Dual Representations of Words with Asymmetric Contexts

KC Tsiolis

Friday August 23, 2019

Abstract

Typically, pre-trained word embeddings are trained with cooccurrence statistics from symmetric context windows around a focal word. Furthermore, it is standard practice to keep only a single representation for each word after training even though separate context vectors are trained. In this work, we show how the use of two vector representations per word allows for the modelling of asymmetric relationships between words, such as ordering and dependency. We also investigate the necessity of training two vector representations of words during the training process.

1 Introduction

Recently, Newell et al. (2019) discovered shared features among some of the most widespread pre-trained word embedding algorithms, including word2vec (Mikolov et al., 2013b), GloVe (Pennington et al., 2014), Swivel (Arora et al., 2015), FastText (Joulin et al., 2016), and the "latent discourse space" model (LDS) Shazeer et al. (2016). One of the commonalities that the authors find is bilinear parametrization of PMI. The latter can be calculated from corpus statistics, while the former comes from inner products between learned word representations. Two word representations are used: context vectors ("covectors") and word vectors. Covectors are typically discarded after training is complete or they are averaged with the word vectors. In downstream applications of word embeddings, the inner product between covector and vector is seldom exploited, in spite of the fact that it measures meaningful information about word cooccurrence by approximating PMI. The authors then derive their own embedder (Hilbert-MLE) based solely on the shared principles in the above mentioned algorithms. It achieves competitive performance with SGNS (Skip-gram with Negative Sampling, a form of word2vec) and GloVe in similarity, analogy and downstream tasks.

In this work, we further investigate the properties of pre-trained word embeddings by making use of Hilbert-MLE. We first introduce modifications to the extraction of cooccurrence

statistics which are fed into Hilbert-MLE by experimenting with different definitions of cooccurrence. These include asymmetric notions of context such as context windows on only one side of the focal word, as well as dependency-based contexts. We quantitatively and qualitatively measure the changes in the resulting embeddings from the case when a symmetric window-based notion of context is used. While we observe small changes in similarity, analogy, and downstream performance, perhaps the most informative result is the change in how covectors and vectors interact. Their inner product is meant to emphasize cooccurrence, and thus changes when the notion of cooccurrence changes. For instance, in the dependency-based case, we observe higher covector-vector inner products for words that appear more frequently (compared to under independence) as dependent-head pairs. Furthermore, the inner product captures the asymmetry inherent to the new contexts. In the dependency-based case, if the roles of head and dependent are switched in a word pair, their inner product also changes.

Secondly, we introduce modifications to the covector-vector inner product itself in an investigation of the necessity of using covectors when training word embeddings. We train a variant of Hilbert-MLE where covectors are excluded from the training process altogether and only vector-vector inner products are taken. We also experiment with replacing covectors by a linear map acting on vectors. The former variant is unable to achieve the same quantitative performance as with covectors, while the latter can do so only when the dimension of the embedding space is increased.

2 Related Work

2.1 Pre-Trained Word Embeddings

Pre-trained word embeddings, also known as distributed (or continuous) word representations allow for words in a corpus to be represented in a meaningful way that is emblematic of the syntactic and semantic similarities between words in a language.

To illustrate this, say we have a large corpus of text. We extract a set of the n most frequently occurring words in this corpus and call this our vocabulary $\mathcal{V}$. How can we represent these words in such a way that we can feed them into a neural network (or any other architecture taking feature vectors as input)? One choice to represent the words in our vocabulary would be to label each one with a unique one-hot feature vector in $\mathbb{R}^n$. However, this maps the vocabulary words to sparse vectors in a high-dimensional space. Furthermore, it does not capture any information about the meaning of the words. To resolve this, we define an embedder $V : \mathbb{R}^n \to \mathbb{R}^d$, where $d << n$, which maps the one-hot representations to dense representations in a low-dimensional space. Now, the set of word vectors is no longer orthogonal, so we can now consider nontrivial angles between elements of the set. A pair of word vectors with high cosine similarity can mean that the two words have a similar meaning. But what does it mean for two words to be similar? And how can this similarity be learned?

There is now a variety of architectures for producing pre-trained distributed/continuous representations of words (a.k.a. word embeddings). These include word2vec (Mikolov et al., 2013a), GloVe (Pennington et al., 2014), Swivel (Arora et al., 2015), FastText (Joulin et al., 2016), and the "latent discourse space" model (LDS) (Shazeer et al., 2016). All of these rely on the distributional hypothesis (Harris, 1954), which states that words that occur in similar contexts have similar meanings. They use word cooccurrence counts in fixed-size context windows as a way of quantifying this.

Consider the most popular form of word2vec, Skip-gram with Negative Sampling (SGNS) (Mikolov et al., 2013b), as an example. It relies on the notion of a context window (typically of size 5) that weighs the words on the left and right of a given focal word. Concretely, if we have a focal word w_0, we consider c "context" words to its left and to its right $\{w_{-c}, w_{-c+1}, ..., w_{-1}, w_1, ..., w_c\}$. Put one way, the objective of SGNS is to be able to predict the context words in the window given the focal word. This is done by training the skip-gram model to map words to a vector space where the word vector for the focal word has a high inner product with "context vectors" that represent each of the words in the window around the focal word.

Given a sequence of training words of length T from a corpus and a context window size of c, the Skip-gram objective is to maximize the following:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j} \mid w_t) \tag{1}$$

They parametrize the probability of a context word w_{t+j} given a focal word w_t as:

$$p(w_{t+j} \mid w_t) = \frac{\exp(v'^{T}_{w_{t+j}} v_{w_t})}{\sum_{w=1}^{n}\exp(v'^{T}_{w} v_{w_t})}, \tag{2}$$

where v_w and v'_w are the "input" and "output" representations of word w, respectively. It should be noted that because of this, two distinct vector representations are used for each word. The input representation represents a focal word, while the output representation represents context words. In applications, only one of the two representations is kept (or the two representations are averaged, as is the case for GloVe).

However, there is value in each of the two representations which are produced. In Equation (2), the inner product $v'^{T}_{w_{t+j}} v_{w_t}$ between the input representation of the focal word and the output representation of the context word measures how likely w_{t+j} is to occur in the context of w_t (more precisely, as we will see later, how likely it is to observe w_{t+j} in the context of w_t relative how likely it would be to see them together if they were independent). This is different from the inner product between two input (word) vectors ($v_i^T v_j$) or two output (context) vectors ($v_i'^T v'_j$), which has been empirically shown to represent how "similar" (as dictated by the distributional hypothesis) words i and j are to each other. If words i and j occur in the same contexts, then their word vectors are trained to be highly selective for

the same context vectors. The word vectors of i and j will then have high cosine similarity (compared to other word pairs where words are not as selective for the same contexts).

