when used in a different sense, are placed in different regions. This effect appeared strongest using the BERT pre-trained model, where example instances even form clusters. This might give rise to future directions of investigation, e.g. unsupervised word sense-induction using clustering techniques." (Wiedemann et al., 2019).

Similar approaches leverage the ability of contextualized word embeddings to capture context (L. Huang et al., 2020; Melamud et al., 2016; Peters et al., 2018). Using a labeled sense dataset, for each sense, an averaged contextualized word sense embedding can be produced. Then the word sense can be predicted for any word in a sentence by using a 1-nearest-neighbor search with the respective contextualized word embedding. This method relies on the training dataset and can only predict word senses available in the training dataset.

To overcome this problem, Scarlini et al. (2020b) used contextualized word embeddings from BERT-large but also knowledge-based methods to be not restricted to labeled word senses. The authors propose a semi-supervised algorithm ARES which generates embeddings for all WordNet word senses. The authors also perform aggregation of contextualized embeddings over multiple contexts, applied k-means clustering to cluster these embeddings, and assigned a word sense to each cluster. With these embeddings, it is possible to predict the word sense of a word in a given sentence by using the word sense of the nearest neighbor embedding. For each synset, sentences containing any word from the synset are crawled. Then the contextualized word embeddings are extracted from BERT and clustered using k-means clustering under the assumption that words are used with the same meaning in sentences in the same cluster. Then using the knowledge-based method UKP the clusters are labeled with a word sense.

3.4 Creation of Sentence Embeddings

Reimers and Gurevych (2019) used the BERT architecture to efficiently create sentence embeddings. The authors state that using BERT for sentence pair regression leads to good results but is also limited in computational efficiency as every sentence pair of interest has to be processed by BERT. This makes the usage of BERT as a cross encoder of sentence pairs unusable in real-world tasks like clustering millions of sentences.

Therefore, they further investigated the possibility to extract meaningful sentence embeddings from the pretrained available BERT model which can be used in more efficient downstream tasks. This was done by two approaches. First by extracting token embeddings from the last layer and a pooling mechanism to create one sentence embedding and second by using the classification token (CLS). The results show that these sentence embeddings often lead to worse performance than using static word embeddings like GloVe. As a consequence, the authors decided to finetune a BERT model on different supervised tasks to create meaningful sentence embeddings.

Depending on the dataset, different training objectives and network architectures were used (see Figure 3.3). For tasks with sentence pairs, a siamese network architecture was used. For tasks with three sentences as the input, a triplet architecture was used. For classification tasks, the training objective was to predict the corresponding label to the input data. The cross-entropy loss was used as the loss function in this case. For regressions tasks, the training objective was to predict a numerical value, indicating the similarity between sentences, e.g. a cosine similarity value. The mean squared error was used as a loss function in this case. The third possible training objective was the triplet objective function where three embeddings from an anchor element, a similar and a dissimilar example are compared, where the similar example is expected to have a small distance to the anchor element and the dissimilar example is expected to have a large distance to the anchor element.

Token embeddings were extracted from the last layer and were pooled by either using the CLS token, average pooling, or max pooling of all tokens. The finetuning was done across different datasets and tasks. One approach was the finetuning on the SNLI (Bowman et al., 2015) and MultiNLI dataset (Williams et al., 2018) using the classification objective where the task is to predict whether two sentences are contradicted, neutral, or entailed. A second approach used this finetuning and a second finetuning on the train set of the STS dataset (Cer, Diab, et al., 2018), where the regression objective was used. In the STS dataset, the task is to predict a numerical value from 0 (dissimilar) to 5 (similar) between two sentences.

For evaluation, the spearman correlation between the similarity vector,

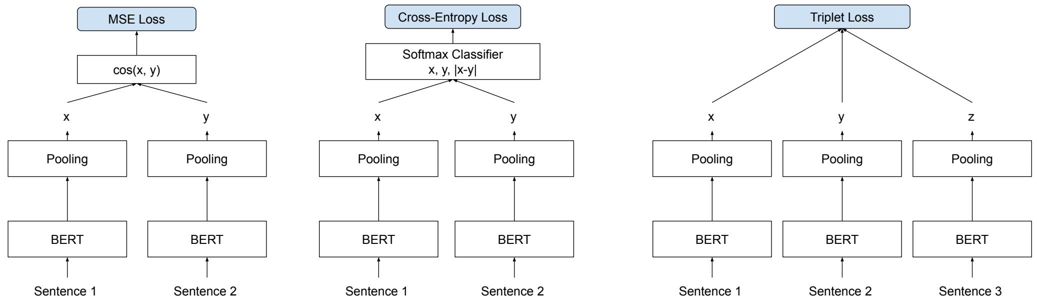


Figure 3.3: Overview of Sentence-BERT architectures using the mean squared error loss (left), cross-entropy loss (middle), and triplet loss (right). Adapted from Reimers and Gurevych (2019).

based on cosine similarity between sentence embeddings, with the true similarity vector from the STS test set was used.

Several baselines were compared. First, average pooled static word embeddings from GloVE were evaluated as they provide fast access to word embeddings. Second, similarities, directly predicted by using BERT as a cross encoder, were evaluated as this approach promised the best results. And third, state of the art sentence embedding models like InferSent (Conneau et al., 2017) and Universal Sentence Encoder (Cer, Yang, et al., 2018) were evaluated.

The results show that BERT used as a cross encoder leads to the best performance. The approaches of the authors show highly competitive performance with the advantage of being more computationally efficient than the cross-encoder approach.

Furthermore, the authors evaluated the usage of these pooled sentence embeddings for other downstream tasks like sentiment prediction where the embeddings were used to train a logistic regression classifier. Though the goal of their work was not to produce embeddings for downstream tasks, the finetuned embeddings had the best performance in most datasets.

Finally, it was shown, that pretrained embeddings extracted from the last layer of BERT are not suited for the sentence similarity task. Their performance was below the performance of static word embeddings. To improve the quality, a finetuning step was performed which resulted in high-quality sentence embeddings. This might indicate a similar expectation for word similarity.

Chapter 4

Datasets, Corpora, and Models

4.1 Models

To generate contextualized embeddings, several Transformer-based models shall be used (Table 4.1). The models BERT, and GPT-2 are chosen as they are widely used Transformer models and are the foundation of lots of bigger language models like GPT-3. As BERT is trained with a masked language modeling objective and GPT-2 is trained with a causal language modeling objective, both of these objectives are covered, too. For both model families, the base version with twelfth layers and the large version with 24 layers are used. Pretrained model weights are used from the *transformers*[1] library. The BERT models are pretrained on the BookCorpus and English Wikipedia, which contain around three billion words together. The GPT-2 models are pretrained on the WebText corpus. Decontextualized embeddings were produced with the same models. The procedure to extract word, word sense, and synset embeddings from Transformer models will be explained later.

Finetuned models are used from the Sentence-BERT release[2]. The original Sentence-BERT implementation from Reimers and Gurevych (2019) is used as a comparison to the pretrained BERT-base model and the Sentence-DistillRoBERTa model is used as it is a recent and high-performing implementation on the Semantic Textual Similarity task.

4.2 Static Word Embeddings

Different kinds of static word embeddings shall be investigated (Table 4.2). One-hot encoding and count-based methods will be ignored as they are out-

[1] https://github.com/huggingface/transformers

[2] https://www.sbert.net/

Table 4.1: Overview of used Transformer models and their characteristics: The number of encoder or decoder layers, the length of input tokens, and the dimensionality of the resulting embeddings

Name	Number layers	Input length	Dimensionality
BERT-base	12	512	768
BERT-large	24	1024	1024
GPT-2	12	512	768
GPT-2-medium	24	1024	1024
Sentence-BERT	12	512	768

Table 4.2: Overview of used static word embeddings and their characteristics: the dimensionality of embeddings and the embedding level, e.g. word or word sense

Name	Embedding dimensionality	Level
Word2Vec	300	Word
GloVe	300	Word
Deconflated	300	Word Sense

dated.

The Word2Vec embedding model (Mikolov et al., 2013), as well as the GloVe embedding model (Pennington et al., 2014), are used as examples for static word embeddings because they are widely used in research and industry. The pretrained embeddings from the *gensim*[3] library are used. For Word2Vec, the word embeddings trained on the Google News corpus with a dimensionality of 300 are used. This corpus contains around three billion words.

The Deconflated embedding model (Pilehvar & Collier, 2016) is used as an example for static word sense embeddings. The embeddings are retrieved from the official release[4].

4.3 Similarity Datasets

Human word similarity judgements have been studied and collected multiple times (Agirre et al., 2009; Luong et al., 2013; Hill et al., 2015) and have been used for intrinsic evaluation of word embeddings (Bommasani et al., 2020; Chronis & Erk, 2020).

The similarity judgements from Wordsim-353 (Agirre et al., 2009) and Simlex-999 (Hill et al., 2015) are used as word similarity judgements as they

[3] https://github.com/RaRe-Technologies/gensim

[4] https://pilehvar.github.io/deconf/

Table 4.3: Examples of similarity ratings from the Wordsim-353 dataset

Word 1	Word 2	Similarity judgements
love	sex	6.77
tiger	cat	7.35
tiger	tiger	10.00
book	paper	7.46
computer	keyboard	7.62

Table 4.4: Examples of similarity ratings from the Simlex-999 dataset

Word 1	Word 2	Similarity judgements
old	new	1.58
smart	intelligent	9.2
hard	difficult	8.77
happy	cheerful	9.55
hard	easy	0.95

are widely used in the area of natural language processing research. The Wordsim-353 dataset consists of 353 word pairs and average human similarity judgements that were collected through an online study with 500 participants. The similarity values range from 0 for dissimilar to 10 for similar (see Table 4.3). The Simlex-999 dataset consists of 999 word pairs and average human similarity judgements that were collected through a study. The similarity values range from 0 for dissimilar to 10 for similar (see Table 4.4). The authors used a more strict definition of similarity and try to reduce the similarity scores for related words like *closet* and *clothes* which are scored more similar in the Wordsim-353 dataset.

A recent similarity judgements dataset from Hebart et al. (2020) contains continuous similarity ratings in the range from 0 (dissimilar) to 1 (similar). They are based on human similarity ratings from 5.983 participants using a triplet odd-one-out task with triplet images and a computational model.

As the definition of similarity can vary, it is important to compare how similarity is specified in the overlap of the datasets. Therefore, Table 4.5, Table 4.6 and Table 4.7 show differences and similarities between Simlex-999, Wordsim-353 and THINGS. THINGS similarity judgements are continuous values, ranging from 0 (dissimilar) to 1 (similar). They seem to be high for word pairs that belong to the same conceptual superordinate category, like *cat* and *dog* or *man* and *woman*. *Cat* and *dog* are both animals and are similar on a range of dimensions like *outdoors-related*. Wordsim-353 has similar ratings

Table 4.5: Differences in similarity ratings between THINGS and Wordsim-353 datasets

Word 1	Word 2	THINGS	Wordsim-353
money	bank	0.39	8.50
train	car	0.96	6.31
bank	money	0.39	8.12
bird	crane	0.23	7.38
football	basketball	0.94	6.81
television	film	0.68	7.72
cucumber	potato	0.86	5.92
gem	jewel	0.87	8.96
television	radio	0.88	6.77
tiger	cat	0.93	7.35

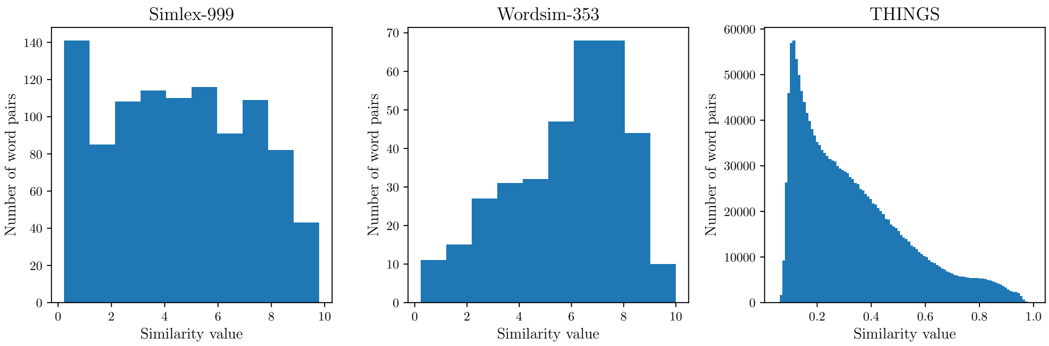


Figure 4.1: Distribution of similarity ratings per dataset

for these word pairs. In contrast to THINGS and Simlex-999, Wordsim-353 similarity judgements for related words seem to be high, e.g. for *money* and *bank*. Both words do not belong to the same concept nor are they synonyms. Thus, this word pair has a medium to low similarity value in THINGS. Further, Wordsim-353 has high similarity values for synonyms like *coast* and *shore*. Similarly, Simlex-999 has high similarity values for synonyms but is more restricted for related word pairs, like *clothes* and *closet*.

Figure 4.1 shows the distribution of similarity ratings per dataset. The Simlex-999 dataset seems to have a more uniform distribution whereas the Wordsim-353 dataset has more similar ratings than dissimilar and the THINGS dataset has more dissimilar ratings.

Table 4.6: Differences in similarity ratings between THINGS and Simlex-999 datasets

Word 1	Word 2	THINGS	Simlex-999
bottle	container	0.31	7.93
dog	cat	0.92	1.75
arm	knee	0.98	2.75
cat	rabbit	0.93	2.37
door	gate	0.65	5.25
purse	bag	0.78	8.33
chair	bench	0.72	6.67
dog	horse	0.92	2.38
boat	anchor	0.90	2.25
wood	log	0.75	7.30

Table 4.7: Differences in similarity ratings between Simlex-999, Wordsim-353, and THINGS datasets

Word 1	Word2	THINGS	Simlex-999	Wordsim-353
woman	man	0.88	3.33	8.3
clothes	closet	0.87	3.27	8.0

4.4 Text Corpora

Several text corpora are used in this study. First, the Wikitext-2 and the Wikitext-103 corpora from Merity et al. (2017) were chosen for their high quality. Wikitext-2 consists of 250.000 tokens and Wikitext-103 of 150.000.000 tokens. The Wikitext corpora are only made of articles that are rated good or featured on Wikipedia. Furthermore, they were cleaned and preprocessed. For example, mathematical terms or HTML artifacts were removed. Words with a frequency less than three were replaced by the *unk* token. An example paragraph can be seen in Figure 4.2.

As the THINGS dataset also samples less frequent objects, the Wikitext corpora did not cover all words from the dataset. This can be seen in Table 4.8. Wikitext-2 and Wikitext-103 do not contain all 1,854 target words from the THINGS dataset. Therefore, the bigger Wikidump[5] dataset is used. Yet, Wikitext-103 already contains all target words from the Simlex-999 and Wordsim-353 datasets. Wikidump was still used for these target words, to use

[5] https://dumps.wikimedia.org/

```
= = = Locations = = =
Gods were linked with specific regions of the universe .  In
Egyptian tradition , the world includes the earth , the sky ,
and the Duat .  Surrounding them is the dark <unk> that existed
before creation .
```

Figure 4.2: Example text paragraph from the Wikitext-2 corpus

Table 4.8: Number of target words covered in corpora per similarity dataset

Text corpus	**THINGS (n=1,854)**	**Wordsim-353 ∪ Simlex-999 (n=1,341)**
Wikitext-2	770	1,247
Wikitext-103	1728	1,341
Wikidump	1,854	1,341

the same text corpus across similarity ratings. Using the Wikiextractor[6] tool, the XML dump was transformed into text.

[6] https://github.com/attardi/wikiextractor

Chapter 5

Methods

Word embeddings from Transformer models are contextualized as they depend on the text that is fed into the model. To create static word embeddings, these contextualized word embeddings have to be decontextualized. A simple approach is to use one context and extract the respective word embeddings. A more sophisticated approach is the usage of multiple contexts as they will contain more information about word meaning. Thus, extracted word embeddings from multiple contexts need to be aggregated to create a single embedding per word.

Decontextualized word embeddings and word sense embeddings are created in this study. Next to the aggregation method, the algorithm can be run over a different number of context sentences that should be used to aggregate. Also, the number of layers or the pooling method of values between the layers can be changed.

Therefore, this chapter will explain how static word embeddings are retrieved, how contextual embeddings are extracted from different models, and how they are decontextualized to create new static Transformer-based word embeddings.

5.1 Creation of Static Word Embeddings

Static word embeddings from Word2Vec and GloVe are extracted from the lookup tables from the *gensim* library. Deconflated word sense embeddings are retrieved from the official dataset[1] from Pilehvar and Collier (2016).

[1] https://pilehvar.github.io/deconf/

5.2 Creation of Transformer-based Embeddings

The process to extract embeddings from Transformer models, as well as the decontextualization, will be explained in this section. First, the retrieval of text paragraphs and the prediction of word senses will be explained. Second, the extraction process across models and hyperparameters will be specified. Last, the decontextualization of these extracted contextualized embeddings will be defined.

5.2.1 Finding Synonyms

To increase the number of available contexts, embeddings are also created for synonyms of the target words. A strict method for generating synonyms is applied when the synset of a word is known. In this case, all words of this synset are used as synonym candidates. Afterwards, all belonging synsets to that word are extracted and the word is considered as a synonym when the most frequent synset is the same as the synset of the target word. If all words of the target synset are used, synonym pairs like *aardvark* and *anteater* would be created as they share at least one synset in WordNet. The proposed approach yields fewer synonym pairs but of more quality like *car* and *automobile*.

Further, other spellings of a word are used. This is done to increase the coverage of words in the corpus, provide enough context for aggregation, and counteract misspellings in the word list. For example, the THINGS dataset contains the words *ice cream* and *iceskate*. For *ice cream*, the spellings *icecream* and *ice-cream* are used. For *iceskate* the spelling *ice skate* is used.

5.2.2 Finding Word Occurrences

Afterwards, paragraphs in the text corpus are searched through whether they contain a word from the word list. This leads to a high number of occurrences for high frequent words like *table* and low number of occurrences for low frequent words *aardvark* (Figure 5.1). For computational efficiency, a maximum number of $n = 1000$ paragraphs is sampled per word (Figure 5.2).

5.2.3 Annotation of Word Senses

As one goal is the creation of word sense embeddings, the word sense of a word in a paragraph is predicted by using the ARES sense embeddings from Scarlini et al. (2020b). All sense embeddings of the word are used as candidates. As the ARES sense embeddings are calculated in the doubled vector space of *BERT-large*, the word embeddings for each occurrence of the word in the

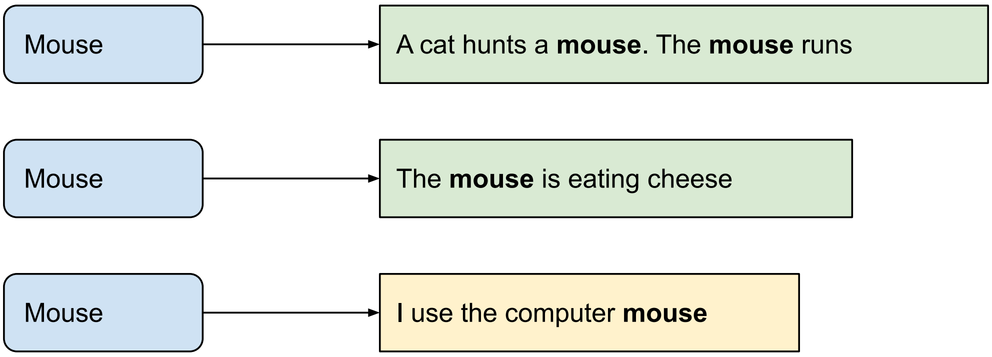

Mouse
A cat hunts a **mouse**. The **mouse** runs
Mouse
The **mouse** is eating cheese
Mouse
I use the computer **mouse**

Figure 5.1: Examples of target words and paragraphs

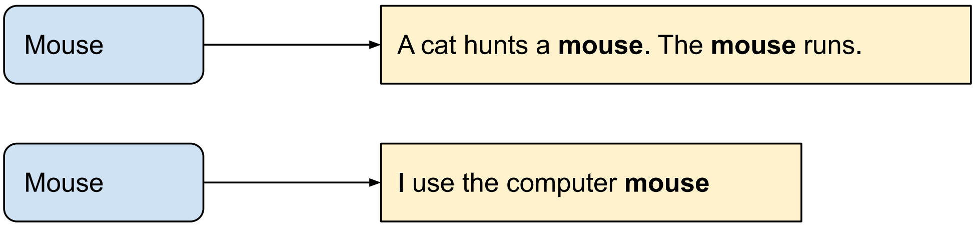

Mouse
A cat hunts a **mouse**. The **mouse** runs.
Mouse
I use the computer **mouse**

Figure 5.2: Examples after sampling n paragraphs per target word

paragraph have to be extracted from the *BERT-large* model. Following the work by Scarlini et al. (2020b), embeddings are extracted from the top 4 layers with mean pooling between layers. The embedding is then concatenated with itself to match the vector space. The word sense of the nearest neighbor sense embedding is then used as the word sense of the target embedding. After this step, for each word in the target list, the paragraph where it occurs and the predicted word sense and synset at each position in the paragraph are known (see Figure 5.3).

