ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Business & Economics: Data Science & Marketing Analytics

How reliable are word embedding evaluators?

Name: Said Salman Sadjadi

Student Number: 622684

Supervisor: dr. Eran Raviv

Second Assessor: dr. Anastasija Tetereva

Date of final version: 03-03-2023

Contents

1 Introduction

My work shows in an exemplary analysis, that many evaluators used for word embedding models are outdated and their ability to reliably compare models with each other is not guaranteed. But first, word embedding models are mathematical models trained on textual data which can capture the semantic and syntactic structure of words from the text. These models find a broad field of application in natural language processing (NLP) tasks, from the auto corrector in text processing programmes to the newly developed and much discussed chatbot ChatGPT by OpenAI. The word embedding is the vectorial expression of a word, which holds its semantic and syntactic information as a combination of numeric values in a vector. The idea of word embeddings bases on the distributional hypothesis of linguistics, which states that words which often occur in same context purport similar meanings (Harris, 1954). So, for the word embedding/vector, words with similar meaning should have similar word vectors. Over the past decades many word embedding models were developed and with it many methods and evaluators were also created to quantifiably compare the goodness of different models' embeddings with each other. Evaluations are mainly done by utilizing the word embeddings (vector representation of a word) of a model to carry out a certain task. Tasks are commonly either extrinsic methods, which are common NLP applications like text classification or intrinsic methods, where word vectors are directly compared with each other. A detailed distinction between intrinsic and extrinsic methods is described in the section 3 Methodology. Depending on how well a task is carried out, a quantifiable performance metric can be calculated to determine the goodness of the model's word embeddings. In this study I will use 8 different evaluation tasks on 3 different word embedding models each to measure the usability of the used tasks. For that I will run significance analyses on the differences between the performances of two models on a same task. The usability or rather reliability of an evaluation task is proxied by measuring whether the difference in the performance between two models on the task is significant on a 5% significance level. This study shows that 4 out of the 8 used evaluation tasks are not usable to reliably compare word embedding models with each other. Drivers for this unreliability of the evaluation tasks are either the simplicity of the tasks and/or the small size of the underlying datasets of the evaluation tasks. My recommendation for future research is to fall back from the usage of many smaller evaluation tasks to only a few bigger and more sophisticated evaluation tasks to significantly measure small differences between models. Following, Section 2 will give an overview of the scientific literature on this topic and explain thoroughly my approach to measure the usability of certain evaluation tasks. In Section 3 the three used word embedding models and the utilized evaluation methods will be introduced. In Section 4 the datasets on which the word embedding models are trained and the 8 underlying datasets for the evaluation tasks are introduced and described. Section 5 covers the implementation and the results of my analysis. Lastly, in Section 6 I will conclude my analysis by summarizing my approach and results and stating important limitations of my work.

2 Theoretical Concept & Research Question

The purpose of word embedding models is mostly to be used in solving practical NLP problems, like language translation, text summarization and, etc. Because of that, evaluating word embedding models comes mostly with the demand to measure how well a model would perform on such a practical downstream NLP task. To do so evaluation tasks for word embedding models are either extrinsic tasks which depict such practical downstream NLP tasks or intrinsic tasks where word vectors are directly assessed and the performance on the intrinsic task should proxy the potential performance on the actual downstream task. A big difference between intrinsic and extrinsic tasks is that first intrinsic tasks are easy and fast to implement, whereas extrinsic tasks are mostly more complex, and second intrinsic tasks try to evaluate the word vectors directly whereas extrinsic tasks use separate models which must be trained. An example for an extrinsic task would be a text classification task, where different texts must be categorized into distinct groups. For that a separate classification model must be trained. A more extensive explanation of the distinction between intrinsic and extrinsic tasks is stated in Section 3 Methodology.

In (Wang et al., 2019) the researchers listed 5 desired properties for effective word embedding evaluators. The first property is that a variety of terms should be included in the underlying test dataset of the evaluator and that the data should be objectively correct. Second, an evaluator should at best test for different properties of the word embedding model and not focus only on one feature. Third, the performance of a word embedding model on an intrinsic task should correlate with the performance on downstream NLP tasks. Fourth, the implementation of the evaluator should be computationally easy and time efficient. Lastly, the performance metrics of different models on an evaluator should be statistically significantly different from each other to reliably rank the models on their goodness of fit.

To analyse commonly used evaluators on the third and fourth property, Wang et al. (2019) ran an extensive correlation analysis between extrinsic and intrinsic evaluation tasks to find computationally efficient tasks which could be used to proxy a model's performance on downstream NLP tasks. Unfortunately, the researchers could not find an ultimate intrinsic evaluator that showed a high correlation with all extrinsic evaluators. Instead, the researchers found that some intrinsic evaluation methods show higher correlation with other specific downstream extrinsic evaluation methods, which concludes that specific intrinsic evaluation methods are better proxies for other specific downstream NLP tasks. In contrast to (Wang et al., 2019), (Faruqui et al., 2016; Schnabel et al., 2015) concluded from their study that if the goal of a word embedding model is to achieve high performances on downstream tasks, one should test the embedding model on that specific task, since they did not find any significant correlation between intrinsic tasks and extrinsic tasks. Another correlation analysis between extrinsic and intrinsic evaluators for word embedding models, built on Chinese texts, (Qiu et al., 2018) shows high correlation values, suggesting that some intrinsic evaluators are good proxies to assess a model's extrinsic performance, affirming the results from (Wang et al., 2019). Again, in contrast to that stands the analysis from Chui et al. (2016) where 10 intrinsic and 3 extrinsic tasks were used in a correlation analysis to show that no correlation between intrinsic and extrinsic tasks existed, which affirms the conclusion from (Faruqui et al., 2016; Schnabel et al., 2015).

Looking at the first property of evaluators as listed in (Wang et al., 2019) which emphasises on the correctness of the underlying test datasets of the evaluators, Faruqui et al. (2016) found that for intrinsic evaluation methods, the underlying datasets could be dependent from the subjectivity of the creator and can even be distorted by misinterpretation. The researchers found for two popular intrinsic tasks that even though the dataset should only be comprised of words with a similarity relation like *car* and *train*, related words were also included in the dataset, like *cup* and *coffee*. It is important to distinguish between similar words and related words, so that the assessment of a model is not biased. The distortion would be that assessors, which were used to build the test dataset by grading two words on their similarity, graded the similarity between *cup* and *coffee* higher than for *car* and *train*, although we now know that *cup* and *coffee* should have a lower score, since they are not similar terms but related terms. In the case of the word analogy task, which is one of many intrinsic evaluation tasks, Wang et al. (2019) stated that there is a difference between human reasoning, or rather logic when it comes to identifying word analogies and the way we want a word embedding model to simulate analogical thinking, which also distorts correctness of the evaluator.

The fifth property of evaluation tasks as listed in (Wang et al., 2019) is that to compare word embedding models with each other, the differences in the performance metrics must be statistically tested. Many researchers use different evaluators to measure the goodness of a model, but most analyses fail to present significance tests to back up their results (Baroni et al., 2014), making their conclusions unreliable. Shalaby & Zadrozny (2015), first implemented the Steiger's Z significance test to measure the statistical significance of differences in performance measures in word similarity tasks. In word similarity tasks the human judgment and the model's judgment on the similarity between two words is compared. The more the model assesses the similarity as the human the better the model. In their analysis the researchers found that even if a model seemed to perform better than another model, if the difference in their performance metrics is not statistically significant the researchers would assess both models' goodness as equal. Another example of accounting for the statistical significance between models' performances is shown in (Rastogi et al., 2015) by introducing the minimum required difference for significance (MRDS) statistic, which also utilizes the Steiger's Z significance test. With the MRDS the researchers were able to compute confidence intervals to determine if a difference between performance metrics is statistically significant or not. Further, Antoniak & Mimno (2018) tested the stability of word embedding models utilizing distance-based word similarity measures. The researchers concluded that the performance of models on word similarity tasks are heavily dependent on the used corpus on which the word embedding model is trained on. Especially for small training corpora the researchers suggest using a bootstrapping algorithm to evaluate a model's performance to get reliable and consistent results. Further, Schnabel et al. (2015) used randomized permutation tests to measure the significance of differences between performance metrics for 2 intrinsic and 2 extrinsic tasks. To summarize the results from past scientific works in this field and to place my analysis in the ranks of others, Table 1 in the Appendix show a non-extensive literature overview.

The goal of my research is to give an overview about the reliability of evaluators for word embedding models utilizing statistical significance testing. In contrast to the above-mentioned and in Table 1 depicted applications of significance testing of performance metrics to validate their newly developed models, I will use these significance tests to measure the usefulness of certain evaluation tasks. The approach of my analysis is to have 3 word embedding models and many evaluators to measure the models' goodness. Finally, I will use significance tests to measure whether the difference in performance metrics of two models one the same evaluator is statistically significant or not. The main idea is to have two word embedding models which I expect to perform similarly well on the evaluation tasks and one embedding model which I expect to perform significantly worse. The minimum requirement for these evaluators would then be to able to distinguish between the good models and the significantly worse model. Better evaluators should also be able to identify significant differences in the performance metrics between the 2 similarly performing models, even if these are small. A good evaluator must meet this requirement, since with the increase in computational power, modern word embedding models will be trained on very large textual data resulting in much more sophisticated and precise word embeddings, so only small differences in performance will determine which model is better or not. Nowadays highly sophisticated and complex NLP models exist, like BERT or GPT-3, but since their complex architecture is based on modern transformer networks and their direct usage in NLP tasks differs from common word embedding models like word2vec I will focus on 3 simpler word embedding models, which are CBOW, GloVe and NMF. I expect that CBOW and GloVe will outperform NMF on the evaluation tasks, since NMF will be based on the term document matrix while GloVe and CBOW are based on the word co-occurrences. Following the expected underperformance of NMF on the evaluation tasks, I will use the NMF model as the lower baseline in my analysis. As shown in Figure 1, which illustratively depicts my conceptual framework, first all 3 models will be trained on the same text data to ensure comparability.

Conceptual Framework

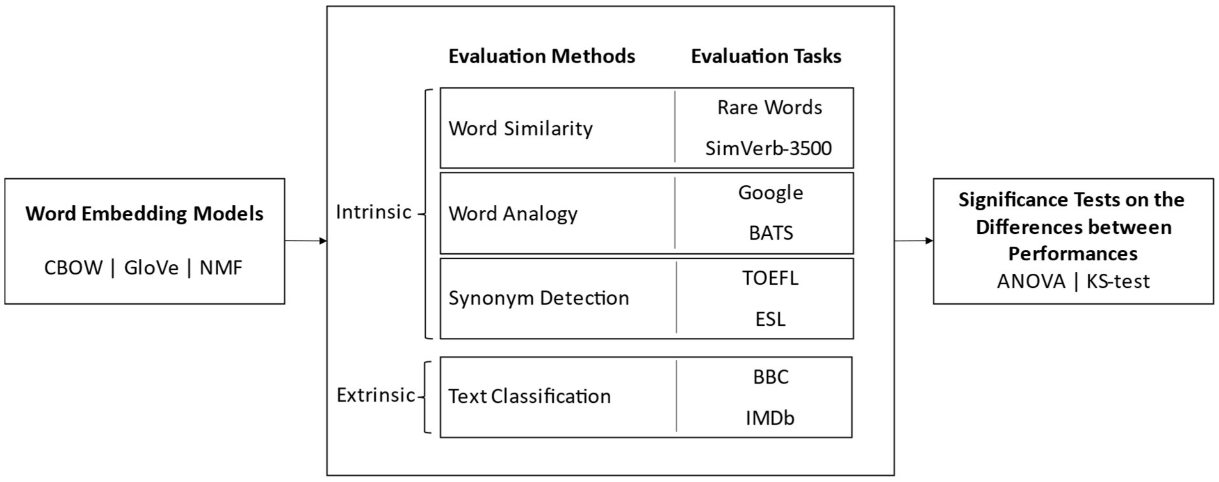

Figure 1: (From left to right) Each of the 3 Word embedding model is evaluated on all 8 evaluation tasks and then differences in performance metrics are statistically tested using either ANOVA or KS test.

Next each model will be evaluated, and performance metrics will be calculated using 4 different evaluation methods, which are spread out on 8 different evaluation tasks, from which 2 are intrinsic tasks and the remaining 6 are intrinsic tasks. An evaluation method is the concept on how to evaluate a word embedding, for example running a classification model to cluster texts into groups using the word embeddings as input features. The task itself is then the practical implementation of a method using a separate dataset. For the above-mentioned example this would be a dataset to train and test the classification model. Finally, to assess the reliability of the evaluation tasks, I will calculate the statistical significance of the differences between the models' performance metrics. This is done by running either one-way ANOVA tests or a two-sample Kolmogorov Smirnov tests, depending on the used evaluation method. Since the significance testing can only be done between 2 models' performance metrics I will run 3 significance tests for each evaluation task, so all 3 models' performance metrics are once compared with each other. The usefulness of the evaluation tasks will be based on the reliability of the results.

