Lexical Relation Mining in Neural Word Embeddings

Aishwarya Jadhav
Heinz College
Carnegie Mellon University, Pittsburgh, PA, USA
ajadhav2@andrew.cmu.edu

Yifat Amir
Electrical Engineering and Computer Sciences
University of California, Berkeley, CA, USA
yifatamir@berkeley.edu

Zachary A. Pardos
Graduate School of Education
University of California, Berkeley, CA, USA
pardos@berkeley.edu

Abstract

Work with neural word embeddings and lexical relations has largely focused on confirmatory experiments which use human-curated examples of semantic and syntactic relations to validate against. In this paper, we explore the degree to which lexical relations, such as those found in popular validation sets, can be derived and extended from a variety of neural embeddings using classical clustering methods. We show that the Word2Vec space of word-pairs (i.e., offset vectors) significantly outperforms other more contemporary methods, even in the presence of a large number of noisy offsets. Moreover, we show that via a simple nearest neighbor approach in the offset space, new examples of known relations can be discovered. Our results speak to the amenability of offset vectors from non-contextual neural embeddings to find semantically coherent clusters. This simple approach has implications for the exploration of emergent regularities and their examples, such as emerging trends on social media and their related posts.

1 Introduction

Word vector models such as Word2Vec (Mikolov et al., 2013a), derived empirically from large corpora of natural language, provide the opportunity to explore what constitutes a linguistic regularity. Conventionally, lexical relations in word vector space have been defined by collections of relatively consistent relationships, or vector offsets, between word-pairs. The presence of these relationships has been established through confirmatory analysis (Levy and Goldberg, 2014), in which a pair of relation examples constructed from prior knowledge is validated to exist in the space by way of analogy (e.g. $geese - goose + mouse \approx mice$) (Finley et al., 2017). The meaning of these collections, or the nature of the relationship between their respective pairs, can be characterized by relation type descriptions such as syntactic (e.g., "plurality") and semantic (e.g., "capital-country") relations.

In this paper, we explore how well lexical relation examples can be clustered using word vectors extracted from state-of-the-art contextual and non-contextual neural word embedding methods. Furthermore, we demonstrate a method for approximating the number of true lexical relations in a noisy offset space. Contextual embeddings such as BERT (Devlin et al., 2018) are the new gold standard on a variety of NLP tasks and have also been shown to possess relational and factual knowledge (Petroni et al., 2019). We perform the unsupervised task of clustering relation examples by using contextual word vectors from BERT; non-contextual word vectors from FastText (Bojanowski et al., 2017) Word2Vec skip-gram (Mikolov et al., 2013a); and explicitly modeled relation vectors (Camacho Collados et al., 2019). Our results show that Word2Vec offsets outperform relation vectors from other embeddings on this task.

We further explore the amenability of the Word2vec offset space to be clustered by evaluating it on the task of discovering new word-pair examples of known lexical relations. The precision of finding new examples using the offset space varied greatly depending on the lexical relation, indicating promise yet demonstrating the need for more nuanced noise rejection in future work.

Proceedings of the 28th International Conference on Computational Linguistics, pages 1299–1311
Barcelona, Spain (Online), December 8-13, 2020

2 Related Work

Relations, in the context of a neural word embedding (Mikolov et al., 2013a), have been described as vector offsets (or differences) between pairs of word vectors that exemplify a relationship. The types of relations and their exemplar pairs have conventionally been established a priori (Mikolov et al., 2013c). Sets of relations and their respective pairs have served as external validations, tested by way of analogy, of the degree to which a vector space encodes a sampling of prior knowledge on the semantics and syntax of the language. In terms of the accuracy of analogy completion (i.e. recall@1), widely-cited neural embeddings have been found to encode upwards of 61% (Mikolov et al., 2013b), and even 75% (Pennington et al., 2014), of related word relationship pairs. At this level of accuracy, it is reasonable to begin to ponder what can be learned from an embedding in a completely unsupervised way, as opposed to only confirming what predefined expert knowledge is encoded.

Several approaches have taken the first steps in an unsupervised direction by exploring if the membership of a word-pair to a relation could be recovered through classification or clustering. Vylomova et al. (2016) use supervised classification to learn an association between vector offsets produced by word-pairs and relation labels (or classes). While their method considers only context-free word vectors such as Word2Vec and GloVe (Pennington et al., 2014), our method also evaluates performance on contextual and explicit relation embeddings.

Levy and Goldberg (2014) surfaced semantics about relations by characterizing what they refer to as a shared *aspect* between words by inspecting the context words, or *features*. Evidence of distributed concepts, also learned from skip-gram models, has been observed in neural embeddings produced from random walks of graphs (Ribeiro et al., 2017) and from sequences of course enrollments (Pardos and Nam, 2020).

Recently, a number of methods have been developed that capture relations between two words in the form of a vector using unsupervised approaches. Espinosa-Anke and Schockaert (2018) learn relation vectors of word-pairs based on averaging their context word vectors and then perform dimensionality reduction using an autoencoder. Jameel et al. (2018) model relationships between words as weighted bag-of-words representations using generalizations of pointwise mutual information. Camacho Collados et al. (2019) develop a latent variable model that aims to explicitly determine what words from given sentences best characterize the relationship between two target words. Wu and He (2019) and Papanikolaou et al. (2019) use contextual models to learn relation embeddings requiring fine-tuning of the base model. Though these methods learn the explicit relation embedding in an unsupervised fashion, they evaluate them on supervised tasks, unlike our approach which evaluates neural embeddings on unsupervised tasks.

