The Relation Between Words and Definitions in Combined Word-Sentence Space

June 23, 2021

Abstract

Methods for learning vector space representations of words have yielded spaces which contain semantic and syntactic regularities. These regularities mean that vector arithmetic operations in the latent space represent meaningful and interpretable relations between words. These word vectors have been so successful in capturing such relations, as well as transfer between domains almost seamlessly, that they have become central in the foundation of modern natural language processing.

There have been multiple proposals for extending these methods to sentences and documents, as well as entirely new approaches based on modern sequence models. So far, none of these methods have demonstrated the same kind of widespread applicability as word vector methods, instead being constrained to a single domain. The question that emerges is then whether these document vector methods yield spaces with the same kind of regularities as are observed in the word vector models.

This work focusses on whether these spaces encode a particular relation, that between a word and its definition. Since most of these methods allow only for a conversion from sentence to vector and not the reverse, the problem has been phrased as a ranking problem over a set of candidate words. In the strict case where the relation is assumed to be linear as with the word vectors this yields a model which ranks the correct vector first 26.6% of the time, with a median rank of the correct answer of 19 out of 2000 options. Relaxing this requirement and using a 3 layer Multi Layer Perceptron yields an improvement in this metric, predicting 37.8% correct, and improves the median rank to 3. The performance of these models suggests that words and sentences can be naturally thought of as occupying a single space.

The results in this work suggest that it may be possible to generate correct definitions of words in a way that comes very close to being unsupervised, needing only a mean difference between words and definitions. When this was attempted, however, the decoder ended up converging to a fixed output.

1 Introduction

Vector representations of words are a foundational element of most modern approaches in varous tasks, such as information retrieval (Galke, Saleh, and Scherp 2017), sentiment analysis (Yu et al. 2017), document classification (Kusner et al. 2015), and question answering (Zhou et al. 2015), where they are used as the input features for more complex models. Much of the early research relied only on angles or distances between vectors as a way to examine and report the quality of their models.

However, recently a new evaluation method has been introduced which takes the entire vector difference between embeddings into account. The first ones to introduce this idea were Mikolov, Yih, and Zweig 2013 who demonstrated that word vectors could be used to solve analogies. These analogies took the form of equations such as ”king” - ”man” + ”woman”, where each word is replaced by its respective word vector, which results in a vector very close to that of ”queen”, examples of this can be seen in figure 1. By designing their word embedding model to perform well on this task they managed to correctly answer 40% of the analogies they created, provided in a dataset named SemEval-2012 Task 2 questions.

This work was improved upon by Pennington, Socher, and Manning 2014 who examined the

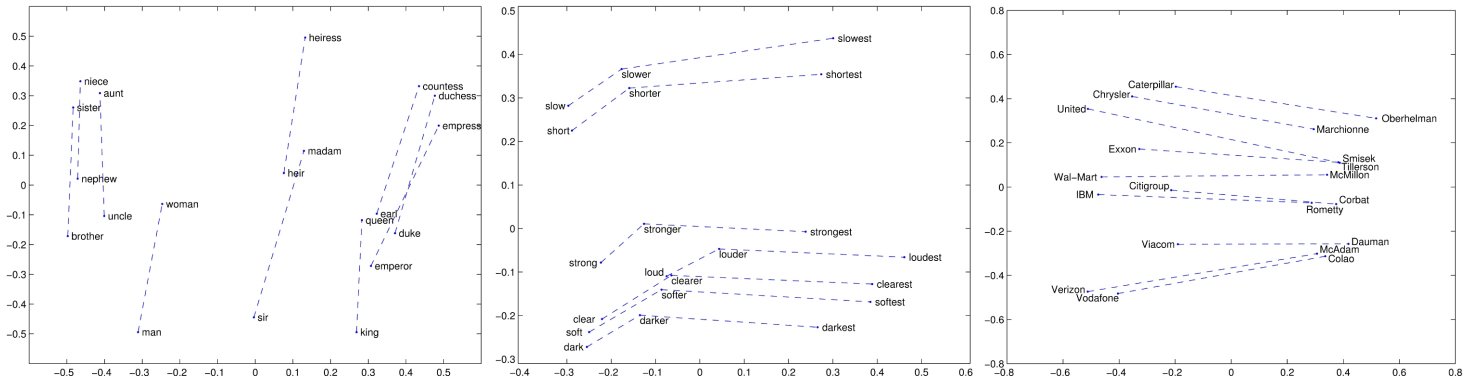

Figure 1: GloVe vectors contain linear substructures that capture the relations between words, figures from Pennington, Socher, and Manning 2014.

properties a model must have to create linear directions of meaning. They make a case for global log-bilinear regression models, and propose a least-squares model which is trained on global word co-occurrence counts. The resulting vectors, which they call Global Vectors (GloVe), achieve a 75% accuracy on the aforementioned word analogy dataset, as well as improving on results in word similarity and named entity recognition benchmarks.

Word vectors stand as one of the most widespread and successful cases of transfer learning, single models often being used across vastly different domains. However, many natural language processing (NLP) tasks do not concern themselves with words but rather sentences and documents. Therefore, ever since word vectors have shown to be successful in pushing forward the state of the art in NLP, there has been research into the possibility of extending such techniques to sequences of words. Such techniques include word vector summarization, in which a document vector is created by averaging over the word vectors in the document. Lately, however, much of NLP has moved to the Transformer architecture, which is in essence a feed-forward architecture that has attention layers which allow for routing information horizontally across the sequence. A particular Transformer named BERT (Devlin et al. 2018) is often used as a starting point for many NLP tasks. BERT is trained to reconstruct sequences of text in which some words have been masked out, predicting these masked words is what forces BERT to encode the structure of language. The set of tasks for which BERT is used includes the creation of sequence embeddings. Sentence-BERT (Reimers and Gurevych 2019) for example uses pre-trained BERT networks (Devlin et al. 2018) to create sentence embeddings.

While words and sentences are usually considered separate entities, it is possible to think of them as part of a single continuum. In encoding schemes such as the Huffman encoding (Huffman 1952) the most common messages are assigned the shortest codes, doing so minimizes the average message length. In language too, the most common words are short while uncommon words are longer on average. But what happens in a language when a message becomes too rare to express even in a long word? Such messages are expressed in sentences. The cutoff point between words and sentences varies between languages, with some languages allowing for the construction of very long words where others prefer sentences. In light of this it is possible to consider character sequences of any length, from the shortest words to the longest documents, to be part of the same space. If this is the case, then an embedding method that maps such sequences into a continuous space may also show the same kind of linear directions of meaning as those observed in word vectors.

This work examines the degree to which several document embedding techniques are capable of solving a specific analogy: that between words and their definitions. The main observations that are made are i) that all examined document embedding methods are significantly better than chance, ii) that document embedding methods have gotten

better at this task in recent years, and iii) that all document embedding methods still perform quite poorly at this task leaving significant room for improvement. The performance of these models also suggests that words and sentences can be naturally thought of as occupying a single space.

2 Related Work

2.1 Word Vectors

A defining feature of the language models used in deep learning is their high-dimensional continuous vector representation of words. Such language models (Bengio et al. 2003; Schwenk 2007) convert words into real-valued vectors using a lookup table, allowing them to be used as input features for neural networks. The GloVe algorithm has been designed specifically for the purpose of creating linear directions of meaning (as in figure 1). It does this by using a learning objective that ensures the following equality holds as well as possible:

$$e^{(\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{w}_k} \approx \frac{p(i|k)}{p(j|k)} \tag{1}$$

where i, j, and k are words in the corpus, $\mathbf{w}_i$ the word vector associated with word i, and $p(i|k)$ being the probability that word i occurs in the context of word k. This probability $p(i|k)$ is the smoothed co-occurrence count of words i and k in the training data set. Smoothing is performed to ensure that $p(i|k) > 0$, which accounts for the possibility of word co-occurrences that are not recorded in the data. Fixing this relation between the three word vectors empirically results in an embedding space that has the desired linear structures.

