On the Learnability of Concepts

With Applications to Comparing Word Embedding Algorithms

Adam Sutton(✉) and Nello Cristianini

University of Bristol, Bristol BS8 1UB, UK
adam.sutton@bristol.ac.uk

Abstract. Word Embeddings are used widely in multiple Natural Language Processing (NLP) applications. They are coordinates associated with each word in a dictionary, inferred from statistical properties of these words in a large corpus. In this paper we introduce the notion of "concept" as a list of words that have shared semantic content. We use this notion to analyse the learnability of certain concepts, defined as the capability of a classifier to recognise unseen members of a concept after training on a random subset of it. We first use this method to measure the learnability of concepts on pretrained word embeddings. We then develop a statistical analysis of concept learnability, based on hypothesis testing and ROC curves, in order to compare the relative merits of various embedding algorithms using a fixed corpora and hyper parameters. We find that all embedding methods capture the semantic content of those word lists, but fastText performs better than the others.

Keywords: Word embedding · Linear classifier · Concepts

1 Introduction

Word embedding is a technique used in Natural Language Processing (NLP) to map a word to a numeric vector, in a way that semantic similarity between two words is reflected in geometric proximity in the embedding space. This allows NLP algorithms to keep in consideration some aspects of meaning, when processing words. Typically word embeddings are inferred by algorithms from large corpora based on statistical information. These are unsupervised algorithms, in the sense no explicit information about the meaning of words is given to the algorithm. Word embeddings are used as input to multiple downstream systems such as text classifiers [19] or machine translations [2].

An important problem in designing word embeddings is that of evaluating their quality, since a measure of quality can be used to compare the merits of different algorithms, different training sets, and different parameter settings.

Published by Springer Nature Switzerland AG 2020
I. Maglogiannis et al. (Eds.): AIAI 2020, IFIP AICT 584, pp. 420–432, 2020.
https://doi.org/10.1007/978-3-030-49186-4_35

Importantly, it can also be used as an objective function to design new and more effective procedures to learn embeddings from data. Currently, most word embedding methods are trained based on statistical co-occurrence information and are then assessed based on criteria that are different than the training ones.

Cosine similarity and euclidean distances have shown the ability to represent semantic relationships between words such as in GloVe where the vector representations for the words man, woman, king and queen are such that [13]:

$$king - queen \approx man - woman \tag{1}$$

Schnabel et al. [16] identifies two families of criteria: intrinsic and extrinsic, the first family assessing properties that a good embedding should have (eg: analogy, similarity, etc.), the second assessing their contribution as part of a software pipeline (eg. in machine translation).

We propose a criterion of quality for word embeddings, and then we present a statistical methodology to compare different embeddings. The criterion would fall under the intrinsic class of methods in the classification of Schnabel et al. [16], and has similarities with both their coherence criterion and with their categorization and relatedness criteria. However it makes use of the notion of "concept learnability" based on statistical learning ideas. We make use of extensional definitions of concepts, as they have been defined by [1]. Intuitively, a concept is a subset of the universe, and it is learnable if it is possible for an algorithm to recognise further members after learning a random subset of its members.

The key part in this study is that of a "concept". If the set of all words in a corpus is called a vocabulary (which can be seen as the universe), we define any subset of the vocabulary as a concept. We call a concept learnable if it is possible for a learning algorithm to be trained on a random subset of its words, and then recognise the remaining words. We argue that concept learnability captures the essence of semantic structure, and if the list of words has been carefully selected, vetted and validated by rigorous studies, it can provide an objective way to measure the quality of the embedding.

In the first experiment we will measure the learnability of Linguistic Inquiry and Word Count (LIWC) lists. We compare LIWC lists to randomly generated word lists for popular pretrained word embeddings of three different algorithms (GloVe [13], word2vec [10], and fastText [9]). We show that LIWC concepts are represented in all embeddings through statistical testing.

In our second experiment we compare the learnability of different types of embedding algorithms and settings, using a linear classifier. We compare three of these embedding methods (GloVe [13], word2vec [10], and fastText [9]) to each other. We use the same method as previous, however for this experiment we train with the same hyper parameters and corpus across all three word embeddings [20]. We show that from this experiment fastText performs the best, performing significantly better than both word2vec and GloVe.