3 Methodology

Word embeddings models are mathematical models, which use different transformation approaches on text corpora to capture the syntactic and semantic structure of a text and save this knowledge in word embeddings. A word embedding or rather a word vector is a is a vector specific to a term, where the values of that vector correspond to the term's syntactic and semantic usage. This means that vectors of words which are similar in any sense, this could be that they originate from the same root word or that both words share similar semantics or that both words are from the same part of speech or etc., should have similar values in their vector, this could be presented through same signs or the same magnitude in the vectors. To generalize and avoid overfitting the corpus, word embedding models use vectors of far lower dimension compared to the size of the vocabulary of a corpus, for example in (Pennington et al., 2014) the researchers used vector dimensions in the range of 100 to 1000 for a vocabulary of 400,000 terms which is built from corpora with more than 1Bn words.

To determine a word embedding model's goodness of fit, one can use different evaluation methods. Evaluations are mainly executed using separate test datasets, called evaluation tasks. In this work, I will compare three word embedding models with each other using 4 different evaluation methods split up into 8 evaluation tasks. Further to measure the validity of the evaluation tasks themselves, I will run a significance analysis on the models' performances.

Following, I will first introduce the word embedding models used in this analysis and second, I will introduce the used evaluation methods.

3.1 Word Embedding Models

3.1.1 GloVe

Global vectors for word representation (GloVe) is a word embedding model which was introduced by Pennington et al. (2014) in their name giving paper. The model, as the researchers claim, should combine the advantages of global matrix factorization models and local context window models, by utilizing the corpus wide term co-occurrence matrix which is based on a given context window.

The term co-occurrence matrix will be denoted as X with X_{ij} representing the observation in the i-th row and j-th column (co-occurrence of term i with term j) of the term co-occurrence matrix. Co-occurrence is based on a given context window, either symmetrical or asymmetrical and can be weighted or not. Symmetric context windows count the co-occurrence of n words before and after a context word, while weighted context windows weight the co-occurrence value higher for words which are less words apart from each other e.g., setting the co-occurrence value for words which are next to each other to 1 and for words which are one word apart to 0.5.

The following explanation of the GloVe model is referenced from (Pennington et al., 2014). Let $P_{ij} = P(j|i) = X_{ij}/X_i$ be the probability that term j occurs in the context of word i , and X_i be the total count of co-occurrences of word i with all other terms. The idea was then to build a model which is based on the ratio of the probabilities which should result to a better understanding to distinguish between relevant and irrelevant words. An example the researchers give in their paper is that $\frac{P(solid|ice)}{P(solid|steam)} = 8.9$ while $\frac{P(water|ice)}{P(water|steam)} = 1.36$, this shows that for words which have a direct link to only one word, here solid to ice but not solid to steam, result to a high value, whereas for words which have a link to both context words result in values near 1. This means that words which result in a value near one show irrelevant word co-occurrences. This idea also holds for words which have no link to both context words. To now link these term probability ratios with word vectors the researchers established the following formula seen in Equation 1.

$$F(w_i, w_j, \widetilde{w}_k) = P_{ik}/P_{jk} \tag{1}$$

with w_i, w_j and $\widetilde{w}_k$ representing the word vectors for a word i, word j and context word k with $i, j, k = 1 \dots V$, where V is the number of unique terms in the vocabulary, and F representing a transformation function to transform the word vectors to a scalar. After further transformation the term can be also written as

$$w_i^T w_j + b_i + b_j = \log(X_{ij}) \tag{2}$$

connecting the vectors with the co-occurrence matrix, with b_i and b_j representing bias terms for the two co-occurring words. Finally, the researchers used a weighted least squares approach to optimize the vectors, which finally results in cost function Equation 3.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \left(w_i^T w_j + b_i + b_j - \log X_{ij} \right)^2 \tag{3}$$

Here $f(X_{ij})$ being a weighting function, which should be zero for word pairs which have no co-occurrence, this should cap the weight of very frequent word co-occurrences to not bias the model but still should give word pairs with lower frequency a smaller weight than word pairs with higher frequency. For that the researchers gave the weighting function the following function:

$$f(x) = \begin{cases} (x/x_{max})^{\alpha} & if\ x < x_{max} \\ 1 & otherwise \end{cases} \tag{4}$$

Finally, after the model is trained and the parameters are optimized, we obtain the word vectors of the vocabulary V by adding up w_j and w_i^T, where $i = j$.

3.1.2 CBOW

Continuous Bag of Words (CBOW) is one of two neural network-based word embedding models build by Google researchers in 2013, which were integrated in their word2vec toolkit. The corresponding paper to publish their research on the two word-embedding models CBOW and Skip-gram was the 2013 released (Mikolov et al., 2013a). CBOW is a sliding window model which tries to predict a focus word based on its surrounding context words. For example, in the sentence: “The office worker drives home from work every day.”, with *drives* as the focus word and a symmetric window of size 2, the context words would be [*office, worker, home, from*].

So, the model would try to predict the word *drives* just from the context words. Since the order of the context words is irrelevant this model is called a Bag of Words model, the prediction only depends on the words and not on their order. Sliding window means that this symmetric window would than shift to the next focus word, which would be *home*. Now the context words are [*worker, drives, from, work*]. A simplified version of the CBOW model architecture is shown in Figure 2 in the appendix. On the left side are context words, which are used as input variables for the neural network, in the middle is a single hidden layer and the output layer consists of the prediction of the focus word.

For the following technical explanation of the CBOW model I will refer to (Rong, 2014). First let's look at a little simplified CBOW model where we use an asymmetric sliding window by only looking at the context word prior to the focus word, as shown in Figure 3.

CBOW model architecture with only one input feature

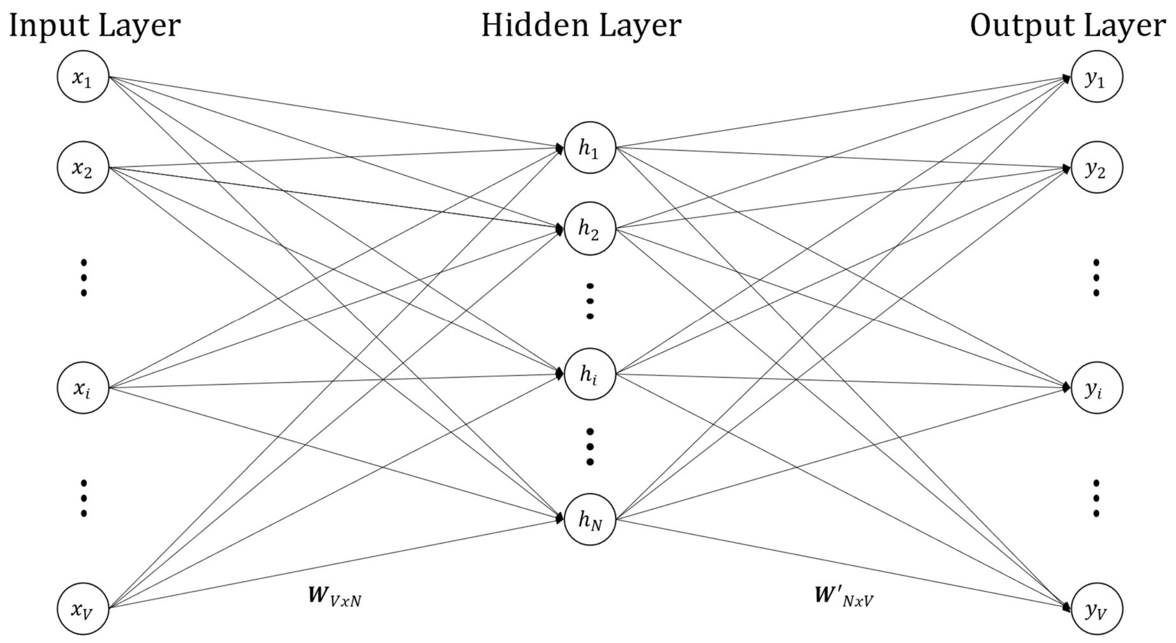

Figure 3: Input layer (left) consists only of a single one-hot encoded context word. The hidden layer (middle) consists of N nodes, which represent the dimensions of the model. Output layer (right) also consists only of a one-hot encoded prediction word. All adjacent layers are fully connected with each other, by utilizing weights which are stored in the two weight matrixes.

The model would then always look at a word of a sentence and try to predict the next upcoming word. Let's also say that the model is built on a corpus with a vocabulary of size V. In that case the input layer of the simplified model would only consist of that one context word, which is a one-hot encoded vector of size V, where all entries are zeros except for the entry of the corresponding term, where it is a one. The hidden layer would then also be only a vector of size N, which represents the number of dimensions used in the word embedding. The output layer is as the input layer a vector of size V. All adjacent layers are fully connected through two weight matrixes, with $\mathbf{W}$ being the weight matrix between input and hidden layer and $\mathbf{W}'$ being the weight matrix between hidden and output layer. The hidden layer is computed by multiplying the input vector with the corresponding weight matrix: $\mathbf{h} = \mathbf{W}^T\mathbf{x} := \mathbf{v}_{wI}^T$, and since we use one-hot encoded words as input variables we can also call $\mathbf{h}$ as $\mathbf{v}_{wI}^T$, which is nothing else than the row of the weight matrix $\mathbf{W}$, which corresponds to the one-hot encoded input word. Since $\mathbf{h}$ is a vector, the corresponding row $\mathbf{v}_{wI}$ needs to be transposed. For models with multiple context words C as input, the equation to compute $\mathbf{h}$ will change to $\mathbf{h} = \frac{1}{C}\mathbf{W}^T(\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_C) := \frac{1}{C}(\mathbf{v}_{w1} + \mathbf{v}_{w2} + \cdots + \mathbf{v}_{wC})^T$. In a situation with multiple context words the average of the rows of the weight matrix, which corresponds to the context words, is taken. Next from the hidden layer to the output layer we need to multiply the hidden layer vector $\mathbf{h}$ with the second weight matrix $\mathbf{W}'$, which will give us $\mathbf{u} = \mathbf{W}'^T\mathbf{h}$, where $\mathbf{u}$ is a V units long vector with each unit representing a word from the vocabulary. Finally, we use softmax function on $\mathbf{u}$ and thus calculate the probabilities of a word occurring in the context of the input context words. These probabilities are calculated as $p(w_i|w_I) = y_i = \frac{\exp(u_i)}{\sum_{j=1}^{V}\exp(u_j)}$, where $p(w_i|w_I)$ is the probability that word w_i appears in the context of word(s) w_I, and y_i being the i-th entry of the V units long output vector $\mathbf{y}$. Finally, a prediction of the focus word is made by picking the word with the highest probability, which is displayed in y_i. Training a CBOW model is carried out by maximizing the conditional probability of observing the correct output word $p(w_i = w_O|w_I)$, by tuning the weight matrixes $\mathbf{W}$ and $\mathbf{W}'$.

Since training such a model on a bigger dataset with billions of words and a vocabulary of many hundred thousand of terms will be computationally too expensive, the researchers introduce two approximation methods for the computations of the output layer, which is hierarchical softmax (hs) and the negative sampling (ns). Since I do not want to get too technical with the explanations of the word embedding models, I will not further explain the technicalities of hs and ns. Since models show better results for infrequent words using hs instead of ns (Mikolov et al., 2013b), I will also use hs in the implementation of CBWO in my analysis.

After training the model and optimizing the weight matrixes, we can use the input weight matrix **W** to obtain our word vectors, where each row of the matrix corresponds to a term of the vocabulary.

3.1.3 NMF

First, matrix factorization represents a method which uses large matrixes like term-document matrixes as input to find clusters in the data. For term-document matrixes as an example, this would mean to cluster documents which have a similarity with each other together and to cluster words which have a similarity with each other together. One could then use that information to identify similar documents or similar words. Matrix factorization achieves this by splitting the input data, here in my analysis the term-document matrix into multiple matrixes, which in that example represent a document matrix and a term matrix. One matrix factorization method is the non-negative matrix factorization (NMF) (Lee, Daniel D. & Seung, 1999; Paatero & Tapper, 1994). Other well-known matrix factorization techniques are PCA, SVD and even GloVe, since GloVe does nothing else than reducing a co-occurrence matrix into two word matrixes. A big difference for my analysis is that in contrast to the application of GloVe on the global term co-occurring matrix for which it is specifically built, NMF which can be applied on any matrix, will be implemented on the term document matrix.