The GloVe algorithm works by first randomly initializing two word vectors $\mathbf{w}$ and $\tilde{\mathbf{w}}$ and two biases b and $\tilde{b}$ for each word in the corpus. The word coocurrence counts are aggregated into a matrix $\mathbf{X}_{i,j}$ which contains the amount of times word i occurs in the context, a fixed width window, of word j. Then the embedding vectors are optimized using AdaGrad until they converge, using the following optimization objective:

$$L = \sum_{i,j=1}^{V} f(\mathbf{X}_{i,j})(\mathbf{w}_i^T \tilde{\mathbf{w}}_j + b_i + \tilde{b}_j - \log \mathbf{X}_{i,j}) \tag{2}$$

where V is the number of words in the corpus, and $f(x)$ is a weighting function that satisfies three conditions i) $f(0) = 0$, ii) it is non-decreasing, iii) it is relatively small for large values of x. As a result of empirical evaluation the GloVe authors chose this function to be:

$$f(x) = \begin{cases} (\frac{x}{100})^{\frac{3}{4}} & \text{if } x < 100 \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

This process yields two sets of word vectors, $\mathbf{W}$ and $\tilde{\mathbf{W}}$ which are then added $\hat{\mathbf{w}}_i = \mathbf{w}_i + \tilde{\mathbf{w}}_i$ in order to obtain the final word vectors (section 4.2 of Pennington, Socher, and Manning 2014).

2.2 Transformers

The Transformer (Vaswani et al. 2017) is a neural network architecture that processes an entire input sequence in parallel, this is in contrast to recurrent neural networks which process the input sequence one step at a time. Transformer encoders consist of a stack of self-attention layers interspersed with fully connected layers. These self-attention layers exist to route information from one part of the sequence to another, while the fully connected layers transform the inputs.

Self-attention layers receive a sequence of inputs $\mathbf{x}_{i,j}$, collectively represented as $\mathbf{X}_j$, where i represents the position in the sequence and j is the position in the layers of the transformer. This input is transformed into a sequence of queries $\mathbf{Q}_j = \mathbf{X}_j^T \mathbf{W}_{1,j}$, keys $\mathbf{K}_j = \mathbf{X}_j^T \mathbf{W}_{2,j}$, and values $\mathbf{V}_j = \mathbf{X}_j^T \mathbf{W}_{3,j}$. These operations are performed once for each head of the attention mechanism, after which the attention head is evaluated as follows:

$$\mathbf{A}_j = \text{softmax}(\frac{\mathbf{Q}_j^T \mathbf{K}_j}{\sqrt{d_k}})\mathbf{V}_j \tag{4}$$

Where d_k is the dimensionality of the key vectors $\mathbf{k}_{i,j}$. The output of the attention head is then added to the input sequence to create the next activations $\mathbf{H}_j = \mathbf{X}_j + \mathbf{A}_j$, all attention heads are added here in parallel. After the evaluation of the attention mechanism a fully connected layer is applied to allow for some extra information processing $\mathbf{X}_{j+1} = \mathbf{H}_j + \mathbf{W}_{FC,j}\mathbf{H}_j$. These two operations, the self-attention layer and the fully connected layer,

are then applied another 11 times to obtain the output sequence, which has the same length and dimensionality as the input sequence. The parameters are not shared between layers but they are shared for each of the elements in the sequence. For this reason Transformers can be seen as a generalization of Convolutional Neural Networks where the kernel is not restricted to a local area, but rather learns which sequence elements are relevant at any point.

2.2.1 BERT

BERT (Devlin et al. 2018) is a transformer network (Vaswani et al. 2017) which achieved state-of-the-art results on several language processing tasks, including the General Language Understanding Evaluation (GLUE) benchmark (Wang et al. 2018), and the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016). It was trained on a large corpus of text using two unsupervised tasks. The first task was masked language modeling in which a percentage of the tokens in the input sequence is masked out, BERT is then tasked to predict the masked tokens. In this prediction the model can make use of both the information preceding as well as the information succeeding the token. This bidirectional nature of the model makes it strictly more powerful than the unidirectional alternatives. The second task was Next Sentence Prediction (NSP) in which BERT is presented with two sentences with the goal to predict whether the second sentence succeeded the first sentence in the corpus. This second task is particularly important for tasks which require knowledge on the relation between sentences, such as Natural Language Inference (NLI) and Question Answering (QA). BERTs architecture consists of 12 (base) self-attention layers which contains 110 million parameters, or 24 (large) layers containing 340 million parameters, and in each of the tasks is trained as a regression model. RoBERTa (Liu et al. 2019) showed that making some modifications to the pre-training protocol could improve the models overall performance.

2.3 Sentence BERT

Because training a document embedding model from scratch is expensive it is useful to take a transfer learning approach. Finetuning a pretrained language model allows levaraging the knowledge of language already present in those models. This approach was taken by Reimers and Gurevych 2019 who used the afforementioned BERT as the basis for a model, called Sentence-BERT (SBERT), that achieved state-of-the-art results on common Semantic Textual Similarity (STS) tasks. One of the challenges of using BERT in STS tasks is that finding the semantic similarity of any pair of sentences requires concatenating the two sentences and performing a full evaluation of the BERT model, none of this computation can be reused.

This means that in order to find the sentence pair with the highest similarity $n(n-1)/2$ evaluations are required, where n is the number of sentences. Instead SBERT evaluates each of the sentences separately and uses the mean of the outputs of each sentence as the vector embedding of the sentence. These vector embeddings are then compared using the cosine similarity, doing so reduces the time complexity of the model from $O(n^2)$ to $O(n)$ while achieving the same performance.

To create a single vector from a sentence the outputs of the final layer of BERT are averaged. This averaging of the output of BERT produces worse sentence representations than averaging GloVe vectors. BERT therefore has to be finetuned before it can be used for this task. The finetuning process involves the use of a dataset containing labeled sentence pairs. Each of the two sentences in the sentence pair is separately encoded by using BERT and its outputs are mean pooled. The two vectors are then compared by computing the cosine similarity between them. The optimization process drives sentences considered 'semantically similar' in the dataset closer together while driving those that are considered dissimilar further apart. This finetuned BERT model is called Sentence-BERT or SBERT.

3 Creating and Evaluating Sentence Vectors

This section highlights the models that are being examined, as well as the criteria used in this examination. Conceptually, this section can be split into three parts. First, section 3.1 explains how the sentence embeddings are created. Second, sections 3.2 and 3.3 detail how these embeddings are evalu-

ated and how words and their definitions are associated. Finally, section 3.4 covers a method which is used to attempt to recover the text from the embeddings. Together these three steps form a pipeline that could take a definition as an input, use the described methods to first create an embedding, then to predict the embedding of the associated word, and then decode this embedding to obtain the text of the word.

3.1 Creating Sentence Vectors

GloVe vectors are only created for words, but they are used in one of the most popular ways of creating sentence vectors. The most straightforward process for creating sentence vectors is to compute an unweighted average of the word vectors of the words in the sentence.

This means that the word vectors belonging to the words in the word-definition pairs are simply reinterpreted as a single-word sentence vector. The definition consisting of m words, whose word vectors are individually labelled $\mathbf{w}_i$, are then converted into a single vector $\mathbf{s}$ by averaging them, as follows:

$$\mathbf{s} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{w}_i \tag{5}$$

This vector $\mathbf{s}$ is taken to represent the sentence as a whole.

Sentence-BERT already performs this averaging procedure internally. Therefore transforming the word, definition pairs into sentence vectors is done by separately passing the word and definition through SBERT.

3.2 Evaluation of Linear Directions of Meaning

Word vector representations are known to have many interesting structures contained in them. A key property of word vectors is that they tend to contain linear relations between concepts in a way that transfers between words (as seen in figure 1).

The question is then whether similar structures exist in sentence vectors. To this end the pairs of words and definitions were transformed to word vectors for words $\mathbf{w}_i$ and sentence vectors for definitions $\mathbf{s}_i$, with $i \in [1, n]$.