Another important observation from Equation (2) is that it is a softmax over the entire vocabulary. For larger vocabulary sizes, this becomes intractable. To counter this, Mikolov et al. (2013b) introduce "negative sampling" as a more efficient way of imposing the unitarity consraint of probability. First, a "positive sample" is drawn from the data distribution. In our case, we draw a word pair based on cooccurrence counts (how often two words appear and are less than c words apart). Then, k "negative samples" are selected from a noise distribution (in this case, we draw a single word from a variant of the unigram distribution and then consider the inner product of its context/output representation with the focal word). They replace Equation (2) with the following:

$$\log p(w_{t+j} \mid w_t) = \log \sigma(v'^{T}_{w_{t+j}} v_{w_t}) + \sum_{i=1}^{k} \mathop{\mathbb{E}}_{w_i \sim P_n(w)} \Big[\log \sigma(-v'^{T}_{w_i} v_{w_t}) \Big], \tag{3}$$

where the second term on the right hand side corresponds to the negative sampling term.

With negative sampling, the goal of maximizing the quantity in Equation (1) can now be interpreted as maximizing the log probability that the observed word-context pairs in the sequence do indeed come from the data distribution and that the negative samples do *not* come from the data distribution (Goldberg and Levy, 2014).

Interestingly, Mikolov et al. (2013b) also find it necessary to undersample frequent words in a corpus. Such words (like "the") are typically less informative and thus should not bear a stiff penalty for lack of fit that is proportional to the number of times they appear in the corpus. With this in mind, they modify negative sampling to draw from the unigram distribution raised to the 3/4rd power. They also discard words in the training set with a probability that increases as unigram frequency increases (see their paper for more details).

2.2 Hilbert-MLE

In a discussion of the properties of pre-trained word embeddings. Newell et al. (2019) show that the success of SGNS, GloVe, Swivel, FastText, and LDS can be attributed to two characteristics that they all share. At their core, these algorithms consist of a multinomial model of word cooccurrence and bilinear parametrization of PMI for word pairs. The authors show empirically that these are sufficient for a model to replicate the performance of SGNS and GloVe in similarity, analogy, and downstream tasks. They do this by deriving a canonical embedder, which they call Hilbert-MLE, based solely on the two principles outlined above. It does away with any superfluous design elements present in other models.

The authors first expand on the proof from Levy and Goldberg (2014b) that SGNS is implicitly a matrix factorization algorithm. It factorizes a $n \times n$ matrix M, where $M_{ij} = \mathrm{PMI}(i, j) - \ln k$, as the product of two $n \times n$ matrices W and V. These can be thought of as matrices of context vectors and word vectors, respectively. Here, $\mathrm{PMI}(i, j)$ is understood to

measure the deviation of the number of cooccurrences of words i and j from how often the two words would occur together under independence. It is defined by the following equation:

$$\text{PMI}(i, j) = \ln \frac{p_{ij}}{p_i p_j}, \tag{4}$$

where p_{ij} is the probability of words i and j cooccurring (with i as a context word of the focal word j), and p_i and p_j are the unigram probabilities of words i and j (i.e. how likely each word is to occur).

Similarly, they show that GloVe, FastText, and LDS also factorize matrices with entries corresponding to a variant of PMI. This result means that all of these algorithms are essentially learning a bilinear parametrization of PMI via context vectors (which the authors refer to as "covectors") and word vectors. This makes the importance of distinguishing between a word as a context word and a word as a focal word clear.

It will now be useful to introduce some notation. We will use the convention proposed by Newell et al. (2019) when discussing covectors and vectors. They denote the covector for the word i as $\langle i|$, the vector for the word j as $|j\rangle$, and their inner product as $\langle i|j\rangle$. $\langle i|$ can be viewed as a linear functional from the embedding space $\mathbb{R}^d$ (commonly $d = 300$) to $\mathbb{R}$. In specific terms, $\langle i|$ maps $|j\rangle$ to $\langle i|j\rangle$, which approximates a variant of $\text{PMI}(i, j)$. In the case of SGNS, we have that $\psi(i, j) = \langle i|j\rangle$ is a bilinear parametrization of $\phi(i, j) = \text{PMI}(i, j) - \ln k$. Once again, we understand that $\langle i|j\rangle$ is meant to designate cooccurrence of words i and j, while $|i\rangle \cdot |j\rangle$ is meant to designate the semantic/substitional similarity of words i and j (how well the words can replace each other).

With only the two aforementioned principles at their disposal, the authors proceed to derive their own embedder, Hilbert-MLE. They build a multinomial model of corpus cooccurrence statistics $\mathcal{D}$ given the sets of vectors and covectors:

$$P(\mathcal{D}|V, W) = N! \prod_{ij} \frac{\hat{p}_{ij}^{N_{ij}}}{N_{ij}!}, \tag{5}$$

where N_{ij} is the number of times words i and j cooccur (with i as the context word and j as the focal word), $N = \sum_{ij} N_{ij}$, and $\hat{p}_{ij}$ is the model's estimate of $p_{ij} = \frac{N_{ij}}{N}$. For all word pairs (i, j), they explicitly parametrize $\hat{\text{PMI}}(i, j)$ bilinearly as the inner product of a covector and a vector. Consequently, we have $\psi_{ij} = \langle i|j\rangle$ and $\phi_{ij} = \text{PMI}(i, j)$ for Hilbert-MLE. From that, the model estimate $\hat{p}_{ij}$ can be computed:

$$\langle i|j\rangle = \hat{\text{PMI}}(i, j) = \ln \frac{\hat{p}_{ij}}{p_i p_j} \tag{6}$$

$$\Rightarrow \hat{p}_{ij} = p_i p_j e^{\langle i|j\rangle} \tag{7}$$

We can also introduce bias terms which aid in modelling PMI. As discussed in Newell et al. (2019), the distribution of PMI across all word pairs in our corpus is approximately normal

with a mean near zero (in fact, the mean is slightly below zero). We can deploy the inner product $\langle i|j\rangle$ to model this distribution as if it was centred at zero, with the bias terms $b_{\langle i|}$ and $b_{|j\rangle}$ providing the necessary shift towards the true mean. Note that there are two bias terms for each word: one "covector" bias and one "vector bias". In our experiments, we consistently observe modest gains when using bias. Hence, we employ the following "inner product":

$$\psi_{ij} = \langle i \cdot j \rangle = \langle i|j\rangle + b_{\langle i|} + b_{|j\rangle} \tag{8}$$