3 Datasets

We use a common validation set[1] from the analogical reasoning task introduced in (Mikolov et al., 2013a), which we call the Google validation dataset, as a source of known lexical relations and their corresponding collections of word-pairs. It contains 550 unique word-pairs belonging to 13 unique relations[2]. We also use another popular lexical relation validation set named DiffVec[3] (Vylomova et al., 2016), which consists of 36 relations and 12,458 word-pairs. The Google dataset is more balanced than the DiffVec dataset in terms of the number of word-pairs per relation[4]. We train all models used in our model comparison experiments on the Wikipedia corpus. We then use the pre-trained Google-News corpus Word2Vec vectors (Mikolov et al., 2013b) for the subsequent task of finding new examples of known relations.

[1] `http://download.tensorflow.org/data/questions-words.txt`

[2] It originally contained 14 relations, but because of the redundancy of "capital-common-countries" with "capital-world," the former relation has been discarded.

[3] `https://github.com/ivri/DiffVec`

[4] While using the DiffVec dataset it is common to leave out relations with less than 10 examples (Jameel et al., 2018).

4 Motivation

Using the Word2Vec model, a lexical relation between a pair of words (*word1*, *word2*) can be represented using a vector offset by subtracting the embedding of *word2* from the embedding of *word1*. Past success in the analogy completion task suggests that offsets generated by word-pairs in the same lexical relation are empirically close to parallel (e.g. $geese-goose \parallel mice-mouse$) (Finley et al., 2017). This naturally presents an opportunity for the clustering of offsets and the exploration of the distributed representation of lexical relations.

A novel visualization of the offsets of the Google dataset can be seen in Figure 1. The figure was produced using t-SNE (Van Der Maaten, 2014) to project the 300-dim offset vectors onto a 2-D space. It depicts offsets naturally grouping by relation and demonstrates the plausibility of a clustering approach capturing relation examples in the offset space. Visualization of offsets had previously been applied only to one relation at a time (Mikolov et al., 2013b). With multiple relation examples represented in a single plot, we can observe relative proximities between groups of relations. For example, "nationality" being closest to "currency," both generated by subtracting a word from a country name. Offsets for "gender" and "opposite" are closest to one another, as are those for "comparative" and "superlative" and for "capital-world" and "city-in-state." "Plural" (nouns) and "plural-verbs" are not only close to one another but also somewhat overlapping.

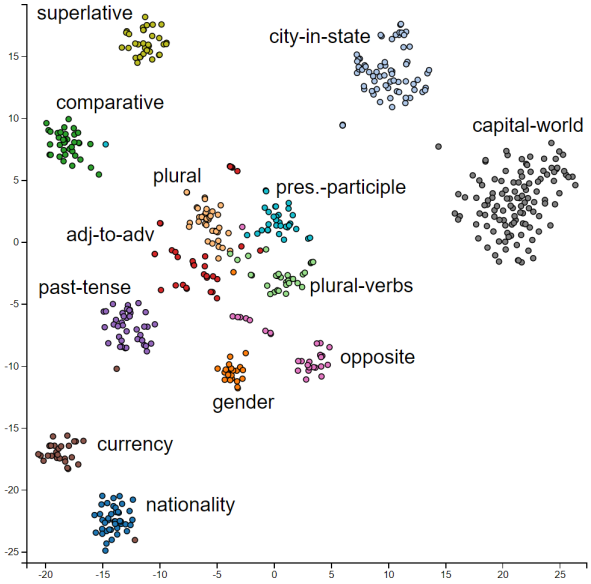

Figure 1: Visualization (via t-SNE) of the Google dataset vector offsets, colored by lexical relation.

Word vector models when used for relation mining are of sociological interest because they are incentivised to capture norms and the exploration of those norms may be potentially revealing. Unsupervised relation mining is also important for many downstream NLP tasks such as automatically building of knowledge graphs. Knowledge bases like WordNet [5] are manually annotated and are used in variety of tasks such as question answering (Hao et al., 2017) and entity and event extraction (Yang and Mitchell, 2019). Recently, Chiang et al. (2020) demonstrated a robustness to the semantic structure of Word2Vec embeddings, showing that relations could be learned even without examples of them having been observed in the training corpus.

5 Relation Embeddings

In this section, we introduce the relation embeddings that will be used for comparison in the unsupervised task of relation clustering.

[5] https://wordnet.princeton.edu

- **Contextual Word Vectors**: Contextualized word representations have recently shown significant improvements on various NLP tasks. We use BERT-Large (Devlin et al., 2018) pre-trained[6] on the Wikipedia corpus. It produces contextualized embeddings of size 1,024 for each input token. The vocabulary size is 30,522 words. To represent the relation between a pair of words using contextualized BERT embeddings, we use the following formulation:

  For a given pair of words, (w_1, w_2), we first collect all the sentences from the Wikipedia corpus containing occurrences of both words. Let S be the set of all such sentences. Let v_1 and v_2 be the word embeddings generated by BERT of words w_1 and w_2 with respect to a sentence $s \in S$. To generate word embedding of each token we first take an average of the embeddings for the top four layers of the BERT model and then again average over multiple occurrences of the words in the sentence s. Let $\overline{v_1}$ and $\overline{v_2}$ be the final embeddings of each of the two words. Then, relation vector r_{12} of the two words is given by:

$$\text{BERT-D (difference)} : r_{12} = \sum_S \frac{\overline{v_1} - \overline{v_2}}{|S|} \tag{1}$$

  This is similar to offsets in Word2Vec. The other form of a BERT relation vector that we use is:

$$\text{BERT-C (concatenation)} : r_{12} = \sum_S \frac{[\overline{v_1}; \overline{v_2}]}{|S|} \tag{2}$$

  where "[;]" denotes concatenation and "$|.|$" denotes cardinality of the set.