This study is a statistical analysis of how a given word embedding affects the learnability of a set of concepts, and therefore how well it captures their meaning. We report on the statistical significance of how learnable various concepts are

under different types of embedding, demonstrating a protocol for the comparison of different settings, data sets, algorithms. At the same time this also provides a method to measure the semantic consistency of a given set of words, such as those routinely used in Social Psychology, eg. in the LIWC technique.

2 Related Work

Word embedding algorithms can be generated taking advantage of the statistical co-occurrence of words, assuming that words that appear together often have a semantic relationship. Three such algorithms that take advantage of this assumption are fastText [9], word2vec [8], and GloVe [13].

There has been a lot of work focused on providing evaluation and understanding for word embeddings. Schnabel et al. have looked at two schools of evaluation; intrinsic and extrinsic [16]. Extrinsic evaluations alone are unable to define the general quality of a word embedding. The work also shows the impact of word frequency on results, particularly with the cosine similarity measure that is commonly used. Intrinsic methods have also had criticisms, with Faruqui et al. calling word similarity and word analogy tasks unsustainable and showing issues with the method [5].

Nematzadeh et al. showed that GloVe and word2vec have similar constraints when compared to earlier work on geometric models [11]. For example, a human defined triangle inequality such as "asteroid" being similar to "belt" and "belt" being similar to "buckle" are not well represented within these geometric models.

Schwarzenberg et al. have defined "Neural Vector Conceptualization" as a method to interpret what samples from a word vector space belong to a certain concept [17]. The method was able to better identify meaningful concepts related to words using non linear relations (when compared to cosine similarity). This method uses a multi class classifier with the Microsoft Concept Graph as a knowledge base providing the labels for training.

Sommerauer and Fokkens have looked at understanding the semantic information that has been captured by word embedding vectors [18] using concepts provided by [3] and training binary classifiers for these concepts. Their proposed method shows that using a pretrained word2vec model some properties of words are represented within the embeddings, while others are not. For example, functions of a word and how they interact are represented (e.g. having wheels and being dangerous), however appearance (e.g. size and colour) are not as well represented.

3 Methods and Resources

3.1 Embeddings

A corpus **C** is a collection of documents from sources such as news articles, or Wikipedia. From **C** we can extract a set of words to be a vocabulary **V**. Each document in **C** is a string of words (in which the ordering of words within the

document is used as part of the embedding algorithm). With a vocabulary $\mathbf{V}$ and a corpus a function $\mathbf{\Phi}$ to be defined such that $\mathbf{\Phi} : \mathbf{V} \rightarrow \mathbb{R}^d$, which is mapping every word in the vocabulary to a d dimensional vector. Word vectors from a word embedding are commonly formalised as w.

Using an embedding method $\mathbf{\Phi}$, we will now define the action of going from words in a vocabulary to an embedded space as: $\mathbf{\Phi}(word_j \in V) = w_j \in \mathbb{R}^d$. A word vector for a given word will now be defined as w. Word vectors are generally normalised to unit length for measurement in word analogy or word similarity tasks:

$$\hat{w} = \frac{w}{||w||} \tag{2}$$

3.2 Concepts

In this paper we make use of the notion of a 'concept' defined as any subset of the vocabulary, that is a set of words. Sometimes we will use the expression "list of words", for consistency with the literature in social psychology, but we will never make use of the order in that list, so that we effectively use "list" as another expression for "set", in this article. We define this as a set of words $\mathbf{L} \subseteq \mathbf{V}$ (or for an embedding a set of points in $\mathbb{R}^d$ such that $\Phi(\mathbf{L}) \subseteq \Phi(\mathbf{V})$).

We use the word vectors from a word list to define this concept in an embedding. In general, a concept is defined as any subset of a set (or a "universe"). We would normally define a concept as an unordered list of words that have been created, validated, and understood by humans that should be learnable by machines. However for the purpose of this paper a concept can be defined as any subset of words from $\mathbf{V}$. This use is consistent with the Extensional Definition of a concept used in logic, and the same definition of concept as used in the probably approximately correct model of machine learning [1].