Taking the negative log likelihood loss and employing a Lagrange multiplier to satisfy $\sum_{ij} \hat{p}_{ij} = 1$ (see paper for details), they arrive at the following loss function:

$$\mathcal{L} = \sum_{ij} \Big(p_{ij} \langle i \cdot j \rangle - p_i p_j e^{\langle i \cdot j \rangle} \Big) \tag{9}$$

Taking the partial derivative with respect to $\langle i \cdot j \rangle$, they obtain:

$$\frac{\partial \mathcal{L}}{\partial \langle i \cdot j \rangle} = p_i p_j \Big[e^{\mathrm{PMI}(i,j)} - e^{\langle i \cdot j \rangle} \Big] \tag{10}$$

Note that when this gradient is zero, we have $\langle i \cdot j \rangle = \mathrm{PMI}(i, j)$.

However, with this particular gradient, Hilbert-MLE suffers from the same problem as word2vec where frequent words are penalized harshly for lack of fit. This comes from the $p_i p_j$ term, which causes a significant imbalance between the contribution of frequent words and that of rare words to the gradient. To resolve this, they add a temperature hyperparameter τ to the gradient:

$$\frac{\partial \mathcal{L}}{\partial \langle i \cdot j \rangle} = (p_i p_j)^{\frac{1}{\tau}} \Big[e^{\mathrm{PMI}(i,j)} - e^{\langle i \cdot j \rangle} \Big] \tag{11}$$

With this approach, increased unigram frequency of words still results in increased contribution to the gradient, but this scales like the τth root function rather than linearly.

One issue with this "dense" version of Hilbert-MLE is that it requires a sum over all word pairs, which is $O(n^2)$. Furthermore, statistics for each word pair must be held in memory. This poses a problem for applications which require large vocabulary sizes in the hundreds of thousands of words. Thus, it is more practical to use a sample-based version of Hilbert-MLE.

The authors derive the sample-based form by considering Equation (9) as a difference of expectations:

$$\mathcal{L} = \mathop{\mathbb{E}}_{(i,j)\sim p_{ij}} \langle i \cdot j \rangle - \mathop{\mathbb{E}}_{i\sim p_i, j\sim p_j} e^{\langle i \cdot j \rangle} \tag{12}$$

This is very similar to negative sampling in SGNS. The positive term is an expectation over the data distribution, while the negative term is an expectation over a noise/unigram distribution. Unfortunately, this formulation has very high variance due to the exponential in the negative sampling term. The authors find the equivalent "balanced sampling" formulation to be far more effective (note that we can still apply the temperature from the dense version):

$$\mathcal{L} = \mathop{\mathbb{E}}_{i \sim p_i^{\frac{1}{T}}, j \sim p_j^{\frac{1}{T}}} \left[e^{\mathrm{PMI}(i,j)} \langle i \cdot j \rangle - e^{\langle i \cdot j \rangle} \right] \tag{13}$$

The authors find that both the dense and sample-based Hilbert-MLE perform on par with SGNS and GloVe on an array of similarity, analogy, and downstream tasks. We employ these same tasks in the evaluation of our embeddings.

2.3 Dependency Parsing

Grammar is what gives language structure by setting rules for how words are allowed to compose and form sentences. The dependency grammar focuses on the notion of a directed dependency relation between a pair of words (Jurafsky and Martin, 2019). For example, in the sentence "Sarah ate a red apple", there is a dependency relation between "red" and "apple", since the role of the adjective "red" in the sentence is to modify the noun "apple". Following the Stanford basic typed dependencies standard for annotation (de Marneffe and Manning, 2008), we call the type of relation between these two words *amod*, which stands for "adjectival modifier". We call "red" the modifier or dependent and "apple" the head. By convention, every dependency relation is designated by an arc from head to dependent. The collection of all words and arcs in a sentence forms a rooted dependency parse tree. For our example sentence, the parse would be:

Notice how the tree is rooted and has a special "ROOT" token which has an outgoing arc to one word of the sentence ("ate"). Typically, the word pointed to by the root is the verb in a verb phrase. Thus, we say that the head of "ate" is the root and the dependency relation involved is of type "root". In addition, each word in the tree has a single head. However, a head can have multiple dependents (as is the case for "ate"). Common dependency relations from the Stanford basic dependencies (aside from the ones already mentioned) include "noun subject" (*nsubj*), "direct object" (*dobj*), "conjunct" (*conj*), "determiner" (*det*), and "preposition" (*prep*).

In recent years, Universal Dependencies (Nivre et al., 2016) has attempted to unite as many languages as possible under the same dependency grammar framework. Its authors define an annotation scheme which is very similar to the Stanford dependencies scheme and that consists of 40 dependency relations (though not all languages use every single relation).

2.4 Dependency-based Word Embeddings

Levy and Goldberg (2014a) discuss how context window-based embeddings from SGNS do

not always capture the most relevant context of a focal word. The use of a fixed-size context window can dilute the most relevant context words with irrelevant information. As an example, they consider the sentence "Australian scientist discovers star with telescope". They note that while "scientist" and "star" are important contexts for "discovers" (being the subject and object of the latter verb), "Australian" is not. Furthermore, small context windows risk missing important contextual information, particularly in the case of dependency relations between words that are far apart from each other. To deal with these two problems, Levy and Goldberg use the dependents and modifiers (along with associated dependency labels) as contexts for each word. They learn context embeddings for all (context word, dependency label) tuples.

Interestingly, Levy and Goldberg find that the dependency-based word vectors exhibit more "functional similarity" (similarity) than window-based SGNS word vectors that exhibit "topical similarity" (relatedness). For instance, the dependency-based representation for "Florida" has the highest cosine similarity with other US states like Texas and Louisiana, while its window-based representation has the highest cosine similarity with cities in Florida, like Tampa and Jacksonville. As Levy and Goldberg put it, dependency-based embeddings find words that behave like the word in question, while window-based embeddings find words that associate with it. This pattern is confirmed by quantitative tests such as the WordSim353 dataset, which tests a model's ranking of similar pairs of words over related pairs of words, and vice versa.