- **Non-Contextual Word Vectors**: We use FastText and the skip-gram model from Word2Vec to obtain non-contextual word vectors. Each of these models has a vocabulary size of 2,145,353 words and generates word embeddings of size 300. We train Word2Vec on the Wikipedia corpus provided by Camacho Collados et al. (2019) using the skip-gram algorithm. To train the model we use a window size of 5, negative sampling, and ignore the words with a frequency lower than 5. We use the FastText model provided in Camacho Collados et al. (2019). Given a word-pair (w_1, w_2), let (v_1, v_2) and (u_1, u_2) be word vectors generated by Word2Vec and FastText models respectively, then the offset that corresponds to the relation vector r_{12} between the words for the Word2Vec model is given by,

$$\text{Word2Vec} : r_{12} = v_1 - v_2 \tag{3}$$

  The offset using the FastText model is given by,

$$\text{FastText} : r_{12} = u_1 - u_2 \tag{4}$$

  We normalize Word2Vec and FastText offsets to unit vectors. We also experimented with normalizing BERT relation vectors but found it did not lead to improvements.

- **Explicit Relation Embeddings**: One of the methods that we use for comparison, *RelPair* (Camacho Collados et al., 2019), is a state-of-the-art method that explicitly calculates relation vectors using a latent variable model. Like many other relation embedding models (Joshi et al., 2018; Washio and Kato, 2018; Espinosa-Anke and Schockaert, 2018), their hypothesis is that the relationship between two words can be characterized by the distribution of words between them in sentences they both appear in. Though this is an unsupervised method to learn relation vectors, they report their performance on supervised tasks. Let the relation vector for two words w_1 and w_2 denoted by r_{12}. We generate the embedding r_{12} by passing the ordered pair of (w_1, w_2) through a pre-trained RelPair model (denoted by "RelPair-model")[7]. This model has a vocabulary size of 1,138,119 pairs and

[6]Pre-trained model used from `https://github.com/Google-research/bert`

[7]Pretrained model used from `https://github.com/pedrada88/relative`

is trained on the Wikipedia corpus. It generates relation embeddings of size 300 given an input word-pair.

$$\text{RelPair} : r_{12} = \text{RelPair-model}(w_1, w_2) \tag{5}$$

6 Experiments

We explore the amenability of various neural word embedding spaces to be clustered using classical methods. We evaluate the homogeneity of clusters in the word-pair (i.e., vector offset) space with respect to a common lexical relation and evaluate the validity of new word-pairs found in the clusters of known relations.

In section 6.1 we will compare the results of unsupervised clustering of the different relation embeddings discussed above. Then in section 6.2 we will evaluate the best performing method from section 6.1 on the task of discovering new examples of known relations.

6.1 Unsupervised Clustering of Relations

Relation clusters mined in an unsupervised way can give insights into prominent lexical relations in a corpus and the typicality of concepts in a language, of potential cognitive and sociological interest. In this section, we perform experiments to evaluate Word2Vec, FastText, BERT, and RelPair relation vectors on this task. We will also study their more practical suitability by analysing their clustering performance in an extended offset validation space producing noisy, dense word-pairs. In section , we further extend the space past the validation dataset vocabulary.

We use three types of clustering algorithms: K-Means clustering (Kanungo et al., 2002), spectral clustering (Zelnik-Manor and Perona, 2005) and hierarchical agglomerative clustering (HAC). The K-Means algorithm is simple and easily scalable. For initializing clusters we use the K-Means++ algorithm given in Arthur and Vassilvitskii (2006) and minimize the euclidean distance between centroids and the relation embeddings. In spectral clustering, we construct the affinity matrix by computing a graph of nearest neighbors and generate a low-dimension embedding using eigenvalue decomposition[8]. For hierarchical clustering, we use the agglomerative (bottom-up) approach with complete linkage that maximizes euclidean distances between all relation embeddings of any two clusters. We have also repeated the experiments for HAC with cosine distance and complete linkage, whose details and results are given in Appendix B.

For measuring clustering performance we use three metrics: Homogeneity (H score), Completeness (C score), V-measure (V score), and Silhouette (S score) score. High homogeneity score indicates that each cluster contains only word-pairs from a single relation. High completeness score indicates that all word-pairs from a single relation are assigned to the same cluster. V-measure is the harmonic mean of homogeneity and completeness scores. While ground truth labels are required to calculate homogeneity, completeness, and V-measure, silhouette scores are calculated without relation labels for the word-pairs. Let d_2 be the mean distance between a word-pair and all other word-pairs in the same cluster and let d_1 be the mean distance between a word-pair and all other word-pairs in the next nearest cluster. The silhouette score s for a single sample is then given as:

$$s = \frac{d_1 - d_2}{max(d_1, d_2)} \tag{6}$$

6.1.1 Results

Since each of the relation embeddings has a different vocabulary size, we first filter the vocabulary to be only the words common among all of the trained models as well as present in the validation datasets.

Table 1 summarizes clustering scores of the relation vectors generated by Word2Vec, FastText, BERT, and RelPair relation vectors on the Google and DiffVec datasets. It shows that Word2Vec offsets outperform the other embeddings with respect to the ground truth labels in the K-Means and HAC algorithms.