NMF works by representing an input matrix V (term document matrix) of size $n \text{ x } m$ as a product of two matrixes $V = WH$ of the sizes of $n \text{ x } k$ and $k \text{ x } m$ respectively for W (word matrix) and H (document matrix), with the main constraints of NMF, that all values of the input matrix must be non-negative so $V_{ij} \geq 0$ and that the resulting basis matrix W and the coefficient matrix H must be non-negative, too, so that W_{ip}, $H_{qj} \geq 0$. One way of representing the optimization problem is by utilizing the Euclidean distance as an approach (Lee, Daniel & Seung, 2000) which results in equation (5).

$$\min_{W_{ip}, H_{qj} \geq 0} \|V - WH\|^2 = d(V, WH)^2 = \sum_{i=1}^{n} \sum_{j}^{m} (V_{ij} - (WH)_{ij})^2 \quad (5)$$

Since Lee & Seung (2000) show that such a global minimum cannot be found and equation (5) is not solvable, the researcher introduced two ways to approximate the minimization problem. The idea is to use numerical optimization so that $V \approx WH$ and approach a local minimum.

By setting k, the dimension of the resulting word vectors can be controlled. The word vectors can be obtained from the word matrix W, where each row of W is a word vector corresponding to a term of the vocabulary.

3.2 Evaluation Methods

In the taxonomy of evaluators for word embeddings, evaluators are first classified as intrinsic or extrinsic methods, second the task is defined and lastly the underlying dataset to the task is identified. First an evaluator is categorized as an intrinsic or extrinsic evaluator. Intrinsic evaluators use tasks to directly compare word vectors with each other, whereas extrinsic evaluators use word vectors as feature variables in downstream tasks like machine translation. A big difference is that extrinsic tasks are much bigger and complex and most of the time require extra model training/building when compared to intrinsic evaluators. Next the evaluator is categorized into their respective task. The task specifies how the word embeddings are used to determine the goodness of fit of a model. Finally, the evaluator's respective test dataset is specified. Over the time researchers developed many test datasets to run specific evaluation tasks. These datasets mainly consist of many prompts which need to be answered utilizing word embeddings. Figure 4 illustrates the common taxonomy of evaluation methods by the example of two test datasets which I used in my analysis.

In (Bakarov, 2018) the author further extends the taxonomy of evaluation methods, with an additional split of the intrinsic evaluator. In my analysis I will only focus on simple intrinsic and extrinsic evaluators since these are the most common evaluation methods. Moreover, I will only focus on 1 extrinsic and 3 intrinsic tasks, since a full extensive overview of the evaluation method landscape would go beyond the scope of a master thesis analysis. Following, I will introduce the 4 used evaluation tasks used in my analysis. The underlying datasets to the evaluation tasks are presented in Section 4 Data.

Taxonomy of evaluation methods

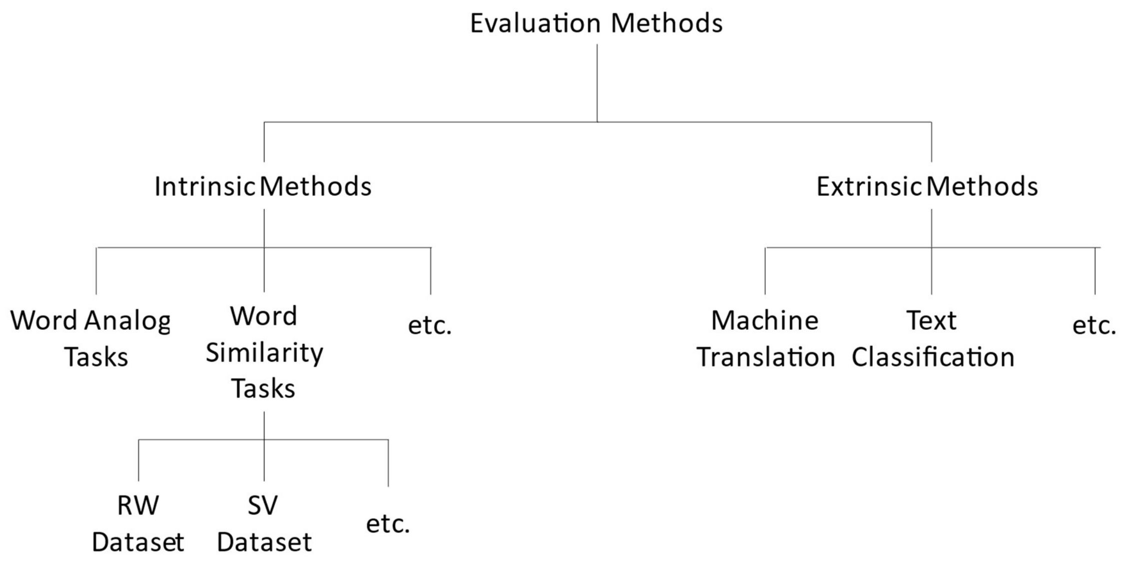

Figure 4: Taxonomy of evaluators shown exemplary on the RW and SV word similarity tasks.

3.2.1 Intrinsic Evaluators

Intrinsic evaluators are methods where the embedding's judgment on word relations is compared to human judgment (Bakarov, 2018). This way one directly evaluates the word embeddings. Words can only be related to each other by being semantically or syntactically similar. One can utilize these relations to measure the goodness of an embedding by using different evaluation tasks. An example which will also be fully explained in the following would be the word similarity task, where using a distance measure a similarity score between two words can be computed using their respective word vectors. This similarity score is then compared to a human judgment of the similarity between the two words. Next, I will introduce the 3 intrinsic evaluation tasks which I used in my analysis.

WORD SIMILARITY

Vijaymeena and Kavitha (2016) say that one can represent the similarity of two words either as character-based or term-based. For each method multiple metrics are available to calculate their similarity. The main idea is always to quantify the difference between two words, or rather determine the distance between both. This distance could be calculated by counting the number of operations needed to transform one string into another string, which would represent the Levenshtein distance as one example for character-based similarity or to calculate the direct distance between two words' vectors in a n-dimensional space, which would represent the Euclidean-distance as an example for term-based similarity (Vijaymeena & Kavitha, 2016). After a similarity metric between two words is calculated, the metric is then either compared to a human judgment or the metric itself is rated by humans (Schnabel et al., 2015). The human judgment often consists of multiple ratings where people were asked to assess the similarity between two words.

The mostly used method to calculate the difference between two words is by computing the cosine similarity of them. The cosine similarity is defined as the cosine of the angle between two word vectors. Since it is the most popular method, used in (Baroni et al., 2014; Pennington et al., 2014; Schnabel et al., 2015; Wang et al., 2019), I will also utilize it also for my analysis. Given that, the range to which the value of the cosine similarity can fall is [1, -1] with values near 1 representing vectors pointing into similar direction, which can be translated to that both words are very similar to each other, or 0 representing vectors which are orthogonal to each other, which can be translated to that both words are not associated with each other, or -1 representing vectors pointing into opposite directions, which can be translated that the two words should have the exact opposite meanings. The formulation of the cosine similarity can be seen in Equation (6),

$$cosine(\vec{A}, \vec{B}) = \cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \times \|\vec{B}\|} \tag{6}$$

with $\vec{A}$ and $\vec{B}$ being the words vectors for word A and B, $\|\vec{A}\|$ and $\|\vec{B}\|$ being the Euclidean norm of the words vectors and θ being the angular difference between the two vectors. One can interpret the cosine similarity as an analogous metric to the Pearson's correlation coefficient with positive values indicating words which have a similar meaning, are related to each other, or occur often together. Vice versa a negative value indicates the opposite. To measure the goodness of a word embedding standard scientific practice is to use a set of word pairs to compute the cosine similarity between them and to get a human judgment on the similarity between these word pairs. Finally, the correlation between the cosine similarity and the human judgment is calculated. For that most scientists use the Spearman correlation coefficient. If the correlation value is high the embedding captured the semantics and syntax nearly as good as humans would do.

The popularity to use the cosine similarity as a similarity measure for word vectors, stems from the very intuitive nature of the measure, which makes it very simple to understand. Further, the word similarity task's many accessible datasets (Bakarov, 2018) and their easy and computational inexpensive implementations, results in that also amateurs can use it as an evaluation method in their work. But with its very simple nature, many weak spots were detected, and its reliability was heavily criticized, summarized in (Bakarov, 2018; Faruqui et al., 2016). One very often mentioned problem is the frequency effect or rather hubness problem (Faruqui et al., 2016). This problem is also depicted in (Schnabel et al., 2015), where words with similar frequency in the training data show a higher value in the cosine similarity and that these higher values can be traced back to the fact that both words show similar frequencies, which distorts the syntactic and semantic similarity values.

WORD ANALOGY

Word analogy tasks mostly consist of queries like, word a is to word b as word c is to word d e.g., *king* is to *queen* as *man* is to *woman*. In this case the analogy is made by looking at the relatedness that connects the words with each other, here it would be the sex of a person. Analogies can also be based on syntactical relations of words like, *run* is to *running* as *walk* is to *walking*. Here the relation is that word a and c is a verb in infinitive form and word b and d are words in the present participle form. The main idea is that we can now use mathematical operations on words so as shown in Equation 7. Using the example from above this would look like: *king – man + woman = queen*

$$a - c + d = \hat{b} \cong b \tag{7}$$

We can do these calculations with words since we use word embeddings to represent words as a collection of values in a n-dimensional vector space.

In the scientific field there are many ways to compute this relation but most prominent one and widely used method will be the 3CosAdd approach (Mikolov et al., 2013c), which can be explained as a direct implementation of the mathematical operation from above seen in Equation 7. Again, using the example from above, first the word vector for *king* is subtracted by the word vector *man* and added to the word vector *woman*. The resulting word vector should be then the word vector for *queen*, calling it $\hat{b}$. Since the used word vectors cannot be perfect and the calculated vector will not have the same values as the *queen* word vector, but should be very similar to it, the idea is then to use the cosine similarity between the calculated vector $\hat{b}$

and the word vectors of the vocabulary. If a word embedding is good, the word vector from the vocabulary with the highest similarity metric should be the vector for the word $queen$. Another approach to compute this analogical relation is by using the 3CosMul approach (Levy & Goldberg, 2014). Instead of taking an additive approach they took a multiplicative approach for this method. Although this approach is the second most used method to implement the analogy task, Wang et al. (2019) showed that, there is no significant difference between 3CosMul and 3CosAdd when used in practice.

Word analogy task datasets mostly consist of prompts containing 2 word pairs which have any analogous relation with each other, so word a is to word b as word c is to word d. Standard practice is then to use 3CosAdd as shown in Equation 7 to compute $\hat{b}$ and cosine similarity to determine the most plausible word for vector $\hat{b}$. The most common evaluation metric would be the accuracy of how often $\hat{b}$ equals b. Since my models are not very sophisticated and all would score very low accuracy values, I do not want to have an evaluation metric which only counts hits but also accounts for how close a hit might be. For my analysis I will use a different evaluation metric which deviates from the standard practice in the scientific field. The approach will be fully explained in section 5 Analysis.

SYNONYM DETECTION

The synonym detection task as the name suggests is a task where the word embedding is given a query word and the model needs to detect the synonym for the query word out of a range of set words. Often this range of words contains 4 words with one being the correct synonym and 3 others. For example, word a is the query word and $[b_1, b_2, b_3, b_4]$ are the given possible synonyms to choose from. The idea is to calculate the difference between word a and the range of words and choose the word with the smallest distance. An example could be that the query word is $huge$ and the given words to choose from are $[fast, small, dirty, big]$, with big being the correct answer. Since a distance between word vectors must be calculated to choose the synonym, a distance measure must be selected. In most cases the usage of the cosine similarity as a distance measure is used. Finally, most studies use the accuracy of the word embedding model on the task as the performance metric to determine the model's goodness of fit (Baroni et al., 2014; Rastogi et al., 2015). In my analysis, I will use a different evaluation metric in my analysis for the synonym detection tasks with the same argumentation as for the word analogy task. I will explain my approach to measure the performance in section 5 Analysis.

3.2.2 Extrinsic Evaluators

In contrast to intrinsic evaluators where the embeddings are directly evaluated and compared with each other, in extrinsic evaluators the word embeddings are implemented as feature variables in downstream tasks. These downstream tasks are mostly common NLP tasks like part-of-speech tagging (Wang et al., 2019) where the embedding is implemented in a classification model to identify the part of speech of a word, neural machine translation (Wang et al., 2019) where the embeddings are used for a language translation model, and etc. The idea of an extrinsic evaluator is to test a model's embedding on tasks and problems which represent possible use cases of word embedding models. The goodness of the word embedding is then measured by looking at common performance metrics of the downstream model.