3 Positionally Asymmetric Embeddings

Recall the notion of ψ_{ij} and ϕ_{ij} discussed in Section 2.2. ψ_{ij} is a function of the model parameters of a word embedder which bilinearly parametrizes ϕ_{ij}, some variant of $\text{PMI}(i, j)$. This notion provides the Hilbert embedder with quite a bit of flexibility, as we can view the partial derivative from Equation (11) as:

$$\frac{\partial \mathcal{L}}{\partial \psi_{ij}} = p_i p_j \Big[e^{\phi_{ij}} - e^{\psi_{ij}} \Big], \tag{14}$$

where $\psi_{ij} = \langle i \cdot j \rangle$ and $\phi_{ij} = \text{PMI}(i, j)$.

As in Section 2.2, note that when this partial derivative is zero, we have $\psi_{ij} = \phi_{ij}$.

This gives us the power to redefine the bilinear parametrization (by altering ψ_{ij} and ϕ_{ij}) without affecting the formulation of the loss.

Consider the definition of $\phi_{ij} = \text{PMI}(i, j)$ from Hilbert-MLE:

$$\text{PMI}(i, j) = \ln \frac{p_{ij}}{p_i p_j} = \ln \frac{\frac{N_{ij}}{N}}{p_i p_j} \tag{15}$$

N_{ij} is defined via cooccurrence counts from a training corpus. The most common definition of cooccurrence consists of fixing a focal word i and weighing the five words to its left and the five words to its right in a "context window". The most common distribution of weights is "dynamic window weighting", which attaches linearly decreasing weights of $[1, 0.8, 0.6, 0.4, 0.2]$ to cooccurrences with context words on each side of the focal word. More precisely, for a context word i that is m words to the left (or right) of a focal word j, a weight of $w_{ij} = 1 - \frac{1}{5}(m-1)$ is added to N_{ij}. Context words which are closer to the focal word get higher cooccurrence weights than context words which are farther from the focal word.

The above definition of N_{ij} is the basis for the embeddings that were trained in the original Hilbert-MLE paper. We name these "symmetric window-based embeddings". From it, we can derive the following property:

Property 1. *In the case of symmetric window-based embeddings, we have $N_{ij} = N_{ji}$ for all $(i, j) \in \{1, ..., n\} \times \{1, ..., n\}$.*

Proof. Let $(i, j) \in \{1, ..., n\} \times \{1, ..., n\}$. Consider the kth cooccurrence of i and j in the corpus (by this we mean that i is the context word, and j is the focal word). If j is m words to the left of i, where $1 \leq m \leq 5$, the weight of this cooccurrence is then $w_{ij}^{(k)} = 1 - \frac{1}{5}(m-1)$. But then i is m words to the right of j. Hence we have a cooccurrence of j and i with weight $w_{ji}^{(k)} = 1 - \frac{1}{5}(m-1) = w_{ij}^{(k)}$. A similar argument can be made if j is to the right of i.

Note that we can interchange the roles of i and j in the above, and so we have that i and j cooccur if and only if j and i cooccur. Combining this with the fact that $w_{ij}^{(k)} = w_{ji}^{(k)}$ for every cooccurrence k, we have that $N_{ij} = N_{ji}$. □

It immediately follows from this result that $\mathrm{PMI}(i, j) = \mathrm{PMI}(j, i)$ (and $\phi_{ij} = \phi_{ji}$). Hence, when the partial derivative in Equation (14) is equal to zero, we have that $\langle i \cdot j \rangle = \mathrm{PMI}(i, j) = \mathrm{PMI}(j, i) = \langle j \cdot i \rangle$. Due to the left-right symmetry in the context window that dictates N_{ij}, we observe a symmetry in covector-vector inner products.

We now observe that altering how N_{ij} is defined would alter $\mathrm{PMI}(i, j)$ and ϕ_{ij}. Doing this while maintaining $\psi_{ij} = \langle i \cdot j \rangle$ allows us to use covectors and vectors to model different notions of cooccurrence. In particular, we can consider cases where $N_{ij} = N_{ji}$ does not hold.

To construct such a case, it is sufficient to construct a context window where there is no longer an equivalence between word i cooccurring with word j and word j cooccurring with word i. We take a "positionally asymmetric window" with context words only on one side of the focal word (either on the left or on the right). For a focal word w_0, we take either $\{w_1, ..., w_5\}$ (right) or $\{w_{-5}, ..., w_{-1}\}$ (left) as its context window.

The use of a positionally asymmetric window as context allows us to explicitly take word ordering into account. For example, the words "red" and "apple" cooccur frequently, and in the majority of cases "red" will be to the left of "apple". Training embeddings with this notion of context will give us the ability to quantify the higher likelihood of "red" being to the left of "apple" than to the right.

To show that $N_{ij} = N_{ji}$ does not hold in this case, we provide a counterexample. Consider a toy corpus which contains only the sentence "The quick brown fox jumped over the lazy dog." When we view "jumped" is a focal word and we restrict ourselves to right contexts only, we have the cooccurrences ("jumped", "over"), ("jumped", "the"), ("jumped", "lazy"), and ("jumped", "dog"), with weights 1, 0.8, 0.6, and 0.4, respectively. However, when we view "over" as a focal word, there is no cooccurrence ("over", "jumped") with weight 1, since the only context words for "over" with a right window are the three words to its right. (Note that if we were using a symmetric context window, such a cooccurrence would exist.) And so, taking word i to be "jumped" and word j to be "over", we have $N_{ji} = 1$, but $N_{ij} = 0$.

The above implies that $\langle i \cdot j \rangle$ and $\langle j \cdot i \rangle$ are no longer necessarily approximately equal. This gives us the power to model both sides of the positional asymmetry with covectors and vectors. If we take right context windows, then $\text{PMI}(i, j)$ measures the deviation from independence of the likelihood of word i being to the right of word j. $\text{PMI}(j, i)$ measures the deviation from independence of the likelihood of word j being to the right of word i, which is the same as that of word i being to the left of word j. These quantities will be approximated by $\langle i \cdot j \rangle$ and $\langle j \cdot i \rangle$ respectively during training. A similar observation can be made about training with left context windows. Since training with either one of the context windows allows for this modelling of both sides of the asymmetry, it is redundant to train a set of embeddings with left contexts and a set of embeddings with right contexts.

We present both qualitative and quantitative results from our training of positionally asymmetric embeddings in Section 6.