[8] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html

	Google Dataset				DiffVec Dataset			
	K-Means Clustering							
	H score	**C score**	**V score**	**S score**	**H score**	**C score**	**V score**	**S score**
RelPair	0.68056	0.59900	0.63718	0.11286	0.29362	0.18716	0.22860	**0.058252**
BERT-C	0.65224	0.51755	0.57714	0.048046	0.19939	0.13661	0.16213	1.33e-06
BERT-D	0.71872	0.57700	0.64011	0.10746	0.18066	0.11543	0.14086	0.024457
Word2Vec	0.88053	**0.77466**	**0.82421**	**0.21603**	**0.48385**	**0.31737**	**0.38332**	0.031839
FastText	**0.88917**	0.70764	0.78809	0.11635	0.45015	0.28728	0.35073	0.030446
	Spectral Clustering							
RelPair	0.6480	0.50430	0.56722	0.042155	0.2749	0.18991	0.2246	**0.045548**
BERT-C	0.6802	0.53339	0.59793	0.0475	0.18287	0.14062	0.15899	0.029026
BERT-D	0.70166	0.56886	0.62832	**0.10423**	0.1510	0.11188	0.12853	0.016731
Word2Vec	0.84318	0.66605	0.74422	0.097274	0.41746	0.3102	0.3559	0.019403
FastText	**0.87754**	**0.69579**	**0.77616**	0.06648	**0.47731**	**0.32279**	**0.38513**	0.02221
	Hierarchical Clustering							
RelPair	0.48506	0.61964	0.54415	0.12495	0.13809	0.14119	0.13962	**0.01977**
BERT-C	0.56393	0.47812	0.51749	0.079464	0.092918	0.1218	0.10541	0.0093276
BERT-D	0.56310	0.526493	0.54418	0.15805	0.16256	0.10543	0.12790	0.0050169
Word2Vec	**0.93608**	**0.90012**	**0.91775**	**0.22690**	**0.39829**	**0.25418**	**0.3103**	0.01018
FastText	0.89452	0.8067	0.84835	0.17006	0.36396	0.22984	0.28175	0.010275

Table 1: Comparison between Word2Vec, FastText, BERT, and RelPair relation vectors. Left: Using Google dataset where the number of clusters is 13 and word-pairs is 207. Right: Using the DiffVec dataset where number of clusters is 34 and word-pairs is 1,512.

When considering the average of all four scores, Word2Vec's average for HAC is 9.5% higher than that for K-Means while clustering the Google dataset. On the other hand, Word2Vec's average score for HAC is 20% lower than that for K-Means on the DiffVec dataset. This suggests that HAC marginally improves the clustering score over K-Means when clustering pairs from the Google dataset but K-Means significantly improves these scores over HAC when clustering pairs from the DiffVec dataset. Thus, K-Means has an advantage over HAC when using the Word2Vec embeddings for clustering relations. For this reason, we proceed using K-Means while analyzing relation clustering in a dense embedding space.

Another advantage of using K-Means clustering for dense space word-pairs is scalability. The standard K-Means clustering algorithm has complexity $O(n^2)$ and that of standard spectral and hierarchical clustering is $O(n^3)$. This shows that it is easier to scale the K-Means algorithm to large datasets. For the next set of experiments where we cluster hundreds of thousands of word-pairs we use K-Means clustering.

We notice that the RelPair model has the smallest vocabulary overlap with both the DiffVec and Google datasets. This limits the size of the vocabulary used in the above experiment. Hence, we also compare Word2Vec and FastText with BERT and with RelPair separately. Details of these experiments are given in Appendix A in the supplementary material, respectively. We find that Word2Vec consistently outperforms both BERT and RelPair for all three clustering algorithms and for both evaluation datasets, with an exception of RelPair having a higher silhouette score on the DiffVec dataset.

Thus far, we have analyzed the performance of various relation embeddings on the word-pairs that are known to be part of a relation as per the two validation datasets. However, in practice, while performing clustering, many noisy word-pair offsets may appear which may not be part of any relation. We will now evaluate the clustering performance of the relation embeddings in the presence of such noisy pairs.

A simple method to determine if a word-pair is unrelated is if the two words do not co-appear in any sentence. Methods such as (Espinosa-Anke and Schockaert, 2018) use pointwise mutual information to determine the relatedness of word-pairs. However, this can hurt low-frequency word-pairs, so we proceed by filtering out the word-pairs that do not occur together in any sentence in the training corpus. This also ensures that the BERT relation embedding exists for the remaining pairs. We do not include RelPair vectors in the following analysis since the model has very limited vocabulary size and does not allow for word-pairs that aren't in its vocabulary. The BERT model allows for word-pairs as long as the two words appear in the same sentence, while the Word2Vec and FastText models allow us to calculate offsets between any two words as long as each word is in the model vocabulary.

We expand our word-pairs to include additional noisy pairs by considering all possible pairwise com-

binations of the words in each of the Google and DiffVec datasets separately. We call these sets of word-pairs the "extended datasets". We then cluster them into *2n+1* clusters, where *n* is the number of original ground truth relations (13 for the Google dataset and 36 for the DiffVec dataset). The additional one cluster represents all the word-pairs expected to be noise which should not represent any relation. The additional *n* represents the ground truth relations in the opposite direction, e.g. the relation "plural-to-singular" for offset of reversed word-pair becomes "singular-to-plural". Out of 104M sentences from the Wikipedia corpus, we found 18.48M sentences containing at least one pair from the extended Google dataset and 10.45M sentences from the extended DiffVec dataset. In these experiments, we use mini-batch K-Means (Sculley, 2010) which converges faster than standard K-Means but has only slightly worse performance. We use initialization similar to K-Means++ and a batch size of 10,000.