3.3 Linear Classification

A classifier is a function that maps elements of an input space (a universe, in our case a vocabulary) to a classification space. A binary linear classifier is a function that classifies vectors of a vector space R^d into two classes, as follows:

$$f : \mathbf{R}^d \rightarrow \{0, 1\}, f(x) = \sigma(\langle x, w \rangle + b) \tag{3}$$

We will learn linear classifiers from data, using the Perceptron Algorithm on a set of labeled data, which is a set of vectors labeled as belonging to class 1 or class 0. As we will learn concepts formed by words, and linear classifiers only operate on vectors, we will apply them to the vector space generated by the word embedding, as follows.

A linear classifier is a simple supervised machine learning model used to classify membership of an input. We will use a single layer perceptron with embeddings as input to see if it is possible for a perceptron to predict half of a word list, while being trained on its other half.

Given a word list $\mathbf{L}$ such that $\Phi(\mathbf{L}) \subseteq \Phi(\mathbf{V}) \subseteq \mathbb{R}^d$ we will define the words from this list as $\mathbf{L}^c = \mathbf{V} \setminus \mathbf{L}$. We will use $\mathbf{L}$ and $\mathbf{L}^c$ to define a train set and test set for our perceptron. We will first uniformly sample half of the words of $\mathbf{L}$, we will then sample in equal amount from $\mathbf{L^c}$. We will then append these two word lists to make $\mathbf{L_{train}}$. To produce a test set $\mathbf{L_{test}}$ we will take the remaining words that haven't been sampled from $\mathbf{L}$, and sample the same number of words again from $\mathbf{L^c}$.

A member of the training set can be defined as $l_i \in \Phi(\mathbf{L_{train}})$. We define our prediction function $\hat{y}$ as:

$$\hat{y} = \sigma((\sum_{i}^{d} \theta_i l_i) + b) \tag{4}$$

where θ and b are the training parameters of the classifier and σ is the sigmoid function. We will then train the perceptron using the cross entropy loss function:

$$J = -\frac{1}{|\mathbf{L}|} \sum_{i}^{|\mathbf{L}|} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \tag{5}$$

where y_i is the correct class of the training sample.

3.4 Linguistic Inquiry and Word Count

This study uses lists of words generated by the LIWC project [12], a long-running effort in social psychology to handcraft, vet and validate lists of words of clinical value to psychologists. They typically aim at capturing concerns, interests, emotions, topics, of psychological significance. LIWC lists are well suited to an experiment of this kind as the words within them are common and relevant to any cross-domain corpus.

Table 1. Sample words from the LIWC word lists used in experiments

Full name	Sample words	List name
Positive emotions	happy, pretty, good	posemo
Negative emotions	hate, worthless, enemy	negemo
Anger processes	hate, kill, pissed	anger
Biological processes	eat, blood, pain	bio
Relativity	area, bend, exit	relative
Affective processes	happy, ugly, bitter	affect
Social processes	talk, us, friend	social
Work concerns	work, class, boss	work
Family concerns	mom, brother, cousin	family
Health concerns	weak, heal, blind	health

Table 1 shows samples of the ten word lists used in this study as well their full names, and what they will be described as when used in the context of this study. Most word lists used have hundreds of words in them. Family is the smallest word list with a total of 54 words being used. These word samples will used to extensionally define word lists as concepts within the embedding.

4 Measuring Performance of Linear Classifiers

We will measure the performance of a linear classifier by using the receiver operating characteristic (ROC) curve, a quantity defined as the performance of a binary classifier as its prediction threshold is changed between the lowest probable prediction and its highest probable prediction. This curve plots the True Positive Rate (also known as the Recall) and the False Positive Rate (also known as the fall-out) at each classification threshold possible. We also show the accuracy of the classifier, and the precision.

Our first experiment will look at the three word embedding algorithms of GloVe, word2vec, and fastText with regards to how they perform using pre-trained word embeddings readily available online. Our second experiment will compare all three algorithms performance under identical conditions, with the same training corpus and hyper parameters.

We will take the input as the embedding representations for words, and the output being a binary classification if the word belongs to that LIWC word set ($\mathbf{L}$) or not. For the training set $\mathbf{L_{train}}$, we will uniformly random sample half of the words from the list $\mathbf{L}$ we are experimenting on. We will then sample an equal number of words from $\mathbf{L^c}$. For the test set $\mathbf{L_{test}}$ we take the remaining words from $\mathbf{L}$, and again sample another equal set of negative test samples from $\mathbf{L^c}$.