4 Dependency-based Embeddings

Here, we expand on the work of Levy and Goldberg (2014a), by bringing our new perspective of explicit bilinear parametrization of PMI to the use of syntactic contexts for pre-trained word embeddings. In our experiments, we adopt a slightly different approach from the one taken by Levy and Goldberg. For our dependency-based contexts, we simply take the head (as defined by a dependency parse) of each focal word as its one and only context word. Thus, during collection of cooccurrence statistics, we add 1 to N_{ij} every time we observe the dependent-head pair (i, j) in the corpus. Consider the same toy example as in the previous section: "The quick brown fox jumped over the lazy dog." (SHOW the parse of the sentence by CoreNLP). Here, we have $N_{ij} = 1$ for (i, j) =("fox", "jumped"). As in the previous section, note the asymmetry in that $N_{ji} = 0$, since dependency relations have directionality attached to them.

Once again, viewing this from the perspective of PMI, we now have that $\text{PMI}(i, j)$ measures the deviation from independence of the probability that word j is the head of word i. $\text{PMI}(j, i)$ measures the deviation from independence of the probability that word i is the head of word j, which is equivalent to word j being the dependent of word i. These are approximated by $\langle i \cdot j \rangle$ and $\langle j \cdot i \rangle$ respectively. With this framework, we have the ability to model both sides of the head-dependent asymmetry, as well as a quantifier for the plausibility

of a dependency relation (with direction) between two words. The latter could prove useful for a task like selectional preference, which we discuss further in Section 8.2.

Our approach differs from Levy and Goldberg in how we exploit the fact that we have two representations per word. Covectors represent words as dependents searching for their heads and vectors represent words as heads searching for their dependents. In the Levy and Goldberg case, both covectors and vectors can represent either heads or dependents (since both the head and dependents of every focal word are taken as context words).

In addition, we do not employ typed dependencies. We obtain a single covector and vector for each word, rather than multiple covectors for each word (one for each type of dependency relation that it appears in). We can directly query our metric of cooccurrence between two words $\langle i \cdot j \rangle$, irrespective of a dependency relation type. We plan on exploring the explicit integration of dependency relation types into the Hilbert inner product in future work (see Section 8.1).

We present both qualitative and quantitative results from our training of dependency-based embeddings in Section 6.

5 Vector-only Embeddings

In this section, we introduce two types of embeddings to investigate the necessity of training covectors. The latter give us the ability to model both symmetric and asymmetric notions of cooccurrence between words via their inner products with vectors. In spite of this, they are often discarded after training (the default for SGNS), or the average between covectors and vectors is taken (the default for GloVe). We attempt to reconcile this practice with how word representations are trained by investigating the performance of embeddings trained without the two distinct representations of vectors and covectors. Returning to the symmetric window-based definition of ϕ_{ij} in the Hilbert-MLE paper, we investigate the effect of manipulating ψ_{ij}.

Our first set of modified Hilbert embeddings eliminates covectors from the training process entirely, replacing $\psi_{ij} = \langle i \cdot j \rangle$ with $\psi_{ij} = |i\rangle \cdot |j\rangle + b_{|i\rangle} + b_{|j\rangle}$. Now, the inner product between two vectors has the dual responsibility that it previously split with the covector-vector inner product. It has the objective $|i\rangle \cdot |j\rangle = \text{PMI}(i, j)$ and is still meant to measure the similarity between words i and j. These goals are not fully compatible since (though there can be some overlap) words that appear together frequently do not necessarily have similar meanings. For example, "young" and "woman" cooccur very often (39,477 times in the corpus we used), but clearly do not have similar meanings.

Our second set of modified Hilbert embeddings also does not train covectors, but instead trains a linear map $R : \mathbb{R}^d \to \mathbb{R}^d$, which maps vectors to representations that are meant to behave like covectors. We set $\psi_{ij} = (|i\rangle R)^T \cdot j\rangle := (|i\rangle R)^T |j\rangle + b_{|i\rangle} + b_{|j\rangle}$. With this, we regain two separate representations for each word i: $|i\rangle$ and $(|i\rangle R)^T$, but the second representation is not trained for each individual word.

6 Experiments

6.1 Implementation

The corpus used for window-based embeddings (both symmetric and positionally asymmetric) is the same as in the Hilbert paper: a combination of Gigaword 3 (Graff et al., 2007) and a lower-cased Wikipedia 2018 dump. For dependency-based embeddings, we use the dependency parse obtained by running Stanford CoreNLP on the Gigaword 3 corpus. (We initially attempted to train on the English Web Treebank (Bies et al., 2012), annotated in Universal Dependencies format by human annotators (Nivre et al., 2016). Unfortunately, it proved to be too small for our embeddings to learn meaningful relations, containing only 254,830 tokens.) For a fair comparison, we also train a second set of symmetric window-based embeddings on the Gigaword 3 corpus only. We collect unigram and cooccurrence statistics for each corpus and for each of the three definitions of cooccurrence (symmetric window-based, positionally asymmetric window-based, and dependency-based). In all cases, the vocabulary size is $n = 50{,}000$.

All embeddings are trained with the sample-based version of Hilbert-MLE with balanced sampling. The dimension of the embedding space is $d = 300$, except for vector-only embeddings, where experiments were done with $d = 300$ and $d = 600$. The number of updates is fixed at 30,000 and the batch size used is 450,000 (except in the case of $d = 600$, where it is decreased to 400,000 to fit onto the GPU). The temperature hyperparameter is set to 2. For all embeddings with $\psi_{ij} = \langle i \cdot j \rangle$, learning rate is between 2e-3 and 3e-3. For vector-only embeddings, learning rate is between 8e-3 and 9e-3. Bias terms ("covector bias" and "vector bias") for each word in the vocabulary are trained and are added to each inner product. In the case of dependency-based embeddings, this means that each word has a bias corresponding to it being a dependent and a bias corresponding to it being a head. This bias is independent of the word's surrounding context.

We run Hilbert-MLE with 1 GPU, 2 CPUs and 32GB of RAM. Reaching 30,000 updates takes approximately 8 hours. We quantitatively evaluate our embeddings on the same intrinsic and extrinsic tasks as in the Hilbert-MLE paper.

6.2 Quantitative Evaluation

In the similarity tasks, we consider the ranking of word pairs given by the cosine similarity between word vectors and measure its correlation with the rankings by human annotators. Spearman's rank correlation coefficient is the metric used for all similarity tasks. These tasks are: Baker Verbs 143 (B143) (Baker et al., 2014), the MEN development set (MENd) and test set (MENt) (Bruni et al., 2012), Radinsky Mechanical Turk (RMT) (Radinsky et al., 2011), Rare Words (RARE) (Luong et al., 2013), the SemEval 2017 Task (SE17) (Camacho-Collados et al., 2017), Simlex999 (S999) (Hill et al., 2015), Wordsim353 (Finkelstein et al., 2001) divided into similarity (WS-S) and relatedness (WS-R) (Agirre et al., 2009), and Yang Powers Verbs 130 (Y130) (Yang and Powers, 2006).