Table 2 shows the performance of the four methods on the Google and the DiffVec datasets. These results show that Word2Vec is the highest performer among all embeddings and across all metrics with the exception of silhouette score on the DiffVec extended dataset, where it places a close second. We find all four scores for all of the methods to be low compared to the scores from the previous experiments on the non-extended datasets. This is likely due to treating the noisy, unrelated word-pairs as a single label and expecting them to cluster as such. Though this seems like an impossible problem, we can easily imagine coming across this task in a real-life application.

To estimate the true number of clusters in these extended datasets without supervision, we perform mini-batch K-Means over a range of numbers of clusters and calculate the silhouette score as shown in Figure 2 and Figure 3. This approach assumes that silhouette score will correlate with ground truth metrics, as it did to a moderate degree in the previous result tables. We find the number of clusters with maximum silhouette score is 1,000 and 5,000 for Google and DiffVec datasets, respectively. In Figure 2, a spike in silhouette score can be observed when the number of clusters assumed is 20, which is very close to 27, the value used in the experiments summarized in Table 2. In Figure 3 no similar spikes can be observed. We also report V-measure in this experiment, assuming 2n+1 labels as explained above. This is an estimation and may not be the set of true ground truth labels. Details of these experiments are given in Appendix C in the supplementary material.

	Google Dataset				**DiffVec Dataset**			
	Mini-Batch K-Means Clustering							
	H score	**C score**	**V score**	**S score**	**H score**	**C score**	**V score**	**S score**
BERT-C	0.2119	0.00141	0.00281	-0.0242	0.08191	0.000351	0.000699	-0.08834
BERT-D	0.1730	0.00115	0.00229	-0.00693	0.03457	0.000137	0.000273	-0.01514
Word2Vec	**0.2372**	**0.00159**	**0.00316**	**0.0320**	**0.1053**	**0.000413**	**0.000824**	0.00995
FastText	0.1982	0.00132	0.00262	0.00065	0.0868	0.000341	0.000679	**0.01086**

Table 2: Left: Using extended Google dataset where number of clusters is 27 and word-pairs is 369,688. Right: Using extended DiffVec dataset where number of clusters is 73 and word-pairs is 9,112,224.

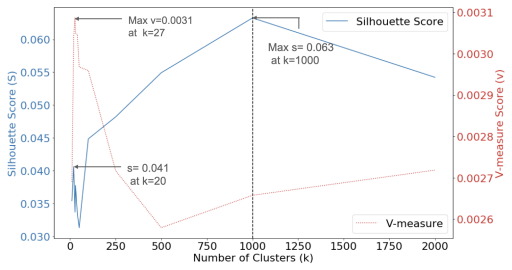

Figure 2: Mini-Batch K-Means clustering on the extended Google dataset.

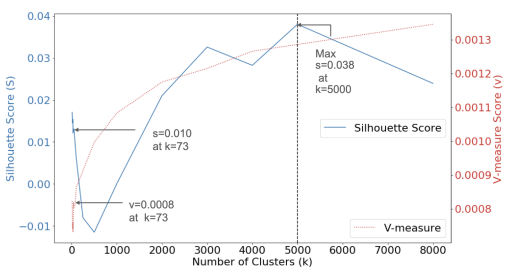

Figure 3: Mini-Batch K-Means clustering on the extended DiffVec dataset.

6.2 Discovering New Examples of Known Relations

In this section, we study if new examples of known lexical relations can be discovered in a further expanded Word2Vec offsets space. This task is ideally performed on the full set of relation vector offsets corresponding to the complete enumeration of all possible ordered word-pairs, which grows exponentially with the size of the vocabulary. For tractability reasons, we use pre-trained Google-News Word2Vec vectors with the Google dataset and limit the size of the vocabulary to the union of the top 10,000 most frequent terms in the Google-News corpus and the 905 unique words which appear in the Google dataset, yielding a set of $N = 10,354$ unique words. This vocabulary generates 107,194,962, or $2 \times \binom{N}{2}$, vector offsets: a dense collection which we call an open-world set.

To tractably expand the offset space, we define 13 hypercones (Figure 4) in the space based around the centroids of ground truth vector offsets representing each of the 13 Google dataset relations. These centroids were used instead of arbitrary points in the space in order to retain a degree of ground truth examples in the open-world that can be used to guide the discovery of new examples. For tractability, we restrict the size of each hypercone by considering only the 2,000 vector offsets closest (using cosine-similarity) to each of the 13 centroids (totalling 26,000 vectors). For each of the 13 hypercones, we perform unsupervised k-search (Zelnik-Manor and Perona, 2005) to yield a clustering of the 2,000 vector offsets. For inspection, we narrow our analysis in the open-world to one cluster in each of the 13 hypercones–the cluster with the highest number of labeled word-pairs.

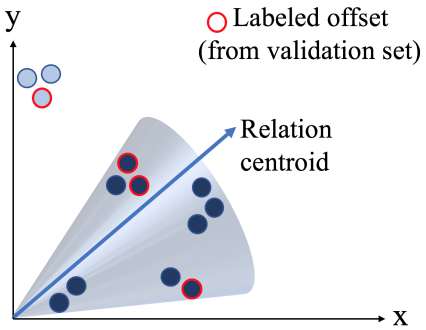

Figure 4: A simplistic illustration of a hypercone in the open-world.

Relation	# Pairs in Cluster	# Pairs from Valid.	Recall
adj-to-adv	18	2	0.063
capital-world	20	15	0.129
city-in-state	21	11	0.162
comparative	30	10	0.270
currency	11	3	0.100
gender	12	5	0.217
nationality	27	8	0.195
opposite	18	5	0.172
past-tense	77	12	0.300
plural	12	5	0.135
plural-verbs	13	4	0.133
pres.-participle	385	17	0.515
superlative	30	13	0.382

Table 3: A summary of the 13 open-world clusters (one per hypercone subspace). For each cluster, the total number of word-pairs, the number of pairs which also appear in the validation set, and the recall (with respect to the validation set) are listed.