If there does exist a linear relation between the words and definitions, then the difference vector $\mathbf{d}_i$ between the word vector $\mathbf{w}_i$ and the definition vector $\mathbf{s}_i$ of its definition would be roughly the same for all these pairs. In other words:

$$\mathbf{d}_i = \mathbf{w}_i - \mathbf{s}_i \tag{6}$$

$$\text{with } \|\mathbf{d}_i - \mathbf{d}_j\| < \varepsilon \text{ for all } i, j \tag{7}$$

For this to be the case the length of the difference vectors separating words from their definitions should be the same too, and different from that of random word vector pairs.

$$|\,\|\mathbf{d}_i\| - \|\mathbf{d}_j\|\,| < \varepsilon \text{ for all } i, j \tag{8}$$

Additionally, all these difference vectors should be pointing in the same direction. This can be measured by comparing the difference vectors to the mean difference vector $\hat{\mathbf{d}}$ using the cosine similarity.

$$\hat{\mathbf{d}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{d}_i \tag{9}$$

3.3 Predicting Words from Their Definitions

If the relation between word vectors and their associated definition vectors is similarly encoded in a linear direction in the document vector space, then the mean difference vector can be interpreted as containing the concept "the definition of". Adding this mean difference vector to the definition vectors would then approximately yield the word vectors. This means that:

$$\mathbf{w}_i \approx \mathbf{s}_i + \hat{\mathbf{d}} \tag{10}$$

Because the embeddings obtained from SBERT can not be turned back into words, the evaluation of the prediction accuracy is done by selecting the word whose word vector has the highest cosine similarity with the predicted word vector.

While word and sentence vector models are often designed specifically to have linear directions

of meaning, it is possible that such models have additional nonlinear structure. In order to uncover these structures, and to improve performance, a nonlinear regression model is used. This nonlinear regression model (NRM) is trained to predict the difference vector between the word and definition vectors as in the following formula:

$$\mathbf{w} = \mathbf{s} + NRM(\mathbf{s}) \tag{11}$$

This word vector $\mathbf{w}$ is then tested against the ground truth $\hat{\mathbf{w}}$ using the cosine embedding loss.

3.4 Recovering Text from the Embeddings

So far it has been explained how it is possible to produce an embedding vector from a piece of text, and how to associate the embedding vectors of a word and its definition. Unfortunately, such an association requires that both the words and their definitions are already known. This leads to the question of whether the text can somehow be recovered from these embedding vectors. If this were possible then it would potentially enable the generation of a word from its definition and vice versa.

This would require a model capable of generating text from such an embedding $\mathbf{e}$, estimating the distribution:

$$p(w_{n+1}|\mathbf{e}, w_1, ..., w_n)$$

Where w_i represent the words in the text. While such decoder models are hard to come by in modern natural language processing, much work has been dedicated to a related model type. Causal Language Models (CLM) predict the probability of the next word in a sequence without relying on some embedding, in other words they approximate the following distribution:

$$p((w_{n+1}|w_1, ..., w_n)$$

Such CLMs have also proven to be very adaptable through finetuning and have therefore been used as a basis for many applications. Perhaps, then, it could be possible to finetune a CLM model to perform the task of decoding a sentence embedding vector.

For this purpose GPT-2 was chosen. Normally the input GPT-2 receives consists of a sequence of vectors each representing a word or word segment, this input is then used to predict the next word or word segment in the sequence. In this case, however, the first element of each input sequence given to GPT-2 is the embedding $\mathbf{e}$ of the full text created using SBERT. GPT-2 is then finetuned to reproduce the text that SBERT used to create the embedding $\mathbf{e}$.

4 Experiments

4.1 Evaluation

To facilitate the aim of examining the relation between words and their definitions an english dictionary was used as a source for word-definition pairs[1]. The data is cleaned by first removing all punctuation, a After removing the word-definition pairs which contain words not available in the GloVe model this dictionary yielded a set of $n = 10161$ pairs of words and definitions. These word-definition pairs form the primary way of evaluating the ability of the various models to encode higher level concepts. Each of the models is used to predict both words from their definitions as well as the inverse. To ensure that the results are stable the prediction is done using 5 fold cross-validation, the reported results are the average of these. This means that the models are evaluated on 2123 word-definition pairs. To determine whether a prediction of a word vector is correct, the predicted word vector is compared to the word vectors for each of the words in the validation set. If the word vector that is closest to the predicted word vector corresponds to the correct word, then the prediction is considered correct. The same holds for the reverse, where definition vectors are predicted from the word vector belonging to the defined word. This is naturally extended to correct-at-k which considers whether the correct vector is in the top k closest vectors.

4.2 Training details

The GloVe vectors used and are included in the `gensim` package for python[2].

The 3 layer MLP used in conjunction with the embeddings generated by the pretrained trans-

[1] https://github.com/adambom/dictionary
[2] https://radimrehurek.com/gensim/

Correct at	1	5	10	median rank	MCS	MSE
Mean difference vector GloVe 50d	$7.9 \pm 0.65\%$	$15.8 \pm 1.4\%$	$20.4 \pm 1.6\%$	175 ± 17	0.34 ± 0.25	0.38 ± 0.18
Mean difference vector GloVe 100d	$11.0 \pm 0.4\%$	$19.3 \pm 1.2\%$	$23.7 \pm 1.3\%$	148 ± 11	0.27 ± 0.22	0.21 ± 0.10
Mean difference vector GloVe 200d	$15.7 \pm 0.9\%$	$26.0 \pm 0.9\%$	$31.8 \pm 1.0\%$	81 ± 7	0.26 ± 0.19	0.16 ± 0.06
Mean difference vector GloVe 300d	$17.7 \pm 1.1\%$	$28.6 \pm 1.1\%$	$33.7 \pm 1.4\%$	68 ± 9	0.27 ± 0.17	0.14 ± 0.04
GloVe 300d	$3.8 \pm 0.4\%$	$6.8 \pm 0.5\%$	$8.4 \pm 0.5\%$	423 ± 16	0.06 ± 0.21	0.17 ± 0.05
RoBERTa base NLI STSb 768d	$22.8 \pm 1.0\%$	$34.6 \pm 0.4\%$	$39.5 \pm 0.7\%$	38 ± 3	0.42 ± 0.19	0.55 ± 0.18
3 layer NN + RoBERTa base NLI STSb 768d	28.7%	42.5%	48.5%	11	0.49	0.36
distilRoBERTa base paraphrase 768d	$26.6 \pm 0.9\%$	$39.4 \pm 0.9\%$	$44.7 \pm 1.1\%$	19 ± 2	0.43 ± 0.14	0.11 ± 0.03
3 layer NN + distilRoBERTa base paraphrase 768d	37.8%	52.2%	58.6%	3	0.44	0.07

Table 1: Percentage of words correctly predicted from their definition vectors through various methods. Followed by the median rank of the correct answer, and three distance metrics between the prediction and the answer, Mean Absolute Error (MAE), Mean Cosine Similarity (MCS), and Mean Squared Error (MSE). 2000 words were available to choose from.

former models has a total of 163600 parameters as a result of the input and output layers containing 768 dimensions and the two hidden layers having 100 dimensions each Each of the hidden layers of the MLP uses the ReLU activation function. This MLP was trained on the word-definition pair dataset with a 80-20 train-test split. To compare the predicted output of the network with the ground truth, mean squared error loss was used. The training consisted of 100 epochs with a learning rate of 0.0001 using the AdamW optimizer included in PyTorch.

Finetuning GPT-2 was done on the same dataset, the input was modified as described in section 3.4. The finetuning consisted of 10 epochs with a learning rate of 0.0001 using the AdamW optimizer included in PyTorch. GPT-2 was separately finetuned on a much smaller set of only 10 word-definition pairs for 100 epochs with a learning rate of 0.0001 using AdamW.

4.3 Results

The results of predicting the word vector from the definition vector can be found in table 1. The GloVe model serving as a baseline shows significant improvement as the dimensionality of the model increases. This improvement, from 7.96% correct-at-1 and a median rank of the correct answer of 175 to a 17.71% correct-at-1 and a median rank of 68 does, however, show that the word-averaging approach of GloVe seems to have diminishing returns for higher dimensionality. It can also be seen that within this model the Mean Absolute Error (MAE) and Mean Squared Error (MSE) show the same pattern of improvement, both decreasing as the dimensionality goes up, as the correct-at-1 and median rank. A similar pattern of improvement can be seen in the Mean Cosine Similarity (MCS) which gets closer to the optimal point of 1 as the dimensionality of the embedding increases.