We repeat this method 1,000 times, and for each iteration of this test we generate new word lists $\mathbf{L_{train}}$ and $\mathbf{L_{test}}$ each time. This method of a linear classifier has been defined in Eq. 4 and Eq. 5. This experiment is performed for the 10 LIWC word lists listed in Table 1. We take their average across all 1,000 iterations of the experiment we performed.

4.1 Learning Concepts from GloVe, Word2vec, and FastText

In this section we will look at the ability of three different word embedding algorithms to capture information in word lists that reflect real world concepts.

To ensure that these metrics are statistically significant, we have created a null-hypothesis of making random concepts based on random word lists ($\mathbf{L_{random}}$) and performing the same classification task on the random concept. We repeat this test 1000 times and take the best performance for each of the metrics we look at for these random lists (which will be defined as $\mathbf{L_{random(max)}}$). Of 1000 tests, we hypothesise no concept defined by a random word list outperforms any of the word lists we test on.

Table 2. Average Performance of a Linear Classifiers using LIWC word lists on GloVe word embeddings to identify members of its own set. Random lists are also tested to obtain a p-value and compare performances. These embeddings perform better than random word lists resulting in a p-value of < 0.001

L	Size	Accuracy	Recall	FPR	Prec	AUC
$\mathbf{L_{posemo}}$	392	0.915	0.902	0.079	0.919	0.964
$\mathbf{L_{negemo}}$	492	0.913	0.913	0.085	0.915	0.965
$\mathbf{L_{anger}}$	184	0.888	0.880	0.103	0.896	0.950
$\mathbf{L_{bio}}$	558	0.895	0.871	0.087	0.909	0.954
$\mathbf{L_{relative}}$	632	0.937	0.935	0.059	0.940	0.979
$\mathbf{L_{affect}}$	908	0.910	0.906	0.085	0.914	0.962
$\mathbf{L_{social}}$	396	0.906	0.887	0.075	0.922	0.962
$\mathbf{L_{work}}$	322	0.899	0.880	0.081	0.916	0.959
$\mathbf{L_{family}}$	54	0.884	0.893	0.125	0.881	0.956
$\mathbf{L_{health}}$	232	0.895	0.880	0.105	0.893	0.953
$\mathbf{L_{random(max)}}$	400	0.547	0.32	0.115	0.617	0.574
$\mathbf{L_{random(avg)}}$	400	0.500	0.198	0.198	0.502	0.501

GloVe. We will set GloVe to be our embedding algorithm ($\mathbf{\Phi}$), with the corpus $\mathbf{C}$ being a collection of Wikipedia and Gigaword 5 news articles. These embeddings are pretrained and available online on the GloVe web-page [6]. These word embeddings are open for anyone to use, and can be used to repeat these experiments.

Table 2 shows the performance and statistics of ten different word lists from LIWC. $\mathbf{L_{random(avg)}}$ shows the average performance of concepts defined from random word lists. $\mathbf{L_{random(max)}}$ shows the best performing random word list for each test statistic.

An accuracy of approximately 0.9 shows a high general performance. The precision and recall show that these word lists are able to accurately discern remaining members of its list and words that are not a part of the concept. After a thousand iterations of random word lists the best performing random lists (shown in $\mathbf{L_{random(max)}}$) were performing worse than each LIWC word list, giving a p-val of < 0.001 for each word list.

word2vec. We will use word2vec as our embedding algorithm ($\mathbf{\Phi}$), with the corpus $\mathbf{C}$ being a dump of Wikipedia from April 2018 [22] using the conventional skip-gram model. These embeddings are available online on the Wikipedia2Vec web-page [22]. These word embeddings are open for anyone to use, and can be used to repeat these experiments.

Table 3 shows the performance and statistics of ten different word lists from LIWC while using the word2vec embedding algorithm. $\mathbf{L_{random(avg)}}$ and $\mathbf{L_{random(max)}}$ again show the average and best performances of random word lists.