5.2.4 Extraction of Transformer-based Embeddings

Using this information, the contextualized word embeddings are extracted from the defined models with defined hyperparameters. Hyperparameters are the layer, where hidden states are extracted and the subtoken pooling method if the word gets tokenized into subwords.

The pretrained Transformer models are used through the *transformers* li-

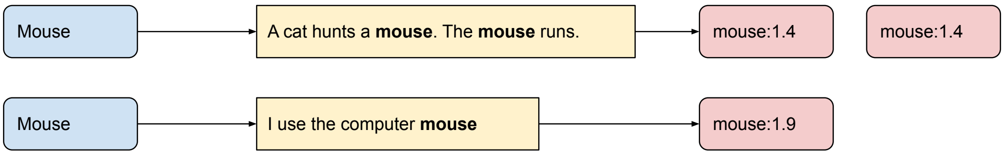

Mouse
A cat hunts a **mouse**. The **mouse** runs.
mouse:1.4
mouse:1.4
Mouse
I use the computer **mouse**
mouse:1.9

Figure 5.3: Examples of the word sense annotation process

brary. To use the Transformer-based models, the input text first has to be tokenized using the matching tokenizer from the transformers library. Each model defines its own tokenization and special tokens. For example, the BERT model family uses a *CLS* token at the beginning of the sentence as a classification token and a *SEP* token to separate and end sentences. The GPT-2 tokenizer considers spaces that will create different tokens depending on whether a word is at the beginning or the middle of a sentence.

To extract the hidden states for the target word, the hidden states of the matching word tokens are extracted. In case a word gets tokenized into multiple subtokens, the hidden states of the subtokens are mean aggregated.

5.2.5 Decontextualization of Embeddings

After the extraction of the contextualized embeddings for all occurrences of all target words, they are decontextualized by aggregating all word embeddings extracted from different contexts. Hyperparameters for the decontextualization are the number of embeddings that are used, the level of aggregation, e.g. whether word embeddings or word sense embeddings shall be created, the aggregation method, e.g. mean aggregation, and the usage of postprocessing methods like dimensionality reduction or anisotropy processing.

Here, a mean aggregation across context and subword representations is used, as it was shown to produce better results under intrinsic evaluation (Bommasani et al., 2020). Using the notation from Bommasani et al. (2020), g and f are set to *mean*.

5.2.6 Produced Embedding Dataset

Table 5.1 shows the total number of word, word sense, and synset embeddings that were extracted for all THINGS objects from the Wikidump text corpus as well as how many objects from the 1,854 THINGS objects have at least one embedding. The creation of word sense and synset embeddings leads to an incomplete embedding dataset where not all objects from the THINGS

Table 5.1: Number of embeddings for the THINGS dataset and the coverage per embedding level.

	Total number embeddings	Coverage of objects of 1,854
word	1,607,468	1,852
word sense	820,555	1,084
concept	1,620,330	1,854
synset	803,915	1,017

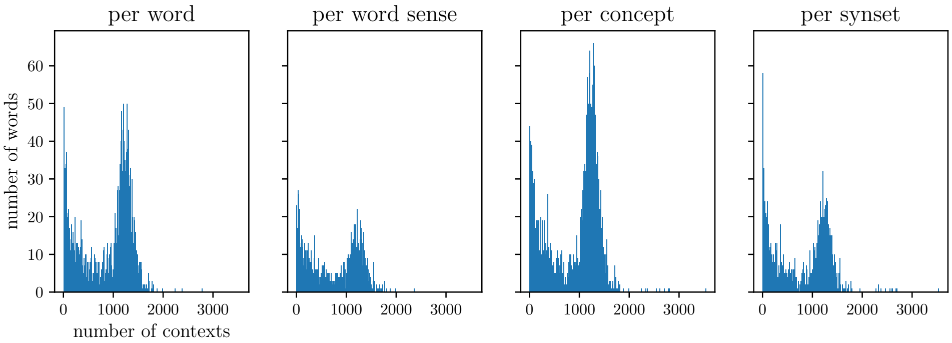


Figure 5.4: Histogram for THINGS embeddings

dataset are covered. This can be either caused by missing word senses in the ARES word sense embedding dataset or by false word sense predictions.

Figure 5.4 shows the histogram of the number of contexts per word, word sense, and synset. It can be seen that the number of contexts is lower for word sense and synset embeddings. For computational performance reasons, a limit of 1000 contexts per word is used. As the sampling is done on the word level, this limit can not be ensured for word senses and synsets. Figure 5.5 shows the matching boxplot for the THINGS dataset. The results show that at least 50% of the 1,854 THINGS objects have 1000 embeddings per word.

Figure 5.6 shows the histogram of the number of contexts per word for words from the Simlex-999 and Wordsim-353 datasets. Figure 5.7 shows the matching boxplot. For almost all words, at least 1000 contexts were found in the text corpus.

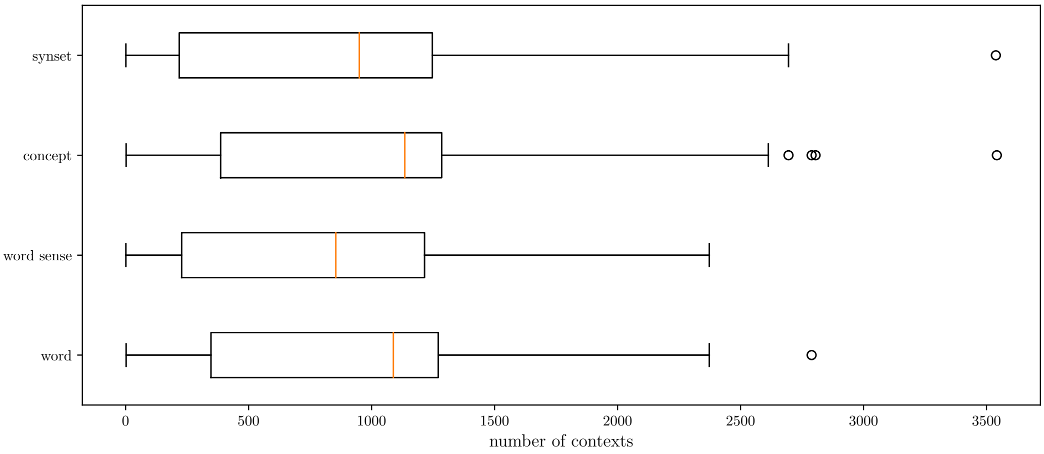


Figure 5.5: Boxplot for word, word sense, and synset embeddings for the THINGS dataset

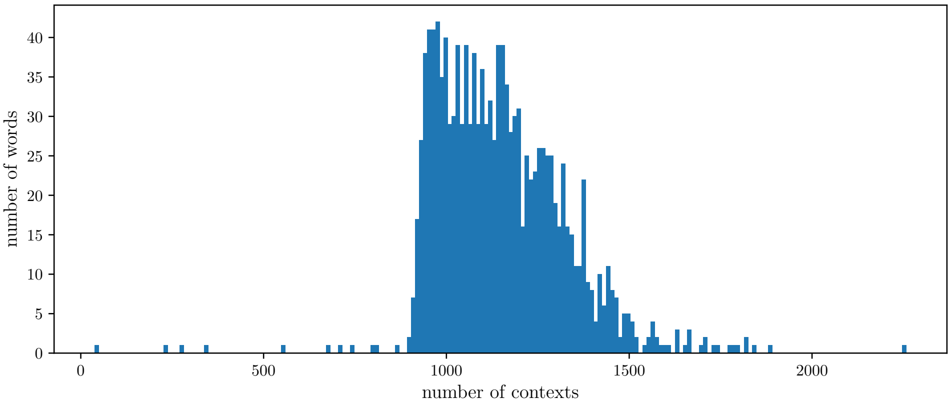


Figure 5.6: Histogram for word embeddings from the Simlex-999 and Wordsim-353 datasets

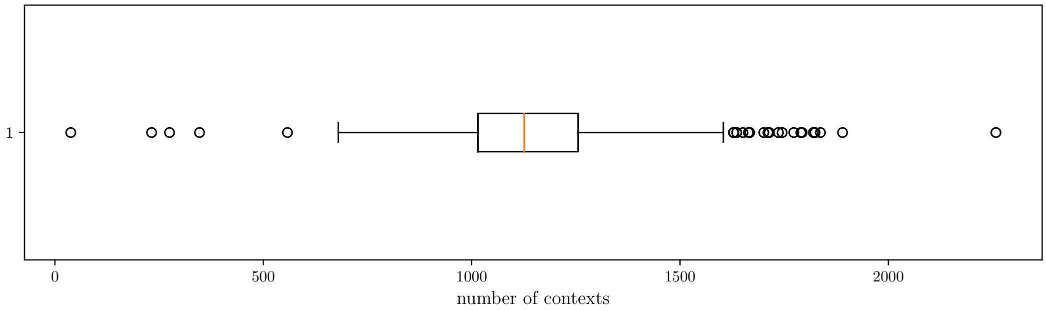


Figure 5.7: Boxplot for word embeddings from the Simlex-999 and Wordsim-353 datasets

Algorithm 1 Postprocessing algorithm to increase isotropy of word embeddings, adapted from Mu and Viswanath (2017)

```
Input : Word representations $\{v(w), w \in V\}$, a threshold parameter D,
Compute the mean of $\{v(w), w \in V\}, \mu \leftarrow \frac{1}{|V|} \sum_{w \in V} v(w), \tilde{v}(w) \leftarrow v(w) - \mu$
Compute the PCA components: $u_1, ..., u_d \leftarrow PCA(\{\tilde{v}(w), w \in V\})$.
Preprocess the representations: $v\prime(w) \leftarrow \tilde{v}(w) - \sum_{i=1}^{D} (u_i^T v(w)) u_i$
Output : Processed representations $v\prime(w)$.
```

5.3 Isotropy Postprocessing

To improve the quality of the generated word embeddings, the method from Mu and Viswanath (2017) shall be used (see Algorithm 1). First, the mean vector is subtracted from all embeddings. Afterwards, the principal components are calculated and the first D components are removed. Static, contextualized, and decontextualized word embeddings are postprocessed using this method. The number of dimensions D to be removed is set to $D = n/100$ with n as the embedding dimensionality which follows the recommendation of the authors. This method is not applied by default and will be covered in its own section in the results chapter.

5.4 Retraining of Embeddings

To increase representational similarity based on word embeddings, a retraining model is created to transform the decontextualized word embeddings. For this, the similarity ratings from the THINGS dataset are used as the ground truth label. The objective of the model is then to learn the similarity defined in this dataset. Word embeddings from the last layer of the BERT-base model are retrained. The last layer is chosen under the assumption that it encodes the most information. The retraining is only applied to the embeddings when it is stated in the result chapter. The retraining is done using a supervised contrastive-learning-based approach.

5.4.1 Model

Similar to the approach by Reimers and Gurevych (2019), a siamese network is used to learn new word embeddings. Their approach uses two BERT models which take a sentence as input each. Afterwards, the word embeddings from the BERT model are extracted and pooled into a sentence embedding. The sentence embeddings are then used in the loss function to optimize the BERT-model to produce new embeddings that satisfy the learning objective.

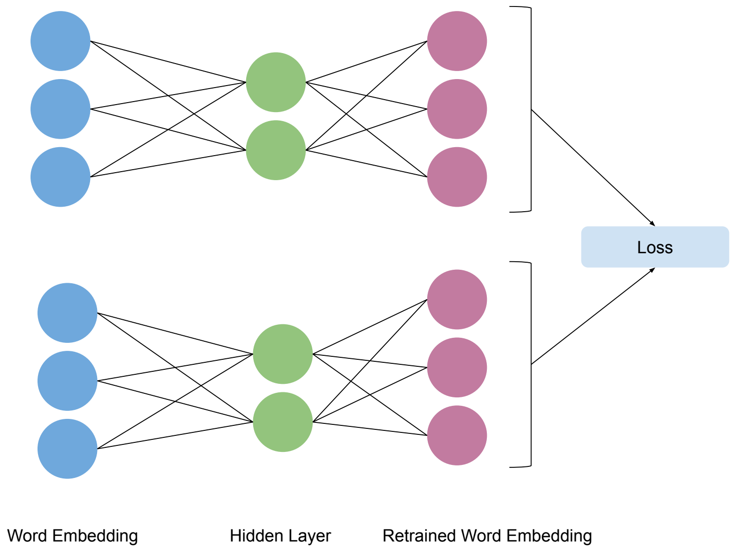


Figure 5.8: Overview about the model architecture. Layer sizes are variable and depend on the embedding dimensionality and the chosen hyperparameters.

In contrast, this model uses word embeddings as input as the embeddings are already extracted. It is built of two feedforward neural networks with shared weights (see Figure 5.8). The model consists of an input layer with a dimensionality of 768 and multiple hidden layers and an output layer. The number of layers and their dimensionality are set as hyperparameters and are chosen based on the hyperparameter search. The output dimensionality was chosen to create embeddings with similar dimensionality to common static word embeddings like Word2Vec. A Rectified Linear Unit (ReLU) activation function was used between all layers.

5.4.2 Dataset

The goal of the retraining approach is the creation of embeddings that better capture the semantic knowledge in the THINGS similarity dataset. For this, a supervised training is performed with the similarity ratings from THINGS as ground truth. The embedding dataset, containing the BERT-base embeddings for all 1,854 THINGS objects, is split into a train, validation, and test fold. Word pairs are only generated within a split (see Figure 5.9). For the train and validation sets, five word embeddings are chosen randomly for each word in a pair. The word pairs from the validation set are used for the hyperparameter search and the word pairs from the test set are used for the final evaluation.

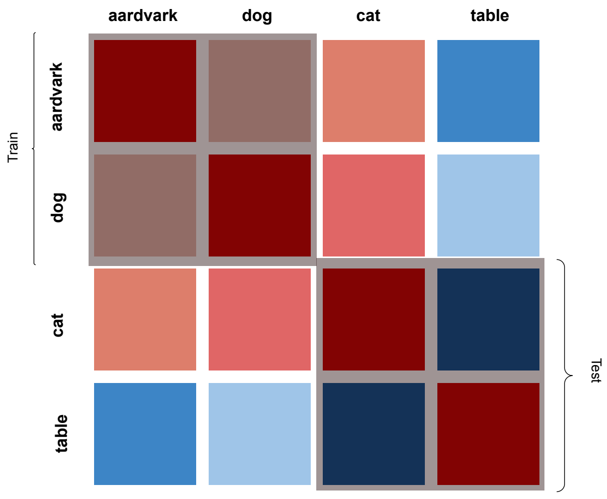


Figure 5.9: Exemplary overview of the dataset split. The validation set is left out for simplicity reasons.

5.4.3 Training and Hyperparameters

The model is trained using supervised methods with different loss functions and hyperparameters.

Three different loss functions are investigated. This follows the work by Reimers and Gurevych (2019) who showed that different loss functions impact the results. First the mean squared error loss:

$$\frac{1}{D}\sum_{i=1}^{D}(\hat{y}_i - y_i)^2 \tag{5.1}$$

with $\hat{y}$ as the cosine similarity, y as the true similarity rating from the THINGS dataset, and D as the number of training samples.

Second, the mean absolute error loss:

$$\frac{1}{D}\sum_{i=1}^{D}|\hat{y}_i - y_i| \tag{5.2}$$

Contrastive loss is used as a third possible loss function. It decreases the distance between similar word pairs and increases the distance between dissimilar word pairs. Given a pair x and y and their true similarity rating Y, the loss minimizes the distance for $Y = 1$ and maximizes the distance for $Y = 0$. The term m defines the distance to which dissimilar pairs are optimized.

$$\frac{1}{D}\sum_{i=1}^{D} Y_i \cdot d(x_i, y_i)^2 + (1 - Y_i) \cdot max(0, m - d(x_i, y_i))^2 \qquad (5.3)$$

As the similarity of object images is a continuous value from 0 to 1, with 0 being dissimilar, contrastive loss can be used with the similarity value as ground truth. The margin value for dissimilar pairs m is set to 1. The distance between two pairs is then defined as the cosine distance with respect to the cosine similarity from equation 2.8:

$$d(x_i, y_i) = cosine\_distance(x, y) = 1 - cosine\_similarity(x, y) \qquad (5.4)$$

The mean squared error loss and mean absolute error loss are chosen because they are widely used loss functions and were also used by Reimers and Gurevych (2019) for finetuning BERT. The contrastive loss is chosen as it is directly based on the distance between word embeddings which is later used for the intrinsic evaluation.

Several hyperparameters were used and evaluated using the hyperparameter sweeping from Weights&Biases[2]. These include the batch size, dropout rate, weight decay, the number of layers, and the number of neurons per layer (see Table 6.2). For each loss function, a hyperparameter search with 100 trainings is performed. All combinations are trained on 20 epochs with an early stopping criteria to prevent overfitting. The final hyperparameters are chosen based on the correlation between similarity ratings, based on the retrained embeddings, and the ground truth similarity ratings from THINGS. Only words from the validation set are used for this evaluation. Dropout is disabled during evaluation mode.

[2] https://wandb.ai/site/sweeps

Table 5.2: Hyperparameters and their ranges and distributions that were sampled from.

Hyperparameter	**Values**	**Distribution**
Learning Rate	$[0.00001, 0.1]$	Log Uniform
Dropout Rate	$[0, 0.9]$	Uniform
Batch Size	$(1000, 5000, 10000)$	-
Weight Decay	$(0, 0.001, 0.0001, 0.00001)$	-
Number of Neurons per Layer	$(10, 50, 100, 300, 600)$	-
Number of Layers	$(1, 2, 3)$	-

Chapter 6

Experimental Results

This chapter first investigates how well human conceptual knowledge is represented in embeddings. The intrinsic evaluation is based on human conceptual knowledge from several human-based similarity ratings and embedding-based similarity ratings. Human-based similarity ratings are used from datasets where humans were asked to rate the similarity between two entities like words or images. Embedding-based similarity ratings are created by calculating the similarity between the embeddings of two words. As similarity ratings are from humans by definition, the embedding-based similarity ratings are not formally ratings but this term will be further used for simplicity.

Afterwards, two downstream tasks are evaluated using an extrinsic approach where the embeddings are used to predict dimension values and superordinate categories from the THINGS dataset.

6.1 Representational Similarity with Human Similarity Ratings

This section explores how well these similarity judgements are modeled by different kinds of word embeddings, especially by Transformer-based word embeddings, and how the process of decontextualization affects this evaluation.

The results are shown first for the Simlex-999 word similarity dataset. Second, the results for the Wordsim-353 word similarity dataset are presented and afterwards the results for the THINGS similarity dataset are shown. For each, a correlation analysis is performed to investigate the representational similarity between humans and different word embeddings. The analysis is first performed for static embeddings and contextualized embeddings from pretrained Transformer-based models. Afterwards, embeddings from finetuned models are investigated as well as the impact of decontextualization, isotropy processing and the number of contexts for aggregation. Finally, the similarity ratings are investigated between extraction layers.

The representational similarity analysis from Kriegeskorte et al. (2008) is used to study how well the similarity ratings from the THINGS dataset are reflected in different representations. The embeddings for the THINGS dataset are further investigated whether the word sense annotation is helpful and whether there is an impact of word frequency and retraining. Using a reweighting method, it is further investigated whether the information is fully exploited in the word embeddings.

Human conceptual knowledge is based on similarity ratings that were either completely collected by studies with human participants or partially collected and modeled with a computational model. The similarity between words is defined as the cosine similarity between the corresponding embeddings. Cosine Similarity is chosen as it is commonly used in natural language processing (Reimers & Gurevych, 2019; Saedi et al., 2018):

$$cos(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{6.1}$$

The intrinsic evaluation is done by calculating a similarity score between all word pairs provided by the datasets like *mouse* and *computer*. As the word similarity datasets do not provide word senses, word embeddings are chosen. The similarity score is based on the respective word embeddings and the used similarity or distance function. Highly similar pairs will get a similarity score of 1 whereas highly dissimilar pairs will get a similarity score of -1. Afterwards, the resulting similarity ratings will be correlated with the similarity ratings from a ground truth similarity rating dataset using the spearman correlation.

6.1.1 Word Similarity Ratings from Simlex-999

Figure 6.1 shows the results of the intrinsic evaluation for several word embeddings based on word similarity from the ground truth dataset Simlex-999.

Pretrained Models and Static Word Embeddings

Classic static word embeddings are widely used in research and industry. The recent success of pretrained Transformer-based models like BERT also showed the importance of these models. Therefore, the investigation of representational similarity with humans is important.

Predicted similarity ratings from all embeddings have high correlations with human word similarity ratings from the Simlex-999 dataset.

Static word embeddings from Word2Vec have a correlation of 0.44 whereas embeddings from GloVe result in a correlation of 0.37 and Deconflated embeddings result in a correlation of 0.45.

Contextualized word embeddings from pretrained Transformer-based models perform similarly. The BERT-base model achieves a correlation of 0.5 with embeddings from the static embedding layer. The correlation decreases afterwards with later layers. The correlation is lowest for the eleventh layer with 0.29. Similarity ratings predicted from embeddings from the BERT-large model have a similar correlation pattern where the correlation of embeddings from the first embedding layer is the highest with 0.51. It decreases with later layers and reaches the minimum of 0.29 at the last layer.

Contextualized word embeddings from the GPT-2 model start with a low correlation of 0.2 from the static embedding layer. The correlation increases with later layers where the third layer has the highest correlation with 0.44. With the next layers, the correlation decreases to almost zero in the last layer. Embeddings from the GPT-2-medium layer follow a similar pattern. In contrast to GPT-2, the embeddings from the static embedding layer start with a high correlation which then decreases at the first decoder layer. The correlation increases until the 15th decoder layer and decreases afterwards until the last layer. The minimum correlation is at the last layer with 0.04.