6.2.1 Results

Table 3 summarizes the 13 open-world clusters by the number of word-pairs in the cluster, the number of those pairs which appear in the corresponding validation set grouping, and the percentage of the validation set pairs contained within the cluster. The "gender" cluster, for example, contains 12 word-pairs, five of which are "gender" pairs from the validation set. These five constitute 21.7% of the original 23 "gender" pairs which appear in the validation set.

We inspect the unlabeled word-pairs (not in the validation set) in each cluster to investigate if the clustering methodology has led to discovery of legitimate new examples of the relations. We manually label the unlabeled word-pairs in each cluster as relevant or not relevant to the relation. We report the precision among unlabeled word-pairs, along with a random sampling of five pairs from each cluster in Table 4. Among others, examples of relevant new pairs are found in the "adj-to-adv" cluster, such as (*adequate*, *adequately*); the "capital-world" cluster, such as (*Pyongyang*, *North_Korea*); and the "nationality" cluster, such as (*Belgium*, *Belgian*).

Of highest precision is the "gender" cluster, containing 100% relevant word-pairs to the relation. This resonates with "gender" as among the higher performing relations in analogy completion (Finley et al.,

Relation	Precision	Unlabeled word-pairs
adj-to-adv	0.125	adequate:properly, fair:promptly, **proper:properly**, **adequate:adequately**, consistent:properly
capital-world	0.200	**Pyongyang:NorthKorea**, Asmara:Ethiopia, Karachi:Pakistan, Anchorage:Alaska, Mogadishu:Ethiopia
city-in-state	0.600	**Raleigh:NorthCarolina**, **LasVegas:Nevada**, Libreville:Louisiana, **Boulder:Colorado**, Calif.:California
comparative	0.200	young:older, tiny:larger, strength:stronger, **strong:stronger**, tiny:bigger
currency	0.250	Finland:krone, Denmark:krona, **Norway:krone**, Sweden:krone, **Iceland:krona**
gender	1.000	**spokesman:spokeswoman**, **actor:actress**, **Mr.:Ms.**, **boys:girls**, **Mr:Ms**
nationality	0.105	**CzechRepublic:Czech**, Iceland:Norwegian, Denmark:Norwegian, **Belgium:Belgian**, Denmark:Swedish
opposite	0.769	great:unbelievable, **able:unable**, **well:poorly**, adequate:insufficient, **strong:weak**
past-tense	0.277	**going:went**, coming:went, running:came, **eating:ate**, running:raced
plural	0.000	pineapple:melons, cake:melons, banana:melons, banana:pineapples, pineapple:melons
plural-verbs	0.889	**drive:drives**, **pick:picks**, **launch:launches**, play:performs, **drop:drops**
pres.-participle	0.525	**take:taking**, pull:pulled, pass:getting, **deliver:delivering**, **predict:predicting**
superlative	0.412	dry:coldest, tiny:smallest, wealthy:richest, **busy:busiest**, **poor:poorest**

Table 4: An inspection of the unlabeled word-pairs in the 13 open-world clusters. Precision is computed after manually labeling every word-pair in each cluster as relevant or not to the respective relation. Five randomly selected unlabeled word-pairs are listed, with the pairs labeled as relevant in bold.

2017). "Plural-verbs" and "opposite" are also high in precision with 88.9% and 76.9% new word-pair relevance, respectively. Additionally, the majority (52.5%) of the 368 unlabeled word-pairs from the "pres.-participle" cluster are relevant to the relation. The lowest precision is found in the "nationality" cluster (10.5%) and in the "plural" cluster, of which 0% of the unlabeled word-pairs are relevant. In the "plural" cluster, three out of five of the examples in the table show *melon* as *word2* of the new pairs. This is a likely case of "default-behavior errors," as introduced by (Levy and Goldberg, 2014), which refers to the incorrect completion of a set of analogies by one specific word, which is described as the "prototypical" dominant word of the *word2*s (e.g. *melon* being the prototypical plural noun).

Aside from the unlabeled word-pairs marked relevant as a result of matching the relations perfectly, there are also some unlabeled pairs that contain a synonym or antonym of one of the words which would appear in the "perfect" pair. For example, the "opposite" cluster contains the near-perfect pair (*adequate*, *insufficient*), which is synonymous with (*sufficient*, *insufficient*). Furthermore, the "comparative" cluster contains (*small*, *larger*), where *small* is the antonym of the ideal *word1*, *large*. The vector offsets of these word-pairs may cluster tightly because synonyms and antonyms tend to have high cosine-similarity in word embeddings due to their use in similar contexts in language (Adel and Schütze, 2014).

7 Conclusions

We explored the amenability of clustering lexical relations in various neural embeddings. On the task of unsupervised clustering of known examples, our experiments showed that baseline non-contextual models (i.e., Word2Vec and FastText) outperformed the relation vectors derived from the contextual model, BERT. Among the clustering methods of spectral, K-Means, and Hierarchical, K-Means yielded the highest supervised and unsupervised scores on both DiffVec and Google validation datasets. Word2Vec with K-Means was best able to cluster the offset vectors into homogeneous clusters with respect to the labeled relations, a trend that continued when expanding the task to an extended, noisy set of word-pairs. In both spaces, the unsupervised metric of silhouette score largely correlated with supervised metrics. This allowed for the possibility of estimating the true number of lexical relations (i.e., clusters), which we attempted in an experiment with Mini-Batch K-Means. The results showed that the maximum silhouette score did not reliably correspond to the estimated true number of clusters; however, the Google dataset showed promise in providing a cluster number (20) close to the true number (27) based on the first silhouette score spike.