Comparing instead the two transformer-based document embedding models it can be seen that the kind of pretraining that is done has a significant impact on performance even if the dimensionality of the models is the same. Again the MAE and MSE mirror this improvement, while the MCS doesn't seem to be useful. Since the neural network was trained to maximize the cosine similarity between prediction and ground truth it was expected to see an improvement of performance on this measure. Significant improvement in all metrics can be made by using a 3 layer fully connected neural network to predict the difference vector from the sentence vector. This leads to the observation that there appears to be additional nonlinear structure in the embeddings that can be exploited.

The results of the inverse task, that of predicting the definition vector from the word vector, can be found in table 2. What immediately stands out is the improvement in performance seen for GloVe vectors, having a 2.34% improvement for the 50 di-

Correct at	1	5	10	median rank	MCS	MSE
Mean difference vector GloVe 50d	$10.3 \pm 0.70\%$	$20.1 \pm 1.3\%$	$25.6 \pm 1.4\%$	113 ± 10	0.53 ± 0.20	0.38 ± 0.18
Mean difference vector GloVe 100d	$14.2 \pm 1.0\%$	$24.9 \pm 0.6\%$	$30.4 \pm 0.7\%$	99 ± 3	0.53 ± 0.17	0.21 ± 0.10
Mean difference vector GloVe 200d	$19.5 \pm 0.9\%$	$31.8 \pm 1.1\%$	$36.9 \pm 1.1\%$	54 ± 5	0.42 ± 0.18	0.16 ± 0.06
Mean difference vector GloVe 300d	$21.3 \pm 1.1\%$	$33.8 \pm 1.4\%$	$38.8 \pm 1.5\%$	44 ± 7	0.32 ± 0.19	0.14 ± 0.04
RoBERTa base NLI STSb 768d	$22.1 \pm 0.6\%$	$33.8 \pm 1.0\%$	$38.7 \pm 0.7\%$	41 ± 5	0.39 ± 0.20	0.55 ± 0.18
3 layer NN + RoBERTa base NLI STSb 768d	33.0%	50.0%	57.4%	4	0.54	0.33
distilRoBERTa base paraphrase 768d	$26.6 \pm 0.9\%$	$39.4 \pm 0.9\%$	$44.7 \pm 1.1\%$	19 ± 2	0.43 ± 0.14	0.11 ± 0.03
3 layer NN + distilRoBERTa base paraphrase 768d	37.2%	53.2%	59.7%	3	0.48	0.07

Table 2: Percentage of definitions correctly predicted from their word vectors through various methods. Followed by the median rank of the correct answer, and three distance metrics between the prediction and the answer, Mean Absolute Error (MAE), Mean Cosine Similarity (MCS), and Mean Squared Error (MSE). 2000 words were available to choose from.

mensional model and a 3.38% improvement for the 300 dimensinoal model. These improvements are mirrored by an increase in the MCS, while the MAE and MSE remain the same due to the symmetry between this task and the previous one. The fact that the GloVe based approaches perform better on this inverse task is somewhat surprising. In fact since definition vectors are the average of a group of word vectors, and since definitions often contain the same words (such as 'the'), it would be expected that definition vectors are closer together and therefore more difficult to predict. For the trasformer based models, in contrast, there does not appear to be much of a difference in performance on this inverse task. While the median correct-at-k and median rank measures do not seem to be impacted much, there is a significant decrease in the MCS.

4.3.1 GloVe Vectors

Taking a more in depth look at the relations between word vectors and definition vectors created using Glove, it can be seen in figure 2 that the lengths of the difference vectors are relatively tightly clustered as would be expected from the fact that the models perform reasonably well. Somewhat surprising is that these vectors do not appear to all point in the same direction, only having a few difference vectors that have a greater than 0.7 cosine similarity with the mean difference vector.

Figure 3 further shows that the cosine similarity between a word vector and its definition vector is

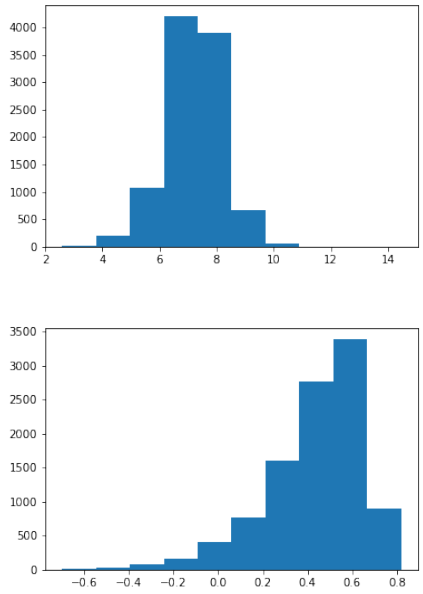

Figure 2: The distribution of the lengths of the difference vectors (top), and the cosine similarity between the individual difference vectors and the mean difference vector (bottom), for 300 dimensional GloVe vectors

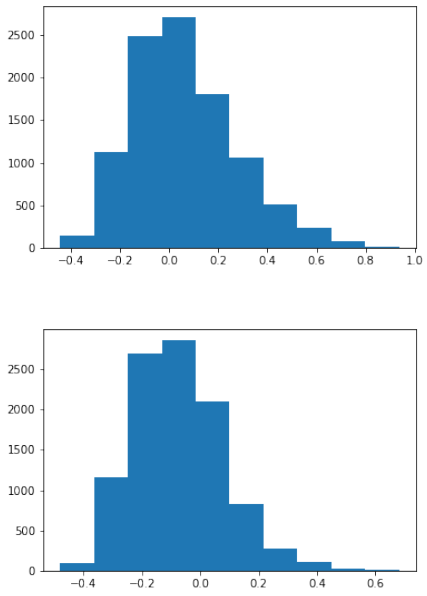

Figure 3: The cosine similarity between a word vector and its definition vector (top) , and between a word vector and a random definition vector (bottom), for 300 dimensional GloVe vectors.

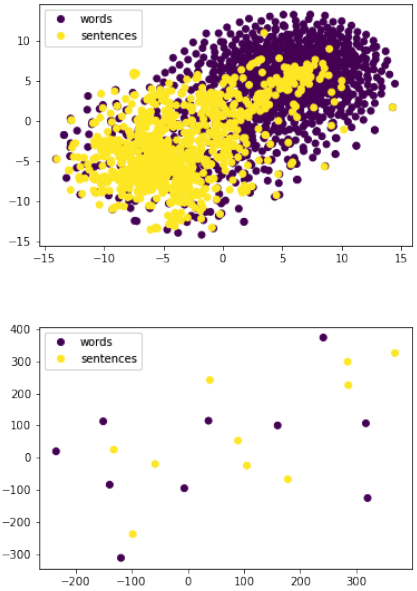

Figure 4: t-SNE plots of the embeddings created using GloVe. The top plot shows the embeddings as they are generated, while the bottom plot shows 10 pairs where the mean difference vector has been added to the sentences.

slightly greater than that between a word vector and a random definition vector. Strangely enough this pattern is not visible when comparing the euclidean distance between these two vectors.

These points are further demonstrated by the t-SNE plot of the embeddings. Which shows that while these embeddings form somewhat distinct clusters for the words and sentences, these clusters merge into one when the mean difference vector is added to the sentence embeddings. It can also be seen that the cluster for sentences is slightly smaller than that for words, which is the result of the averaging of the individual vectors in the sentences. Note however, that the fact that the sentence embeddings are shorter does not impact the accuracy of the predicions seen in tables 1 and 2, since the predictions use cosine similarity which does not take the length of the vector into accounts.