Figure 6.2 shows the scatter plot of similarity ratings from humans with similarity ratings based on decontextualized word embeddings that were extracted from the last layer of BERT-base and GPT-2 and similarity ratings based on static word embeddings from Word2Vec. It shows a similar distribution between BERT-base and Word2Vec embeddings. It can be further seen that word embedding pairs from GPT-2 are almost all highly similar.

Finetuned Models

Pretrained Transformer-based models are trained with self-supervised methods. Therefore, it is possible that semantic similarity is not encoded as well as possible. It was shown that finetuning these models can lead to better performance while some information retains (Merchant et al., 2020). Therefore, public available finetuned models on similarity tasks might encode word similarity better.

Contextualized word embeddings from BERT models that were finetuned on the semantic text similarity task have a similar correlation with pretrained BERT models. Embeddings from the finetuned Sentence-BERT model start with a similar correlation as the pretrained BERT-base model. The correlation decreases with later layers but is higher than the correlation of static word embeddings until the eleventh layer. The last layer has the lowest correlation of 0.23. The second layer has the highest correlation of 0.53. Embeddings from the finetuned Sentence-Distill-RoBERTa model have similar correlations compared to embeddings from the Sentence-BERT model. The correlation of predicted similarity ratings is lowest for the last layer but higher than the correlation from static word embeddings.

Decontextualization

Following the last two sections, the investigation of decontextualized embeddings shows whether the aggregation across contexts increases the representational similarity with humans. For this, decontextualized word embeddings are created from pretrained and finetuned Transformer-based models.

Similarity ratings, based on decontextualized word embeddings, have higher correlations across all models and extraction layers.

Decontextualized word embeddings from the pretrained BERT-base model have a similar correlation through all layers. The difference between the correlation between contextualized and decontextualized word embeddings increases with later layers. The correlation of decontextualized word embeddings from the BERT-large model increases with later layers, reaching the maximum of 0.58 at the 19th layer. Afterwards, the correlation decreases slightly until the last layer. Similar to the BERT-base model, the difference in correlation between contextualized and decontextualized embeddings increases with later layers.

Decontextualized embeddings from the GPT-2 model have higher correlations in all layers than contextualized embeddings. This effect is higher in earlier layers where the correlation increases from 0.2 to 0.46 in the static embeddings layer. Yet, decontextualized embeddings from the last layer have a low correlation compared to previous layers. Similarly, correlations for decontextualized embeddings from the GPT-2-medium model are higher in all layers compared to contextualized embeddings. The maximum correlation of 0.55 is reached at the 12th layer.

Predicted similarity ratings from decontextualized embeddings of finetuned Sentence-BERT models have the highest correlations with human similarity ratings. The correlation for embeddings from the Sentence-BERT model increases with later layers, reaching the maximum at 0.65 at the eighth layer. The correlation for embeddings from the finetuned Sentence-Distill-RoBERTa model increases with later layers, reaching the maximum at the last layer at 0.62.

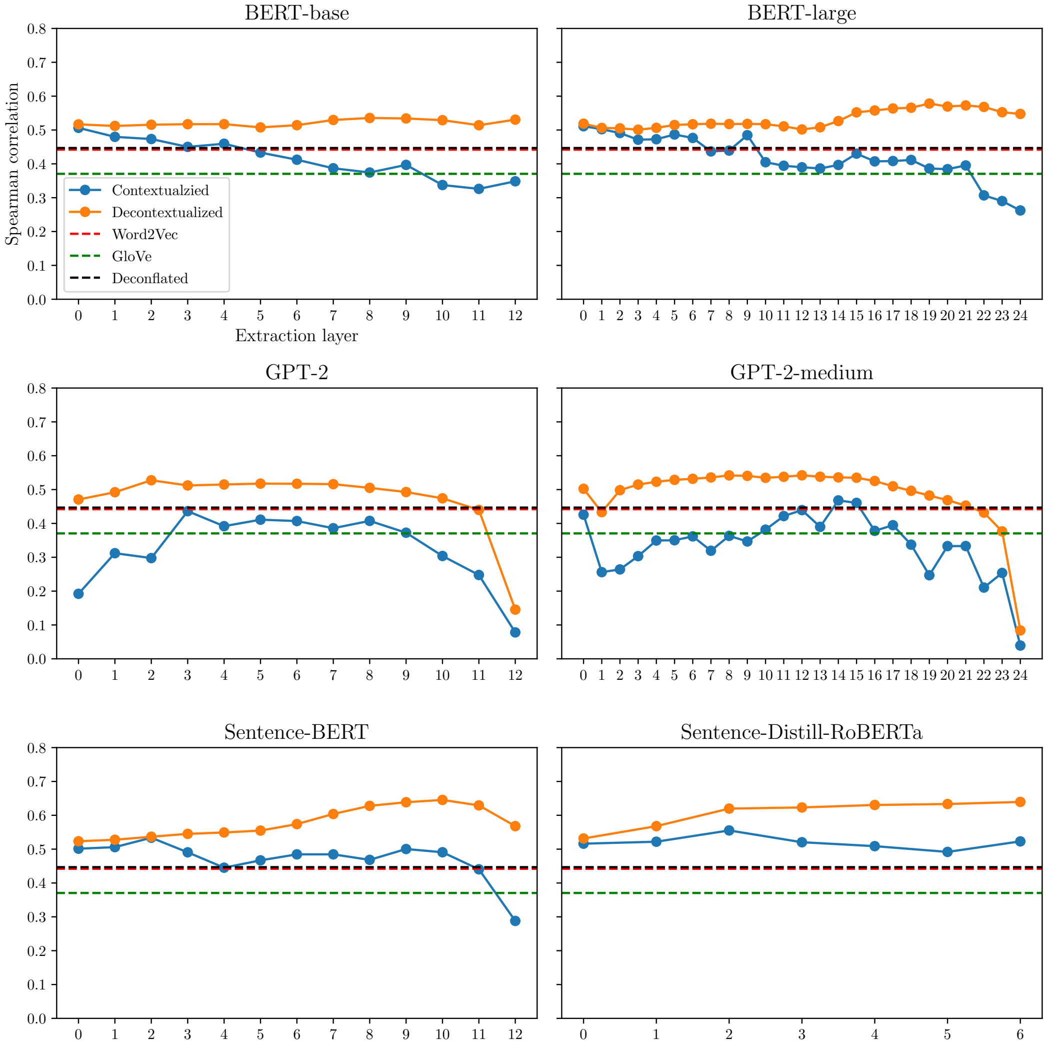


Figure 6.1: Spearman correlation of similarity ratings, based on embeddings, with word similarity ratings from the Simlex-999 dataset

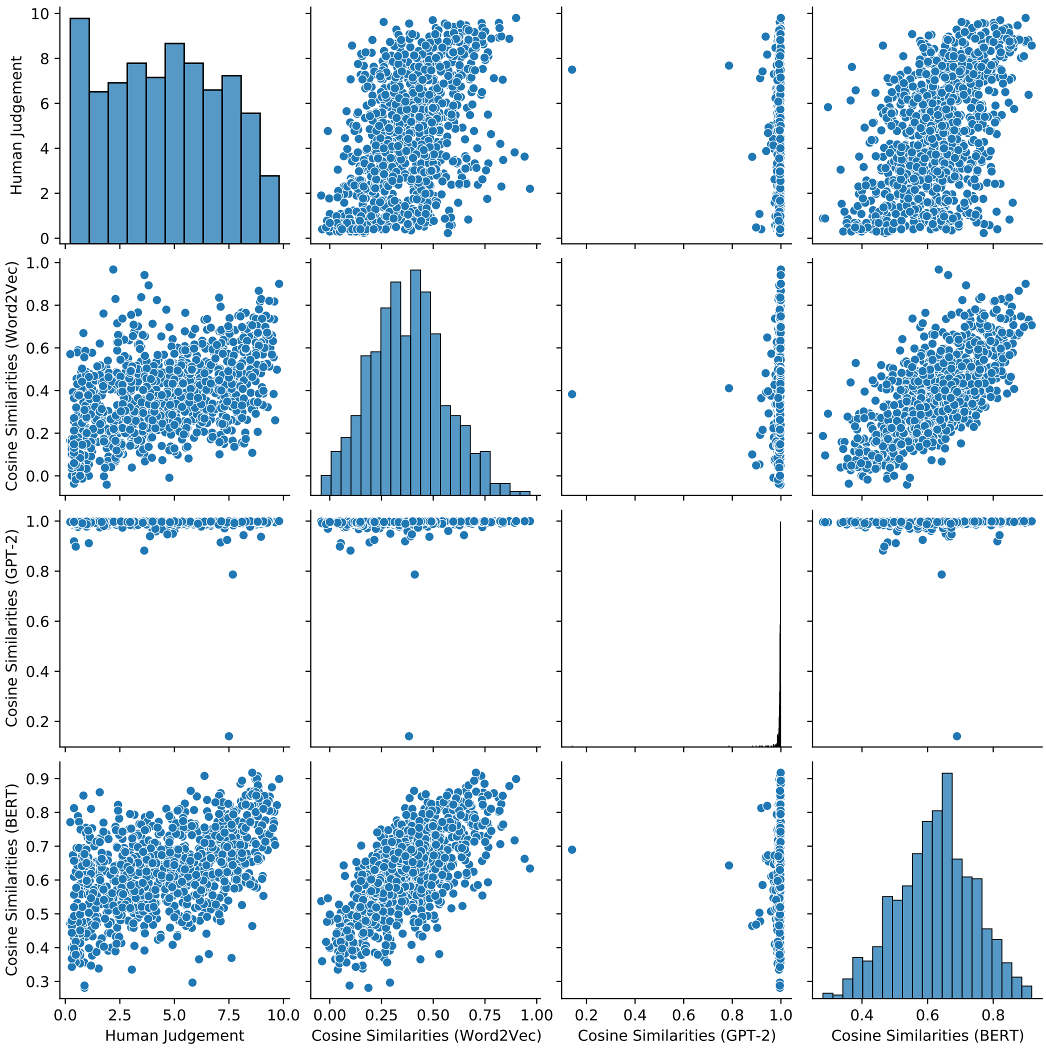


Figure 6.2: Similarity ratings by humans from the Simlex-999 dataset and similarity ratings based on decontextualized word embeddings from the last layer of BERT-base, GPT-2, and static word embeddings from Word2Vec.

Correlations per Layer and Number of Contexts

Decontextualization by aggregation over contexts depends on the number of contexts. Therefore, it might be possible that only a few contexts are enough to create high-quality static embeddings.

To investigate the impact of the number of contexts that were used for the aggregation of contextualized word embeddings, the evaluation is repeated with a fixed number of contexts per word.

Instead of using all available contexts, the number of contexts is set to a value in the range $[1, 10, 50, 100, 300, 500, 1000]$. The analysis is done for word embeddings from all Transformer-based models and all extraction layers.

The results are shown in Figure 6.3. For all models, the correlations increase for most layers with a higher number of contexts. Models that are based on the BERT architecture, like BERT-base, BERT-large, Sentence-BERT, and Sentence-Distill-RoBERTa show a higher increase of correlation for later layers than for early layers. There is no big improvement after 50 contexts are used for aggregation. Models that are based on the GPT architecture do not show a clear influence of the number of contexts on the layer hierarchy. Similar to the BERT models, most correlations do not improve after 50 contexts.

Similarity Predictions across Layers

Transformer models might encode different information across their layers and might encode different semantics. To investigate this question, similarity ratings, based on the word embeddings from all model layers, are compared to each other. This is done by calculating the spearman correlation between the similarity ratings from all layer pair combinations.

The results are shown in Figure 6.4. It can be seen that the correlation between similarity ratings, based on contextualized word embeddings from BERT-like models, decreases with later layers. This indicates that encoder layers encode different information about semantic similarity. In contrast, the correlation between similarity ratings, based on decontextualized word embeddings from BERT-like models, stays high across all layers. This indicates that the decontextualization process preserves information about semantic similarity.

Contextualized word embeddings from GPT- like models produce similarity ratings that are less correlated than similarity ratings from BERT-like models. Though, the decontextualization process leads to similar results where similarity ratings are highly correlated between all layers except the last layer which produces similarity ratings that are uncorrelated to similarity ratings from all other layers.

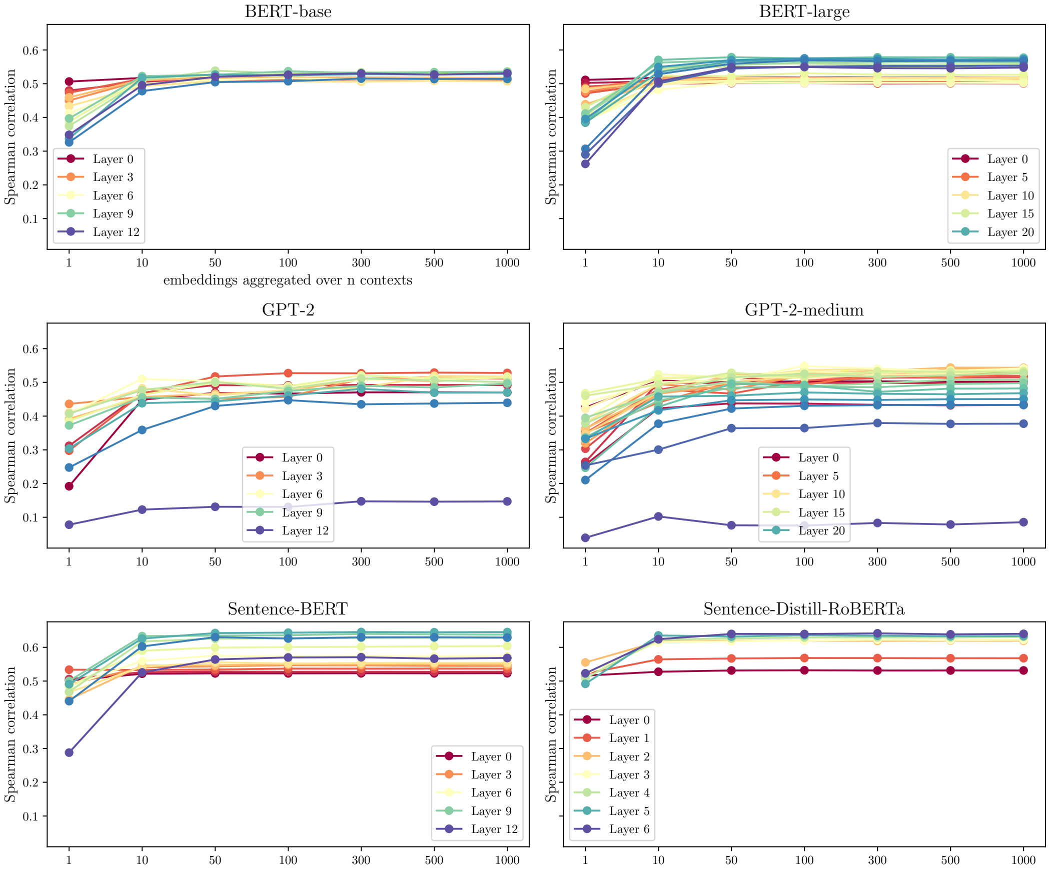


Figure 6.3: Correlation of similarity ratings, based on embeddings, with word similarity ratings from Simlex-999 per layer and number of contexts. Correlations are shown for all layers.

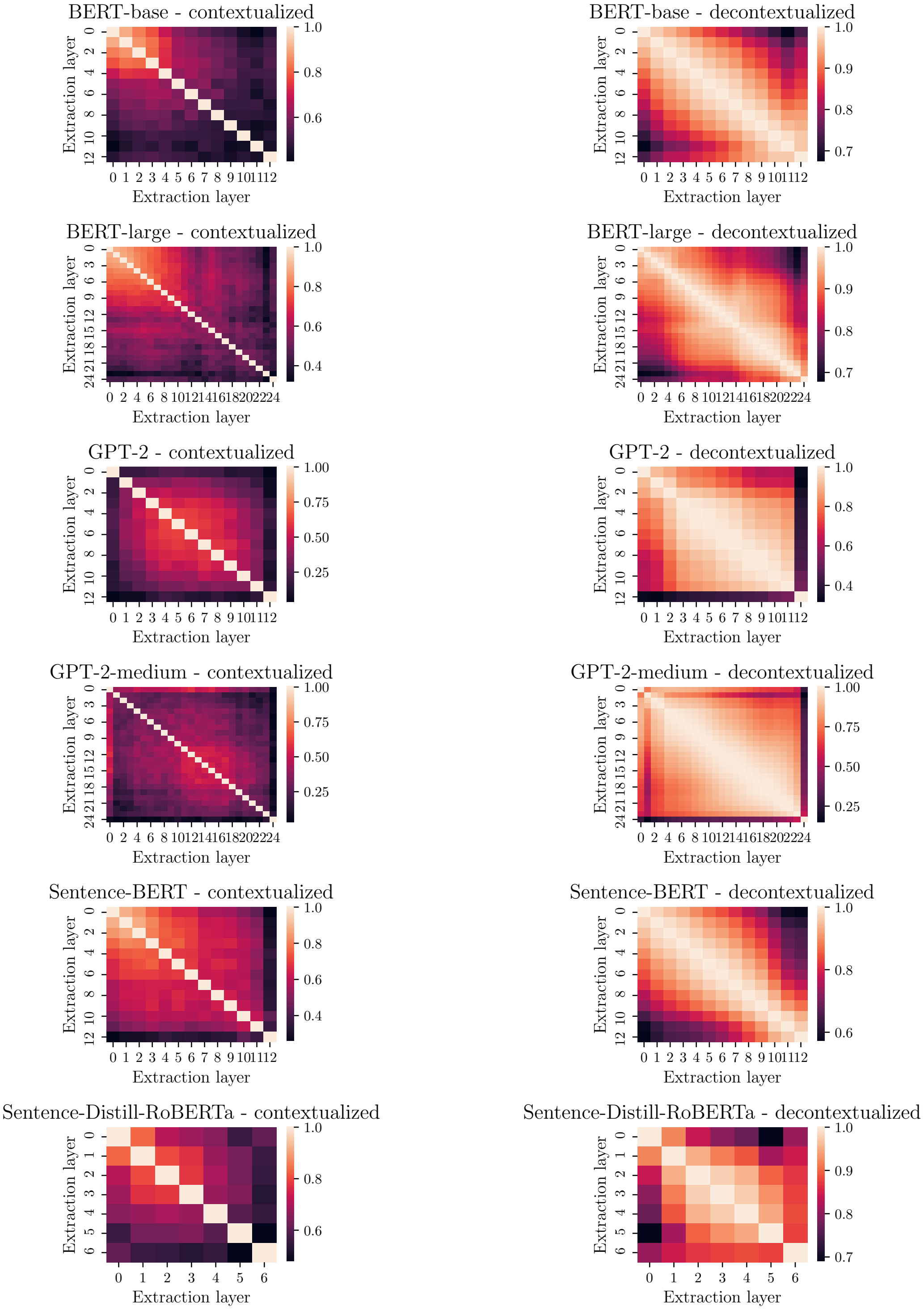


Figure 6.4: Correlation of similarity ratings, based on embeddings, between all extraction layers

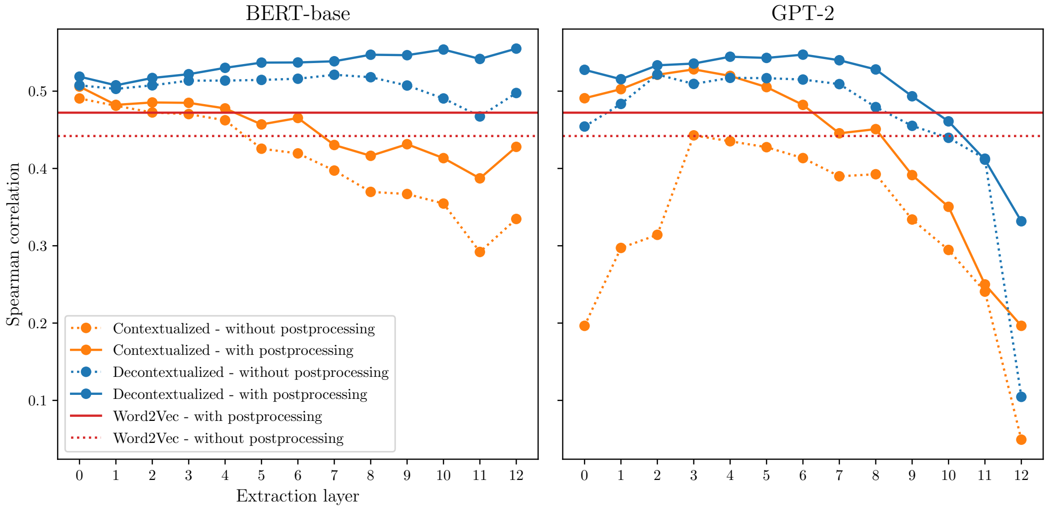


Figure 6.5: Effect of isotropy postprocessing on the correlation of similarity ratings, based on embeddings from BERT-base and GPT-2, with similarity ratings from the Simlex-999 dataset

Impact of Isotropy Postprocessing

It was shown by Mu and Viswanath (2017) and Ethayarajh (2019) that static and contextualized word embeddings suffer from anisotropy. Therefore, the isotropy postprocessing method from Mu and Viswanath (2017) is used to increase the correlation with similarity ratings from the Simlex-999 dataset.