In the analogy tasks, the goal is to resolve analogies like "Paris is to France as Berlin is to X" using operations on word vectors. We evaluate on the Google Analogy dataset (Google) (Mikolov et al., 2013a) and the Bigger Analogy Test Set (BATS) (Gladkova et al., 2016). We use the *3CosMul* selection rule, which was found to perform best in the Hilbert-MLE paper.

There are two types of downstream tasks: text clasification and sequence labelling. For text classifcation, the IMDB movie reviews dataset for sentiment analysis (Maas et al., 2011) and the AGNews news classification dataset, preprocessed by Kenyon-Dean et al. (2018) are used. We use the same BiLSTM-max sequence encoder model (Conneau et al., 2017) as the one used in the Hilbert paper. For sequence labelling, we use our embeddings for supersense tagging (Ciaramita and Altun, 2006) on the Semcor 3.0 corpus (Miller et al., 1993), as well as part-of-speech (POS) tagging on the Penn Treebank Wall Street Journal Corpus (Marcus et al., 1993) and the Brown corpus (distributed by NLTK). We use the same biLSTM sequence labelling model from Huang et al. (2015) employed in the Hilbert-MLE paper.

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
SC	0.518	0.392	0.714	0.530	0.635	0.657	0.353	0.718	0.443	0.701	0.566
RC	0.514	0.391	0.687	0.569	0.623	0.623	0.284	0.697	0.468	0.681	0.554
LC	0.542	0.401	0.697	0.499	0.654	0.628	0.303	0.697	0.327	0.679	0.542

Table 1: Table summarizing the similarity task performance for embeddings with different choices of ϕ_{ij}. Performance is measured with Spearman's rank correlation coefficient, signifying correlation with human annotator scores. "SC" designates "symmetric window-based with covectors", "LC" designates "left window-based with covectors", and "RC" designates "right window-based with covectors". The last column is the average over all similarity task performances for each embedding.

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
SCG	0.170	0.224	0.453	0.497	0.443	0.508	0.307	0.577	0.272	0.419	0.387
DCG	0.493	0.402	0.689	0.520	0.617	0.623	0.302	0.694	0.348	0.653	0.534

Table 2: Table summarizing the similarity task performance for embeddings trained on Gigaword 3 only (symmetric window-based and dependency-based). Performance is measured with Spearman's rank correlation coefficient, signifying correlation with human annotator scores. "SCG" designates "symmetric window-based with covectors trained on Gigaword 3", "DCG" designates "dependency-based with covectors trained on Gigaword 3". The last column is the average over all similarity task performances for each embedding.

6.3 Qualitative Evaluation

To better understand the behaviour of covectors and vectors under changes of ψ_{ij} and ϕ_{ij}, we construct a qualitative analysis of our embeddings, which we call "selection tables". This is similar to the qualitative analysis of dependency-based embeddings by Levy and

	RARE	S999	MENt	WS-R	SE17	RMT	B143	WS-S	Y130	MENd	Avg
SC	0.518	0.392	0.714	0.530	0.635	0.657	0.353	0.718	0.443	0.701	0.566
SR	0.529	0.391	0.696	0.587	0.637	0.676	0.337	0.728	0.448	0.689	0.571
SV	0.462	0.351	0.749	0.597	0.606	0.659	0.234	0.709	0.468	0.732	0.557

Table 3: Table summarizing the similarity task performance for embeddings with different choices of ψ_{ij}. Performance is measured with Spearman's rank correlation coefficient, signifying correlation with human annotator scores. "SC" designates "symmetric window-based with covectors", "SR" designates "symmetric window-based with R (600-dimensional)", and "SV" designates "symmetric window-based vector-only (without R, 600-dimensional)". The last column is the average over all similarity task performances for each embedding.

	Google	BATS
SC	0.563	0.283
RC	0.515	0.224
LC	0.513	0.251

Table 4: Accuracy scores for embeddings with different choices of ϕ_{ij} on Google Analogies and the Bigger Analogy Test Set.

	Google	BATS
SCG	0.359	0.199
DCG	0.348	0.130

Table 5: Accuracy scores for embeddings trained on Gigaword 3 only on Google Analogies and the Bigger Analogy Test Set.

	Google	BATS
SC	0.563	0.283
SR	0.539	0.265
SV	0.602	0.213

Table 6: Accuracy scores for embeddings with different choices of ψ_{ij} on Google Analogies and the Bigger Analogy Test Set.

Goldberg (2014). In that analysis, they fix a word vector and list the five word vectors that have the highest cosine similarity with it, as well as the five covectors that have the highest inner product with it. We perform the same analysis and, since we have embeddings capable of modelling two sides of an asymmetric relation, we also generate tables where a word's covector is fixed and we list the vectors that have the highest inner product with it. For embeddings where $N_{ij} = N_{ji}$ does not hold, we expect to observe that the tables for fixed vector and fixed covector to be different. For instance, in the case of dependency-based embeddings, fixing a word vector $|j\rangle$ is akin to fixing a head that is looking for its dependents. The list for that fixed vector should be a list of five words which have the highest PMI with the word represented by $|j\rangle$ when the latter is viewed as a head and they are viewed as dependents. To facilitate interpretability, we only rank the top 2000 words in the vocabulary when generating selection tables.

j	$\text{argmax}_i \langle i \cdot j \rangle$	$\text{argmax}_i \langle j \cdot i \rangle$	$\text{argmax}_i \cos(\|i\rangle, \|j\rangle)$
company	insurance, profit, owned, holding, largest	owned, insurance, bought, oil, inc.	companies, firm, industry, businesses, business
president	vice, w., george, bush, clinton	vice, w., clinton, bush, george	vice, presidential, chairman, administration, met
florida	supreme, state, university, governor, texas	beach, supreme, georgia, southern, carolina	carolina, texas, virginia, jersey, ohio
small	businesses, town, arms, amount, village	businesses, arms, town, amount, large	large, smaller, larger, huge, largest
arrived	here, afternoon, shortly, saturday, sunday	here, troops, morning, ship, refugees	returned, met, sent, headed, visit

Table 7: Selection table for symmetric window-based embeddings trained on Gigaword and Wikipedia.