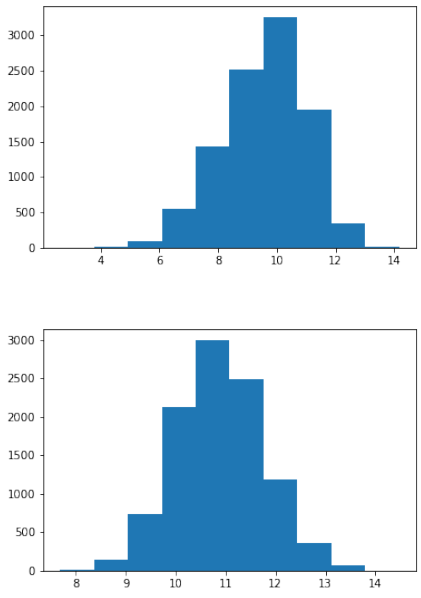

Figure 5: Plotting the length of the difference between a word vector and the sentence vector representing its definition (top), and the difference between a word vector and a random sentence vector (bottom). Using the model distilRoBERTa.

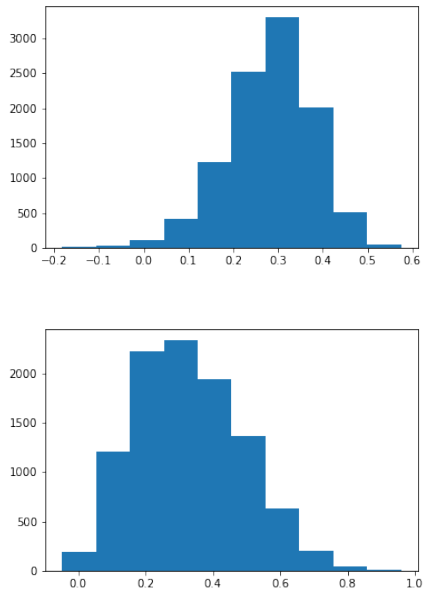

Figure 6: The cosine similarity between the individual difference vectors and the mean difference vector (top), and the cosine similarity between a word vector and its definition vector (bottom). Using the model distilRoBERTa.

4.3.2 Sentence-BERT

When examining the relation between word embedding and definition embedding it became immediately clear that there was a much clearer relation in these embeddings than those created using GloVe. In figure 5 this can be seen from the fact that the difference vectors between correct word-definition pairs have a different distribution than that between random word-definition pairs.

In figure 6 it can be seen that the cosine similarity between the individual difference vectors and the mean difference vector is fairly low. This is somewhat surprising given that the use of this mean difference vector nonetheless leads to the best performance for predicting both the word vectors from their definition vectors as well as the reverse. On the other hand, the distribution of cosine similarites between the word vectors and their definition vectors is clearly positive for distilRoBERTa, contrast this with the same plot for GloVe (figure 3) where the vectors have significantly lower cosine similar-

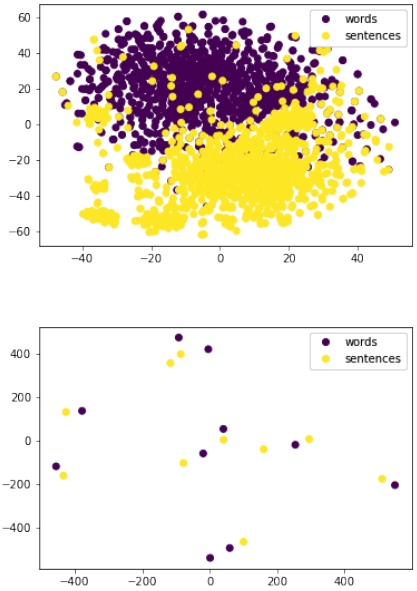

Figure 7: t-SNE plots of the embeddings created using the distilRoBERTa paraphrase model. The top plot shows the embeddings as they are generated, while the bottom plot shows 10 pairs where the mean difference vector has been added to the sentences.

ity.

Similar to how the t-SNE plot showed somewhat distinct clusters for the words and sentences, this pattern also holds for the distilRoBERTa embeddings. And again the clusters merge into one once the mean difference vector is added to the sentence embeddings. The improvements the MLP made in prediction accuracy can be seen in figure 8.

4.4 Finetuning GPT-2

Unfortunately, the results of finetuning GPT-2 do not show any ability to learn to make use of the embedding vector to recover a piece of text. Instead the finetuning process seems to have caused the model to learn to ignore this embedding. This leads the model to either generate random entries from the set of text sequences it was finetuned on, or to generate the same sequence each time. Examples of this behaviour can be seen in 3.

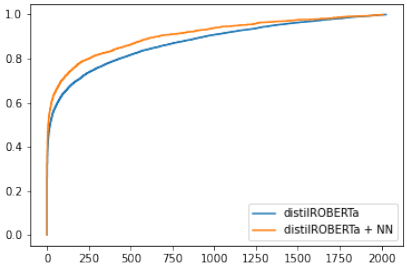

Figure 8: The cumulative percentage of predictions correct at the given rank for distilRoBERTa and distilRoBERTa with MLP

Input	Output
Diploblastic	Bahaism
Defigure	Bahaism
Lombard	Bahaism

Table 3: Examples of outputs generated by GPT-2. In each case, the input was embedded using SBERT after which GPT-2 was given the task of generating text given said embedding. In this case the model ended up generating the same output each time.

5 Conclusion

This work explores the relation between words and their definitions in the embedding spaces of various sentence embedding techniques. In doing so it is shown that these embedding techniques create embedding spaces in which words end up lying close to their definitions. Nonetheless the words and sentences still make up separate but somewhat overlapping clusters. By adding the mean difference vector between the two clusters it is possible to predict, with reasonable accuracy, the embedding vector of the word from the embedding vector of the definition and vice versa. This makes it possible to pick the correct word for a definition from a pool of thousands of candidates. The best models for other sentence embedding tasks are also the models that perform best at embedding words and their definitions in such a way that these predictions can be made at the highest accuracy. Despite the good performance these models have when simply using their embeddings and mean differences, significant improvement in prediction accuracy can

be achieved by training a 3 layer neural network to predict the difference vector for each word definition pair. Since the best performing model has a prediction accuracy of 37.8% and a median rank of 3, a lot of further improvement can be made by future sentence embedding methods.

The performance of these models suggests that words and sentences can be naturally thought of as occupying a single space. This is because if they were best considered separate then such embedding techniques would not place words and their definitions so close together and structured in such a way that they can be easily associated with one another. Future sentence embedding techniques should make use of this fact in the same way that word embedding techniques such as GloVe have been designed to explicitly model the relations between words.

The attempt to finetune GPT-2 to to decode the embedding vectors created by distilRoBERTa did not ultimately pan out. It instead ended up converging to outputting either unrelated text of a single element of the input set, depending on how long it was trained for. In encoder-decoder models the encoder and decoder are usually trained together. Should such a model be available, the results in this work suggest that it may be possible to generate correct definitions of words in a way that comes very close to being unsupervised, needing only a mean difference between words and definitions.

6 Discussion

6.1 Hyperbolic Embedding Spaces

The creation of embeddings in machine learning usually takes place in euclidean spaces. However, embedding tree-like and hirarchical structures in a euclidean space while maintaining the distances between related concepts turns out to be very difficult. Hyperbolic space can, informally, be thought of as the continuous version of the tree-like structure, and is therefore naturally capable of accomodating such an embedding. In fact, it has been demonstrated that any tree can be embedded in a hyperbolic space while maintaining distances (Gromov 1987). The use of hyperbolic spaces for creating word embeddings to embed the structure of WordNet, which is a dataset containing hirarchical relations between words, has show that a 5-dimensional hyperbolic space can better preserve distances between words than a 200-dimensional euclidean space (Nickel and Kiela 2017).

6.2 Violating Model Assumptions

While GPT-2 has been finetuned for various purposes involving generation of text, reconstructing text from an embedding generated by another model appears to be a step too far. A possible explanation for this outcome stems from some of the most basic design principles of neural network architectures.

Consider for example images. Regardless of whether an object appears in the top left or the bottom right of an image it should be seen as the same. This translational symmetry in the data can therefore naturally be captured in a model by using a convolutional layer, a layer that applies the same linear transformation to each part of the image. The inputs and outputs of a convolutional layer have a translational equivariance as a result of its design.

This relation between symmetries, parameter sharing, and equivariance can be seen everywhere in neural network design. Graph convolutional networks make use of the permutation symmetry present in the graphs by sharing the parameters for each of the nodes, leading to an equivariance between the graph and its embedding. Recurrent neural networks make use of the temporal symmetry present in sequence data, sharing parameters for all sequence elements and creating an equivariance between the input sequence and the hidden state of the RNN.