For my analysis I will run two **text classifications**, with the first being a multiclass topic classification and the second a dichotomous sentiment classification as shown in (Schnabel et al., 2015). The classification model for both tasks will be a Random Forest and performance will be measured using Matthews Correlation Coefficient. The full implementation of the extrinsic evaluators will be explained in section 5 Analysis.

3.2.3 Significance Tests

The heart of my analysis is to measure the significance of the difference between two word embedding models' evaluation metrics and to infer from that the usability of the tested evaluation tasks. The significance test will be used as a proxy to determine the reliability/usability of the evaluation tasks. As shown in Table 1 different significance tests were already used for different evaluation tasks. For my analysis I wanted to use a different approach. I chose one way ANOVA (ANOVA) and two sample Kolmogorov Smirnov test (KS test) to measure the significance of the difference between evaluation metrics. One way ANOVA was only used for the word similarity tasks. For all other tasks I used the KS test as significance measure. The KS test tests whether two samples come from the same distribution whereas ANOVA tests whether the mean of two samples is significantly different from each other. The actual implementation of the significance tests in my analysis is explained in Section 5 Analysis.

4 Data

The data used for my analysis is divided into a training and testing part. The training data is used to train the models to obtain the word embeddings, while the testing data is used to evaluate these word embeddings. The training data consists of a combination of different, freely accessible and cleaned corpora. The testing data is split into an intrinsic and extrinsic evaluator part. The intrinsic part consists of two datasets for each intrinsic evaluation method, of which there are three, resulting in 6 intrinsic testing datasets in total. The extrinsic part also consists of two datasets for the extrinsic evaluation method, resulting in 8 different test datasets in total for the testing part.

4.1 Training Data

To obtain the desired word vectors, I trained the three word embedding models on a self-combined corpus, containing the full BAWE (British Academic Written English) corpus, the full Brown corpus, "Matt Mahoney's first billion characters" corpus, parts of the OANC (Open American National Corpus) and a small corpus of several Wikipedia articles. A detailed list of the used corpora is shown in the Appendix Table 2.

4.1.1 BAWE

Through a collaborative project between the universities of Warwick, Reading and Oxford Brookers, called "An investigation of genres of assessed writing in British Higher Education" taken place between 2004 and 2007, the BAWE corpus (Nesi, 2008) was created. It consists of 2761 pieces of student writing resulting in a total of 6,506,995 words. The writings cover 4 different genres, namely Arts and Humanities, Social Science, Life Science and Physical Science, and the study level of the students can be divided into four groups, from first year up to master level students.

4.1.2 Brown

The Brown University Standard Corpus of Present-Day American English (Kucera et al., 1967) , also known as Brown Corpus, was first created in 1961 by the department of linguistics at Brown university. The corpus was made for comparative studies of written English, thereby consisting of a total of 500 samples of text with each sample being over 2000 words big. The samples can be assigned to either one of 2 main groups with the first consisting of 9 genres and the second 6 genres, ranging from financial press reports to science fiction short stories. The creators of the corpus only wanted written discourse for the corpus, hence genres like drama were left out from the corpus since most drama publications consist of dialogue or rather spoken discourse. Overall, the corpus consists of 1,014,312 words.

4.1.3 OANC

The OANC (Ide, 2001) is a subset of the American National Corpus, which is a in 2005 provided project by the Linguistic Data Consortium, which is an open consortium of universities, research laboratories and companies from the United States. The ANC is meant to be a counterpart to the British National Corpus which totals around 100 million words, compared to the 22 million words from the ANC. Since the OANC is only a subset of the ANC it only contains 14,623,927 words, collected from spoken recordings and written discourse, ranging from police reports and technical journals up to recorded telephone calls.

4.1.4 Matt Mahoney’s first 100 million bytes

In the official web publication of the word2vec project, the Google researchers reference their used training data to be from several sources, for some of which they used the Wikipedia dump pre-processing code from Matt Mahoney, a retired senior scientist at Dell, specializing at data compression. The Google researchers also reference an already pre-processed Wikipedia dataset from Matt Mahoney’s Large Text Compression Benchmark (Mahoney, 2011) for a simpler implementation of the data. This dataset consists of, the alphabetically ordered, first Wikipedia articles truncated to 100MB, resulting in a corpus of 17,005,207 words.

4.1.5 Wikipedia

The final dataset used to build my word embedding models is an accumulation of Wikipedia articles from handpicked genres/domains. For the articles collection, I used a free to use Wikipedia web crawler, published by data scientist Michael Hainke, on Github (Hainke, 2019). The crawler is initialized, with a starting article, to find related articles to the initial article, and then to find further related articles to the related articles and so on. Every time a new starting article is used, a new dataset is created. To cover different genres, I used 16 distinct initial articles, ranging in topic from wildlife, electronics and up to clothing. A detailed list for each dataset is shown in Table 2 in the appendix. To limit the size of each resulting dataset the crawler had only a runtime of 30 seconds after initializing it, resulting in datasets of ranging size from around 500 thousand up to 2 million words, each. Taking all Wikipedia crawler datasets together results in a corpus of 27,530,351 words in total.

4.1.6 Final Training Dataset

Joining all 5 datasets and cleaning it leaves a corpus with a total of 55,493,383 words. Since the corpus cannot be cleaned perfectly and many rare non-defined terms, which can be described as noise, are still in the corpus. To account for this noise, all terms which do not occur more than 10 times will be dropped from the corpus. Doing this, most phrases, and terms, which are meaningless, sparse and pollute the data are removed, which results in a corpus of 54,620,463 words and 87,390 unique terms spread over 5,583 documents. I decided on a minimum word count threshold of 10, as a trade-off between keeping rare words and removing non-defined terms as in (Wang et al., 2019). The cleaning process keeps plurals and declinations, but removes symbols, numbers, punctuation, and abbreviations. This way, the word embedding models should learn to distinguish between singular and plural, adjectives and adverbs, adjectives, and nominalization, etc. A list of all training corpora with the most important statistics is presented in Table 3.

Complete training dataset

Name	Number of Genres	Number of Documents	Word Count (Raw)	Word Count (Cleaned) [a]
BAWE	4	2761	6,506,995	6,359,203
Brown	15	500	1,014,312	886,839
OANC	12	2305	14,623,927	2,605,863
Matt Mahoney's 1st 100Mill. Bytes	-	-	-	17,005,207
Wikipedia	16	16	28,452,800	27,763,451
Total	47	5583	67,603,241	54,620,463

Table 3: Here the most important statistics of the training corpora are listed. Since Matt Mahoney's 1st 100Mill. Bytes is a fully cleaned datasets build by many Wikipedia articles, and the author did not list the number of genres and documents covered by his dataset, their cells were kept empty.

[a] After cleaning and removing all words occurring less than 10 times.

4.2 Test Data

As stated in the methodology section, the way of testing a word embedding model's goodness of fit can be split into two categories, which are either intrinsic, where word vectors are evaluated directly, or extrinsic, where we utilize the word vectors as feature variables for downstream tasks. In my analysis I will only focus on 6 intrinsic and 2 extrinsic evaluation tasks. I must mention that there are many more intrinsic and extrinsic evaluation methods and tasks, which could be used to conduct a much more extensive analysis.

4.2.1 Intrinsic Evaluation Datasets

WORD SIMILARITY

For the word similarity task, the first dataset I used was the **Stanford Rare Word Similarity (RW)** dataset, which was first introduced and developed for the analysis in (Luong et al., 2013). The dataset consists of 2034 rows, representing word similarity question prompts. Each prompt consists of a word pair, 10 individual ratings of assessors who were asked to give a score from a scale of [0, 10] on how similar the two words are to each other, with 0 representing the lowest similarity and 10 the highest and an average score out of the ten.

Since the dataset should contain only rare words, word pairs were constructed by first searching for words which have a low frequency of occurrence in Wikipedia texts (Luong et al., 2013) and selecting a random synset of that word based on WordNet (Miller, 1995). From this synset a range of related words is set, and one word is randomly chosen as the second word.

The dataset contains 2951 unique terms, with 50% of all terms occurring once, the mean occurrence of a term being 1.3 and the maximum occurrence of a term being 41, while the minimum is 1. All words in RW datasets are either nouns, verbs, adjectives, or adverbs, which could be conjugated or declined.

The second dataset used for the word similarity task is the **SimVerb-3500 (SV)**, which as the name suggests only contains 3500 verb pairs. The dataset was first introduced in (Gerz et al., 2016), as a new and superior addition to the existing word similarity datasets. As in RW each row corresponds to a question prompt, consisting of a word pair, individual ratings and an averaged rating score. In SV each word pair was assessed by at least 10 assessors, where the assessors were asked to give a similarity score from the scale of [0, 6], with 0 meaning that an assessor does not sees a similarity between the two words and 6 meaning that the assessor sees a high similarity between the two words. For the dataset the researchers mapped all scores linearly to a scale of [0, 10] to make the dataset more comparable to other datasets. Finally, an averaged score over the individual ratings is also presented.

In total the dataset consists of 827 unique verbs, with 50% occurring at least 7 times, and the minimum occurrence of a word being 1 and the maximum being 56 and a mean occurrence of 8.4. All verbs are in infinitive form.

WORD ANALOGY

For the word analogy task, I used 2 datasets. The first one is the **Google** word analogy test set, which was first introduced in (Mikolov et al., 2013a). As many other analogy test sets, each row of the dataset corresponds to a word analogy question prompt, consisting of 4 words, where an analogous relationship between two word pairs is given. In total the dataset contains 19,544 prompts, whereby each prompt belongs to one of 14 unique genres. The genres can be divided into 2 categories, where either a semantic relation or a morphological relation is presented, with 5 genres being semantic and 9 being morphological. 8,869 out of the 19,544 total prompts, so 45,38% show a semantic relation and the remaining 54,62% of all prompts have a morphological relationship. All 14 genres together with examples are presented in Table 4 in the appendix. It can be seen from that table that the morphological prompts are well balanced among the genres and their relative share to all prompts. In contrast to that, we see that the genre Capital-world has a very high relative share to the total prompts and therefore resulting in a 51% relative share to all semantic relational prompts, showing a big imbalance.

The next test set is the **Bigger Analogy Test Set (BATS)**, which was first introduced in (Gladkova et al., 2016), as a bigger and better structured analogy test set compared to the common ones to that time. The whole dataset is split into 4 main categories, which represent distinct word relations and each main category is then also split into different subcategories. Each main category consists of 10 datafiles, which then contain 50 word pairs, whereby each word pair shows a word relation. An overview of the dataset is shown in Table 5 in the appendix. The first main category contains inflectional word relations, split up for nouns, verbs, and adjectives. Inflectional relations of words are for example declination of nouns and adjectives. The second main category is comprised of derivational word relations, divided into stem word changes and no stem word changes. One example for no stem change would be the relation between *life* and *lifeless*. The next main category contains lexicographic word relations. These relations can be hypernyms, hyponyms, meronyms, synonyms, and antonyms. The final main category contains encyclopediatic word relations. These relations could be the colour of things like *water* to *blue* or sounds of things like *dog* to *bark*. In contrast to inflectional and derivational relations which are morphological word relations, lexicographic and encyclopediatic relations, which are semantic world relations are given multiple correct words for the analogy task (e.g., *cat* to *meow/ meu/ purr/ caterwaul*). Since the dataset only provides word pairs with their relations, one has built a usable word analogy dataset with question prompts by themselves. Building combinations of two word pairs without repetition and by taking the order into account, the maximum count of unique word analogy question prompts, which can be built is 98,000, resulting in one of the biggest word analogy test datasets.

For my analysis, I only used word relations which have unique answers, resulting in utilizing only the semantic word relations and the capitals relations file from the encyclopediatic category, containing word relations like *Athens* to *Greece*. Finally, I built a dataset in the size of 25,725 analogy question prompts consisting only of word pair combinations without repetition and order. An example would be that the test data contains an analogy question like *arm* is to *armless* like *art* is to *artless*. Since the order is not taken into account, the other way around (*art* is to *artless* like *arm* is to *armless*) is not in the dataset. In total, my built dataset contains 790 unique word pairs and is mainly comprised of morphological analogy questions.

SYNONYM DETECTION

Lastly, as stated I also used two common test datasets for the synonym detection task. Both datasets are built the same way, where each row represents a synonym question prompt. Each prompt gives a query word and 4 possible answer words, from which only one is the correct synonym to the query word.

The first dataset is the **English as a Second Language (ESL)** test set, which originates from the Basic 2000 words – Synonym Match 1 internet quiz made by Donna Tatsuki (Tatsuki, 1998). The quiz is part of many other activities presented by The Internet TESL Journal, which is an international organization of teachers to help people learn English. The dataset was first used as a test dataset for a synonym detection task in (Turney, 2001). In total the dataset consists of 50 questions, covering 206 unique words which are nouns, verbs, adjectives, and adverbs, whereby the maximum occurrence of a term is 5. All words are not declined or conjugated.