On the second task of discovering new relation examples in a space of over 100M word-pairs, results varied greatly by relation. Thirteen subspaces of offsets were created centered around the known lexical relation examples. After clustering within each of these subspaces, the best performing cluster within

each subspace was able to capture existing ground truth examples with recall ranging from 0.063 (adj-to-adv) to 0.515 (pres.-participle). When evaluating new examples collected in these top performing clusters, precision ranged from 0 (plural) to 1 (gender).

Our results suggest that linear, non-contextual embeddings have an advantage over contextual embeddings on these lexical relationship mining tasks when using classical clustering techniques, but that modeling improvements are necessary to reduce noise to a level of practical utility in noisy real-world scenarios. These improvements may come from developing work exploring linearities in contextual model embeddings (Reif et al., 2019; Jawahar et al., 2019) and would have the potential to identify emergent semantic regularities in large corpora.

References

Heike Adel and Hinrich Schütze. 2014. Using mined coreference chains as a resource for a semantic task. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1447–1452.

David Arthur and Sergei Vassilvitskii. 2006. k-means++: The advantages of careful seeding. Technical report, Stanford.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jose Camacho Collados, Luis Espinosa-Anke, Shoaib Jameel, and Steven Schockaert. 2019. A latent variable model for learning distributional relation vectors.

H.-Y. Chiang, J. Camcho-Collados, and Z.A. Pardos. 2020. Understanding the source of semantic regularities in word embeddings.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Luis Espinosa-Anke and Steven Schockaert. 2018. Seven: Augmenting word embeddings with unsupervised relation vectors. *arXiv preprint arXiv:1808.06068*.

Gregory Finley, Stephanie Farmer, and Serguei Pakhomov. 2017. What analogies reveal about word vectors and their compositionality. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pages 1–11, 01.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231.

Mohammad Jameel, Zied Bouraoui, and Steven Schockaert. 2018. Unsupervised learning of distributional relation vectors.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language?

Mandar Joshi, Eunsol Choi, Omer Levy, Daniel S Weld, and Luke Zettlemoyer. 2018. pair2vec: Compositional word-pair embeddings for cross-sentence inference. *arXiv preprint arXiv:1810.08854*.

Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. 2002. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the eighteenth conference on computational natural language learning*, pages 171–180.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.

Yannis Papanikolaou, Ian Roberts, and Andrea Pierleoni. 2019. Deep bidirectional transformers for relation extraction without supervision. *arXiv preprint arXiv:1911.00313*.

Zachary A. Pardos and Andrew J. H. Nam. 2020. A university map of course knowledge. *PLoS ONE*, 15(9).

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In *Advances in Neural Information Processing Systems*, pages 8594–8603.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394. ACM.

David Sculley. 2010. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2016. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1671–1682.

Koki Washio and Tsuneaki Kato. 2018. Neural latent relational analysis to capture lexical semantic relations in a vector space. *arXiv preprint arXiv:1809.03401*.

Shanchan Wu and Yifan He. 2019. Enriching pre-trained language model with entity information for relation classification. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2361–2364.

Bishan Yang and Tom Mitchell. 2019. Leveraging knowledge bases in lstms for improving machine reading. *arXiv preprint arXiv:1902.09091*.

Lihi Zelnik-Manor and Pietro Perona. 2005. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608.

A Additional Results of Clustering Experiments

A.1 Comparing Word2Vec, BERT-C, BERT-D and FastText

Table 5 compares the clustering of relation vectors generated by Word2Vec, FastText and BERT on the Google and DiffVec datasets. For experiments with the Google dataset, we have 387 word-pairs common to all model vocabularies consisting of 13 relations and from the DiffVec dataset, we have 7,782 word-pairs consisting of 36 relations. These word-pairs are common among the models and datasets. It can be observed that Word2Vec offsets outperform BERT-C and BERT-D offsets in all the experiments.

	Google Dataset				**DiffVec Dataset**			
	K-Means Clustering							
	H score	**C score**	**V score**	**S score**	**H score**	**C score**	**V score**	**S score**
BERT-C	0.64199	0.6438	0.6429	0.045362	0.16965	0.094723	0.12157	-0.0283
BERT-D	0.65349	0.63626	0.64476	0.10506	0.10774	0.058451	0.075787	0.013537
Word2Vec	0.77054	0.75713	0.7637	**0.12684**	0.46482	0.25064	0.32567	0.042768
FastText	**0.82701**	**0.84071**	**0.83380**	0.078917	**0.47010**	**0.26855**	**0.3358**	**0.0467**
	Spectral Clustering							
BERT-C	0.61109	0.6299	0.62035	0.035247	0.11650	0.10530	0.11062	-0.00111
BERT-D	0.6750	0.66663	0.67083	0.10153	0.067005	0.062730	0.064797	-0.0144
Word2Vec	0.76312	0.7683	0.7657	**0.12939**	0.33523	**0.26321**	**0.2948**	-0.0153
FastText	**0.79287**	**0.7883**	**0.79059**	0.055707	**0.34117**	0.24842	0.28750	**0.02019**
	Hierarchical Clustering							
BERT-C	0.44533	0.61259	0.51574	0.094548	0.051867	0.067373	0.058612	-0.0125
BERT-D	0.52488	0.53252	0.52867	0.095725	0.074991	0.040040	0.052206	-0.00904
Word2Vec	**0.75826**	**0.7643**5	**0.7648**	**0.1253**	**0.36074**	**0.20077**	**0.28776**	**0.0101**
FastText	0.69180	0.69768	0.69473	0.085226	0.29674	0.15846	0.20660	0.005262

Table 5: Comparison between Word2Vec, FastText and BERT relation vectors. Left: Using the Google dataset, where number of clusters is 13 and word-pairs is 387. Right: Using DiffVec dataset where number of clusters is 36 and word-pairs is 7,782.