7 Discussion

As shown in Table 1, the positionally asymmetric embeddings are competitive with the symmetric window-based embeddings. On top of the benefit of being able to query word ordering, there is little decrease in intrinsic performance with this asymmetric context. This is likely because the positionally asymmetric embeddings still rely on the notion of a fixed size context window with dynamic weighting. Even though left context windows may not capture a context word j to the right of a focal word i (which a symmetric window would catch), they will still capture i as a context word for j (which the symmetric window also catches), since i is to the left of j. In a sense, the positionally asymmetric windows capture the same information, but do not "double count" coocurrence statistics, and thus the PMI

j	$\text{argmax}_i \langle i \cdot j \rangle$	$\text{argmax}_i \langle j \cdot i \rangle$	$\text{argmax}_i \cos(\|i\rangle, \|j\rangle)$
company	founded, owned, earnings, bought, shares	insurance, holding, telephone, entertainment, phone	companies, firm, corp., co., businesses
president	w., clinton, bush, george, saddam	vice, elected, former, iraqi, appointed	vice, leaders, leader, candidate, presidential
florida	supreme, state, attorney, governor, coast	beach, beat, fort, south, university	carolina, texas, ohio, jersey, georgia
small	businesses, amount, arms, village, town	very, large, too, quite, contains	large, smaller, larger, largest, huge
arrived	here, scene, afternoon, morning, evening	ships, troops, ship, arafat, refugees	return, occurred, leave, visit, trip

Table 8: Selection table for positionally asymmetric embeddings with right context windows.

j	$\text{argmax}_i \langle i \cdot j \rangle$	$\text{argmax}_i \langle j \cdot i \rangle$	$\text{argmax}_i \cos(\|i\rangle, \|j\rangle)$
company	insurance, holding, telephone, software, oil	earnings, owned, bought, profit, founded	companies, firm, co., businesses, business
president	vice, elected, former, senior, iraqi	w., clinton, bush, george, bill	vice, prime, government, chairman, minister
florida	beach, south, texas, fort, university	supreme, state, georgia, attorney, coast	texas, carolina, georgia, virginia, ohio
small	very, too, contains, large, quite	town, amount, village, arms, size	large, few, smaller, larger, huge
arrived	ship, refugees, soldiers, plane, troops	here, afternoon, scene, baghdad, moscow	met, sent, entered, returned, took

Table 9: Selection table for positionally asymmetric embeddings with left context windows.

j	$\text{argmax}_i \langle i \cdot j \rangle$	$\text{argmax}_i \langle j \cdot i \rangle$	$\text{argmax}_i \cos(\|i\rangle, \|j\rangle)$
company	insurance, holding, telephone, software, oil	earnings, owned, bought, profit, founded	companies, firm, co., businesses, business
president	vice, elected, former, senior, iraqi	w., clinton, bush, george, bill	vice, prime, government, chairman, minister
florida	beach, south, texas, fort, university	supreme, state, georgia, attorney, coast	texas, carolina, georgia, virginia, ohio
small	very, too, contains, large, quite	town, amount, village, arms, size	large, few, smaller, larger, huge
arrived	ship, refugees, soldiers, plane, troops	here, afternoon, scene, baghdad, moscow	met, sent, entered, returned, took

Table 10: Selection table for positionally asymmetric embeddings with left context windows.

j	$\text{argmax}_i (\|i\rangle R)^T \cdot j \rangle$	$\text{argmax}_i (\|j\rangle R)^T \cdot i \rangle$	$\text{argmax}_i \cos(\|i\rangle, \|j\rangle)$
company	insurance, owned, largest, inc., founded	insurance, owned, holding, earnings, software	companies, firm, co., industry, businesses
president	vice, w., clinton, george, bush	vice, w., clinton, george, bush	
florida	supreme, georgia, beach, voters, bay	beach, supreme, texas, georgia, fort	carolina, texas, california, georgia, virginia
small	businesses, amount, fish, farm, numbers	businesses, amount, numbers, town, farm	large, smaller, larger, huge, nearby
arrived	here, shortly, afternoon, ship, troops	shortly, here, paris, afternoon, refugees	met, returned, visit, sent, reporters

Table 11: Selection table for vector-only embeddings with R.

j	$\text{argmax}_i \lvert i\rangle \cdot \lvert j\rangle + b_{\lvert i\rangle} + b_{\lvert j\rangle}$	$\text{argmax}_i \cos(\lvert i\rangle, \lvert j\rangle)$
company	firm, co., companies, corp, inc.	companies, firm, co., owned, industry
president	vice, w., met, leader, former	vice, w., met, clinton, bush
florida	carolina, beach, texas, virginia, state	carolina, texas, beach, georgia, virginia
small	large, smaller, fish, businesses, larger	large, smaller, larger, businesses, fish
arrived	met, ambassador, visit, sent, saturday	met, visit, returned, monday, saturday

Table 12: Selection table for vector-only embeddings (without R).

j	$\text{argmax}_i \langle i \cdot j\rangle$	$\text{argmax}_i \langle j \cdot i\rangle$	$\text{argmax}_i \cos(\lvert i\rangle, \lvert j\rangle)$
company	executives, insurance, software, bought, largest	software, executives, insurance, inc., operating	companies, firm, firms, businesses, industry
president	vice, w., george, yeltsin, clinton	vice, w., george, clinton, elected	vice, met, administration, chairman, elected
florida	supreme, coast, university, beach, georgia	supreme, miami, beach, coast, university	miami, texas, arizona, california, georgia
small	businesses, amount, size, bowl, arms	businesses, town, amount, bowl, arms	large, smaller, huge, big, size
arrived	delegation, here, baghdad, scene, afternoon	delegation, here, shortly, baghdad, afternoon	visited, met, returned, visit, sent

Table 13: Selection table for symmetric window-based embeddings on Gigaword 3.

j	$\text{argmax}_i \langle i \cdot j \rangle$	$\text{argmax}_i \langle j \cdot i \rangle$	$\text{argmax}_i \cos(\|i\rangle, \|j\rangle)$
company	executives, earnings, executive, shares, corp.	holding, insurance, software, giant, phone	firm, companies, firms, unit, businesses
president	clinton, milosevic, bush, yeltsin, hussein	vice, association, executive, marketing, senior	vice, governor, candidate, lawyer, chairman
florida	university, voters, republican, miami, coast	south, arizona, georgia, miami, southern	california, arizona, texas, jersey, state
small	businesses, arms, bowl, large, amount	too, very, size, numbers, pretty	large, smaller, huge, big, largest
arrived	delegation, soon, shortly, soldiers, refugees	delegation, visit, evening, scene, afternoon	visited, returned, came, met, took

Table 14: Selection table for dependency-based embeddings on Gigaword 3.

matrix is not symmetric. We also note that the right asymmetric embeddings perform better than the left asymmetric embeddings on similarity tasks, which is consistent with the findings of Lison and Kutuzov (2017). In addition, the positionally asymmetric embeddings perform worse on analogy tasks (Table 4) compared to their symmetric counterpart.