The second dataset is the **Test of English as a Foreign Language (TOEFL)** synonym test set, which was first publicized from Test of English as a Foreign Language by the Educational Testing Service. In (Landauer & Dumais, 1997) the dataset was first used for a synonym detection task to measure the goodness of their newly developed word embedding model. Following their work, many scientists used the dataset to evaluate new word embedding models. The dataset itself consists of 80 synonym question prompts, covering 391 unique words, ranging from nouns to verbs, adjectives, and adverbs. Nouns are in singular or plural form, while verbs are in infinitive or past participle form. Adjectives and adverbs are not declined. The maximum occurrence of a term is 2, which means that almost all terms occur only once in the dataset.

Finally, since the models are only trained on a specific set of corpora, which only covers a specific number of terms (the vocabulary), it may happen in the analysis that certain prompts in some tasks cannot be processed by the models because query words do not occur in the models' vocabulary. An example would be, if for a word similarity task one prompt would be to calculate the cosine similarity between the word vectors of the words *squishing* and *squeezing*. If a model does not have the word *squishing* and/or squeezing in their vocabulary, because it maybe never occurred in any of the training corpora or it was excluded due to its lack of occurrence, the model has then no word vector for that specific word and cannot run the calculation. When I encountered this problem in any tasks, I skipped the prompt, resulting in not fully utilized testing datasets. Below Table 6 presents for each task how much of the test datasets were used due to this problem.

Composition of the used test datasets

Datasets	**… out**	**of …**	**% of missing prompts**
Word Similarity			
RW	1209	2034	40.5%
SV	3375	3500	3.5%
Word Analogy			
GOOGLE	17,569	19,544	10.1%
BATS	19,917	25,725	22.6%
Synonym Detection			
ESL	48	50	4%
TOEFL	76	80	5%

Table 6: The table shows how many prompts of each test dataset were used for the analysis out of the actual number of prompts given by each dataset. On the left all 6 intrinsic evaluation tasks are listed. From the middle of the table on the left side the number of used prompts is presented while on the right side the potential number of prompts is listed. On the right side of the table the percentage of missing observations to the original count of observations is depicted.

Of course, this phenomenon had a bigger impact in my analysis since I could not use much larger and more varied training corpora, due to my computational limitation to train the models on big datasets and the monetary limitations since many sophisticated and professional training corpora are behind very expensive paywalls.

4.2.2 Extrinsic Evaluation Datasets

TEXT CLASSIFICATION

The **BBC text classification (BBC)** test set is a collection of BBC news articles from the BBC news website. All rights, including copyright, in the content of the original articles are owned by the BBC. The dataset was first introduced in (2006) by Derek Greene & Pádraig Cunningham as part of an empirical evaluation of a model. The dataset consists of 2225 individual news articles from 2004 and 2005, grouped into 5 distinct genres. Each article is comprised of a title and the main text. A descriptive summary of the BBC dataset is presented in Table 7.

BBC dataset composition

Genre	Document count	Avg. word count	Min to max word count
Business	510	337	142 to 915
Entertainment	386	340	143 to 3564
Politics	417	461	91 to 4486
Sport	511	337	117 to 1721
Tech	401	513	163 to 2994

Table 7: Here important statistics for each genre of the BBC dataset are listed. The last column shows the smallest and largest word count of all articles from a specific genre.

The mean word count of an article is 393, while the median value stands at 340 words, showing that all genres are well represented by the articles. To train the classification model, which will be used to run the extrinsic tasks, I will use the first 350 articles from each genre as training data while the remaining articles are used as testing data, resulting in a 80/20 training to testing split with 160 business, 36 entertainment, 67 politics, 161 sport and 51 tech articles as test articles.

The last testing dataset is the **IMDb** film review dataset (Maas et al., 2011), which was used for sentiment analysis by the Stanford researchers and will also be used in my analysis for sentiment classification. The original dataset consists of 50,000 individual movie reviews and their corresponding ratings, which were collected from the internet movie database (IMDb). Original movie ratings, which were given as star ratings in a scale of 1 to 10 stars were mapped linearly by the researchers to a binary scale [0 ,1] with 0 meaning that the review has a negative sentiment and 1 that the revies has a positive sentiment. The data set is perfectly balanced with 25,000 positive and 25,000 negative movie ratings. The used dataset itself was downloaded from Kaggle, which is a data science and machine learning online community platform.

For my analysis I only used 10% of the original dataset, resulting in a balanced dataset of 5,000 movie reviews and their ratings. I only used a fraction of the original dataset since the computational effort of preparing the data for the analysis was not feasible. Each review only consists of a plain text, with the shortest review only being comprised of 14 words, and the longest review text being over 1500 words long. For training and testing the cropped dataset was divided into an 80/20 split, where a sample of 4000 randomly chosen, reviews were used to train the classification model and the remaining 1000 reviews were used as testing set to measure the performance of the model.

5 Analysis & Results

5.1 Analysis

The whole analysis in this thesis can be broken down into 4 steps. First, the chosen word embedding models are trained and the word vectors are extracted. Second, these extracted embeddings are then tested on different kinds of evaluation tasks. Third the significance of the difference between the evaluation metrics is calculated. Lastly, these significance values are used to infer the usability and reliability of the evaluation tasks. Following, the first 3 steps will be presented and explained.

5.1.1 Model Training

The aim of the analysis is to assess the reliability of different evaluators, by looking at the performance of word embedding models, which are applied on these evaluators. I chose GloVe and CBOW as word embedding models since these two are the most prominent word embedding methods in the NLP community, excluding pre-trained models based on a transformer architecture like BERT. GloVe was utilized in many analyses, from papers presenting the most prominent evaluation methods (Schnabel et al., 2015), to analyses studying the relationship between intrinsic and extrinsic evaluators (Wang et al., 2019).

Conforming with my conceptual framework I wanted to use word2vec as the second better performing model. From the word2vec toolkit I chose CBOW over Skip-gram since CBOW is computationally more time efficient (Mikolov et al., 2013a). As the third and final model I selected NMF, which was also implemented in (Baroni et al., 2014). Since the basis of the analysis is to have 2 competing models which are similar in performance and one model which should be significantly worse, I chose to implement NMF on a term document matrix, whereas GloVe was built on a term co-occurrence matrix and CBOW was built directly on the corpus. All three models were then trained on the total and cleaned training data as shown in the last cell from Table 3. Since I had to rely on freely accessible data to train the models, I used a combination of different corpora. The Brown and BAWE corpora were utilized in their full extend. From the OANC I had to remove some parts since it also consists of police reports and telephone recordings, resulting in spoken discourse containing colloquial language and abbreviated sentences, which is not optimal for training a word embedding model, which will be tested on written discourse. To increase the word count, I also included the first 100 megabyte of cleaned Wikipedia articles, from Matt Mahoney's website, which was also presented in the word2vec toolkit website from Google. Finally, to broaden the spectrum of different topics in my combined corpus, I also implemented Wikipedia articles from handpicked topics, using a web crawler.

Since in (Mikolov et al., 2013a) the researchers used 50 dimensions for their word vectors and Schnabel et al. (2015) also used 50 dimensions on a 100K term vocabulary, I will also use 50 dimensions for my models, since my vocabulary amounts to around 87K unique terms. Higher dimensions would only make sense for greater training datasets, with higher word count and much bigger vocabulary. A too high dimension for my model could result in less generalization and maybe overfitting. Next, GloVe and CBOW were built on a 5-word symmetric window, meaning that for each focus word, 5 prior and 5 subsequent words will be taken into account when training the word vectors. A too low window size would result in word vectors mainly capturing words' syntax in sentences, while a too high window size would result in word vectors being biased in capturing mostly word semantics. I took a window size of 5 since it was recommended by Mikolov et al. (2013b). Since NMF bases on the term-document matrix the window size depends on each document, meaning that the window size is variable. This will ultimately result in more generalized word vectors capturing general semantics. All other hyperparameters of all word embedding models are set to default, as given in their respective packages[1] in R. Default values are based on the recommendations of the models' creators, which are mostly from the origin papers of the models. As shown in the section 3Methodology, the word vectors used for the analysis are the word matrix for the NMF model, the input weight matrix for the CBOW model and the summation of the two weight matrixes for the GloVe model.

5.1.2 Model Evaluation & Significance Testing

For my analysis, I mainly chose popular and straightforward evaluation methods to keep the analysis simple. Since the most prominent evaluators are mainly intrinsic tasks, and I wanted to broaden the scope of my analysis I will also look at extrinsic tasks. Following, I will explain step-by-step how all evaluation tasks are executed and how the significance tests are carried out.

For the intrinsic word similarity evaluation task, I chose the RW and the SV datasets. First for each word pair in the datasets the cosine similarity is calculated using the word vectors from each embedding model. Next, for each embedding model the Pearson correlation coefficient between the cosine similarity values and the average human judgments is calculated to measure the goodness of the models' embeddings. Finally, to calculate the significance of differences between the models' evaluation scores I implemented a one-way ANOVA test. For each dataset (RW & SV) three ANOVA tests were made to determine if, for example the cosine similarity values calculated by the GloVe model are significantly different from the CBOW model. In this case for example, the input values would be the calculated cosine similarity values using the GloVe word vectors and the calculated cosine similarity values using the CBOW word vectors. The significance testing is executed using a 95% confidence interval.

The word analogy task will be executed on the Google and the BATS dataset as shown in Table 6. Further, only the 3CosAdd approach will be considered since it is the most intuitive and the most widely used approach. For each analogy question in the dataset the cosine similarity between $\hat{b}$, form Equation 7 and each word from the vocabulary will be calculated. Next, all words from the vocabulary will be then ordered by their cosine similarity value from highest to lowest and the rank of the correct word b will be written down. To then measure the goodness of an embedding model I will look at the distribution of the ranks of word b and calculate a weighted score out of it. Since, the rank could potentially range from 1 to 54,620,463, because $\hat{b}$ is compared with all words from the vocabulary, I will only look at the distribution of the ranks from 1 to 10. A score is calculated by first weighting counts of a rank, and then adding these up. The higher the value of that score the better the model. The weighting is linearly in 0.1 increments, so the first rank is weighted with 1 and the second rank is weighted with 0.9 until the 10th rank is weighted with 0.1. To illustrate the evaluation process, I will demonstrate it on the GloVe word embedding evaluation on the BATS dataset. The distribution of ranks from 1 to 10 looks like this [1877, 2239, 1010, 614, 424, 335, 242, 197, 165, 157]. This would mean that 1877 times the nearest word to the word vector $\hat{b}$ is the searched word b, this is the hit rate. 2239 times the searched word b is the second nearest word to the calculated word vector $\hat{b}$, and so on till the 10th rank. Next the 10 numbers are weighted. The first number is taken as it is and weighted with a 1 resulting in 1877. The 2239 is multiplied with 0.9 and the 1010 is multiplied with 0.8 and so on until the 157 is multiplied with 0.1. After that the weighted values are summed up and divided by the total count of observations in the dataset. This way we computed a weighted accuracy score which not only counts the hit rate, where the nearest word vector to $\hat{b}$ is b but also instances where b was close to $\hat{b}$. Since all 3 models were not sophisticated and resulted in low accuracies for the word analogy tasks, I chose this way of computing the evaluation metric so a bigger picture of the embeddings performance can be seen. For the above example of the GloVe evaluation the evaluation metric on the BATS dataset would be 0.289. Finally, I will run a Two Sample Kolmogorov-Smirnov test (KS-test) to determine if the distributions of ranks from model 1 to model 2 are statistically significantly different from each other or not. The KS-test compares two distributions and tests if both come from the same population. The used distributions are the distributions of the ranks from 1 to 10 for each embedding model on both datasets. This means that for each dataset (Google & BATS) I will run three different KS tests, namely GloVe to CBOW, GloVe to NMF and CBOW to NMF.

As the final intrinsic evaluation method, I chose the synonym detection task, since it is very simple and was also used in influential scientific papers (Baroni et al., 2014; Schnabel et al., 2015). I will use the TOEFL, and the ESL datasets as presented in Table 6 to run the synonym detection task. In both datasets, each row consists of a query word and 4 given words from which the synonym of the query word must be found. First, the similarities between the query word and the 4 given words are calculated using the cosine similarity measure. Then, as in the word analogy task, for each synonym question the similarity scores are ranked, and the rank of the correct synonym is written down. Since each row consists only of 4 words the ranks can only be 1, 2, 3 or 4, with 1 meaning that the model correctly identified the synonym. To measure the goodness of the model using this task, I will look at a weighted score just as in the word analogy task. Since, it only consists of 4 ranks the weights are [1, 0.75, 0.5, 0.25]. For example, the ranking distribution of the GloVe embedding on the ESL dataset looks like this [16, 9, 14, 9]. This would mean that the 16 is multiplied by 1, the 9 is multiplied by 0.75, the 14 is multiplied by 0.5 and the last 9 is multiplied by 0.25. Next, the weighted values are summed up and divided by the total count of used observations from the dataset. The resulting evaluation metric would then be 0.6667.