This same temporal symmetry is assumed in the data received by standard transformers, as evidenced by the fact that transformers share parameters for all sequence elements. Attempting to condition such a transformer on the output of another model by prepending a sequence embedding to the input of the transformer violates the temporal symmetry that forms the basis for the design of the model. This makes it exceptionally difficult for a transformer to deal with, because by design each sequence element is treated the same. For this reason it is likely that finetuning a model like GPT-2 for such a task will result in failure.

References

Bengio, Yoshua et al. (2003). "A neural probabilistic language model". In: *Journal of machine learning research* 3.Feb, pp. 1137–1155.

Devlin, Jacob et al. (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Galke, Lukas, Ahmed Saleh, and Ansgar Scherp (2017). "Word embeddings for practical information retrieval". In: *INFORMATIK 2017*.

Gromov, Mikhael (1987). "Hyperbolic groups". In: *Essays in group theory*. Springer, pp. 75–263.

Huffman, David A (1952). "A method for the construction of minimum-redundancy codes". In: *Proceedings of the IRE* 40.9, pp. 1098–1101.

Kusner, Matt et al. (2015). "From word embeddings to document distances". In: *International conference on machine learning*, pp. 957–966.

Liu, Yinhan et al. (2019). "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692*.

Mikolov, Tomáš, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic regularities in continuous space word representations". In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 746–751.

Nickel, Maximillian and Douwe Kiela (2017). "Poincaré embeddings for learning hierarchical representations". In: *Advances in neural information processing systems*, pp. 6338–6347.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

Rajpurkar, Pranav et al. (2016). "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250*.

Reimers, Nils and Iryna Gurevych (2019). "Sentence-bert: Sentence embeddings using siamese bert-networks". In: *arXiv preprint arXiv:1908.10084*.

Schwenk, Holger (2007). "Continuous space language models". In: *Computer Speech & Language* 21.3, pp. 492–518.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems*, pp. 5998–6008.

Wang, Alex et al. (2018). "Glue: A multi-task benchmark and analysis platform for natural language understanding". In: *arXiv preprint arXiv:1804.07461*.

Yu, Liang-Chih et al. (2017). "Refining word embeddings for sentiment analysis". In: *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 534–539.

Zhou, Guangyou et al. (2015). "Learning continuous word embedding with metadata for question retrieval in community question answering". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 250–259.

A Figures

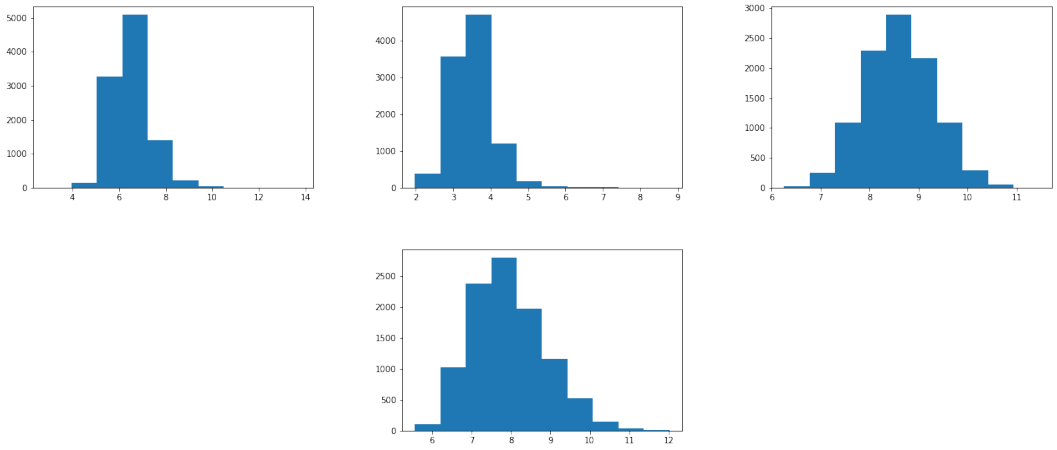

Figure 9: In reading order: word vector length in GloVe 300d, definition vector length in GloVe 300d, word vector length in distilROBERTa, definition vector length in distilROBERTa.

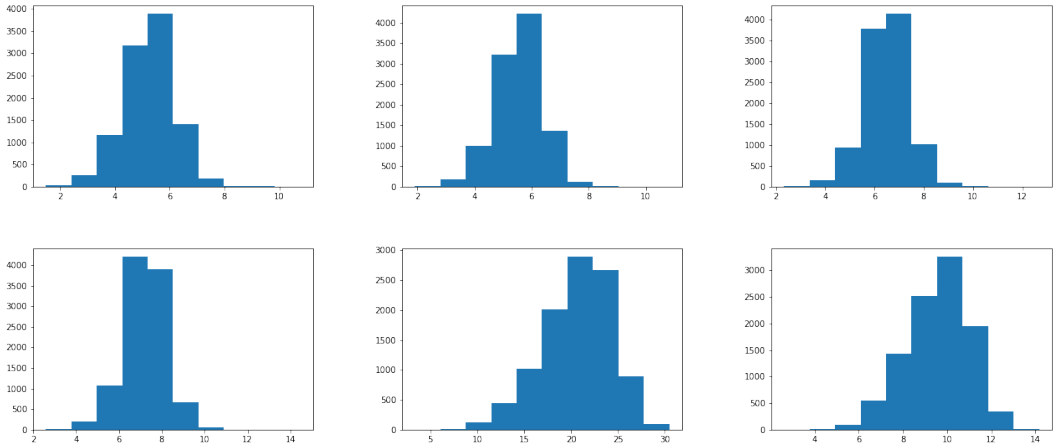

Figure 10: Plotting the length of the difference between a word vector and the sentence vector representing its definition for each of the models. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

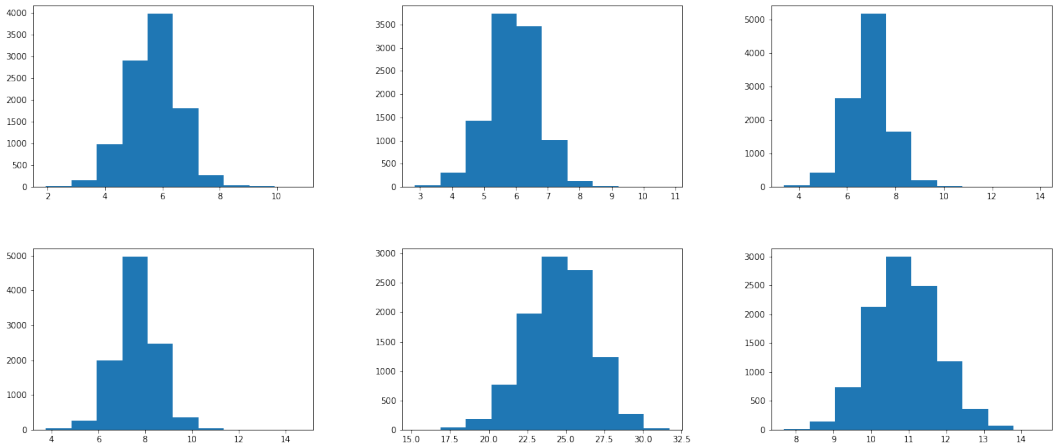

Figure 11: Plotting the length of the difference between a word vector and a random definition vector for each of the models. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