Word embeddings from BERT-base and GPT-2 are exemplarily used as Transformer-based embeddings. Word2Vec embeddings are exemplarily used as static word embeddings.

The postprocessing leads to increased correlation for all word embeddings across all models and layers (see Figure 6.5). The improvement of the correlation with contextualized and decontextualized word embeddings from BERT-base gets higher with later layers.

6.1.2 Word Similarity Ratings from Wordsim-353

Figure 6.6 shows the results of the intrinsic evaluation of word embeddings based on word similarity from the ground truth dataset Wordsim-353.

Pretrained Models and Static Word Embeddings

Predicted similarity ratings from all embeddings have high correlations with human word similarity ratings from the Wordsim-353 dataset. Generally, they show a similar correlation pattern compared to correlations with similarity ratings from the Simlex-999 dataset.

Static word embeddings from Word2Vec have a correlation of 0.69 whereas embeddings from GloVe result in a correlation of 0.6. Deconflated embeddings result in a correlation of 0.65.

Contextualized word embeddings from the BERT-base model have the highest correlations at the static embedding layer with 0.66 and the lowest correlation of 0.37 at the penultimate layer. Embeddings from the BERT-large model have the maximum correlation of 0.69 at the second layer. It decreases with later layers, reaching the minimum of 0.4 at the penultimate layer.

Contextualized word embeddings from the GPT-2 model start with a correlation of 0.3 at the static embedding layer. It increases to the maximum of 0.55 at the fifth layer and decreases to the minimum of 0.12 at the last layer. In contrast, embeddings from the static embedding layer from the GPT-2-medium model have the highest correlation of 0.62. It decreases to the minimum of 0.11 at the last layer.

Finetuned Models

Contextualized word embeddings from the finetuned Sentence-BERT model have the highest correlation at the first encoder layer with 0.66 which decreases then with later layers until the last layer to 0.36. The finetuned Sentence-Distill-RoBERTa model has the highest correlation of 0.67 at the first encoder layer. It decreases slightly, reaching the minimum at the third layer with 0.56. It increases slightly until the last layer to 0.61.

Decontextualization

Similar to the correlations with similarity ratings from the Simlex-999 dataset, the process of decontextualization increases the correlation across all models and extraction layers.

For embeddings from the BERT model family, the correlations range between 0.58 and 0.73. The difference in correlations with contextualized word embeddings gets higher with later layers. The maximum correlation is reached at 0.73 at the sixth layer from the BERT-large model. Embeddings from the GPT-2 model have the highest correlation of 0.67 at the first decoder layer which decreases with later layers until the minimum of 0.29 at the last layer. For the GPT-2-medium model, the embeddings from the static embedding layer have the highest correlation of 0.7 which then decreases to the minimum of 0.31 at the last layer.

Similarly to the pretrained BERT models, the correlation of decontextualized embeddings from the finetuned models of Reimers and Gurevych (2019) is higher in later layers compared to the correlations from contextualized word embeddings. Similarity ratings based on embeddings from the Sentence-BERT model reach a maximum correlation of 0.7 and ratings based on embeddings from the Sentence-Distill-RoBERTa model reach a maximum correlation of 0.72.

Correlations per Layer and Number of Contexts

In the previous evaluation, all available contexts were used to create decontextualized embeddings. The results showed higher correlations for decontextualized embeddings compared to contextualized embeddings using one context. Bommasani et al. (2020) showed that the total number of contexts increases the correlation with several word similarity ratings, too. To further investigate the effect of the number of contexts on the correlation of similarity predictions, based on embeddings, with human similarity ratings, the previous evaluation was repeated with a different number of contexts for aggregation.

Only words with at least 1000 unique contexts were chosen to avoid upsampling of context for low frequent words. Afterwards, n contexts from [1, 10, 50, 100, 300, 500, 1000] were used to create and evaluate decontextualized word embeddings.

In Figure 6.7 the correlations are shown per model, extraction layer and number of contexts for aggregation. For pretrained and finetuned BERT models, the correlations converge after ten contexts. The correlation increases more at later layers. For pretrained GPT-2 models, the correlations converge after 50 contexts.

Similarity Predictions across Layers

Predicted similarity ratings, based on embeddings, are different depending on the extraction layer of the embeddings as seen in the previous analyses. To investigate whether different model layers encode similar or different information about the semantic meaning, the predicted similarity ratings for word pairs are correlated using the spearman correlation. For this analysis, contextualized word embeddings as well as decontextualized word embeddings are used to further investigate the influence of aggregation on layer encoding.

The results in Figure 6.8 show the correlation matrices between layers. For contextualized word embeddings from models based on the BERT architecture, the predicted similarity ratings from early layers have high correlations. The correlations decrease with later layers to a minimum of around 0.4. For decontextualized word embeddings, the predicted similarity ratings have generally higher correlations across layers. For embeddings from the BERT-base, BERT-large, and Sentence-BERT models, the correlations of predicted similarity ratings are high between all layer pairs until the last three layers. For embeddings from the GPT-2 and GPT-2-medium model, the correlations of predicted similarity ratings are high between all layer pairs except for the last layer.

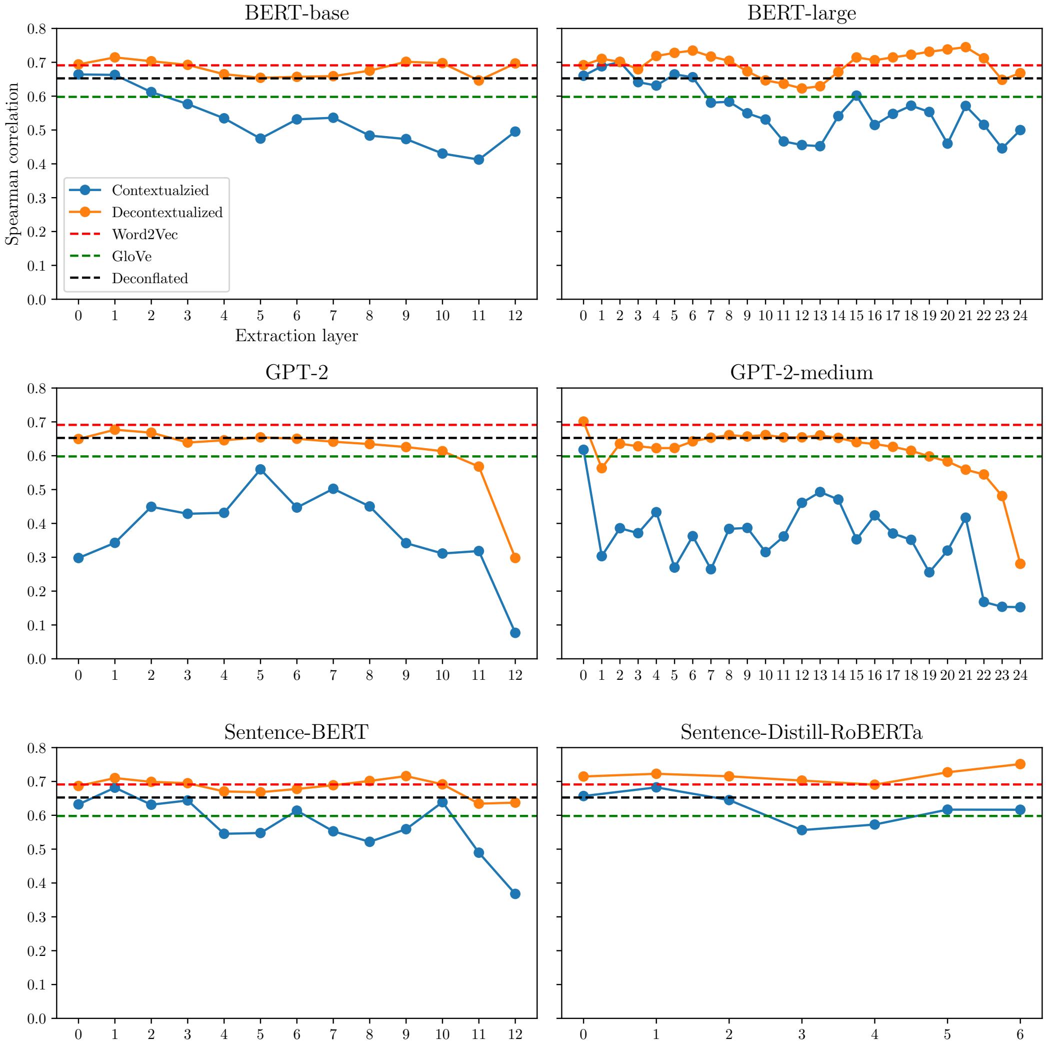


Figure 6.6: Spearman correlation of similarity ratings, based on embeddings, with word similarity ratings from the Wordsim-353 dataset

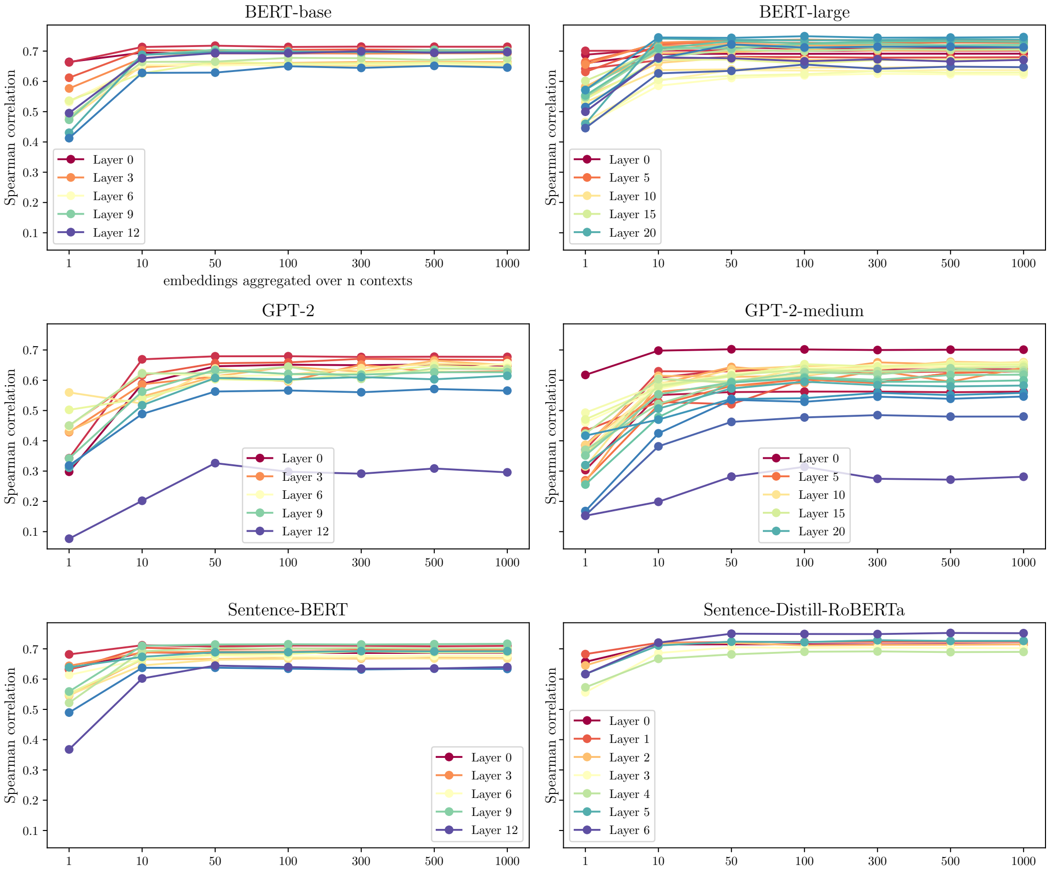


Figure 6.7: Correlation of similarity ratings, based on embeddings, with word similarity ratings from Wordsim-353 per layer and number of contexts

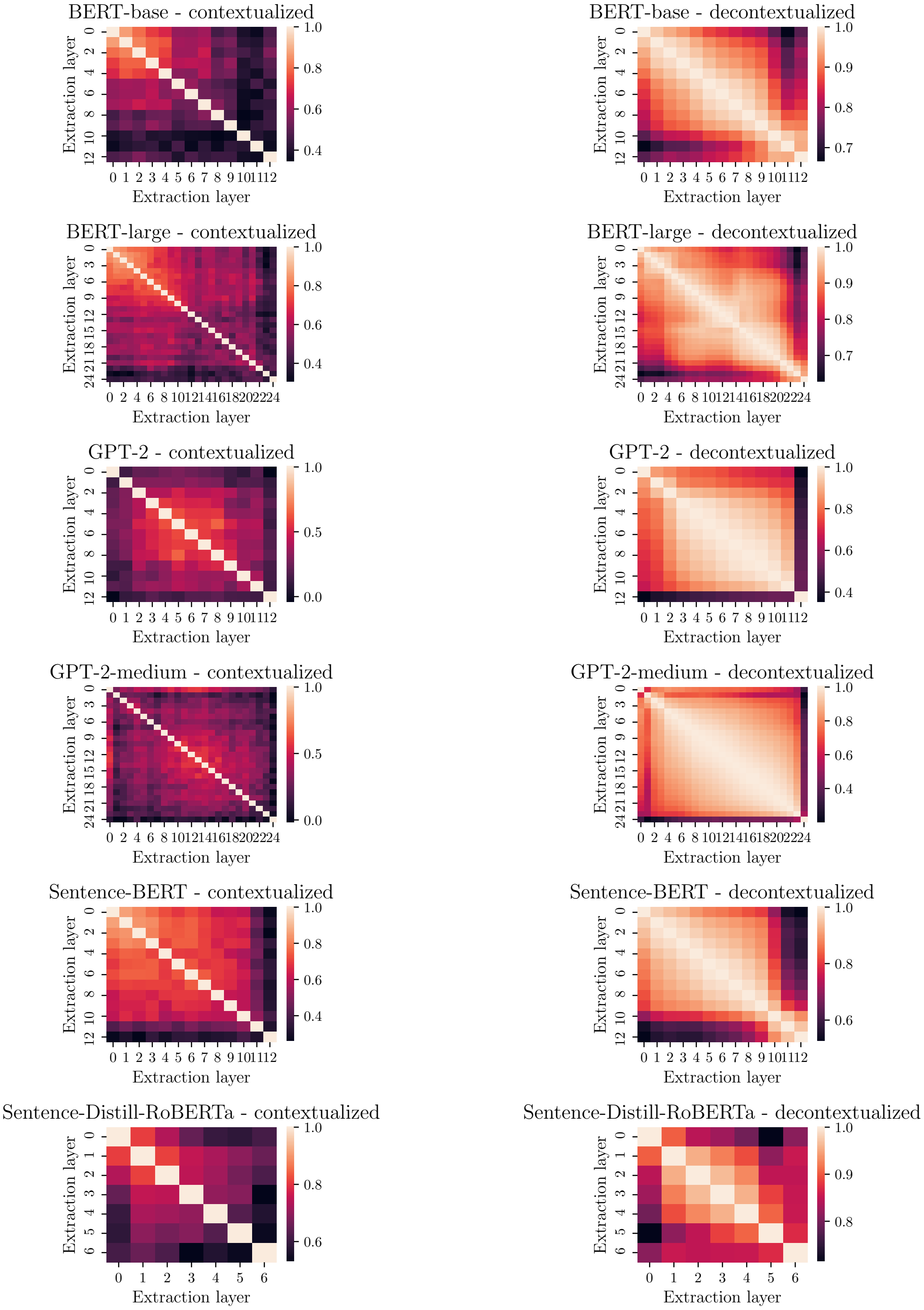


Figure 6.8: Correlation of similarity ratings, based on embeddings, between all extraction layers

6.1.3 Similarity Ratings of Object Images from THINGS

The previous analyses were based on word similarity which measures the similarity between two words without context. It is based on similarity ratings between word pairs based on human judgements. As these word similarity datasets are unsuitable to determine why two words are rated similar or dissimilar, a new similarity dataset from Hebart et al. (2020) shall be used for evaluation. The evaluation is based on the Representational Similarity Analysis from Kriegeskorte et al. (2008) which is an analysis method from neuroscience research. For a set of objects, the similarities of all pairs are calculated based on a similarity function. Compared to the classical intrinsic evaluation with word similarity datasets like Simlex-999 and Wordsim-353, it is ensured that all possible pairs are evaluated.

The resulting similarity matrix is then correlated with a target similarity matrix to access how well representational similarity is reflected in the base matrix. Representational Similarity Analysis is usually done between similarities based on a computational model, similarities based on brain activity, e.g. fMRI data, and similarities based on human behavior. In this evaluation, word embeddings will be used as the computational model and the similarity judgements from Hebart et al. (2020) will be used as human behavioral data.

Impact of Word Sense Annotation

As described in the introduction, embeddings can be created for words, word senses, and synsets. An advantage of using word sense or synset embeddings is that homonyms get different embeddings. Therefore, they capture their diverse meaning. To create these embeddings, a word sense prediction is performed. For this, the ARES embeddings from Scarlini et al. (2020b) are used to predict the word sense of a target word by choosing the word sense of the closest ARES embedding to a BERT-large embedding of the word. By using the *NLTK* library, the synsets are matched to the word senses. Afterwards in the decontextualization process, embeddings are aggregated per word sense and synset.

The representational similarity analysis is then used to access the quality of the embeddings. As the THINGS dataset has word sense and synset annotations, only the matching embeddings are used. A fixed number of $n = 10$ contexts is used for decontextualization for a fair comparison as word and word senses have a different total number of embeddings. Objects without matching embeddings are removed. Embeddings are exemplarily used from the BERT-base model because the word sense annotation process only influences the decontextualization process and not the contextualized embeddings. Therefore, similar results are expected for other extraction models.

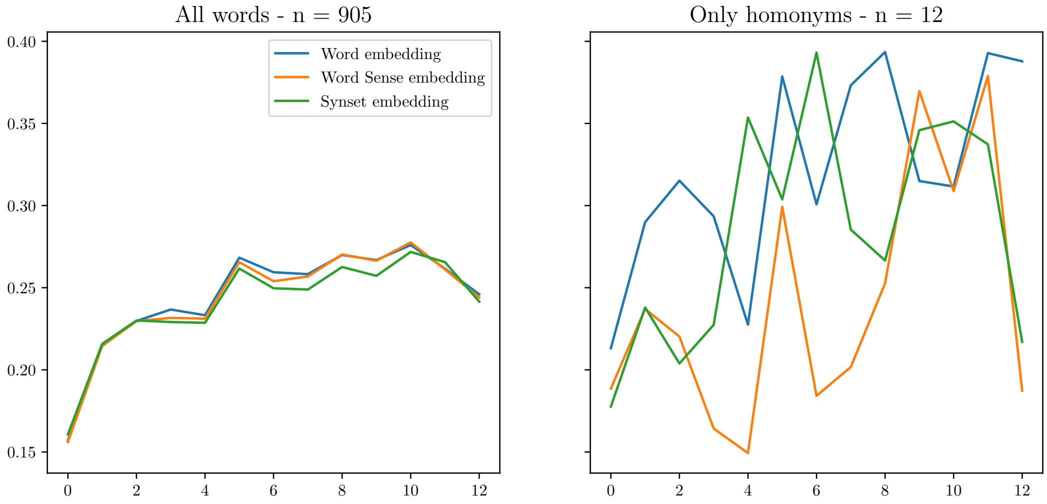


Figure 6.9: Correlation of similarity ratings, based on word, word sense, and synset embeddings, with object image similarity ratings from the THINGS dataset.

The results can be seen in Figure 6.9. It shows the correlation of similarity ratings, based on word, word sense, and synset embeddings with the object image similarity ratings from the THINGS dataset. For all three embedding types, the correlations increase with later layers but are highly similar. The same analysis is performed for homonymous words that occur multiple times in the THINGS dataset like *mouse* or *bat*. As the sample is not big enough, the results are not significant. Word embeddings are further used in the following experiments based on this result.

Pretrained Language Models and Static Word Embeddings

Figure 6.10 shows the results of the intrinsic evaluation for all word embeddings. Static word embeddings have the highest correlation with the human ground truth similarity judgements whereas embeddings from large language models perform worse.

Word2Vec embeddings lead to a correlation of 0.37, GloVe embeddings to 0.29, and Deconflated embeddings to 0.46.

Single contextualized word embeddings show different performances across models. For BERT-base embeddings, the correlations are similar across layers with the maximum of 0.16 at the first encoder layer and the minimum of 0.12 at the eleventh layer. Embeddings from the BERT-large model have the highest correlation of 0.18 at the sixth layer and the lowest correlation of 0.08 at layer 23. GPT-2 embeddings result in a maximum correlation of 0.12 at the eighth layer and a minimum correlation of 0.03 at the static embedding layer. Embeddings from the GPT-2-medium model result in a maximum correlation of 0.12 at layer 15 and a minimum correlation of 0.03 at the last layer.

Finetuned models

Single contextualized word embeddings from finetuned models have similar correlations compared with embeddings from pretrained models.

Similarity ratings, based on embeddings from the Sentence-BERT model, have a maximum correlation of 0.19 at the eleventh layer and a minimum correlation of 0.1 at the last layer. Embeddings from the Sentence-Distill-RoBERTa model result in a maximum correlation of 0.23 at the sixth layer and a minimum correlation of 0.08 at the fourth layer.