Just as in the word analogy task I will run a total of 6 KS-test for the synonym detection tasks to determine the statistical significance between differences in performance metrics. The distributions to run the significance tests are the ranking distributions of each embedding model for both datasets.

To further broaden the focus of the analysis I also incorporated 2 extrinsic text classification evaluation tasks for which I used two test datasets. For the first dataset (BBC dataset) the task is to make a multiclass classification, where articles from different genres need to be correctly classified. For the second dataset (IMDb dataset) a two-class classification must be made, where IMDb movie reviews need to be correctly assigned a positive or negative sentiment. Both tasks are very common and simple NLP tasks and were used and analysed by other researchers before (Lai et al., 2015; Wang et al., 2019). For both tasks I used the same training algorithm, where first the datasets (articles and movie reviews) are cleaned and stopwords are removed. For the IMDb dataset I kept words of negation [*not, cannot, etc.*] since these are important to correctly assign a sentiment of a review. Next, each article, or rather movie review is reduced to the average word vector of all words in that article or movie review. This pre-processing step is also illustrated in Figure 5. These averaged word vectors are then used as the input data for the classification model, for which I chose a random forest model where each entry of a word vector represents one input feature, resulting into a random forest model with V (size of vocabulary) input variables. As stated in the Section 4 Data the datasets were divided into a training and testing dataset with an 80/20 split.

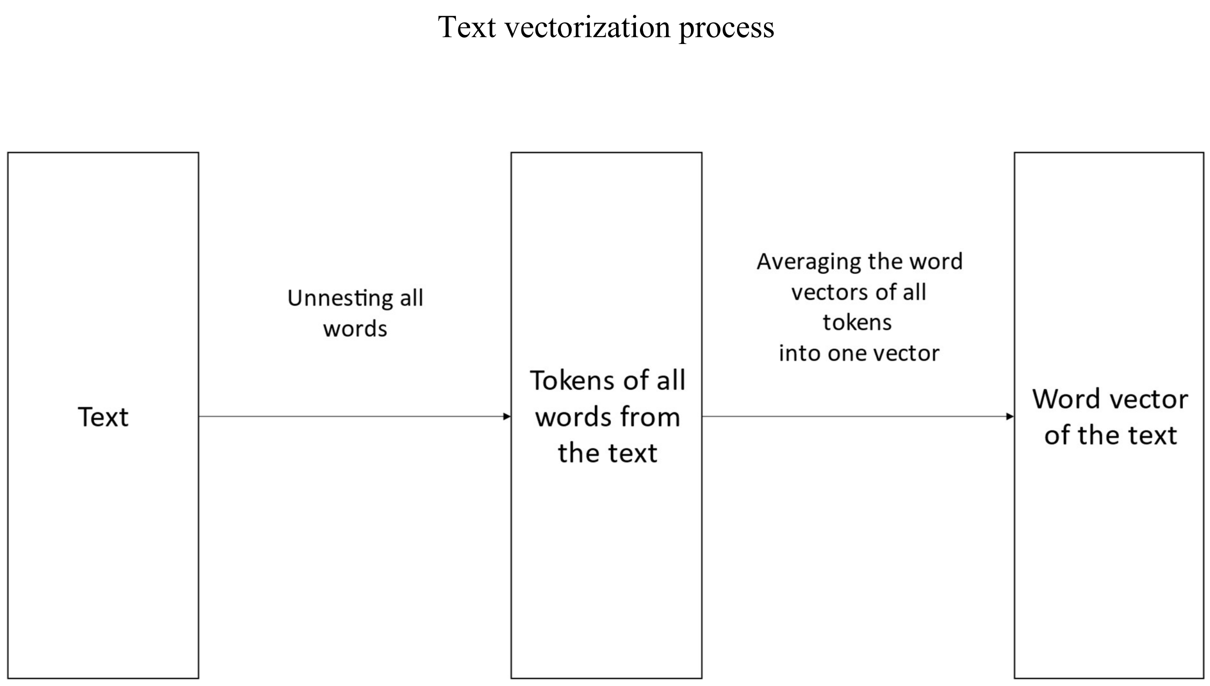
Text vectorization process
Text
Unnesting all words
Tokens of all words from the text
Averaging the word vectors of all tokens into one vector
Word vector of the text

Figure 5: The figure shows exemplary how a single text is reduced to a single vector. From left to right, first the text is deconstructed into its single words (unnesting). Next, the word vectors from all single words from the text are averaged into a single vector. Doing this for all movie reviews and news articles, then these vectors are implemented in the text classification models.

Hyperparameters of the random forest model are set to default values which are pre-set in the utilized package ("randomForest"), which was used to implement the model. All final random forest models consist of 500 individual classification trees, where for each tree only a third of the input features were used and all trees were built to the fullest with the size of a leaf being one. Performance of a model is measured by looking at the Matthew's Correlation Coefficient (MCC), which is a performance metric to assess the performance of binary classification models. The score ranges from [-1, 1], with a score of 1 indicating perfect successful prediction performance and -1 indicating perfectly failing prediction performance of a model. Since for the sentiment analysis task only a binary classification must be made, I used the standard MCC score as performance metric. For the multiclass text classification task using the BBC dataset, I used a macro averaged MCC score to measure the performances. Finally, to measure the significance of the differences between models' performances, I ran multiple KS-tests using the distribution of the resulting confusion matrixes of the classification models. All KS-tests are calculated with a 95% confidence interval.

Finally, all implemented tasks, used test datasets, used performance metrics and what significance tests were applied are summarized in Table 8 to give an overall view of my analysis before I go into the results.

Evaluation task overview

Dataset	Task	Metric	Significance Tests	Dataset Source
RW	Word similarity	Pearson	one way ANOVA	Luong et al. (2013)
SV	Word similarity	Pearson	one way ANOVA	Gerz et al. (2016)
ESL	Synonym detection	Weighted score	2 sample KS-test	Turney (2001)
TOEFL	Synonym detection	Weighted score	2 sample KS-test	Landauer & Dumais (1997)
BATS	Word analogy	Weighted score	2 sample KS-test	Gladkova et al. (2016)
GOOGLE	Word analogy	Weighted score	2 sample KS-test	Mikolov et al. (2013a)
BBC	Text classification	Weighted MCC	2 sample KS-test	Greene & Cunningham (2006)
IMDb	Text classification	MCC	2 sample KS-test	Maas et al. (2011)

Table 8: From left to right are the abbreviated names of the datasets, the type of task for each dataset, the performance metric used for that task, the consistency metric to determine the significance between performances of models and the origin and/or first appearance of the dataset in scientific literature.

5.2 Results

Before I showcase the performances of the different models on the tasks, I will first present the nearest neighbouring words of different query words, so a first impression of the models' embedding quality is given. For the query words I will look at 2 nouns, verbs, and adjectives each, where one is a more common and the other a relative rarer word. In Table 9 the 8 nearest neighbouring words to the 6 query words are presented for all 3 word embedding models. Neighbouring words are determined using cosine similarity on the whole vocabulary of the model.

Closest surrounding words

Query Word	GloVe	CBOW	NMF
piano	piano, violin, solo, harpsichord, trio, trumpet, orchestra, cello	piano, violin, cello, sonata, orchestral, sonatas, quartet, concertos,	piano, dune, golan, beethoven, yahweh, jedi, browns, rock
rust	rust, tar, grease, biomarkers, tubing, pineapple, tobacco, foam	rust, rusting, corrosion, spray, abrasion, stains, corrosive, coating,	rust, fluorinated, jars, yb, lamellae, nucleation, halogens, wreck
Shine	shine, shines, insulate, bite, emit, stray, absorb, shining	shine, glow shining, sky, clouds, light, rainbows, shadows	shine auditory bonobo pausing kymlicka westgate reciting suckle
police	police, officers, officer, officials, guard, personnel, secret, army,	police, prosecutors, officers, policemen, unarmed, personnel, civilians, omon	police, easton, offence, imaginings, depriving, advice, sultry, matrimony
big	big, little, bang, game, played, like, playing, boy	big, coens, crazy, terrific, weird, hollywood, lebowski, scary	big, belgica, padres, inbreeding, sneak, largest, fest, redneck
run	run, running, runs, ran, up, out, home, off	run, running, shoot, stand, move, go, walk, catch	run, rate, stable, supply, imperfect, decreased, stickiness, steady

Table 9: On the left 6 query words are listed, 2 nouns, 2 verbs, and 2 adjectives, respectively, with the first 3 being relatively fewer common words. The next columns, present the 8 nearest neighbouring words for each query word per word embedding model. The neighbouring words are all selected using cosine similarity to measure the distance.

For all nearest neighbours the nearest word is always the word itself since it will automatically have a perfect similarity score. Looking at the query word *piano* one can see that GloVe and CBOW embedded the word in a similar environment resulting in reoccurring neighbouring words like *violin* and *cello*. Looking at the query word *run* we see that GloVe conjugates the word while CBOW lists verbs which have a similar meaning or are related to *run*. Overall, Table 9 demonstrates that GloVe and CBOW show plausible neighbouring words which are either related or similar with the query words. But looking at the neighbouring words generated by NMF the model mostly presents words which have no semantic or syntactic connection to each other. This could be since the word matrix is built from the term-document matrix, resulting in word vectors which are similar when the words occur in the same documents. The model does not fail completely, since it can still link words like *largest* to *big*, *offence* to *police*, *piano* to *beethoven* or *rust* to *wreck*. Further, one can see that non-defined terms are still in the vocabulary, like *yb* and *omon*, which do not carry any meaning and distort the models. The table should demonstrate that in the case of my analysis the NMF embeddings captured the syntax and semantics of words worse than compared to the other 2 models.

Finally, Table 10 showcases the performances of the 3 word embedding models on the 8 previously mentioned evaluation tasks. All scores are scaled to a range of [-1, 1], so values can be compared with each other although different performance measures are used. Since Pearson correlation coefficient and Matthew's correlation coefficient are already on a scale of [-1, 1], only the weighted scores from the synonym detection and the word analogy tasks are scaled from a range of [0, 1] to [-1, 1]. Further, bold written scores represent the best performing model out of the 3.

Model performances on evaluation tasks

Datasets	**GloVe**	**CBOW**	**NMF**
RW	0.1870	**0.3540**	0.1048
SV	0.0919	**0.1476**	0.0094
ESL	0.1112	**0.3195**	0.0693
TOEFL	0.5701	**0.6141**	0.0877
BATS	-0.4220	**-0.2886**	-0.9860
GOOGLE	-0.3298	**-0.3110**	-0.9878
BBC	0.8949	**0.9161**	0.8392
IMDb	0.3405	**0.3810**	0.3110

Table 10: This table shows the performance of each word embedding model on the 8 different evaluation tasks. Since different evaluation metrics were used among the tasks, all performance metrics were scaled to a constant range of [-1, 1]. Values near 0 represent models which have no prediction power and could be seen analogous to random guessing. Values near -1 represent models which choose mostly the worst outcome, while values near 1 represent models which's predictions are mostly correct.

As it can be seen CBOW performs best in all tasks. The reason for the dominance of CBOW over the other models could be that either CBOW is the superior model, as also shown in (Schnabel et al. 2015) or that the hyperparameters are set better for CBOW compared to GloVe and NMF, since Levy et al. (2015) even have shown that traditional simple matrix factorization techniques like SVD could outperform novel models like GloVe and Skip-Gram if tweaked correctly. For my analysis I did not emphasize on tweaking the hyperparameters and keeping most of them at default values and certain hyperparameters like dimensions and minimum word count constant over all models. Further, in Table 10 highlighted cells represent performances of models which are not statistically significantly different to each other on a 95% confidence level, in contrast to that not highlighted cells represents performances which are statistically significantly different from the other performances. Since the KS-test was used to assess the significance of differences in scores, a high observation count and a high difference between scores contributed to the significance of the values. Since the ESL and TOEFL datasets only consist of 50 and 80 observations, respectively, also bigger differences in the scores are not statistically significant from each other. On the other hand, the small difference between CBOW and GloVe in the GOOGLE word analogy task is still significant, which is mostly contributed by the size of the dataset of around 19K analogy questions. Additionally, to the significance testing using ANOVA for the word similarity tasks, correlation tests were also carried out, which showed that all scores are statistically significantly different from 0, even though the value for NMF at the SV dataset is very close to 0.