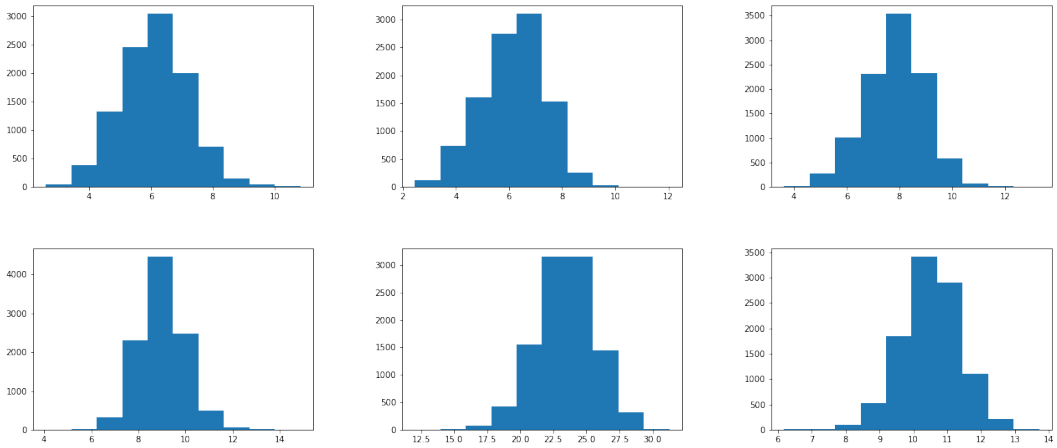

Figure 12: Plotting the length of the difference between a word vector and a random different word vector for each of the models. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

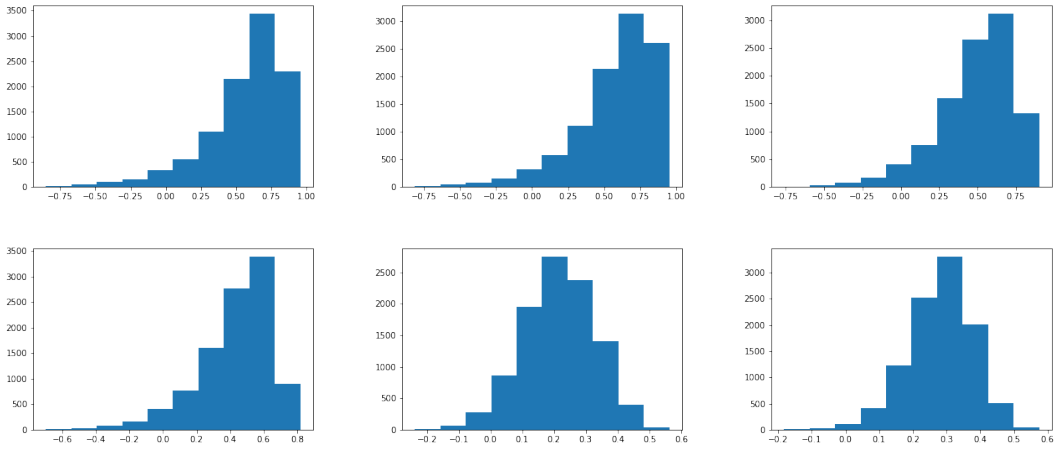

Figure 13: Histogram of the cosine similarity between the individual difference vectors and the mean difference vector. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

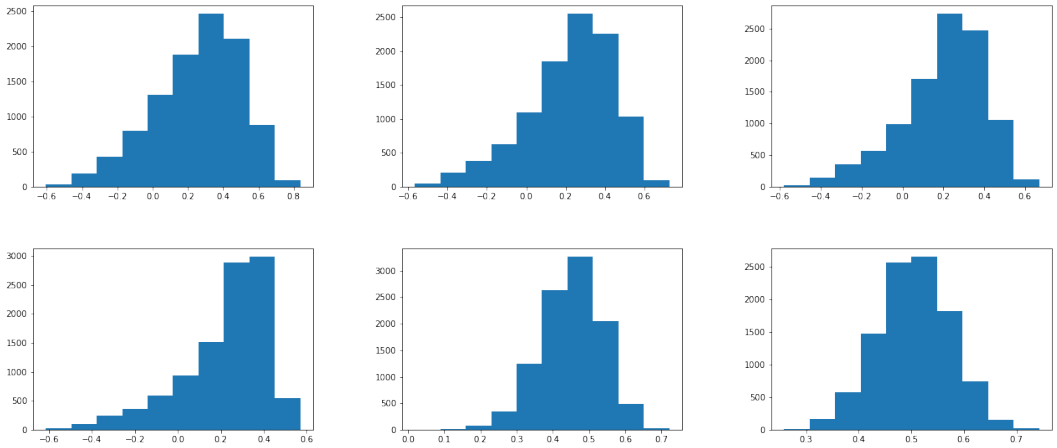

Figure 14: Histograms of the cosine similarity between the individual difference vectors and the mean word vector in the dataset. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

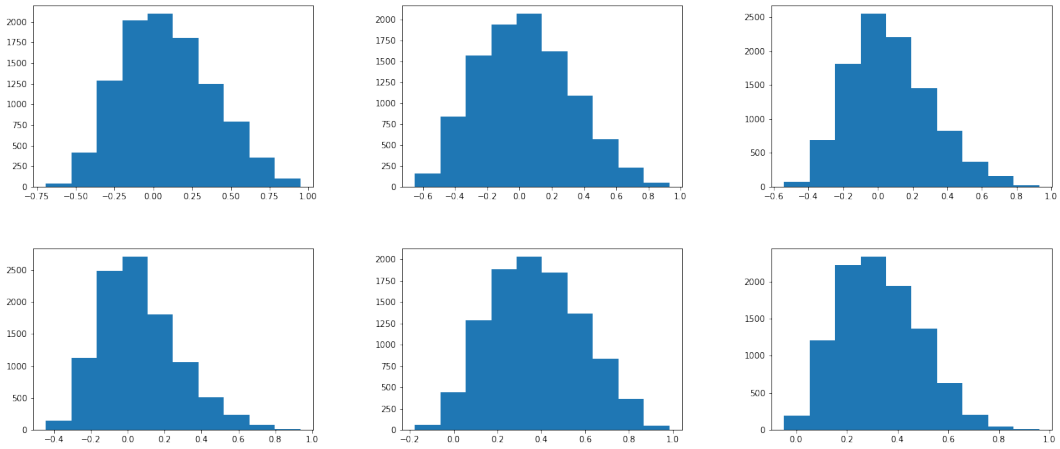

Figure 15: Histograms of the cosine similarity between a word vector and the sentence vector representing its definition. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

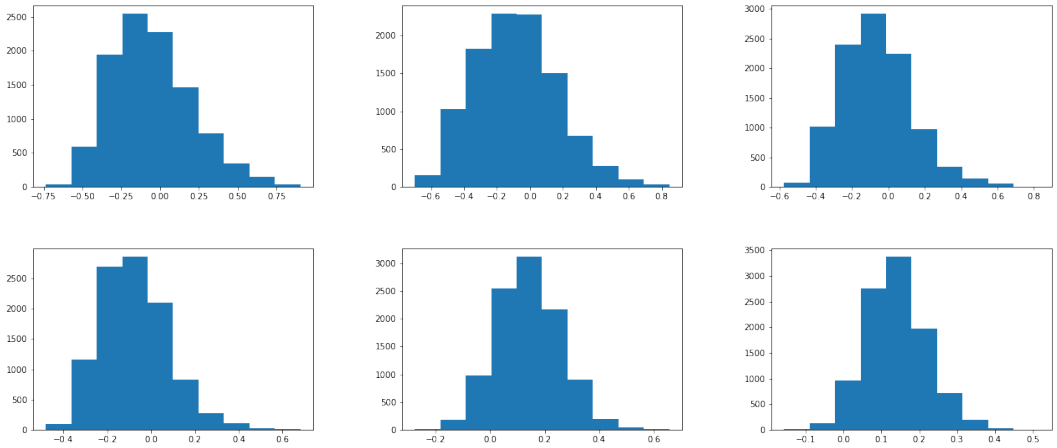

Figure 16: Histograms of the cosine similarity between a word vector a random sentence vector. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