Impact of Decontextualization

Decontextualized embeddings have higher correlations than contextualized embeddings across all models. For BERT-base, decontextualized embeddings result in a maximum correlation of 0.25 at the tenth layer and a minimum correlation of 0.15 at the static embedding layer. For BERT-large, decontextualized embeddings result in a maximum correlation of 0.25 at layer 21 and a minimum correlation of 0.14 at the second layer. In both GPT-2 models, the correlations increase with later layers, resulting in a maximum correlation of 0.22 for GPT-2 and 0.25 for GPT-2-medium in the last layers. Yet, the correlation of the final decoder layer decreases to 0.08 for GPT-2 and 0.06 for GPT-2-medium. Embeddings from the Sentence-BERT model have the maximum correlation of 0.27 at the eleventh layer and the minimum correlation of 0.15 at the static embedding layer. Embeddings from the Sentence-distill-RoBERTa model have the maximum correlation of 0.23 at the last layer and the minimum correlation of 0.12 at the fourth layer.

Qualitative Analysis

The rank differences d_i^2 from the spearman rank calculation can be used to further analyze the embeddings. A high rank difference of a word pair indicates a difference in representational similarity compared with humans.

In Table 6.1, word pairs with the highest rank difference are shown with the respective similarity ratings from humans and based on decontextualized word embeddings from BERT-base. The list contains homonyms that are not directly marked as homonyms in the THINGS dataset. For example, the word *gyro* is used for food in the THINGS dataset, but it also has the meaning of a

Table 6.1: Word pairs where similarity ratings based on cosine similarity between word embeddings and object image-based similarity ratings from humans differ the most. Pairs are sorted by ascending spearman rank difference.

Word 1	Word 2	Human Judgement	Cosine Similarity
piglet	ratchet	0.07	0.70
gyro	periscope	0.07	0.69
cornucopia	unicycle	0.07	0.72
beaver	cardinal	0.94	0.25
scaffold	scallop	0.08	0.73
cardinal	vulture	0.96	0.26
target	volleyball	0.94	0.26
cornucopia	tripod	0.07	0.68
gyro	tripod	0.08	0.76
blouse	tag	0.92	0.24
eggbeater	seagull	0.07	0.68
cobra	moth	0.91	0.23
gyro	gyroscope	0.08	0.88
beetle	cobra	0.92	0.25
jackhammer	meatball	0.08	0.69
beetle	earwig	0.95	0.27
jackhammer	pancake	0.07	0.68
sweater	tag	0.93	0.26
skunk	whisk	0.07	0.66
fox	lizard	0.92	0.26

playing toy. The word *cardinal* is used for the bird but also has the meaning of a leading dignitary in the catholic church.

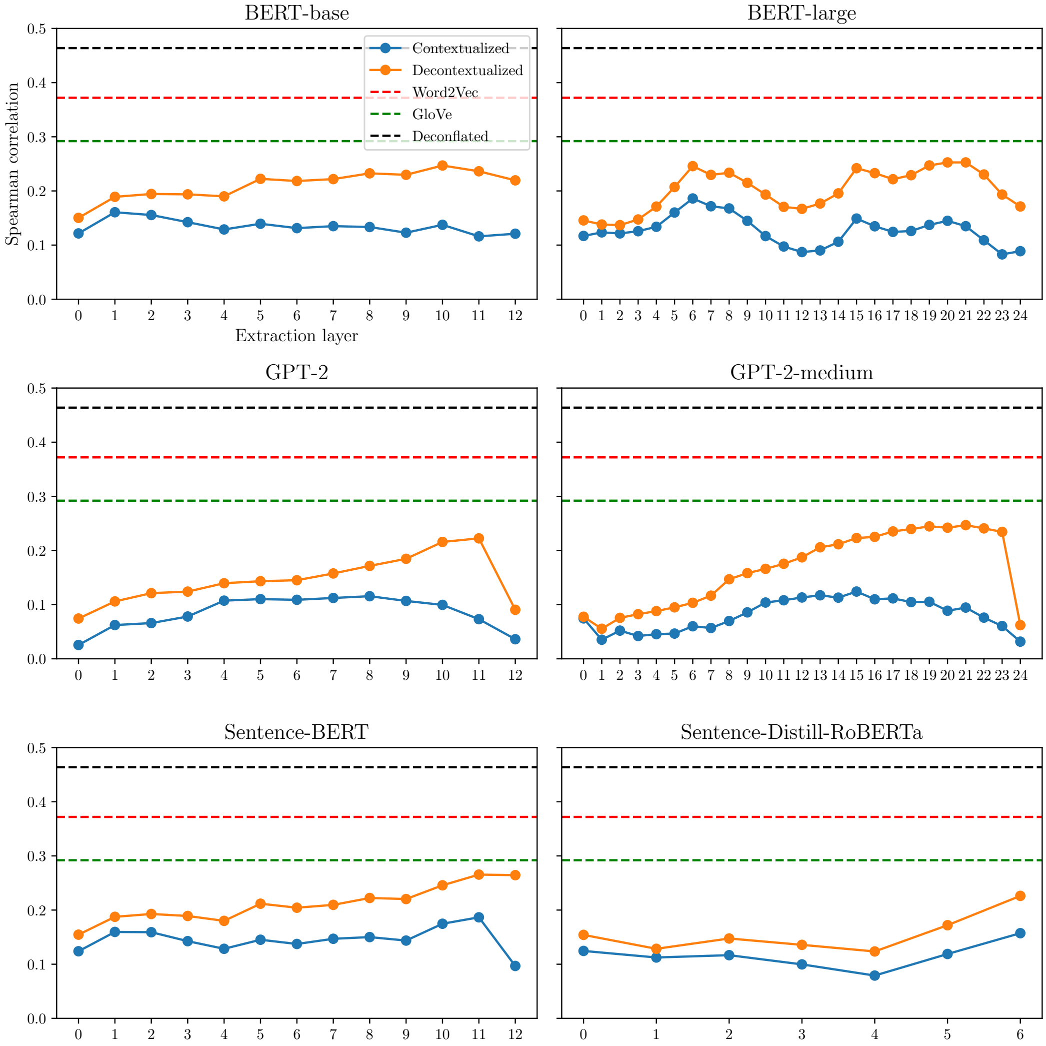


Figure 6.10: Correlation of similarity ratings, based on embeddings, with object image similarity ratings from the THINGS dataset.

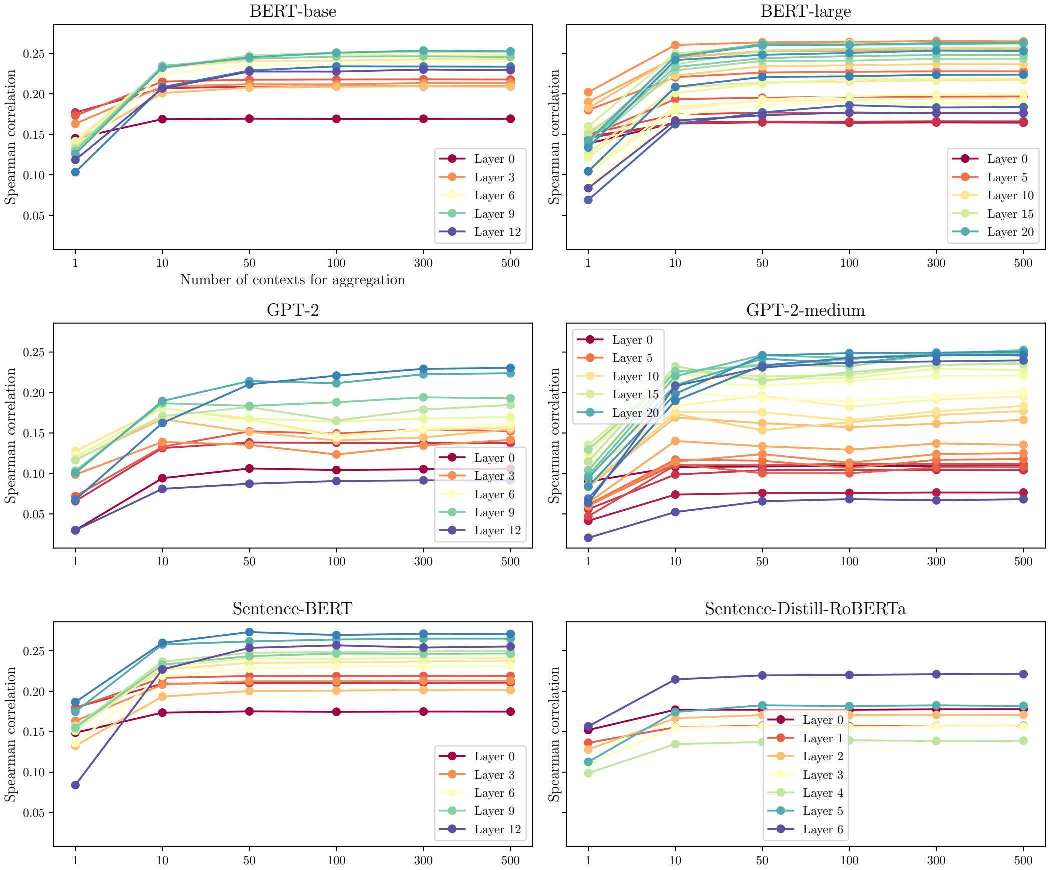


Figure 6.11: Correlation of similarity ratings, based on embeddings, with object image similarity ratings from the THINGS dataset per layer and number of contexts

Impact of Number of Contexts

To investigate the importance of the number of contexts used for aggregation, the intrinsic evaluation was performed on a fixed size of contexts. For a fair comparison, only word pairs that have more than 1000 unique contexts were used. Further, all layers were used for extraction to investigate the dynamic between the number of contexts and the extraction layer.

As seen in Figure 6.11, the number of contexts affects the representational similarity. The more contexts are used, the higher the correlation. This effect stops after 100 contexts for the GPT models and after ten to 50 contexts for BERT models. Afterwards, the correlation converges.

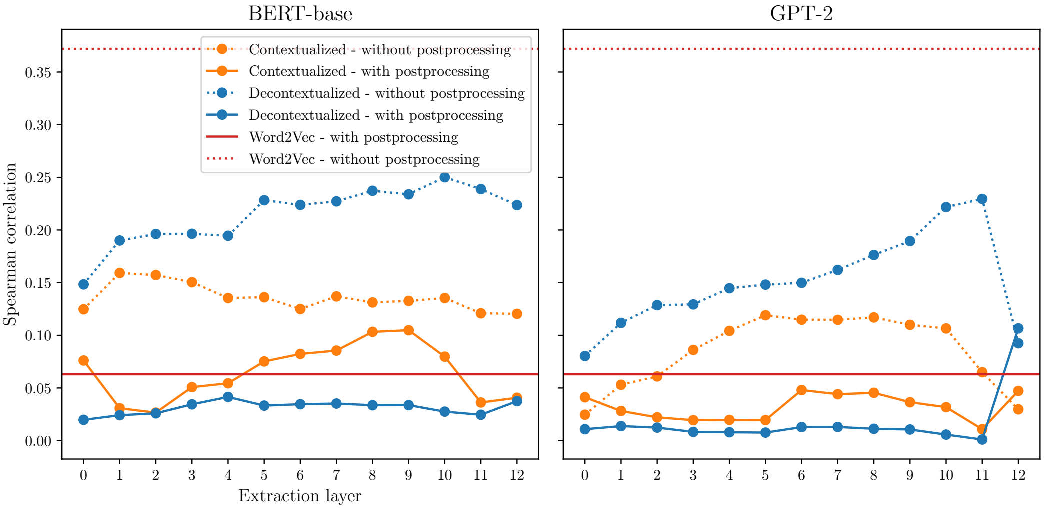


Figure 6.12: Effect of isotropy postprocessing on the correlation of similarity ratings, based on embeddings from BERT-base and GPT-2, with object image similarity ratings from the THINGS dataset.

Impact of Isotropy Postprocessing

Figure 6.12 shows the effect of the postprocessing step to increase isotropy. In contrast to the correlations with similarity ratings from the word similarity datasets, it decreases the correlation with the similarity ratings from the THINGS dataset. For Word2Vec embeddings, the correlation decreases from 0.37 to 0.06. Similarly, the correlation decreases for contextualized and decontextualized word embeddings from BERT-base and GPT-2.

Impact of Low Frequent Words

The correlations of predicted similarity ratings between word pairs with similarity ratings from the THINGS dataset are lower than compared with word similarity ratings from the Simlex-999 and Wordsim-353 datasets.

One reason for this difference might be the composition of the THINGS dataset. It contains living and non-living concrete objects covering high frequent objects as well as low frequent objects whereas the word similarity datasets Simlex-999 and Wordsim-353 seem to be composed of more high frequent words. This can be seen in the distribution of the number of contexts per word in Figure 5.6 where it is possible to sample 1000 contexts for almost all target words from the word similarity datasets. It was further shown by Li et al. (2020) that word frequency might bias word embeddings and the predicted similarity as it is based on the cosine similarity between embeddings. The frequency of a word might influence the training of an embedding model or influence the aggregation as less context are available. Static word embeddings like Word2Vec are trained on a bigger corpus than Transformer-based models which results in more contexts per word.

The influence on the aggregation was shown in Figure 6.11. To investigate the effect of word frequency on the training, the object frequency metadata from the THINGS dataset is used to approximate the frequency of words during training as this information is not available. The 1,854 words are first sorted by descending frequency and then split into 30 batches. For each non-overlapping batch, a similarity matrix, based on the cosine similarity between word pair embeddings, is calculated. In a second analysis, the batches are merged, starting with the batch containing the most frequent words and ending with a batch containing all words. Decontextualized word embeddings from BERT-base are used. BERT-base is chosen as similarity ratings based on the model embeddings have high correlations and are fast to compute.

The results can be seen in Figure 6.13. On the left side, the correlation for each batch is shown as well as the mean frequency, based on the COCA frequency. On the right side, the correlation of similarity ratings based on cumulative batches is shown. Both analyses do not show a significant impact of the frequency of words on the representational similarity.

Similarity Predictions across Layers

Similar to the analysis of word similarity ratings, based on word embeddings, the predicted similarity of object images, based on word embeddings, shall be compared across extraction layers.

The same method was applied where the spearman correlation between similarity predictions, based on word embeddings, from two layers is calculated

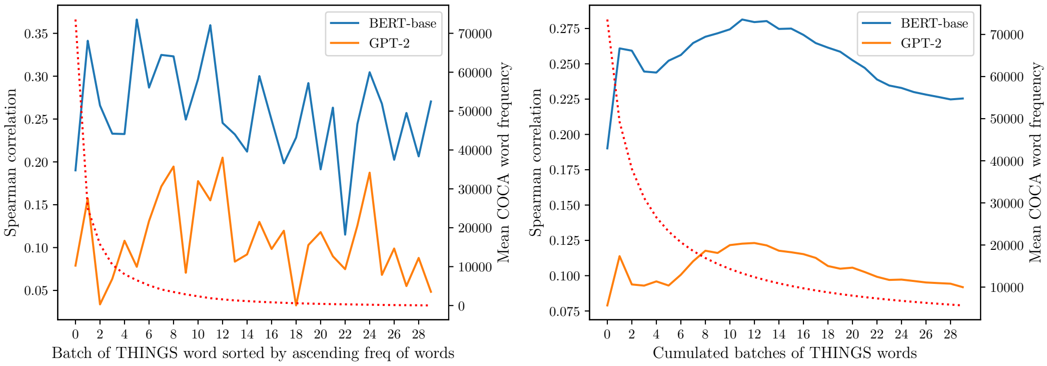


Figure 6.13: Effect of low frequent words on the correlation of similarity ratings, based on embeddings from BERT-base and GPT-2, with object image similarity ratings from the THINGS dataset.

as a measure whether the similarity is encoded in a similar or different way. It is expected to match the results based on the Wordsim-353 and Simlex-999 datasets as only the vocabulary changes. Yet, the focus of the THINGS dataset on living and non-living concrete objects might influence the result.

In Figure 6.14, overall low correlations can be seen between similarity ratings based on contextualized word embeddings. Especially for embeddings from the GPT-2 and GPT-2-medium model. Similarity ratings based on decontextualized word embeddings have higher correlations between layers across all models.

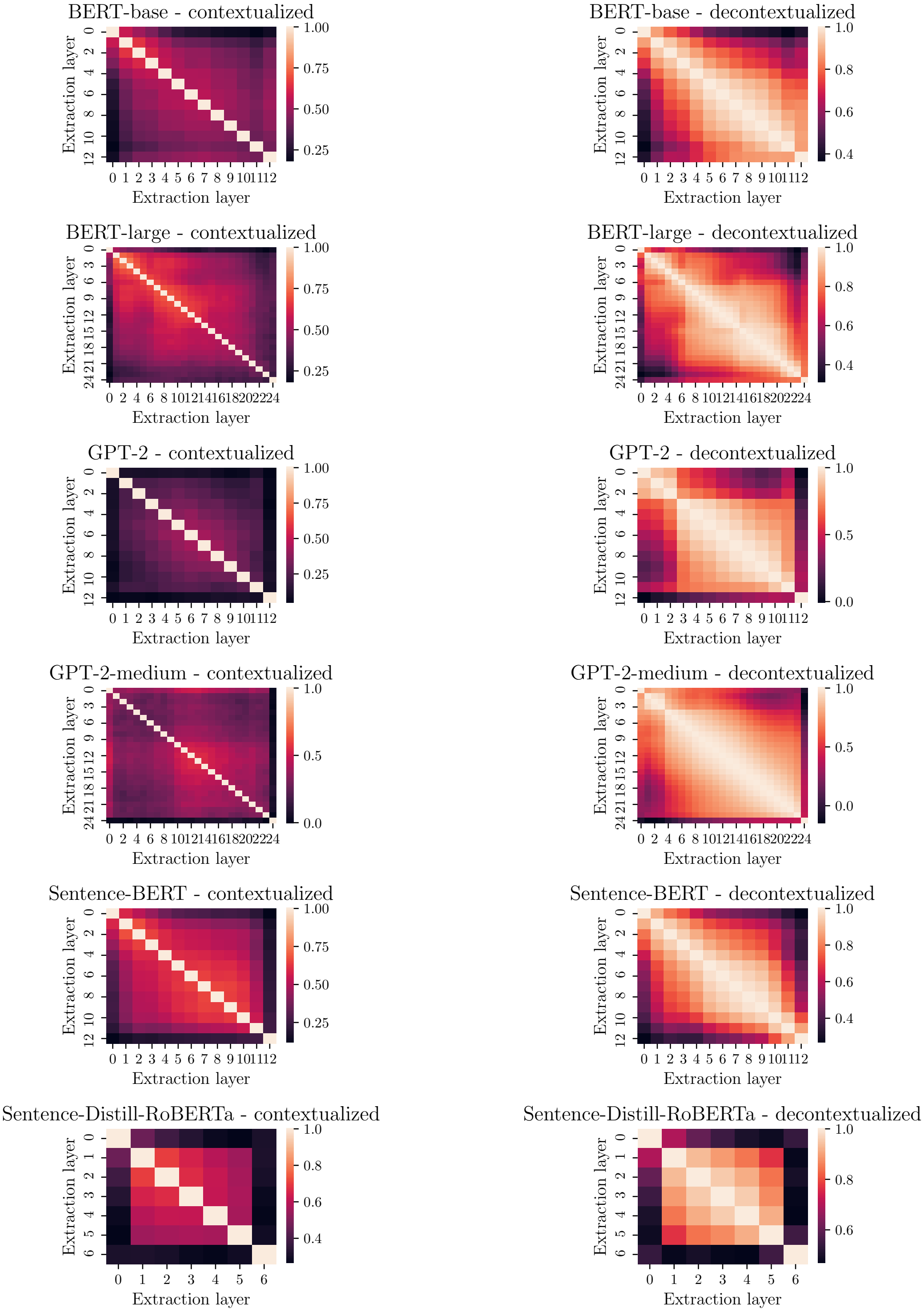


Figure 6.14: Correlation of similarity ratings, based on embeddings, between all extraction layers

Table 6.2: Hyperparameters and their ranges and distributions that were sampled from.

	MSE	MAE	Contrastive loss
Learning Rate	0.000019	0.000045	0.000057
Dropout Rate	0.048		0.1948
Batch Size	1000	-	5000
Weight Decay	0.000001	-	0.00001
Number of Neurons per Layer	600	600	600
Number of Layers	1	1	1

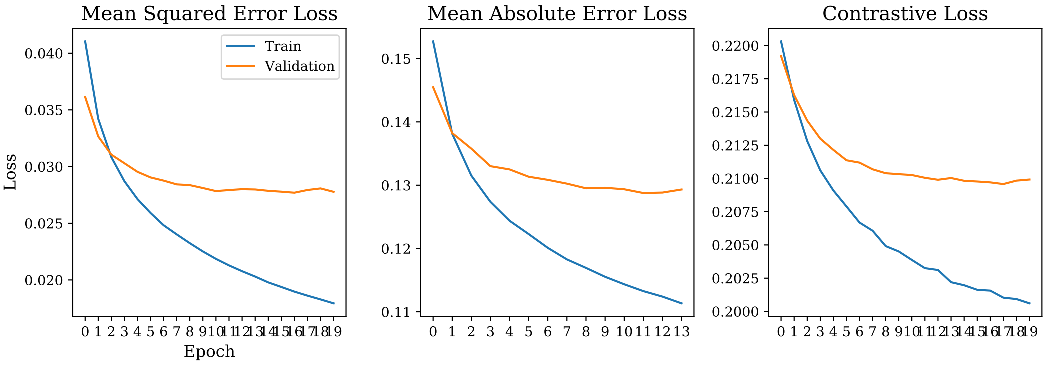


Figure 6.15: Training and validation losses per epoch for training with mean squared error loss, mean absolute error loss, and contrastive loss.