For most tasks all models perform either statistically significantly different from each other or the opposite, expect for the TOEFL synonym detection task. Here the low count of observations and the small difference between scores of CBOW and GloVe result in a statistical insignificance of the two performances, while NMF performed so bad at the task that despite the low observation count the score from NMF is statistically significantly different from CBOW and GloVe.

Further, for the intrinsic tasks it was expected that NMF underperforms compared to CBOW and GloVe, since we saw in Table 9 that for NMF neighbouring words of query words were mostly not semantically or syntactically related with each other. Even though NMF performs observably worse than the other two models on the intrinsic tasks, for the extrinsic tasks all 3 models perform on a similar level. The reasons why two models could perform similarly well on a task is either that the two models' embeddings are similarly good or that the underlying task is so easy to perform that even with a bad embedding one can achieve high performance. Judging from this observation I conclude that, either the chosen extrinsic evaluators are not reliable evaluators, meaning that the tasks themselves were too easy resulting in that a bad word embedding model (NMF) performs relatively well, or that the intrinsic evaluation tasks are not good proxies to assess the quality of a word embedding model used for extrinsic downstream problems.

Another phenomenon is that for the first three intrinsic evaluation tasks, CBOW performs relatively better than GloVe, with a ~100%, ~50% and ~200% increase in the scores of the first, second and third task respectively. While for the last three intrinsic evaluation tasks CBOW only merely outperforms GloVe if looked at the relative increase in the scores. A reason for these differences can be seen if the structures of the datasets are considered. Although the original GOOGLE dataset contains 19,544 observations it is only built of 551 unique word pairs which are intertwined, the same holds for the original BATS dataset which is 25,725 observations big but is only based of 790 unique word pairs. In contrast to that the RW dataset is 2034 observations big but consists of 2951 unique words, and the SV dataset is 3500 observations big and consists of 827 unique words. So, this intertwining of word pairs with other word pairs to increase the observation count may be a reason why CBOW and GloVe perform more similar. If the embeddings of a model are good for on one word pair, then this will result in situations where the model performs also good on all observations where this particular word pair was used. This would mean that the effective observation count GOOGLE and BATS would not be the actual count of around ~20k and ~25k observations but rather the unique word pairs count which is under 1K for both datasets. Comparing that to the word similarity datasets, the relative share of words which only occur once in the word similarity dataset is much higher than in the word analogy datasets. But I could not find a plausible reason why the difference between scores of CBOW and GloVe are so dramatic for the ESL dataset compared to the TOEFL dataset.

Since the TOEFL task is at least able to distinguish between good and very bad performing models, resulting in significantly different scores for NMF compared to GloVe & CBOW, it meets the minimum requirements for an evaluation method, which the ESL task does not even reach. Still, the TOEFL task was not sophisticated enough to detect statistically significant differences in performances between CBOW and GloVe. From these results and until a more sophisticated dataset for synonym detection tasks is developed, since the ESL and TOEFL dataset are until now the only commonly known synonym detection tasks, I would rate the synonym detection task to be not suited to assess the quality of a word embedding model. The low count of observations will most likely result in insignificant results, especially when the compared word embedding models are much more sophisticated and the hyperparameters are correctly tweaked.

Next, when looking at the text classification tasks, we saw that NMF performed relatively well compared to the intrinsic tasks, although the model was meant to function as a lower benchmark. As mentioned, this could either be reason to a bad dataset, which is not sophisticated enough or an unsophisticated approach from my side to solve the classification problems. Further, all scores from both classification tasks were not statistically significantly different from each other. The insignificance of the scores is either based on the low observation count in the test set, or due to the small differences in scores. The BBC dataset was only 2,225 observations big and with its imbalance in the labels only a small testing test of 475 observations was possible to be built. The original IMDb dataset is 50K observations big which would be more than enough to run a sophisticated analysis, but due to the computationally expensive and time extensive process to run the extrinsic classification tasks, I was bound to only use 5,000 observations from which 1,000 were in the test set. Although a test set of 1,000 observations should also be enough to run a sophisticated analysis, the small differences between scores still resulted in insignificant performance metrics. For my analysis not only was I constrained in the size of the test sets of the classification tasks, due to lack of observations or lack of time and computational power, I was also constrained in the adequacy of the classification models due to the complexity and the computational costs of running the classification tasks. From these results I would conclude that text classification tasks are not suited to measure the goodness of a word embedding model, since an adequately classification model with a large enough test set is necessary to get meaningful and significant results. And still a correlation with the performances in intrinsic tasks is not guaranteed (Schnabel et al., 2015). I only recommend extrinsic evaluation methods if the needed resources like computational power and expertise are guaranteed. Further, as Schnabel at al. (2015) also stated, extrinsic evaluators are the best fitting evaluators when the purpose of the evaluated word embedding is an implementation in a downstream model of same task as the evaluator.

Finally, the word similarity and the word analogy tasks resulted in statistically significantly different performance metrics for the word embedding models, even though some differences were minimal. Of course, the sizes of the datasets had a big contribution to the significance of the results, but in the same manner did the sophistication of the datasets, especially for the RW word similarity dataset. Despite the small difference between scores of GloVe and CBOW in the word analogy tasks, due to the statistical significance the order of the best performing models stays the same, with GloVe as second and NMF as least best performing model. Also, as said earlier above, that this intertwining of word pairs to increase the observation count of the word analogy tasks, could lead to a decrease in the complexity of the tasks. From this perspective, the presented word analogy tasks are still a reliable measure to assess the goodness of a word embedding model, but the two presented word similarity tasks seem more sophisticated. On the other hand, Wang et al. (2019) showed that word analogy tasks had the highest correlation with extrinsic evaluation tasks and therefore function as the best proxies to determine the goodness of a word embedding model on real world problems. Overall, the RW & SV datasets and the BATS & GOOGLE datasets were reliable evaluation datasets to measure the goodness of the word embedding models.

Ultimately, some evaluation tasks, many of which were used in former studies show insignificant results, mostly driven by the small size of the datasets. From that it can be derived that it may be more progressive for future analyses to drop not sophisticated enough tasks and datasets and to concentrate on more sophisticated and bigger datasets and tasks to measure the goodness of a word embedding model. Especially with time passing by and the quality of word embedding models increasing, more sophisticated evaluation models and tasks are needed to significantly measure small differences between models, which small datasets are mostly not capable to do. The idea I present for future analyses is to shift from using many different evaluation tasks for measuring the goodness of word embedding models to only a handful of tasks which should be much more mature and sophisticated, leading to significant and reliable results.

As a final addition I must mention that the choice of using the KS-test and ANOVA-test as significance tests of course has influenced the results, and that the usage other significance tests may lead to other results.

6 Conclusion

My work answers the question of which evaluators are reliable for comparing and evaluating word embeddings, through an exemplary analysis of 6 intrinsic and 2 extrinsic evaluators. From the analysed tasks, which were word similarity, word analogy, synonym detection and text classification tasks, the text classification and especially the synonym detection tasks produced unreliable results. Drivers for this unreliability of the evaluators were either their simplicity of the tasks and/or the small size of the tasks' datasets.

The concept of this analysis was to have two qualitatively similar word embeddings and one that is visibly worse, for which I chose CBOW, GloVe and NMF as embedding models respectively. All models were built on the same 54 million word count corpus and their word embeddings' quality was tested on the same exemplary set of 8 evaluators. Finally, I used ANOVA and Kolmogorov-Smirnov tests to measure the significance of the difference in performance metrics of different models' embeddings. The statistical significance is used as proxy for the evaluator's ability to significantly distinguish between better and worse performing embeddings. The minimum requirement would be to distinguish between the 2 good and the bad model. Further a good evaluator should be able to significantly distinguish differences in performance metrics between the 2 similarly performing models. Many studies only relied on the quantity of used evaluators to present the quality of their newly developed word embedding models while the quality of the evaluators was never taken into account. My recommendation is to shift from focussing on the quantity of used evaluators to the quality and only use evaluators which are bigger, more sophisticated and cover a broader spectrum of word relations to get reliable results. If computational resources and expertise are given I would recommend to evaluate embeddings on specific extrinsic tasks fitting to the real world NLP problem.

Lastly, I need to clearly state that my analysis, and with that my results are bound to certain limitations. For my work I had to train my word embedding models on freely accessible texts since much more sophisticated and better cleaned corpora are behind very expensive paywalls. Further with the computational limitations of my personal desktop computer, which I used to run my analysis, I was not able to use bigger training corpora and with that, higher dimensions for the embedding space. Further constrained by my computational limitations I could only use a fraction of the IMDb text classification dataset for the extrinsic task. Second, even beyond my limitations it was not feasible for me to run my analysis on a wider spectrum of embedding models and evaluators to further strengthen my results, since this analysis is only built for a master thesis. So, I recommend for future research on this topic to also account for the mentioned limitations of mine.

References

Antoniak, M., & Mimno, D. (2018). Evaluating the stability of embedding-based word similarities. *Transactions of the Association for Computational Linguistics, 6*, 107-119.

Bakarov, A. (2018). A survey of word embeddings evaluation methods. *arXiv Preprint arXiv:1801.09536,*

Baroni, M., Dinu, G., & Kruszewski, G. (2014). Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. Paper presented at the *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* 238-247.

Chiu, B., Korhonen, A., & Pyysalo, S. (2016). Intrinsic evaluation of word vectors fails to predict extrinsic performance. Paper presented at the *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP,* 1-6.

Faruqui, M., Tsvetkov, Y., Rastogi, P., & Dyer, C. (2016). Problems with evaluation of word embeddings using word similarity tasks. *arXiv Preprint arXiv:1605.02276,*

Gerz, D., Vulić, I., Hill, F., Reichart, R., & Korhonen, A. (2016). *Simverb-3500: A large-scale evaluation set of verb similarity* [Data set]. Association for Computational Linguistics. http://doi.org/10.18653/v1/D16-1235

Gladkova, A., Drozd, A., & Matsuoka, S. (2016). *Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't* [Data set]. Paper presented at the Proceedings of the NAACL Student Research Workshop*,* 8-15. https://vecto.space/projects/BATS/

Greene, D., & Cunningham, P. (2006). *Practical solutions to the problem of diagonal dominance in kernel document clustering* [Data set]. Paper presented at the Proceedings of the 23rd International Conference on Machine Learning*,* 377-384. https://www.kaggle.com /c/learn-ai-bbc

Hainke, M. (2019). *Wikipedia_API* [Computer Software]. https://github.com/michael-hainke/Wikipedia_API

Harris, Z. S. (1954). Distributional structure. *Word, 10*(2-3), 146-162.

Ide, N., & Macleod, C. (2001). *The american national corpus: A standardized resource of american english* [Data set]. In Proceedings of corpus linguistics (Vol. 3, pp. 1-7). Lancaster, UK: Lancaster University Centre for Computer Corpus Research on Language. https://anc.org/data/oanc/download/

Kučera, H., & Francis, W. N. (1967). *Computational analysis of present-day american english*[Data set]. Brown university press.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. Paper presented at the *Twenty-Ninth AAAI Conference on Artificial Intelligence,*

Landauer, T. K., & Dumais, S. T. (1997). *A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge* [Data set]. Psychological Review*, 104*(2), 211.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*(6755), 788-791.

Lee, D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems, 13*

Levy, O., & Goldberg, Y. (2014). Linguistic regularities in sparse and explicit word representations. Paper presented at the *Proceedings of the Eighteenth Conference on Computational Natural Language Learning,* 171-180.

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics, 3*, 211-225.

Luong, M., Socher, R., & Manning, C. D. (2013). *Better word representations with recursive neural networks for morphology* [Data set]. Paper presented at the Proceedings of the Seventeenth Conference on Computational Natural Language Learning, 104-113. https://nlp.stanford.edu/~lmthang/morphoNLM/

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). *Learning word vectors for sentiment analysis* [Data set]. Paper presented at the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 142-150. https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Mahoney, M. (2011). *Large text compression benchmark* [Data set]. http://mattmahoney.net /dc/text8.zip

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). *Efficient estimation of word representations in vector space* [Data set]. arXiv Preprint arXiv:1301.3781. http://download.tensorflow.org/data/questions-words.txt

Mikolov, T., Yih, W., & Zweig, G. (2013b). Linguistic regularities in continuous space word representations. *Paper presented at the Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 746-751.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems, 26*

Miller, G. A. (1995). WordNet: A lexical database for english. *Communications of the ACM, 38*(11), 39-41.

Nesi, H., Gardener, S., Thompson, P., & Wickens, P. (2008). *British Academic Written English Corpus* [Data set]. Oxford Text Archive. http://hdl.handle.net/20.500.12024/2539

Paatero, P., & Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics, 5*(2), 111-126.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. Paper presented at the *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP),* 1532-1543.