Table 3. Average Performance of Linear Classifiers using LIWC word lists on word2vec embeddings to identify members of its own set. Random lists are also tested to obtain a p-value and compare performances. These embeddings perform better than all random word lists resulting in a p-value of < 0.001

L	Size	Accuracy	Recall	FPR	Prec	AUC
$\mathbf{L_{posemo}}$	392	0.904	0.914	0.115	0.888	0.959
$\mathbf{L_{negemo}}$	492	0.923	0.920	0.081	0.919	0.970
$\mathbf{L_{anger}}$	184	0.890	0.906	0.126	0.879	0.953
$\mathbf{L_{bio}}$	558	0.890	0.901	0.120	0.882	0.954
$\mathbf{L_{relative}}$	632	0.911	0.952	0.135	0.876	0.963
$\mathbf{L_{affect}}$	908	0.886	0.947	0.177	0.842	0.950
$\mathbf{L_{social}}$	396	0.893	0.911	0.123	0.881	0.957
$\mathbf{L_{work}}$	322	0.877	0.910	0.154	0.855	0.947
$\mathbf{L_{family}}$	54	0.874	0.912	0.164	0.853	0.953
$\mathbf{L_{health}}$	232	0.893	0.899	0.113	0.889	0.959
$\mathbf{L_{random(max)}}$	400	0.545	0.27	0.055	0.68	0.576
$\mathbf{L_{random(avg)}}$	400	0.498	0.128	0.130	0.494	0.500

An accuracy of approximately 0.9 shows a high general performance, although it performs slightly worse than GloVe's pre-trained embeddings. This shows that the word2vec embedding algorithm $\mathbf{\Phi}$ applied to the corpus $\mathbf{C}$ yields word vectors that represent the real world meaning of words. The AUC is extracted from the scores of the sigmoid within the classifier. Overall word2vec performs slightly worse than GloVe embeddings in most metrics. However while the source corpora is very similar, GloVe has additional sources of information. The p-values for these word lists in comparison to random word lists is again <0.001 showing that these word lists that have a real world representation are represented accurately within the embedding.

fastText. The third and final word embedding algorithm ($\mathbf{\Phi}$) we will test is fastText [7]. The corpus $\mathbf{C}$ is a collection of Wikipedia, "UMBC WebBase corpus" and statmt.org news [9]. These embeddings are also pretrained word embeddings that are available from the fastText website.

Table 4 shows the performance statistics of the fastText word embeddings using our proposed method to evaluate word embeddings. $\mathbf{L_{random(avg)}}$ and $\mathbf{L_{random(max)}}$ show the random performance, while the other lists are LIWC word lists and their respective performances.

A precision of 1 in the best performing random word lists are insignificant as the recall is shown to be poor, due to predicting most samples to be negative. The p-val of all of the word lists defined by LIWC is <0.001 as after one thousand iterations no random list outperformed any of LIWC lists. This again means that

Table 4. Average Performance of Linear Classifiers using LIWC word lists on fastText embeddings to identify members of its own set. Random lists are also tested to obtain a p-value and compare performances. These embeddings perform better than random word lists resulting in a p-value of < 0.001

L	Size	Accuracy	Recall	FPR	Prec	AUC
$\mathbf{L_{posemo}}$	392	0.928	0.925	0.068	0.931	0.977
$\mathbf{L_{negemo}}$	492	0.937	0.934	0.067	0.932	0.978
$\mathbf{L_{anger}}$	184	0.940	0.965	0.084	0.919	0.981
$\mathbf{L_{bio}}$	558	0.917	0.933	0.098	0.905	0.970
$\mathbf{L_{relative}}$	632	0.933	0.966	0.099	0.907	0.977
$\mathbf{L_{affect}}$	908	0.886	0.947	0.177	0.842	0.950
$\mathbf{L_{social}}$	396	0.927	0.920	0.074	0.925	0.973
$\mathbf{L_{work}}$	322	0.918	0.914	0.077	0.922	0.970
$\mathbf{L_{family}}$	54	0.966	0.975	0.041	0.960	0.995
$\mathbf{L_{health}}$	232	0.931	0.940	0.078	0.924	0.980
$\mathbf{L_{random(max)}}$	400	0.51	0.04	0.0	1.0	0.562
$\mathbf{L_{random(avg)}}$	400	0.500	0.007	0.006	0.427	0.505

these word lists represent a real world concept, and that the embeddings are able to capture this information of this concept by using members of the set within the embedding to define it.