Impact of Retraining

The previous analyses showed that Transformer-based word embeddings encode the similarity of object images worse than static word embeddings. Therefore, a retraining method is applied to the Transformer-based word embeddings to create new embeddings and to investigate whether it is possible to improve the embeddings.

The final retraining is based on the optimized hyperparameters. The training and validation losses per epoch are shown in Figure 6.15.

Figure 6.16 shows the effect of retraining on the representational similarity of embeddings for words from the train, validation, and test set. The correlation of similarity ratings from retrained word embeddings with similarity ratings from the THINGS dataset is higher for all extraction layers compared to decontextualized word embeddings without retraining. Similar to previous results, the correlation is higher with later layers. It is possible to reach a

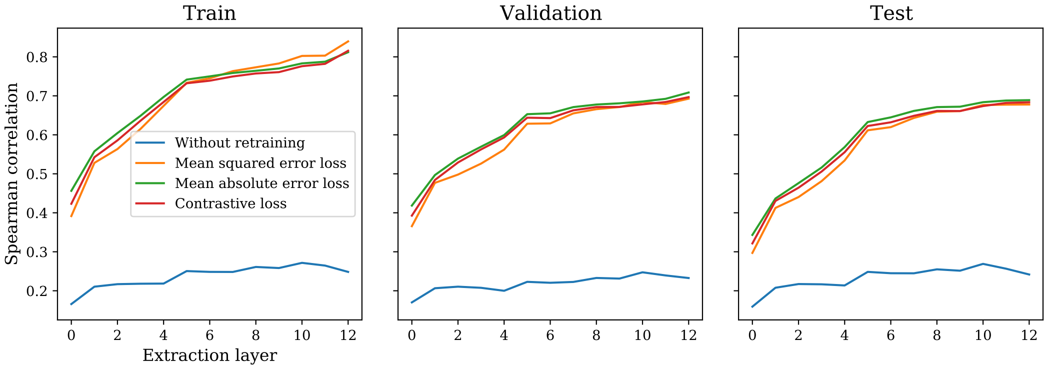


Figure 6.16: Correlation of similarity ratings, based on embeddings, with object image similarity ratings from THINGS before and after retraining on train, validation, and test sets

maximum correlation of 0.7 with all three loss functions.

Figure 6.17 shows the distribution of similarity ratings for embeddings of word pairs from the validation set before and after retraining and the scatter plot of true similarity ratings and embedding-based similarity ratings. The retraining adjusts similar as well as dissimilar pairs which leads to a more diagonal aligned scatter plot.

Table 6.3 shows wrongly optimized word pairs from the test set that are sorted by decreasing squared rank difference ${d_i}^2$ from the spearman correlation. It can be seen that the retraining leads to similar word pairs containing animals and the word *wok*.

Table 6.3: Word pairs from the test set sorted by spearman rank difference after retraining. Human judgements range from 0 (dissimilar) to 1 (similar). Cosine Similarities range from -1 (dissimilar) to 1 (similar).

Word 1	**Word 2**	**Human Judgement**	**Cosine Similarity**
chinchilla	wok	0.07	0.75
llama	waffle_iron	0.07	0.74
poodle	wok	0.07	0.71
cocktail	funnel	0.79	0.03
llama	tupperware	0.08	0.73
mongoose	wok	0.07	0.70
bongo	chinchilla	0.08	0.76
paper_plate	wok	0.78	0.04
plate	soup	0.79	0.05
ottoman	turkey	0.09	0.86
llama	wok	0.07	0.68
bongo	hamster	0.08	0.71
cinnamon	peg	0.85	0.08
possum	wok	0.07	0.69
bus	rim	0.83	0.08
cilantro	hedge	0.91	0.10
crow	wok	0.07	0.68
dish	llama	0.08	0.72
tape_measure	wax	0.09	0.73
bongo	ferret	0.08	0.69

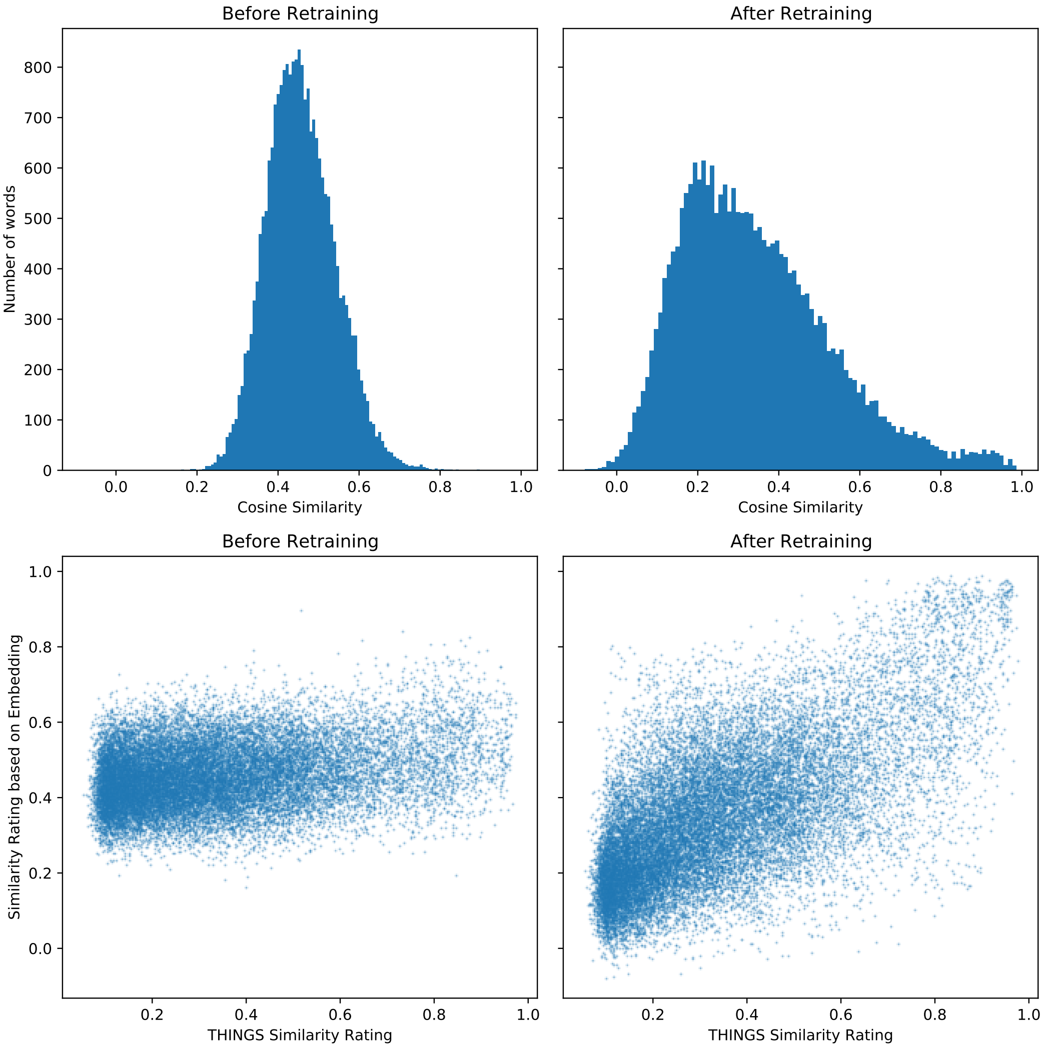


Figure 6.17: Similarity distribution of the validation set before and after retraining

Feature-Reweighted Representational Similarity Analysis

As the previous analysis assumed the same importance of all embedding dimensions, it is possible that the encoded information is not fully exploited as different dimensions can have a different high influence on the similarity. For example, some dimensions might be more important for the encoding of lexical meaning rather than semantic meaning.

To investigate this question, a feature-reweighted representational similarity analysis is performed to predict the dimension-based similarity. The framework from Kaniuth and Hebart (2020) is used to perform this analysis. The term feature refers to the dimensionality of the word embedding, e.g. 768 for embeddings from the BERT-base model. Instead of calculating the similarity between word embeddings based on all features, this algorithm first calculates the similarity between words based on a single feature. Then the target similarity is predicted using a multiple regression model with y as the target similarity based on all features, β as the regression coefficients, p as the length of the word embeddings, e.g. 768 for a BERT-base based word embedding, and x_{ij} as the similarity value based on a single feature j of a word pair i:

$$y_i = \beta_0 + \sum_{j=0}^{p} \beta_j \cdot x_{ij} \tag{6.2}$$

The model then uses predictor variables for every single feature and the similarity ground truth as the target y_i. A mean squared error term and ridge regularization are used as the objective term:

$$\frac{1}{m}\sum_{i=0}^{m}(y_i - \hat{y_i})^2 + \lambda \cdot \sum_{j=0}^{p} \beta_j^2 \tag{6.3}$$

Word similarity is not investigated with this approach, as the algorithm is based on similarity matrices and therefore requires all possible pairs to be compared.

As the ground truth is based on similarity rather than dissimilarity, a similarity measure has to be used to perform the reweighting. The dot product $u \cdot v$ is used to calculate similarity based on a single feature. K-fold cross-validation with $k = 5$ is used. The feature-reweighting process is applied on decontextualized word embeddings from all layers of BERT-base and GPT-2 and static word embeddings from Word2Vec and the Deconflated embedding model.

The results in Figure 6.18 show a high increase in correlation of predicted similarity ratings for all feature-reweighted embeddings. The correlation increases from 0.39 to 0.59 for Word2Vec. The correlation is slightly higher than the correlation for Transformer-based embeddings. The correlation of similarity ratings from Deconflated embeddings increases from 0.62 to 0.72. For feature-reweighted embeddings from BERT-base, the correlation starts at 0.25 in the first static embedding layer and increases with later layers. It reaches the maximum at the eleventh layer with a correlation of 0.58. The effect of feature-reweighting improves the correlation, especially in later layers. For the GPT-2 model, the correlation of feature-reweighted embeddings starts at 0.3 and reaches the maximum at the eleventh layer with 0.58.

The results indicate that some embedding dimensions are more useful for calculating similarity. The difference in representational similarity between static word embeddings and decontextualized Transformer-based word embeddings is smaller when feature-reweighting is used. This leads to the assumption that both, static and Transformer-based, word embeddings represent the similarity of object images in a similar way. The higher dimensionality of Transformer-based embeddings and the encoding of more information, not relevant to the similarity of object images, might lead to worse correlations when all embedding dimensions have the same weight for calculating the similarity.

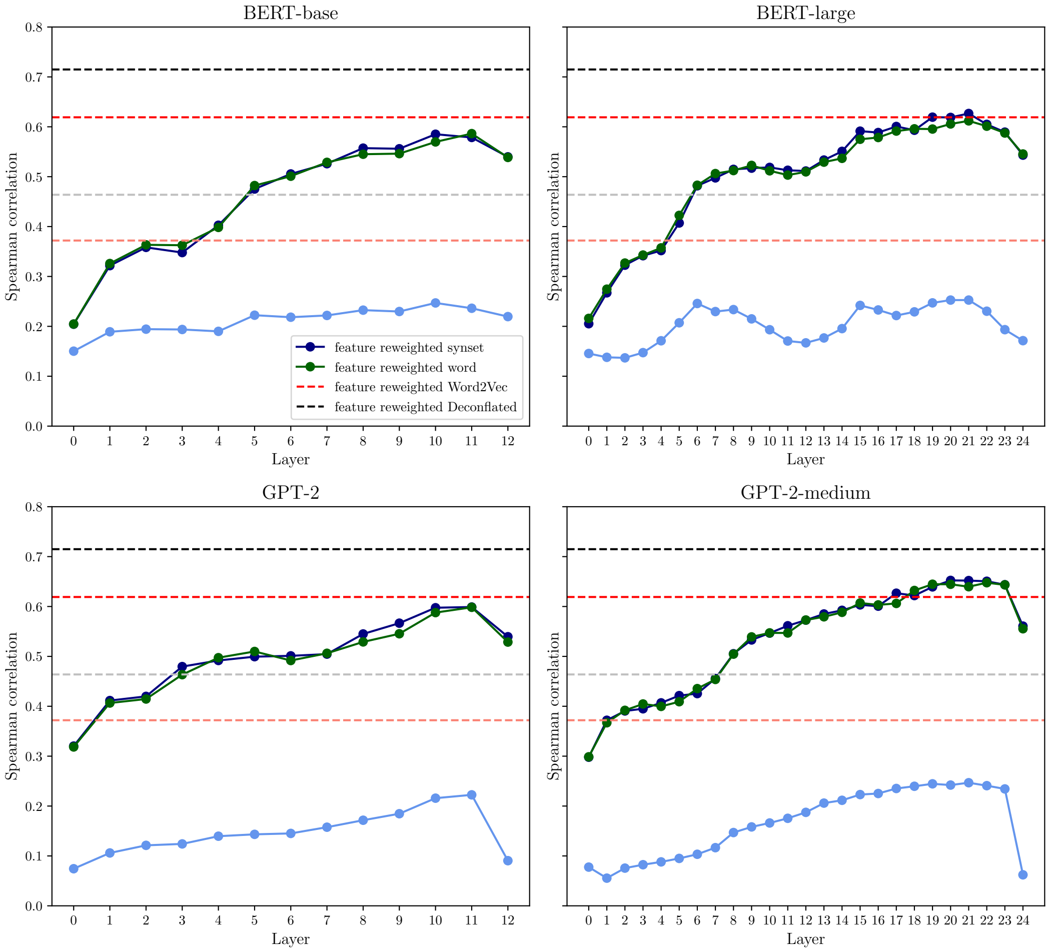


Figure 6.18: Correlation of similarity ratings, based on embeddings, with object images similarity ratings from THINGS. Feature-reweighting is used to calculate the similarity between object pairs.

6.2 Prediction of THINGS Dimension Embeddings

The similarity metric from Hebart et al. (2020) is based on 49 interpretable object dimensions (see Table A.1). These dimensions meaningfully reflect conceptual and perceptual properties. For example, they measure how important the dimension *animal-related* is for an object like *dog* or how unimportant the dimension *colorful* is for an object like *airplane*.

To assess which dimensions are encoded in the word embeddings, a multiple regression model is trained on the word embeddings to predict the dimension values with y as the dimension value, β as the regression coefficients, p as the length of the word embeddings, e.g. 768 for a BERT-base based word embedding and i as the index of the THINGS object.

$$y_i = \beta_0 + \sum_{j=0}^{p} \beta_j \cdot x_{ij} \tag{6.4}$$

For this purpose, an Elasticnet model is chosen which uses Ridge and Lasso regularization and minimizes the following loss:

$$\frac{1}{m} \sum_{i=0}^{m} (y_i - \hat{y}_i)^2 + \lambda(\frac{1-\alpha}{2} \cdot \sum_{j=0}^{p} \beta_j^2 + \alpha \cdot \sum_{j=0}^{p} |\beta_j|) \tag{6.5}$$

The model is then trained for each of the 49 dimensions using nested cross-validation to first choose the best hyperparameters and then predict the dimension values.

Different scores are then used to define the predictiveness of the word embeddings (see Table 6.4). First, the coefficient of determination R^2 is used as a measure of explained variance:

$$R^2 = 1 - \frac{RSS}{TSS} = \frac{\sum_i (y_i - \hat{y})^2}{\sum_i (y_i - \bar{y})^2} \tag{6.6}$$

with $\hat{y}$ as the predicted dimension value, $\bar{y}$ as the mean dimension value for all objects and y_i as the true dimension value for object i.

Second, the spearman correlation between the predicted and true dimension values is used (C1).

Third, the representational similarity is calculated based on all predicted dimension values (C2). The spearman correlation between the predicted similarity ratings of all pairs from the test set with the similarity ratings from the THINGS dataset is calculated.

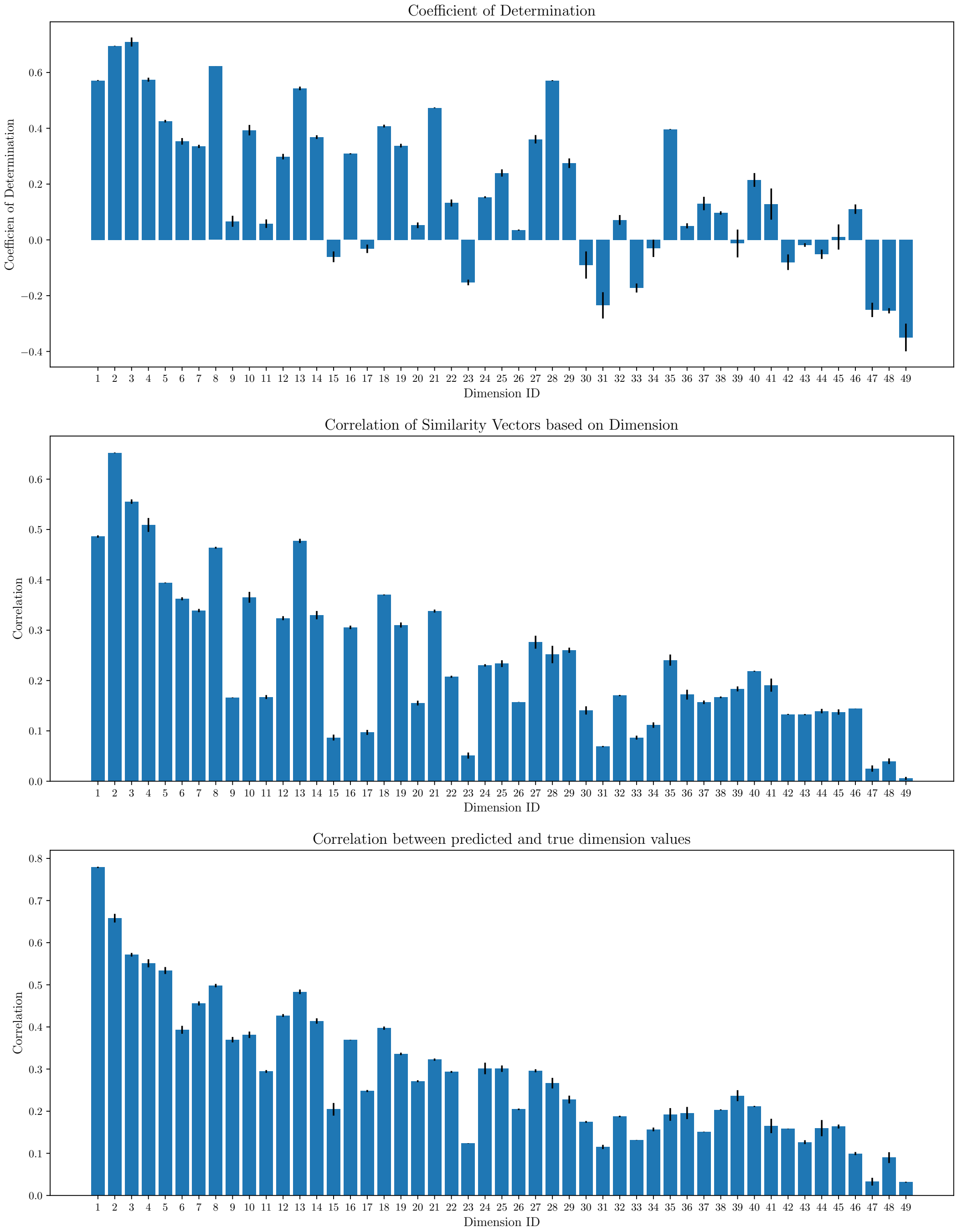
Table 6.4: Cross-validated results for prediction of dimensions

Model	**C1**	**C2**	R^2	**C3**
Word2Vec	0.29	0.64	0.15	0.24
GloVe	0.25	0.55	0.09	0.20
Deconflated	0.32	0.70	0.24	0.27
BERT-base	0.28	0.62	0.18	0.24
BERT-large	0.32	0.63	0.28	0.26
GPT-2	0.15	0.20	0.04	0.04
GPT-2-medium	0.10	0.11	0.02	0.02

Fourth, the representational similarity is calculated based on single predicted dimensions (C3). For this, the spearman correlation between the predicted similarity ratings of all pairs with the similarity ratings, based on a single dimension, from the THINGS dataset is calculated.

The metrics R^2, C1, and C3 are calculated separately for each dimension and averaged across test folds. The metric C2 is calculated across all dimensions.

In Figure 6.19, the average cross-validated scores C1, C3, and R^2 can be seen per dimension for decontextualized word embeddings from the BERT-base model. In Table 6.4, the average cross-validated scores C1, C3, and R^2 across all dimensions are shown as well as the cross-validated C2 score. The Deconflated word sense embeddings have the highest correlations and R^2 values which indicate the highest quality for the prediction of these dimension values.



Coefficient of Determination

Correlation of Similarity Vectors based on Dimension

Correlation between predicted and true dimension values

Figure 6.19: Cross-validated averaged results for dimension prediction with de-contextualized BERT-base word embeddings. The bars include a 95% confidence interval based on the cross-validation.