Qiu, Y., Li, H., Li, S., Jiang, Y., Hu, R., & Yang, L. (2018). Revisiting correlations between intrinsic and extrinsic evaluations of word embeddings. *Chinese computational linguistics and natural language processing based on naturally annotated big data* (pp. 209-221). Springer.

Rastogi, P., Van Durme, B., & Arora, R. (2015). Multiview LSA: Representation learning via generalized CCA. Paper presented at the *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 556-566.

Rong, X. (2014). Word2vec parameter learning explained. *arXiv Preprint arXiv:1411.2738,*

Schnabel, T., Labutov, I., Mimno, D., & Joachims, T. (2015). Evaluation methods for unsupervised word embeddings. Paper presented at the *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing,* 298-307.

Shalaby, W., & Zadrozny, W. (2015). Measuring semantic relatedness using mined semantic analysis. *arXiv Preprint arXiv:1512.03465,*

Tatsuki, D. (1998). *Basic 2000 words – Synonym Match 1* [Data set]. http://a4esl.org/q/j/dt /mc-2000-01syn.html

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. Paper presented at the *European Conference on Machine Learning,* 491-502.

Vijaymeena, M. K., & Kavitha, K. (2016). A survey on similarity measures in text mining. *Machine Learning and Applications: An International Journal, 3*(2), 19-28.

Wang, B., Wang, A., Chen, F., Wang, Y., & Kuo, C. J. (2019). Evaluating word embedding models: Methods and experimental results. *APSIPA Transactions on Signal and Information Processing, 8*

Appendix

Literature overview table

Study	Main Findings	Training Data	Word-embedding Models	Evaluation Methods	Consistency Measures
Schnabel T. et al. (2015)	Embeddings should be compared with respect to the given task. Frequency effects shed doubt on the utility of cosine similarity as similarity measure.	All models are built on a 103K Terms 2007 Wikipedia dump	Usage of the 6 different embedding models	Intrinsic: 6 word-similarity tasks 4 categorizing tasks 2 preference selection tasks 3 word-analogy tasks 1 comparative intrinsic evaluation task Extrinsic: 1 coherence analysis 1 sentiment classification 1 noun phrase chunking task	Using **random permutation test** for comparative intrinsic evaluation, coherence analysis, sentiment classification and noun phrase chunking to measure the **significance of resulting metrics**
Antoniak M. et al. (2018)	Corpora's content has significant influence on the model's performance in similarity tasks. Bootstrapping allows to compare models while accounting for the significant influence of the composition of the corpus	6 Different corpora with varying sizes from 2 million up to 20 million words, resulting in a total of 59 million words	PPMI, SGNS, GloVe, LSA	Intrinsic: Only focus on one word similarity tasks	Measuring the **stability of word similarity task performances**, against different training corpora **using bootstrapping** to create multiple training corpora

Literature overview table continued

Study	Main Findings	Training Data	Word-embedding Models	Evaluation Methods	Consistency Measures
Shalaby & Zadrozny (2015)	Introduction of a novel distributional semantics model MSA, which holds up in performance to state-of-the-art models.	52GB 2015 Wikipedia Dump	Total of 13 different word embedding models	Intrinsic: 6 word-similarity tasks	First time implemented the **Steiger's Z score** to measure **difference between different Pearson correlation coefficients** in word similarity tasks
Wang B. et al. (2019)	Giving practical guidance in choosing the optimal word embedding evaluation methods for specific extrinsic tasks. Semantic word analogy tasks result in the overall highest correlation with extrinsic evaluators.	10 GB wiki2010 corpus	6 unique WE methods 16 models in total	Intrinsic: 13 word-similarity tasks 2 word-analogy tasks 3 concept categorization tasks 2 outlier detection tasks 1 QVEC task Extrinsic: 1 POS tagging 1 chunking task 1 NER task 2 sentiment-analysis tasks 1 neural machine translation task	Measuring **consistency of evaluation methods** by calculating **correlation between intrinsic and extrinsic** evaluation metrics

Literature overview table continued

Study	Main Findings	Training Data	Word-embedding Models	Evaluation Methods	Consistency Measures
Chiu et al. (2016)	No and even negative correlation between intrinsic evaluators and extrinsic evaluators. Suggestion to implement extrinsic evaluators when assessing quality of embeddings.	Combination of different English corpora Total of 3.8 billion tokens	Usage of only Skip-gram but with varying window sizes	Intrinsic: 10 word-similarity tasks Extrinsic: 1 POS tagging 1 chunking 1 NER task	Calculation of **Pearson's correlation coefficient** between **intrinsic and extrinsic task** performances over different window sizes
Rastogi et al. (2015)	Introduction of a new word embedding model MVLSA with a fast scalable algorithm Introduction of MRDS to measure significance between correlation values	Combination of different training corpora	Comparison of MVLSA with GloVe and Skip-gram	Intrinsic: 13 word-similarity tasks	Computation of a **MRDS using Steiger's Z score**, to calculate **significance between correlation values** and determine the best performing models
My Work (2023)	Many evaluators for word embeddings, used in previous scientific works are outdated and mostly result unreliable findings	Combination of different training corpora	CBOW GloVe NMF (lower benchmark)	Intrinsic: 2 word-similarity tasks 2 word-analogy tasks 2 synonym detection tasks Extrinsic: 2 text classification tasks	Usage of **ANOVA** for word similarity tasks and **KS-test** for remaining intrinsic and extrinsic tasks

Table 1: The table gives an overview of how previous scientific work used significance testing for word embedding models, and how my work is to be classified compared to previous work.

Depiction of a simple neural network model

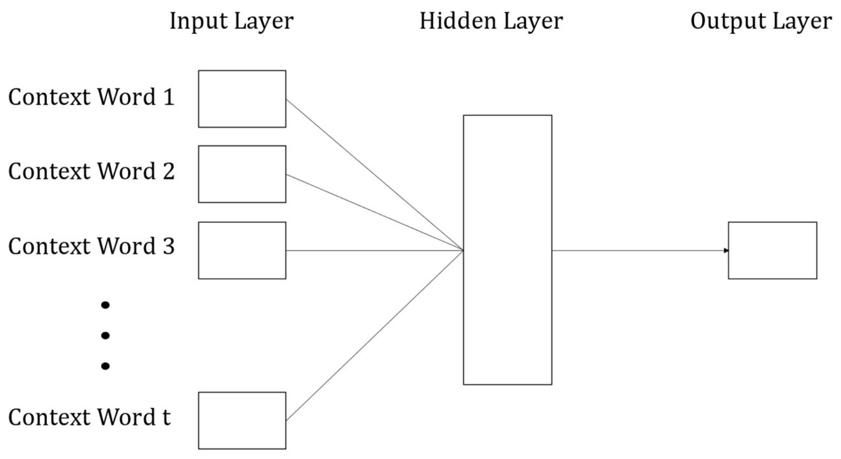

Figure 2: Depiction of a simple neural network model, (from left to right) with multiple inputs, a single hidden layer, and a single output.

Detailed list of the composition of training corpora

BAWE word count					
Genre	Year 1	Year 2	Year 3	Master	Total
Arts and Humanities	468,353	583,617	427,942	234,206	1,714,118
Life Science	299,370	408,070	263,668	441,283	1,412,391
Physical Science	300,989	314,331	426,431	339,605	1,381,356
Social Science	371,473	475,668	440,674	688,921	1,999,130
Total Word Count	1,440,185	1,781,686	1,558,715	1,704,015	6,506,995

Brown document count

Informative Prose

Reportage

Political	Sport	Society	Spot News	Financial	Cultural	Total
14	7	3	9	4	7	44

Editorial

Institutional	Personal	Letters to the Editor	Total
10	10	7	27

Reviews

Total
17

Religion

Books	Periodical	Tracts	Total
7	6	4	17

	Books	Periodicals	Total
Skills & Hobbies	2	34	36
Lore	23	25	48
Belles Letteres, Biography, Memoirs, etc.	38	37	75

Miscellaneous

Government Documents	Foundation Reports	Industry Reports	College Catalogue	Industry House Organ	Total
24	2	2	1	1	30

Detailed list of the composition of training corpora continued

Learned							
Natural Science	Medicine	Mathematics	Social Science	Political Science	Humanities	Technology	Total
12	5	4	14	15	18	12	80

Imaginative Prose

	Novels	Short Stories & Essays	Total
General Fiction	20	9	29
Mystery	20	4	24
Science Fiction	3	3	6
Adventures	15	14	29
Romance	14	15	29
Humour	3	6	9

OANC word count

Spoken

Name	Domain	Word Count
Charlotte	Face to Face	198,295
Switchboard	Telephone	3,019,477
Total	-	3,217,772

Written

Name	Domain	Word Count
911 Reports	Government	281,093
Berlitz	Travel Guide	1,012,496
Biomed	Technical	3,349,714
Eggan	Fiction	61,746
ICIC	Letters	91,318
OUP	Non-Fiction	330,524
PLOS	Technical	409,280
Slate	Journal	4,238,808
Verbatim	Journal	582,384
Web Data	Government	1,048,792
Total	-	11,406,155
Corpus Total	-	14,623,927

Detailed list of the composition of training corpora continued

Wikipedia word count

Domain	Word Count
Politics	1,780,960
Wildlife	570,703
Materials	1,844,023
Physics	1,517,040
Clothing	781,867
Electronics	1,259,865
Fashion	954,265
Literature	885,906
Paper	1,335,596
Safety	2,823,224
Agriculture	2,269,449
Nutrition	2,979,198
Philosophy	2,048,109
Psychology	5,902,674
Rubber	906,576
Water	1,308,787
Total	28,452,800

Table 2: The table shows how the different corpora used for my model training are structured. The "Matt Mahoney's first 100 Million bytes" corpus is not listed in this table, since it only consists of the first 100 million bytes of Wikipedia text and it is not disclosed which genres were looked at. The listed corpora are mainly divided into different genres, domains, points in time, etc. Further, I need to highlight that for all listed corpora in the table, only for the Brown corpus the composition of texts is depicted in document count whereas for the rest it is depicted in word count.

GOOGLE dataset composition

Genre	Example	% To total questions	Unique words pairs
Capital-common-country	Athens : Greece; Baghdad : Iraq	2.58%	23
Capital-world	Abuja : Nigeria; Accra : Ghana	23.14%	116
Currency	Algeria : Dinar; Europe : Euro	4.31%	29
City-in-state	Huston : Texas; Tampa : Florida	12.62%	47
Family	Dad : mom; He : She	2.58%	23
Adjective-adverb	Calm : Calmly; Lucky : Luckily	5.07%	32
Opposite	Clear : Unclear; Logical : Illogical	4.15%	29
Comparative	Big : Bigger; Short : Shorter	6.81%	37
Superlative	Big : Biggest; Tall : Tallest	5.74%	34
Present participle	Dance : Dancing; Play : Playing	5.40%	33
Nationality-adjective	Albania : Albanian; France : French	8.18%	41
Past tense	Knowing : Knew; Moving : Moved	7.98%	40
Plural-nouns	Bird : Birds; Onion : Onions	6.81%	37
Plural-verbs	Eat : Eats; Walk : Walks	4.45%	30

Table 4: The table shows the structure and important statistics of the GOOGLE word analogy test set. The first column shows the different genre of analogy questions, with the first 5 rows being semantic questions and the rest morphological. Next, example questions from the dataset are shown. In the last two columns the relative share of the questions to all questions and the number of unique word pairs are presented.

BATS dataset composition

Main Category	Subcategories	Examples
Inflectional Morphology	Nouns	student:students, wife:wives
	Adjectives	strong:stronger, strong:strongest
	Verbs	follow:follows, follow:following, follow:followed, following:follows, following:followed, follows:followed
Derivational Morphology	Stem Change	bake:baker, edit:editable, continue:continuation, argue:argument
	No Stem Change	create:recreate, home:homeless, mad:madness, able:unable, usual:usually, used:overused
Lexicographic Semantics	Hypernyms	turtle:reptile, peach:fruit
	Hyponyms	color:white
	Meronyms	car:engine, sea:water, player:team
	Antonyms	up:down, clean:dirty
	Synonyms	sofa:couch, cry:scream
Encyclopedic Semantics	Animals	cat:kitten, dog:bark, fox:den
	Geography	Athens:Greece, Peru:Spanish, York:Yorkshire
	People	Lincoln:president, Lincoln:American
	Other	blood:red, actor:actress

Table 5: Adapted from "Analogy-based detection of morphological and semantic relations with word embeddings: What works and what doesn't.," by A. Gladkova, A. Drozd, and S. Matsuoka, 2016, *Proceedings of the NAACL Student Research Workshop,* p. 10. Copyright 2016 by Association for Computational Linguistics.