4.2 Comparing Embeddings

In this section we will be comparing the performance of the three word embedding algorithms used in the previous experiment. However, for this experiment the hyper parameters and the corpora trained will be fixed for the purpose of direct comparison. All embeddings have been generated by ourselves using the three word embedding algorithms word2vec (skip-gram), GloVe, and fastText.

The AUC metric we have previously shown can be viewed as a measure of the learnability of an embedded concept. This compares the true positive rate (also known as the recall) and the false positive rate and shows the performance at each threshold that is possible within the classifier on for a given word lists test set.

This AUC could be seen as the performance of that binary classifier, and also as a measure of the quality of each embedding and a measure of the quality of each word list. The better the performance of an embedding, the higher perceived quality of that embedding. The better a list performs on all embeddings, the higher the quality of that list.

To accurately compare the performance of the embedding algorithms, we perform the same test as shown in Sect. 4.1. However we ensure that a number

of parameters are kept the same for each embedding, to maintain fairness. For this test, we will ensure that the corpus used to train will be identical between all embeddings. The corpus (**C**) used for all three embedding algorithms will be a dump from the English Wikipedia taken from the first of July, 2019 [20]. The embedding dimension d will be set to 300. A word must appear a minimum of five times to be embedded, and the context window of all words is five.

In Table 5 we show the AUC performance of all three embedding algorithms used in the paper. The fastText embedding algorithm is shown to have the highest performing embedding for 8 of the 10 lists that have been tested. Glove performs best on two lists, and generally performs better than word2vec overall. These performances are consistent with previous comparisons of these word embeddings [9,13]. The word list $\mathbf{L_{relative}}$ is shown to have the best overall performance across all three non-random embeddings, demonstrating the quality of that list.

Table 5. AUC performance of word lists for each embedding algorithm used in these experiments, along with the average AUC for an embedding across all lists. Bold denotes the embedding algorithm that performs best for a given word list. Italic denotes the best performing list for each embedding algorithm.

L	GloVe	word2vec	fastText
$\mathbf{L_{posemo}}$	0.961	0.929	**0.965**
$\mathbf{L_{negemo}}$	0.965	0.945	**0.973**
$\mathbf{L_{anger}}$	0.957	0.928	**0.970**
$\mathbf{L_{bio}}$	0.960	0.935	**0.974**
$\mathbf{L_{relative}}$	***0.971***	*0.927*	*0.961*
$\mathbf{L_{affect}}$	**0.960**	0.944	0.958
$\mathbf{L_{social}}$	0.960	0.925	**0.973**
$\mathbf{L_{work}}$	0.947	0.909	**0.970**
$\mathbf{L_{family}}$	0.948	0.864	**0.963**
$\mathbf{L_{health}}$	0.952	0.923	**0.975**
Mean	0.958	0.922	**0.968**
Median	0.960	0.927	**0.970**

We tested the statistical significance of the performance differences observed between GloVe and fastText. To this purpose we performed a Wilcoxon signed-rank test, using the median of the AUCs from each embedding as the test statistic [21]. We use the Wilcoxon signed-rank test as the fastText mean AUCs shown in Table 5 do not represent a normal distribution.

We propose a null hypothesis that the median difference of fastText and GloVe AUCs (as shown in Table 5) are 0. We use a sample size of 10 as the difference of no pairs are equal to zero. We set our alpha to 0.01 for a one sided

(right) tail test, where the test statistic W_{crit} is 5. We find our resulting W_{test} to be 3, which leads us to reject the null hypothesis and show that fastText outperforming GloVe is statistically significant, for the word lists that we are testing. This gives us a p-value of 0.0088.

5 Conclusion

In this paper, we have shown that word embeddings are able to capture the meaning of human defined word lists. We have shown the ability of embedding algorithms in learning concepts from word lists. In particular we have shown this quality in word2vec, GloVe and fastText. We have shown that learning embeddings from real data can represent real world concepts defined extensionally, utilising word lists provided by LIWC.

We have also shown the relative performance of GloVe, fastText, and word2vec when using LIWC word lists to form concepts using similar corpora that derive most of their corpora from Wikipedia. fastText performs better in the majority of situations for all word lists we have tested from LIWC, while GloVe outperforms word2vec generally. However as all algorithms use slightly different corpora, this result may change depending on the corpora used.