6.3 Prediction of Superordinate Categories from THINGS

This experiment investigates how well different word and word sense embeddings perform when they are used in downstream tasks. The task to predict superordinate categories of objects also investigates how well human conceptual knowledge is represented in the embeddings.

For this, superordinate categories from the THINGS dataset for all 1,854 objects are used. A Support Vector Classifier is then trained on top of the embeddings for all 1,854 objects and cross-validated using k-Fold cross-validation with $k = 5$. The predictiveness of the embeddings is accessed as the prediction accuracy over the test fold. The static word embeddings from Word2Vec, GloVe, and the Deconflated embedding model are used. Further, BERT-base, BERT-large, GPT-2, GPT-2-medium, Sentence-BERT, and Sentence-Distill-RoBERTa are used as Transformer-based models.

The results are shown in Figure 6.20. It shows the mean accuracy over test folds for the static word and word sense embeddings as well as contextualized and decontextualized word embeddings. Static word and word sense embeddings from Word2Vec and the Deconflated embedding model have the highest accuracy in predicting categories. Contextualized word embeddings have the lowest accuracy. Their accuracy increases in later layers in BERT-base and BERT-large. For GPT-2, the accuracy increases slightly but decreases in later layers.

Decontextualized word embeddings have a higher accuracy than contextualized word embeddings. The decontextualization process is especially increasing the accuracy in later layers in all large language models.

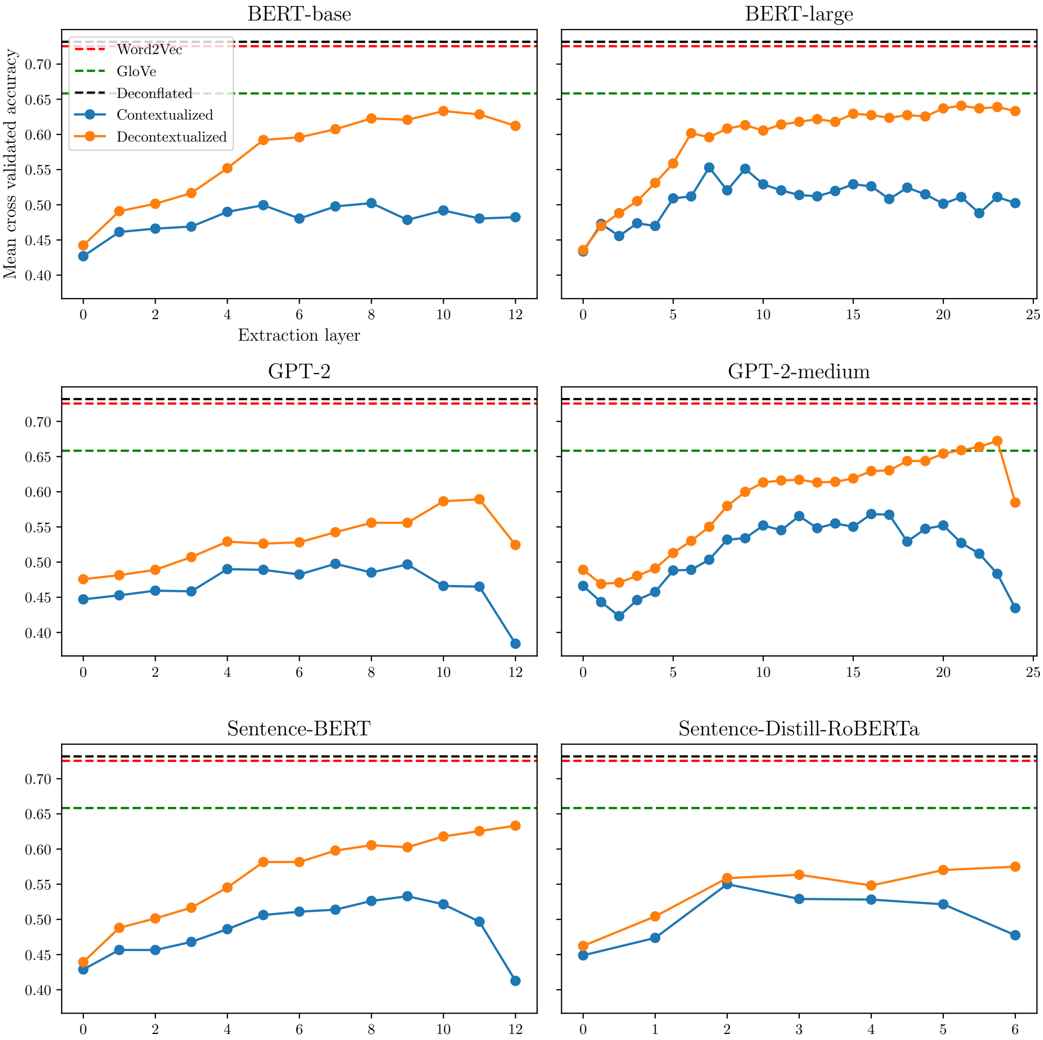


Figure 6.20: Cross-validated accuracy for superordinate category prediction of objects from the test set

Chapter 7

Discussion

The results show that it is possible to generate word, word sense, and synset embeddings from Transformer-based models. These contextualized and decontextualized embeddings can be of high quality, depending on the task and dataset.

The intrinsic evaluation was performed with representational similarity analysis to investigate how well human knowledge about semantic meaning, similarity, and relatedness is encoded in different embeddings. The results show that different human similarity ratings are encoded in static, contextualized, and decontextualized word and word sense embeddings. Word similarity is well encoded in all embedding types whereas similarity of object images is best encoded in static word sense embedding and worse encoded in Transformer-based embeddings.

The extrinsic evaluation with two downstream tasks was used to investigate the quality of embeddings for specific tasks. The first task was the prediction of THINGS dimension values and the second task was the prediction of superordinate categories. In both tasks, static word and word sense embeddings performed better compared to Transformer-based embeddings.

Creation of Static Word Embeddings from Large Language Models

It is possible to extract and decontextualize word embeddings from large language models to create new static word embeddings that can be efficiently used in other tasks. Decontextualization further improves the representational similarity with humans leading to more qualitative embeddings. Aggregation might counteract the anisotropy or contextualization effects shown by Ethayarajh (2019).

While contextualized word embeddings capture the meaning of a word in a given text context well, it is difficult to use them as static embeddings as they are highly sensitive to the context. Thus, aggregation over multiple contexts recovers the general meaning of a word. The ability to reflect different meanings for homonyms is lost as a word gets a fixed embedding. To circumvent this problem, a word sense prediction method was applied based on the ARES approach from Scarlini et al. (2020a).

The postprocessing method from Mu and Viswanath (2017) was further applied to create new embeddings. While this created embeddings with higher representational similarity with humans, based on word similarity datasets, it created embeddings with worse representational similarity, based on the similarity of object images.

A retraining method was developed to create word embeddings with higher representational similarity with humans, based on the similarity of object images. This training step was performed in alignment with previous research where several problems with pretrained contextualized word embeddings were shown (Reimers & Gurevych, 2019). The results showed that the representational quality can be improved by supervised methods. Still, the direct training on the similarity dataset did not improve the representational similarity as much as expected. This might indicate that Transformer-based embeddings do not encode all information for the similarity of object images.

Classic static word embeddings are still competitive against Transformer-based word embeddings when intrinsic evaluation is performed. It is possible to produce high-quality static word embeddings from Transformer-based models when intrinsic evaluation with word similarity is performed. Though, this does not generalize to other similarity definitions. Transformer-based word embeddings can still be used for downstream tasks where they can lead to better performance, compared to static word embeddings (C. Wang et al., 2020).

The results further show that the last layer often produces worse embeddings when pretrained models are used. Especially the last decoder of GPT-2 has low representational similarity with humans when intrinsic evaluation is performed. Under extrinsic evaluation, embeddings from the last layer result in higher performance than embeddings from early layers but have lower performance compared to the previous last layers.

Word Sense Disambiguation

The process of word sense disambiguation was used to create word sense embeddings. Though, this did not increase the representational similarity.

It might be possible that the sampling of paragraphs, used to create decontextualized embeddings, led to poor candidates for the word sense prediction. The random sampling was based on the word level rather than the word sense level as the prediction of the word sense for all word occurrences would have been too computationally expensive. Low frequent word senses are then harder to predict if only paragraphs with high frequent word senses are sampled.

It is further possible that the method by Scarlini et al. (2020a) does not work well or that homonyms do not influence the representational similarity.

Word Similarity

Word similarity is well encoded in static word embeddings. Transformer-based word embeddings encode word similarity similar but with differences across model architectures, layers, and the number of contexts for aggregation. Embeddings from encoder models seem to perform better than embeddings from decoder models, indicating a stronger representational encoding in encoder models. This might be caused by the bidirectional training of encoder models in contrast to the unidirectional training in decoder models.

Single contextualized embeddings have a decreasing representational similarity in later layers. This seems counterintuitive as later layers incorporate more context. Contextualized word embeddings have been shown to strongly reflect the context rather than encoding strict word meaning (Ethayarajh, 2019). This could cause a broad distribution of word embeddings leading to similar word pairs when single contextualized embeddings are chosen. As shown by Ethayarajh (2019), the effect of contextualization might cause or might be correlated with the effect of anisotropy where the embeddings form a narrow cone and the cosine similarity across words gets higher. This effect is especially shown for later layers. This could also affect the representational similarity as the cosine similarity is not a suitable similarity metric anymore. Similarly, Reimers and Gurevych (2019) showed that the vector space of BERT embeddings is unsuitable for similarity measures like cosine similarity. This effect has the highest impact in decoder models like GPT-2 where embeddings from the last decoder layer result in similarity ratings with a very low correlation with human similarity ratings. Furthermore, the decrease in representational similarity could be caused by later layers encoding more information relevant to the training objective. For example, later layers in the BERT model may encode more information concerning the masked language model objective.

The aggregation of multiple embeddings with different contexts improves the representational similarity across all models and layers. Especially embeddings from later layers encode more relevant information with this method. Decontextualized word embeddings from the BERT-large model outperform static word embeddings when similarity ratings, based on the embeddings, are compared with human-made similarity ratings. This approach might counteract the anisotropy effect or the contextualization effect in later layers as it improves the representational similarity of later layers. Further, aggregation might have a positive effect as it incorporates the semantic meaning of words across different contexts and different relationships to other words. As context is more encoded in later layers, the aggregation then has the biggest effect on embeddings extracted from later layers.

Though, there seems to be a limit to the number of contexts that are used in aggregation. After around 50 contexts per word, there is no improvement in the representational similarity. The results show further that plain static word embeddings as well as contextualized word embeddings encode semantic meaning based on word similarity well. Decontextualization and the usage of large models result in better performance. The Wordsim-353 dataset measures similarity and relatedness. As all word embeddings have a high correlation, this indicates that relatedness is well encoded across embeddings and models. The Simlex-999 dataset measures strict similarity and the correlations are lower for all embeddings. This indicates that the encoding of semantic similarity is harder than the encoding of relatedness.

Postprocessing to increase isotropy in word embeddings has a positive influence on the representational similarity across all embedding types. The correlation of similarity ratings, based on static, contextualized, and decontextualized word embeddings, increased across all layers and models.

Similarity of Object Images

To investigate how well the similarity of object images is encoded in word embeddings, a new representational similarity analysis was performed between similarity ratings from word embeddings and similarity ratings from the THINGS dataset.

In contrast to word similarity, the similarity of object images is worse reflected in word embeddings. Static word embeddings show the highest representational similarity whereas Transformer-based word embeddings perform worse. This seems counterintuitive as the training objectives of static word embedding models like Word2Vec and Transformer models like BERT are similar.

Similar to word similarity, single contextualized word embeddings are influenced by anisotropy and contextualization processes. Therefore, they might reflect the similarity of object images worse in later layers.

Aggregation over multiple contextualized word embeddings improved the representational similarity across all models and layers.

The usage of postprocessing to increase isotropy in word embeddings decreased the representational similarity across all models and layers. At this point, the reason for this effect is not clear.

These results indicate that the similarity of object images is not well represented in word embeddings, especially Transformer-based word embeddings. To further investigate whether the semantic information needed for this similarity task is not fully exploited in Transformer-based models, different downstream tasks were performed.

Using the word embeddings in a decoding model to predict superordinate categories showed, that later layers also encode more information needed to predict the global structure of THINGS objects. Again, static word embeddings performed better than contextualized and decontextualized word embeddings.

Feature-reweighted representational similarity analysis was used to investigate the influence of single embedding features on representational similarity. The results showed almost equal representational similarity between the static word embeddings from Word2Vec and GloVe and Transformer-based word embeddings. An increasing representational similarity with later layers was further shown. As the reweighting process assigns weights to the embedding dimensions, this could mean that Transformers-based models may encode more information that is not relevant to the similarity of object images (Tenney et al., 2019). All Transformer-based embeddings have a higher dimensionality than all static word embeddings. This might provide more possibilities to encode other information. Still, the embeddings from the Deconflated embedding model showed the highest representational similarity. This might be caused by the accounting for different word senses or by encoding more taxonomic and functional information about objects, based on the WordNet graph.

The embeddings were further used to directly predict the dimension values of THINGS. A cross-validated regression approach was used for prediction. The static word sense embeddings from the Deconflated word embedding model have the best performance whereas embeddings from the BERT model family show higher predictiveness than embeddings from the GPT model family.

Overall, the results indicate that static word and word sense embeddings are still highly competitive with regard to modeling human conceptual knowledge. New Transformer-based embeddings may provide better representational similarity but this does not generalize to all similarity measures and might be dependent on the dataset.

Differences in Representational Similarity across Similarity Datasets

The similarity of object images seems to be worse encoded than word similarity. This could be caused by several differences in the THINGS dataset, for example, a focus on nouns, visual dimensions, the usage of rare objects, or the usage of more homonyms.

By focusing on nouns of concrete objects, the power of large language models might not be fully exploited as fewer linguistic properties might be needed for similarity prediction. In contrast, the semantic comparison of two verbs might be easier to encode in text and therefore, easier to encode in word embeddings. As word similarity datasets are composed of adjectives, verbs, and nouns and not only nouns, it might be easier for text-based embedding approaches to encode the relevant information.

As the data collection is based on a triplet image odd-one-out task, participants are also biased to use visual properties for similarity comparison. This can be seen in the resulting interpretable dimensions where visual dimensions like *colorful*, *red* and *white* emerged. The approach to directly predict these dimension values indicates that these dimensions are harder to predict by word embeddings as it might be harder to encode visual information from text. Humans might use other information in text when they describe an object than when they see an image of the same object. For example, a cardinal (bird) might be described more taxonomic or conceptual in text but might be more described with the color red when it is seen in an image.

The usage of rare objects in the THINGS dataset might lead to embeddings with less training data as rare objects are also less frequent in text. Yet, the analysis of the influence of word frequency showed no strong effect. Future analyses could use the Rare-Words dataset to further investigate this question.

The usage of homonyms in the THINGS dataset might lead to worse representational similarity. For example, the THINGS dataset contains homonyms like *mouse*, *bat* and *calf* where multiple word senses are used in the dataset. Further, other words like *cardinal*, *beetle*, or *gyro* are present in the dataset where only one word sense is used. The usage of word embeddings leads to the same embeddings for multiple word senses. This might be a reason why the word sense embeddings from the Deconflated embedding model have the highest correlation with similarity ratings from the THINGS dataset. Future investigations using the number of word senses per word could give more insights into this question.

Chapter 8

Conclusion

This work evaluated several word and word sense embeddings from different embedding models. Static word and word sense embeddings as well as Transformer-based contextualized and decontextualized word and word sense embeddings were created and evaluated using intrinsic and extrinsic evaluations.

It was shown that human similarity ratings can be well encoded in different word embeddings. Yet, this depends on the similarity rating dataset. Word similarity can be well represented by word embeddings while similarity of object images is not well represented across all embedding types. Transformer-based embeddings reflect this worse than standard static embeddings. The extrinsic evaluation showed that static word embeddings can still outperform newer Transformer-based embeddings.

Several methods to improve Transformer-based embeddings and to create new static decontextualized word embeddings were performed. It was shown that decontextualization of Transformer-based embeddings can lead to improved performance. This depends on the chosen model, extraction layer, and the number of contexts used for aggregation.

A method to create decontextualized word and word sense embeddings was developed which can be used with any Transformer model and text corpus. The creation of word sense embeddings did not yield improvements.

Chapter 9

Future Work

Future work could further investigate newer language models like the DeBERTa or GPT-3 models as they can achieve higher performance on natural language tasks.

Transformer models use a higher context window, e.g. BERT-base and GPT-2 use 512 tokens compared to five to ten tokens in Word2Vec. Therefore, the influence of different context window lengths could be investigated. A shorter context length could influence the contextualization effect on word embeddings which could lead to a smaller difference in representational similarity between contextualized and decontextualized word embeddings.

Word embeddings could further be used as a replacement for humans in the method from Hebart et al. (2020) where the triplet odd one out task could be based on triplet word embeddings. The odd one out would then be defined as the object with the lowest word embedding similarity. This could lead to more insights into which dimensions are used by large language models. Further, the original data collection process of Hebart et al. (2020) could be repeated with the usage of words instead of images. This could lead to a higher representational similarity between large language models and humans as there will be no visual bias.

Most Transformer-based models use a tokenizer in the processing of text which can lead to identical subtokens in words. This can have a positive influence on similar words like *gyro* and *gyroscope* when both words refer to the word sense of a tool but can have a negative influence on dissimilar words like *teepee* and *t-shirt*.

The isotropy postprocessing method from Mu and Viswanath (2017) has a positive influence on representational similarity with humans based on word similarity. This effect could further be studied using new methods, e.g. the BERT-flow method from Li et al. (2020), to circumvent the anisotropy of contextualized word embeddings.

The influence of the used text corpora could be further investigated. For example, domain-specific corpora might increase the representational similarity of words belonging to a domain.

These future steps could lead to more insights about the encoding of information in word embeddings and language models as well as new embeddings with higher quality.

References

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., & Soroa, A. (2009). A study on similarity and relatedness using distributional and wordnet-based approaches. In *Naacl hlt 2009 - human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics, proceedings of the conference.* doi: 10.3115/1620754.1620758

Akbik, A., Bergmann, T., & Vollgraf, R. (2019). *Pooled contextualized embeddings for named entity recognition* (Vol. 1; Tech. Rep.). Retrieved from `https://github.com/zalandoresearch/flair` doi: 10.18653/v1/n19-1078

Bakarov, A. (2018). *A Survey of Word Embeddings Evaluation Methods* (Tech. Rep.).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*. doi: 10.1162/tacl_a_00051

Bommasani, R., Davis, K., & Cardie, C. (2020). Interpreting Pretrained Contextualized Representations via Reductions to Static Embeddings. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 4758–4781). Association for Computational Linguistics. Retrieved from `https://tedunderwood.com/2019/07/15/` doi: 10.18653/v1/2020.acl-main.431

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Conference proceedings - emnlp 2015: Conference on empirical methods in natural language processing.* doi: 10.18653/v1/d15-1075

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. In *Advances in neural information processing systems* (Vol. 2020-Decem).

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2018). SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation.. doi: 10.18653/v1/s17-2001

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St John, R., ... Kurzweil

Google Research Mountain View, R. (2018). Universal Sentence Encoder. *AAAI*.

Chronis, G., & Erk, K. (2020). When is a bishop not like a rook ? When it's like a rabbi! Multi-prototype BERT embeddings for estimating semantic relationships. *Proceedings ofthe 24th Conference on Computational Natural Language Learning, Online, November 19-20, 2020*, 227–244. Retrieved from https://doi.org/10.18653/v1/P17 doi: 10.18653/v1/P17

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., & Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. In *Emnlp 2017 - conference on empirical methods in natural language processing, proceedings.* doi: 10.18653/v1/d17-1070

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6). doi: 10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

Devereux, B. J., Tyler, L. K., Geertzen, J., & Randall, B. (2014, dec). The Centre for Speech, Language and the Brain (CSLB) concept property norms. *Behavior Research Methods*, *46*(4), 1119–1127. Retrieved from http://link.springer.com/10.3758/s13428-013-0420-4 doi: 10.3758/s13428-013-0420-4

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pretraining of deep bidirectional transformers for language understanding. In *Naacl hlt 2019 - 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies - proceedings of the conference* (Vol. 1).

Ethayarajh, K. (2019). *How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings* (Tech. Rep.).

Firth, J. (1957). A Synopsis of Linguistic Theory 1930-55. *Studies in Linguistic Analysis: Special Volume of the Philological Society*.

Grand, G., Blank, I. A., Pereira, F., & Fedorenko, E. (2022). Semantic projection recovers rich human knowledge of multiple object features from word embeddings. *Nature Human Behaviour*. Retrieved from https://doi.org/10.1038/s41562-022-01316-8 doi: 10.1038/s41562-022-01316-8

Gupta, P., & Jaggi, M. (2021, aug). Obtaining Better Static Word Embeddings Using Contextual Embedding Models. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)* (pp. 5241–5253). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/

2021.acl-long.408 doi: 10.18653/v1/2021.acl-long.408

He, P., Liu, X., Gao, J., & Chen, W. (2021). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *ArXiv*, *abs/2006.0*.