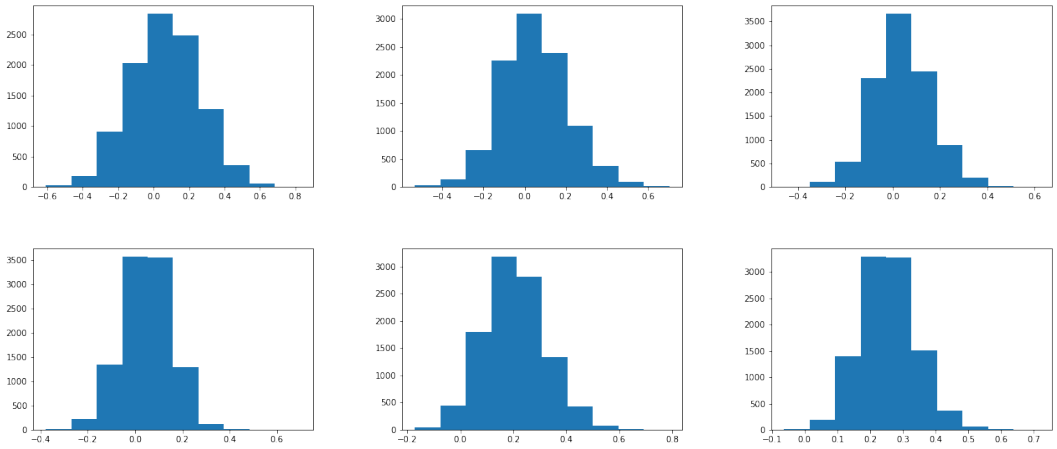

Figure 17: Histograms of the cosine similarity between a word vector a random different word vector. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

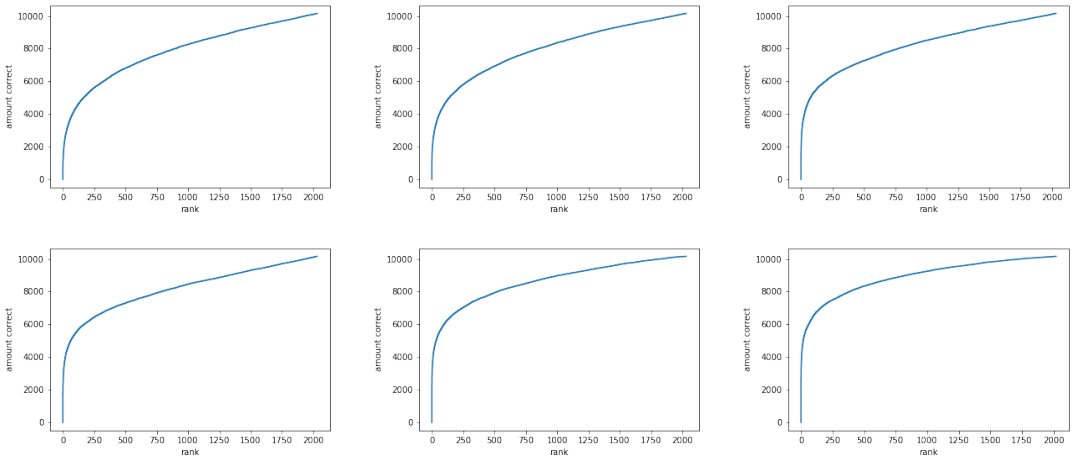

Figure 18: Amount of correct predictions at each rank for each model. In reading order from the top left: GloVe 50D, GloVe 100D, GloVe 200D, GloVe 300D, RoBERTa STSb, distilRoBERTa paraphrase.

B Words Predicted from Definitions

The following are a number of words predicted from the given definitions. Predictions made by the model 3 layer NN + distilRoBERTa baseparaphrase. Of the displayed examples 50% were predicted correctly.

Truth	Prediction	Definition
nucleolar	indecisively	of or pertaining to the nucleolus of a cell
tost	tost	imp p p of toss
rearmost	rearmost	farthest in the rear last
foxhound	foxhound	one of a special breed of hounds used for chasing foxes
rescuer	barque	one who rescues
blooming	barque	the process of making blooms from the ore or from cast iron
cranny	rudderless	a tool for forming the necks of bottles etc
turkle	prosecutable	a turtle obs or illiterate
interspace	interspace	intervening space bp hacket
splotch	splotch	a spot a stain a daub r browning
drily	analgesia	see dryly thackeray
adulterate	adulterate	to commit adultery obs
onwards	onwards	onward
masonic	prosecutable	of or pertaining to freemasons or to their craft or mysteries
twelfth	outward	an interval comprising an octave and a fifth
coarctation	moly	a stricture or narrowing as of a canal cavity or orifice
renovator	rankle	one who or that which renovates foster
supercilium	moly	the eyebrow or the region of the eyebrows
tuna	tuna	the opuntia tuna see prickly pear under prickly
strich	moly	an owl obs spenser
friending	friending	friendliness obs shak
avenge	adulterate	to take vengeance levit xix 18
custodianship	drily	office or duty of a custodian
winrow	moly	a windrow
unsubstantial	outward	lacking in matter or substance visionary chimerical
turkeys	drow	turkish obs chaucer
fish	fish	a counter used in various games
surge	sly	to slip along a windlass
promisee	promisee	the person to whom a promise is made
fand	moly	imp of find spenser
daddy	engineman	diminutive of dad dryden
scuff	scuff	the back part of the neck the scruff prov eng ld lytton
squarely	squarely	in a square form or manner
ethologist	ethologist	one who studies or writes upon ethology
narrowness	narrowness	the condition or quality of being narrow
khanate	khanate	dominion or jurisdiction of a khan
dimensioned	rudderless	having dimensions r
spectrogram	outward	a photograph map or diagram of a spectrum
middy	middy	a colloquial abbreviation of midshipman
hoop	moly	the hoopoe see hoopoe
resister	resister	one who resists
transgressive	transgressive	disposed or tending to transgress faulty culpable

nodule	prosecutable	a rounded mass or irregular shape a little knot or lump
typology	typology	a discourse or treatise on types
deterrence	deterrence	that which deters a deterrent a hindrance r
ethel	moly	noble obs
potent	potent	a staff or crutch obs
compiler	moly	one who compiles esp one who makes books by compilation
illegally	engineman	in a illegal manner unlawfully
engineman	engineman	a man who manages or waits on an engine
rudderless	rudderless	without a rudder
botts	botts	see bots
prosecutable	prosecutable	capable of being prosecuted liable to prosecution
looby	looby	an awkward clumsy fellow a lubber swift
concurrently	nucleolar	with concurrence unitedly
racemate	moly	a salt of racemic acid
dow	dow	a kind of vessel see dhow
restitute	restitute	to restore to a former state r dyer
seche	systemization	to seek obs chaucer
negress	negress	a black woman a female negro
drow	drow	of draw obs chaucer
assailant	assailant	assailing attacking milton
indecisively	prosecutable	without decision
sheard	sheard	see shard obs
skinless	concurrently	having no skin or a very thin skin as skinless fruit
keg	keg	a small cask or barrel
hemi	hemi	a prefix signifying half
esox	esox	a genus of freshwater fishes including pike and pickerel
disablement	disablement	deprivation of ability incapacity bacon
ambassadorial	moly	of or pertaining to an ambassador h walpole
gleeful	rudderless	merry gay joyous shak
checker	moly	one who checks
laddie	laddie	a lad a male sweetheart scot
rankle	rankle	to cause to fester to make sore to inflame r beau fl
moly	moly	a kind of garlic allium moly with large yellow flowers called also golden garlic
destin	destin	destiny obs marston
appear	moly	appearance obs j fletcher
analgesia	moly	absence of sensibility to pain quain
warmness	warmness	warmth chaucer
slotted	friending	having a slot
conclusory	prosecutable	conclusive r
outward	outward	external form exterior rso fair an outward and such stuff within shak
indictee	indictee	a person indicted
reinspect	moly	to inspect again
diuresis	ethel	free excretion of urine
steeplechasing	moly	the act of riding steeple chases
groove	groove	a shaft or excavation prov eng
beaver	beaver	an amphibious rodent of the genus castor

pelting	pelting	mean paltry obs shak
systemization	systemization	the act or process of systematizing systematization
curation	centaur	cure healing obs chaucer
male	prosecutable	see mal
sly	sly	slyly obs or poetic spenser
donator	donator	one who makes a gift a donor a giver
barque	moly	same as 3d bark n
emplace	sheard	to put into place or position to fix on an emplacement
centaur	moly	a fabulous being represented as half man and half horse
dreaminess	curation	the state of being dreamy
coon	sheard	a raccoon see raccoon
recommence	hoop	to commence again or anew