This measure of performance of word embeddings can be used in the future as a measure of "quality" of word embeddings. While there are other methods that look at the performance of word embeddings by evaluating their performance in a specific task [15], our method differs in that it looks at an embeddings general ability to understand human defined concepts. There has also been criticism of evaluating word embeddings using only word similarity tasks [5]. This method can also be used in another way as a measure of the quality of word lists and their ability to accurately describe a concept, providing an assumption or proof that an embedding is performing suitably to the users needs.

Future work with this method would involve extensive testing of the method using with varying differing hyper parameters to see the optimal performance of these embedding algorithms. An example of this is the impact of embedding dimension on performance. Another experiment could be looking at the performance of this test on deep contextualized embeddings such as ELMo [14] and BERT [4]. These embeddings have been shown to have better performance on many tasks that employ word embeddings. While these embeddings are optimized for their specific end tasks, they train embeddings before that tuning process takes place. There is potential to compare these embeddings by testing the extracted embedding with a linear classifier, or fine tuning their full model to our task. However a key benefit for sentence embeddings is the context of words around them, which our task will not benefit from.

Further work could be focused on the performance of different word lists and concepts within word embeddings. The benefit of this could be to validate word lists that are not as carefully curated as LIWC word lists. These word lists may come from different fields, as LIWC is focused on clinical psychology other word lists may perform differently. Different source corpora may also change the

performance of these word lists due to the meaning of some words changing from domain to domain.

References

1. Anthony, M., Biggs, N.: Computational Learning Theory, vol. 30. Cambridge University Press, Cambridge (1997)
2. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
3. Devereux, B.J., Tyler, L.K., Geertzen, J., Randall, B.: The centre for speech, language and the brain (CSLB) concept property norms. Behav. Res. Methods **46**(4), 1119–1127 (2014)
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Faruqui, M., Tsvetkov, Y., Rastogi, P., Dyer, C.: Problems with evaluation of word embeddings using word similarity tasks. arXiv preprint arXiv:1605.02276 (2016)
6. Pennington, J., Socher, R., Manning, C.D.: Wikipedia 2014 + Gigaword 5 pretrained word embeddings. http://nlp.stanford.edu/data/glove.6B.zip, Accessed 07 Oct 2019
7. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
9. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pretraining distributed word representations. In: Proceedings of the International Conference on Language Resources and Evaluation, LREC 2018 (2018)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp. 3111–3119 (2013)
11. Nematzadeh, A., Meylan, S.C., Griffiths, T.L.: Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In: CogSci (2017)
12. Pennebaker, J.W., Francis, M.E., Booth, R.J.: Linguistic inquiry and word count: Liwc 2007, Mahway: Lawrence Erlbaum Associates 71 (2001)
13. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP, pp. 1532–1543 (2014)
14. Peters, M.E., et al.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
15. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016)
16. Schnabel, T., Labutov, I., Mimno, D., Joachims, T.: Evaluation methods for unsupervised word embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 298–307 (2015)
17. Schwarzenberg, R., Raithel, L., Harbecke, D.: Neural vector conceptualization for word vector space interpretation. arXiv preprint arXiv:1904.01500 (2019)
18. Sommerauer, P., Fokkens, A.: Firearms and tigers are dangerous, kitchen knives and zebras are not: testing whether word embeddings can tell. arXiv preprint arXiv:1809.01375 (2018)

19. Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., Qin, B.: Learning sentiment-specific word embedding for twitter sentiment classification. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1555–1565 (2014)
20. Wikimedia: enwiki dump on 20190701. https://dumps.wikimedia.org/enwiki/20190701/. Accessed 07 Jul 2019
21. Wilcoxon, F.: Individual comparisons by ranking methods. In: Kotz, S., Johnson, N.L. (eds.) Breakthroughs in Statistics. Springer Series in Statistics (Perspectives in Statistics). Springer, New York (1992). https://doi.org/10.1007/978-1-4612-4380-9_16
22. Yamada, I., Asai, A., Shindo, H., Takeda, H., Takefuji, Y.: Wikipedia2vec: an optimized tool for learning embeddings of words and entities from wikipedia. arXiv preprint 1812.06280 (2018)