A.2 Comparing Word2Vec, RelPair, and FastText

Tables 6 compares the clustering of relation vectors generated by Word2Vec, FastText, and RelPair on the Google and DiffVec datasets. For experiments with the Google dataset, we have 272 word-pairs common to all model vocabularies consisting of 11 relations. Similarly for the DiffVec dataset, we have 1,889 word-pairs consisting of 34 relations.

	Google Dataset				**DiffVec Dataset**			
	K-Means Clustering							
	H score	**C score**	**V score**	**S score**	**H score**	**C score**	**V score**	**S score**
Relpair	0.7718	0.6373	0.6981	0.09062	0.2789	0.1750	0.2151	**0.04085**
Word2Vec	**0.8989**	**0.6534**	**0.7568**	**0.1222**	**0.4901**	**0.3047**	**0.37583**	0.02583
FastText	0.8957	0.6441	0.7493	0.09928	0.4673	0.2927	0.3599	0.02629
	Spectral Clustering							
Relpair	0.680	0.4836	0.5655	0.03209	0.2577	0.1806	0.2124	**0.02332**
Word2Vec	**0.8792**	**0.6365**	**0.7384**	**0.1214**	**0.4354**	**0.3215**	**0.3699**	0.0162
FastText	0.8494	0.6350	0.7267	0.0466	0.4277	0.3062	0.3569	0.01749
	Hierarchical Clustering							
Relpair	0.6385	0.6981	0.6670	0.1019	0.1426	0.1436	0.1431	**0.01463**
Word2Vec	0.9467	**0.9007**	**0.9231**	**0.2009**	**0.4241**	**0.2672**	**0.3278**	0.01409
FastText	**0.9475**	0.7975	0.8661	0.1031	0.3584	0.2235	0.2753	0.00650

Table 6: Comparison between Word2Vec, FastText and RelPair. Left: Using Google dataset where number of clusters is 11 and word-pairs is 272. Right: Using DiffVec dataset where number of clusters is 34 and word-pairs is 1,889.

B Hierarchical Clustering Experiments

We have repeated the experiments for HAC with cosine distance and complete linkage. Results of these experiments are given in Table 7. We can see that Word2Vec offsets outperform BERT-C, BERT-D and

RelPair in all the three experiments.

	Experiment 1							
	H score	**C score**	**V score**	**S score**	**H score**	**C score**	**V score**	**S score**
Word2vec	**0.758267**	**0.764359**	**0.761301**	**0.125302**	**0.360746**	**0.200775**	**0.257973**	0.0102531
FastText	0.691806	0.697683	0.694732	0.0852269	0.296747	0.158467	0.206605	-0.00318162
BERT-C	0.445339	0.612599	0.515747	0.0945483	0.0332689	0.0532383	0.0409487	**0.0225003**
BERT-D	0.524882	0.532527	0.528677	0.0957254	0.0749914	0.0400409	0.0522066	-0.0128421
	Experiment 2							
Word2vec	0.946708	**0.900798**	**0.923183**	**0.200932**	**0.424111**	**0.267235**	**0.327874**	0.0129567
FastText	**0.947553**	0.797549	0.866104	0.103135	0.358428	0.223505	0.275325	0.00535484
RelPair	0.63855	0.698174	0.667033	0.101908	0.142603	0.143651	0.143125	**0.0146898**
	Experiment 3							
Word2vec	**0.936082**	**0.900127**	**0.917753**	**0.226901**	**0.398299**	**0.254189**	**0.31033**	0.0118096
FastText	0.894529	0.80671	0.848353	0.170067	0.363967	0.229845	0.281759	0.00965932
BERT-C	0.563931	0.478125	0.517496	0.0794643	0.0929183	0.1218	0.105417	0.00832806
BERT-D	0.563107	0.526493	0.544185	0.158053	0.162561	0.105435	0.127909	0.00318062
RelPair	0.485066	0.619641	0.544157	0.124935	0.138093	0.141195	0.139626	**0.0165833**

Table 7: Results on hierarchical clustering with cosine distance and complete linkage. Left: Using Google dataset. Right: Using DiffVec dataset

Details of these experiments are:

- Experiment 1: Comparison between Word2Vec, FastText and BERT offsets. Using the Google dataset where the number of clusters is 13 and word-pairs is 387. Using the DiffVec dataset where the number of clusters is 36 and word-pairs is 7782.
- Experiment 2: Comparison between Word2Vec, FastText, and RelPair. Using the Google dataset where the number of clusters is 11 and word-pairs is 272. Using DiffVec dataset where the number of clusters is 34 and word-pairs is 1,889.
- Experiment 3:Comparison between Word2Vec, FastText, BERT, and RelPair. Using the Google dataset where the number of clusters is 13 and word-pairs is 207. Using DiffVec dataset where the number of clusters is 34 and word-pairs is 1,512.

C Unsupervised Experiments with Noisy Data

We perform grid search on the number of clusters to find the one with the maximum silhouette score. The grid ranges are [10, 20, 27,30, 40, 50, 100, 250, 500, 1000, 2000] for extended Google dataset and [10, 20,30, 40, 50, 73,100, 250, 500, 1000, 2000,3000,4000,5000,8000] for the extended DiffVec dataset.