Hebart, M. N., Dickter, A. H., Kidder, A., Kwok, W. Y., Corriveau, A., Van Wicklin, C., & Baker, C. I. (2019). THINGS: A database of 1,854 object concepts and more than 26,000 naturalistic object images. *PLoS ONE*, *14*(10). doi: 10.1371/journal.pone.0223792

Hebart, M. N., Zheng, C. Y., Pereira, F., & Baker, C. I. (2020). Revealing the multidimensional mental representations of natural objects underlying human similarity judgements. *Nature Human Behaviour*, *4*(11). doi: 10.1038/s41562-020-00951-3

Hill, F., Reichart, R., & Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (Genuine) similarity estimation. *Computational Linguistics*, *41*(4). doi: 10.1162/COLI_a_00237

Hollenstein, N., De La Torre, A., Langer, N., & Zhang, C. (2019). CogniVal: A framework for cognitive word embedding evaluation. In *Conll 2019 - 23rd conference on computational natural language learning, proceedings of the conference.* doi: 10.18653/v1/k19-1050

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012). Improving word representations via global context and multipleword prototypes. In *50th annual meeting of the association for computational linguistics, acl 2012 - proceedings of the conference* (Vol. 1).

Huang, L., Sun, C., Qiu, X., & Huang, X. (2020). *Glossbert: BERT for word sense disambiguation with gloss knowledge* (Tech. Rep.). Retrieved from https://github.com/HSLCY/GlossBERT

Jang, B., Kim, I., & Kim, J. W. (2019). Word2vec convolutional neural networks for classification of news articles and tweets. *PLoS ONE*, *14*(8). doi: 10.1371/journal.pone.0220976

Jurafsky, D., & Martin, J. (2009). Speech and Language Processing. In *Speech and language processing.* (Vol. 2).

Kaniuth, P., & Hebart, M. N. (2020). Tuned representational similarity analysis: Improving the fit between computational models of vision and brain data. *Journal of Vision*, *20*(11). doi: 10.1167/jov.20.11.1076

Khattak, F. K., Jeblee, S., Pou-Prom, C., Abdalla, M., Meaney, C., & Rudzicz, F. (2019). *A survey of word embeddings for clinical text* (Vol. 4). doi: 10.1016/j.yjbinx.2019.100057

Khatua, A., Khatua, A., & Cambria, E. (2019). A tale of two epidemics: Contextual Word2Vec for classifying twitter streams during outbreaks. *Information Processing and Management*, *56*(1). doi: 10.1016/j.ipm.2018.10.010

Kilgarriff, A., & Fellbaum, C. (2000). WordNet: An Electronic Lexical

Database. *Language*, *76*(3). doi: 10.2307/417141

Kriegeskorte, N., Mur, M., & Bandettini, P. (2008). Representational similarity analysis - connecting the branches of systems neuroscience. *Frontiers in Systems Neuroscience*, *2*(NOV). doi: 10.3389/neuro.06.004.2008

Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems* (Vol. 3).

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., & Li, L. (2020). On the sentence embeddings from pre-trained language models. In *Emnlp 2020 - 2020 conference on empirical methods in natural language processing, proceedings of the conference.* doi: 10.18653/v1/2020.emnlp-main.733

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, *abs/1907.1*.

Lucy, L., & Gauthier, J. (2017). Are distributional representations ready for the real world? evaluating word vectors for grounded perceptual meaning. In *Proceedings of the 1st workshop on language grounding for robotics, robonlp 2017 at the 55th annual meeting of the association for computational linguistics, acl 2017.* doi: 10.18653/v1/w17-2810

Luong, M. T., Socher, R., & Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. In *Conll 2013 - 17th conference on computational natural language learning, proceedings.*

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Acl-hlt 2011 - proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (Vol. 1).

McRae, K., Cree, G. S., Seidenberg, M. S., & McNorgan, C. (2005). Semantic feature production norms for a large set of living and nonliving things. *Behavior Research Methods*, *37*(4). doi: 10.3758/BF03192726

Melamud, O., Goldberger, J., & Dagan, I. (2016). context2vec: Learning generic context embedding with bidirectional LSTM. In *Conll 2016 - 20th signll conference on computational natural language learning, proceedings.* doi: 10.18653/v1/k16-1006

Merchant, A., Rahimtoroghi, E., Pavlick, E., & Tenney, I. (2020). What Happens To BERT Embeddings During Fine-tuning? In *Proceedings of the third blackboxnlp workshop on analyzing and interpreting neural networks for nlp* (pp. 33—-44). Association for Computational Linguistics. doi: 10.18653/v1/2020.blackboxnlp-1.4

Merity, S., Xiong, C., Bradbury, J., & Socher, R. (2017). Pointer sentinel mixture models. In *5th international conference on learning representations, iclr 2017 - conference track proceedings.*

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *1st international conference on learning representations, iclr 2013 - workshop track proceedings.* International Conference on Learning Representations, ICLR.

Mu, J., & Viswanath, P. (2017, feb). All-but-the-top: Simple and effective post-processing for word representations. *arXiv*. Retrieved from `http://arxiv.org/abs/1702.01417`

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Emnlp 2014 - 2014 conference on empirical methods in natural language processing, proceedings of the conference.* doi: 10.3115/v1/d14-1162

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Naacl hlt 2018 - 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies - proceedings of the conference* (Vol. 1). doi: 10.18653/v1/n18-1202

Pilehvar, M. T., & Collier, N. (2016). De-conflated semantic representations. In *Emnlp 2016 - conference on empirical methods in natural language processing, proceedings.* doi: 10.18653/v1/d16-1174

Radford, A., Narasimhan, T., Salimans, T., & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training. In *Preprint.*

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners..

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Emnlp-ijcnlp 2019 - 2019 conference on empirical methods in natural language processing and 9th international joint conference on natural language processing, proceedings of the conference.* doi: 10.18653/v1/d19-1410

Rubinstein, D., Levi, E., Schwartz, R., & Rappoport, A. (2015). How well do distributional models capture different types of semantic knowledge? In *Acl-ijcnlp 2015 - 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing of the asian federation of natural language processing, proceedings of the conference* (Vol. 2). doi: 10.3115/v1/p15-2119

Saedi, C., Branco, A., António Rodrigues, J., & Silva, J. (2018). WordNet Embeddings. In *Proceedings of the third workshop on representation learning for NLP* (pp. 122–131). Melbourne, Australia: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/W18-3016` doi: 10.18653/v1/W18-3016

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.* arXiv. Retrieved from http://arxiv.org/abs/1910.01108 doi: 10.48550/ARXIV.1910.01108

Scarlini, B., Pasini, T., & Navigli, R. (2020a). SensEmBERT: Context-Enhanced Sense Embeddings for Multilingual Word Sense Disambiguation. In *Aaai 2020 - 34th aaai conference on artificial intelligence* (Vol. 34, pp. 8758–8765). doi: 10.1609/aaai.v34i05.6402

Scarlini, B., Pasini, T., & Navigli, R. (2020b). With More Contexts Comes Better Performance: Contextualized Sense Embeddings for All-Round Word Sense Disambiguation. In *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* (pp. 3528–3539). Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.285

Sien, S. K. (2015). Adapting word2vec to Named Entity Recognition. *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*(Nodalida).

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., Thomas McCoy, R., ... Pavlick, E. (2019). What do you learn from context? Probing for sentence structure in contextualized word representations. In *7th international conference on learning representations, iclr 2019.*

Trask, A., Michalak, P., & Liu, J. (2015). sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings. *CoRR*, *abs/1511.0*. Retrieved from http://arxiv.org/abs/1511.06388

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (Vol. 2017-Decem).

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th international conference on learning representations, iclr 2019.*

Wang, C., Nulty, P., & Lillis, D. (2020). A Comparative Study on Word Embeddings in Deep Learning for Text Classification. In *Acm international conference proceeding series.* doi: 10.1145/3443279.3443304

Wiedemann, G., Remus, S., Chawla, A., & Biemann, C. (2019). *Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings* (Tech. Rep.).

Williams, A., Nangia, N., & Bowman, S. R. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Naacl hlt 2018 - 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies - proceedings of the conference* (Vol. 1). doi: 10.18653/v1/n18-1101

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *2015 ieee international conference on computer vision (iccv)* (pp. 19–27). doi: 10.1109/ICCV.2015.11

Appendix A

Dimension Prediction

Table A.1 shows the 49 interpretable dimensions of the THINGS dataset. The dimension names can be used in section *Prediction of THINGS Dimension Embeddings* where the dimension ID is shown in the plots.

Figure A.1 shows the results for the BERT-large model, Figure A.2 for the GPT-2 model and Figure A.3 for the GPT-2-medium model.

Table A.1: Number and names of THINGS dimensions

Dimension ID	Dimension Name
1	made of metal / artificial / hard
2	food-related / eating-related / kitchen-related
3	animal-related / organic
4	clothing-related / fabric / covering
5	furniture-related / household-related / artifact
6	plant-related / green
7	outdoors-related
8	transportation / motorized / dynamic
9	wood-related / brownish
10	body part-related
11	colorful
12	valuable / special occasion-related
13	electronic / technology
14	sport-related / recreational activity-related
15	disc-shaped / round
16	tool-related
17	many small things / course pattern
18	paper-related / thin / flat / text-related
19	fluid-related / drink-related
20	long / thin
21	water-related / blue
22	powdery / fine-scale pattern
23	red
24	feminine (stereotypically) / decorative
25	bathroom-related / sanitary
26	black / noble
27	weapon / danger-related / violence
28	musical instrument-related / noise-related
29	sky-related / flying-related / floating-related
30	spherical / ellipsoid / rounded / voluminous
31	repetitive
32	flat / patterned
33	white
34	thin / flat
35	disgusting / bugs
36	string-related
37	arms/legs/skin-related
38	shiny / transparent
39	construction-related / physical work-related
40	fire-related / heat-related
41	head-related / face-related
42	beams-related
43	seating-related / put things on top
44	container-related / hollow
45	child-related / toy-related
46	medicine-related
47	has grating
48	handicraft-related
49	cylindrical / conical

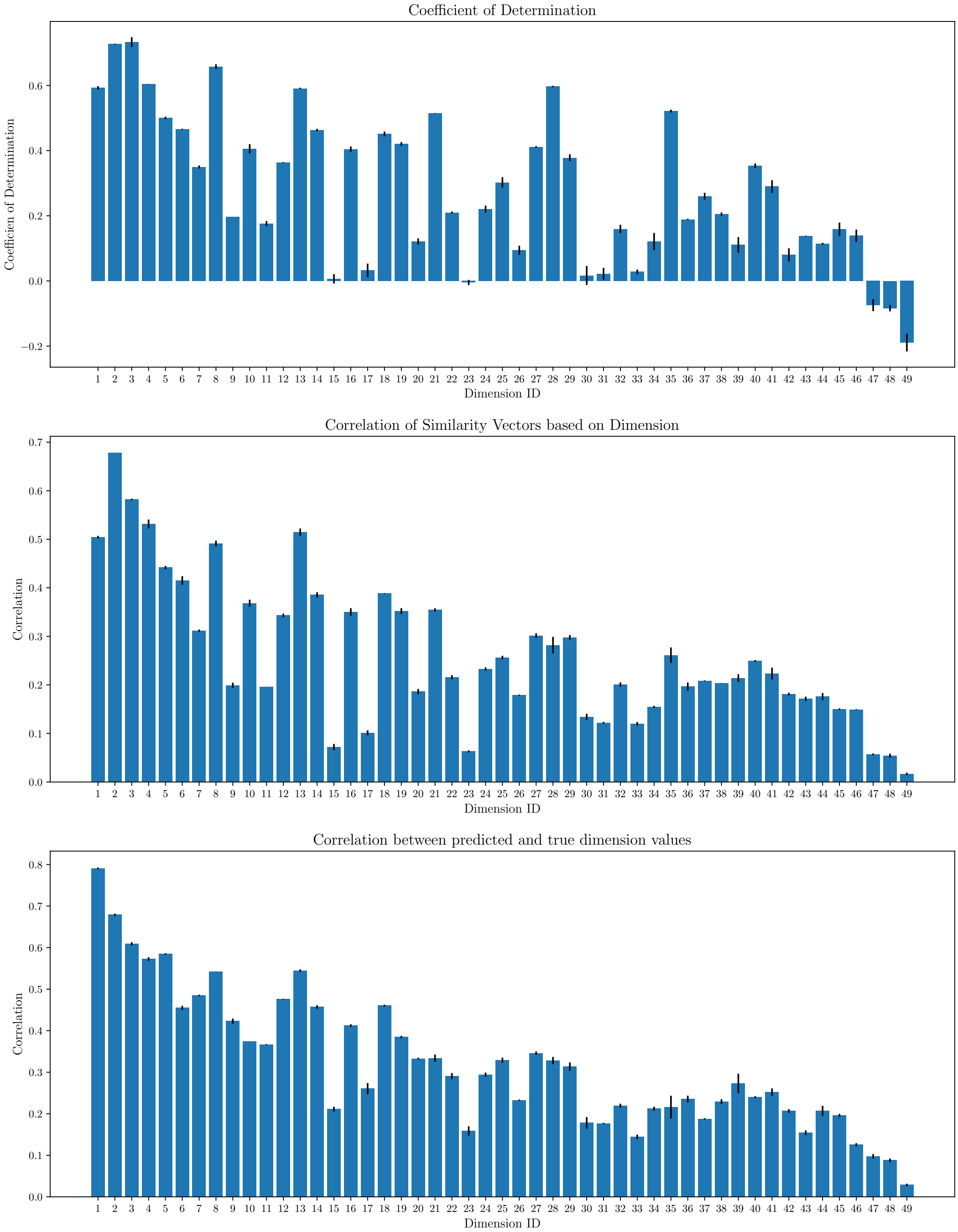


Figure A.1: Metrics for dimension prediction for decontextualized BERT-large word embeddings

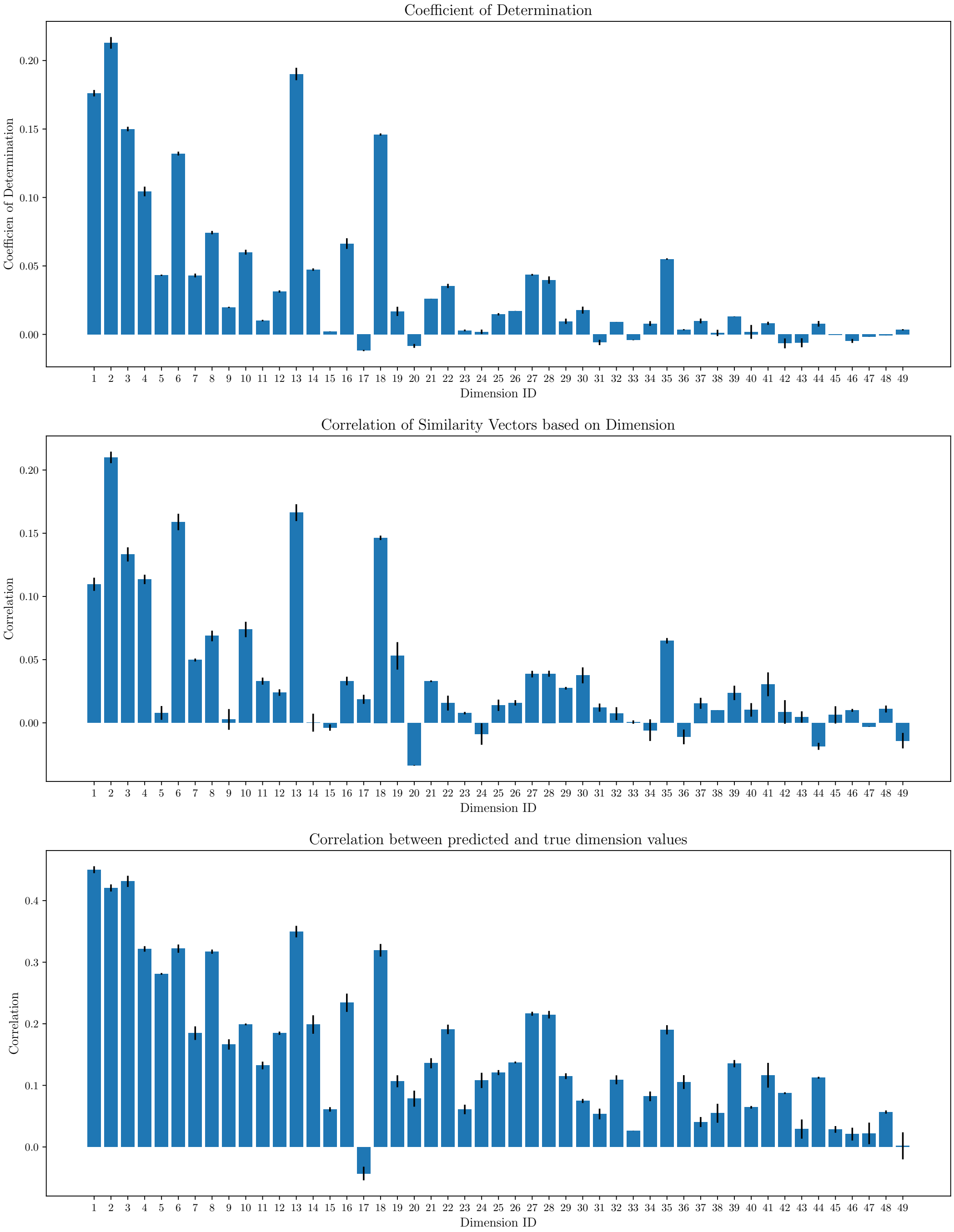


Figure A.2: Metrics for dimension prediction for decontextualized GPT-2 word embeddings

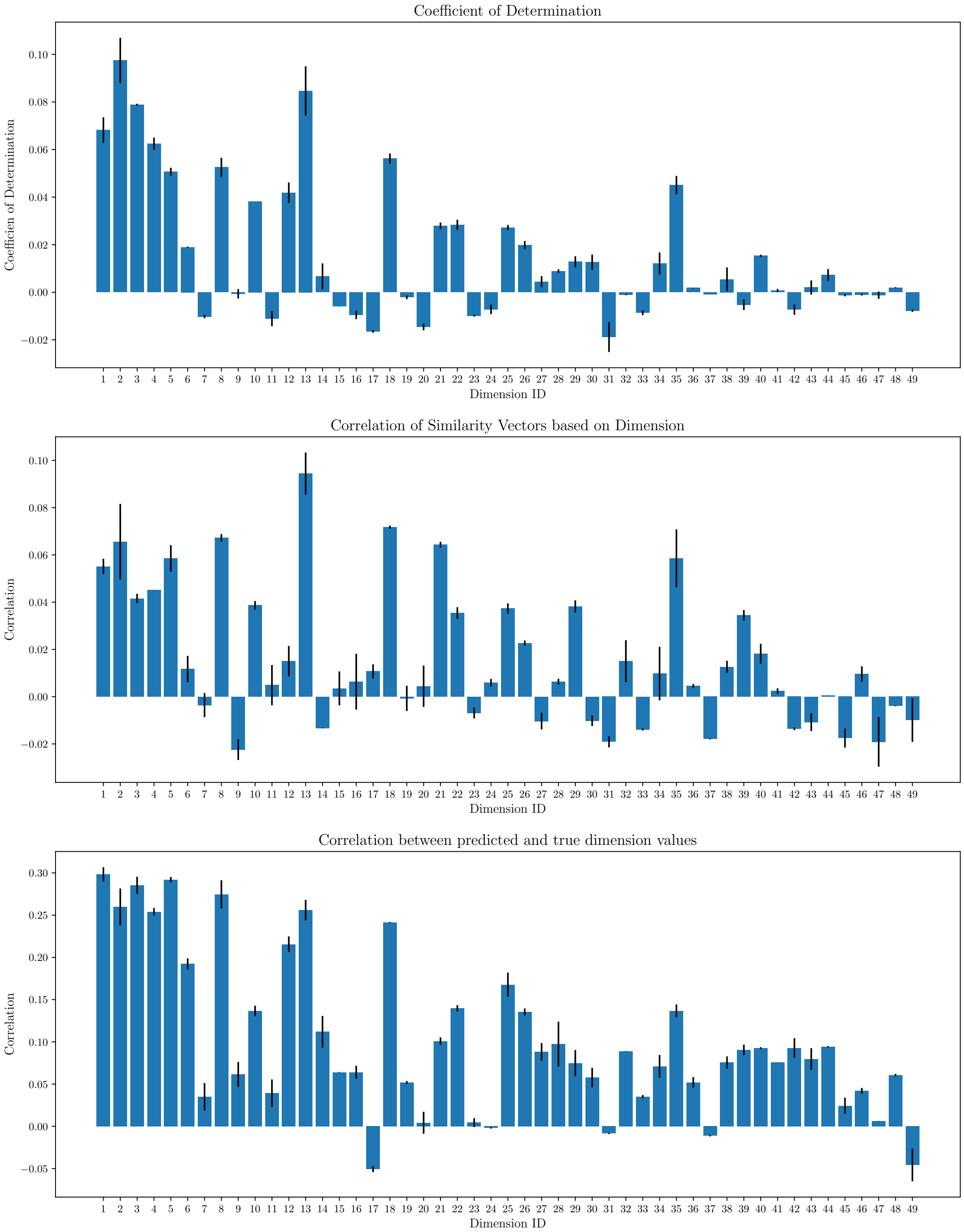


Figure A.3: Metrics for dimension prediction for decontextualized GPT-2-medium word embeddings