On the similarity tasks, we find that the dependency-based embeddings perform much better than the symmetric window-based embeddings trained on the Gigaword 3 corpus, as shown in Table 2. However, as was the case in Levy and Goldberg (2014a), dependency-based embeddings perform very poorly on analogical reasoning tasks (Table 5).

Removing covectors (without replacing them by a linear map R) causes an average drop of 0.04 in the Spearman coefficient across all tasks (see Table 3) (including in the 300-dimensional case, which we do not report here). The selection table sheds some light on this, as the highest cosine similarity between word vectors often reflect cooccurrence rather than similarity. For instance, the words "went" and "awry" have a relatively high cosine similarity. However, this drop in correlation is not dramatic. There are still word vector pairs with high cosine similarity that do reflect word similarity. The word "gone" appears as one of the words having the highest cosine similarity with "went". This can be explained by the fact that both the vector for "went" and the vector for "gone" have high cosine similarity with "awry". This shared selectivity for the same word also makes the similar words "went" and "gone" selective for each other. Thus, inner products between vector-only embeddings (without R) can reflect both cooccurrence and similarity, but provide no explicit way of distinguishing between the two. We cannot determine whether a pair of vector-only embeddings having high cosine similarity means that the two words appear often together or that they have a similar meaning. The use of covectors provides the ability to make this distinction explicit.

Replacing covectors with a linear map acting on vectors lags behind when the embedding

dimension is kept at 300. However, increasing the embedding dimension to 600 is able to achieve an average score which is slightly better than that of symmetric window-based with covectors. Note that the R case uses nearly the same amount of parameters as the case with covectors (actually $300^2 = 90,000$ more because of the training of R), while the former uses half as many. The 600-dimensional case allows for an embedder to learn embeddings that have a "covector" and a "vector" part (though this might not necessarily be the case, we can view an embedding as a concatenation of a covector and a vector). Then R can mask the "vector" part of an embedding and reverse the positions of the "covector" and "vector" part, thus allowing $(|i\rangle R)^T |j\rangle$ to emulate a covector-vector inner product. However, this is not possible in the 300-dimensional case, as the dimension of the subspaces for the "covector" and "vector" parts would be too low. This preliminary result suggests that the effect of covectors can successfully be emulated by a linear map (and this is further backed up by the selection tables).

We observe very small differences (most are within 1% of each other) between the embeddings in downstream task performance. Performance is slightly weaker for vector-only embeddings, and we hypothesize that the reason for this is once again the conflation of cooccurrence with similarity. Overall, the symmetric window-based embeddings perform better (albeit marginally) than the other variants.

That said, we hypothesize that the covector-vector and vector-covector inner products in embeddings with asymmetric contexts could prove useful for such downstream tasks. We plan on exploring this in the future. The modelling of different notions of cooccurrence may be more suited for these tasks than the modelling of similarity and/or relatedness.

Arguably the most significant analysis of the embedding variants is the selection table. In spite of its qualitative nature, it provides the most insight into the behaviour of a set of embeddings.

In the symmetric window-based case ($\psi_{ij} = \langle i \cdot j \rangle$ and $\phi_{ij} = \mathrm{PMI}(i, j)$), we indeed observe that columns for fixed vector and fixed covector are nearly identical. This is a reflection of the fact that $\mathrm{PMI}(i, j) = \mathrm{PMI}(j, i)$, which leads to $\langle i \cdot j \rangle \approx \langle j \cdot i \rangle$. Unlike in Levy and Goldberg (2014), we find that cosine similarity between word vectors can reflect functional similarity quite well. For instance, the top five selections for "Florida" are all other US states.

In the positionally asymmetric case with left contexts, we observe how fixing a word vector results in it having high selectivity for covectors of words that frequently appear to its left. As an example, the word "company" is selective for "insurance" and "holding". Meanwhile, fixing a word covector results in it having high selectivity for vectors of words that frequently appear to its right. In this case, "company" becomes selective for "earnings" and "profit". In the case of right contexts, we see the reverse effect. Hence, these embeddings are capable of modelling both sides of the positional asymmetry. Furthermore, since we can interchange covectors and vectors, the left and right embeddings are equivalent in terms of their behaviour. We also observe that the cosine similarity column for both of these embeddings is close to that of the symmetric window-based embeddings. This further explains the close performance of the three variants on similarity tasks.

Similarly, dependency-based embeddings successfully model both sides of the dependent-head asymmetry. Fixing a words vector results in the selection of covectors that represent plausible dependents for that word. For example, "president" is selective for "vice" and "executive". Fixing a words covector results in the selection of plausible heads for that word. Here, "president" becomes selective for names of actual presidents, like "bush" and "clinton". Unlike Levy and Goldberg (2014a), we observe less of an emphasis on the *conj* (conjunction) relation in the covector-vector selections. For cosine similarity between word vectors, we see the same pattern as in Levy and Goldberg, as pairs of words that reflect functional similarity are ranked higher. We see this with the verb "arrived" being selective for relevant past tense verbs like "visited" and "returned". However, we observe this ability to pick out functional similarity with symmetric context windows as well. This suggests the greater benefit from training dependency-based embeddings lies in the bilinear parametrization of the plausibility of a head-dependent relation (and its direction).

Finally, vector-only embeddings with R exhibit selection tables which are very close to embeddings with covectors. Vector-only embeddings without R have no notion of covector, so only the vector-vector inner product and cosine similarity can be evaluated. We do not observe major differences between the later two columns. Focusing in on the cosine similarity column, we see how the vector-only embeddings conflate similarity and cooccurrence. The verb "arrived" is indeed selective for similar verbs like "met" and "returned", but it is also selective for dissimilar words which it cooccurs with often like "monday" and "saturday". The following example illustrates how vector-only embeddings are able to capture both cooccurrence and similarity:

The word vector for "gone" is highly selective for the vectors of "awry" and "went". The former reflects cooccurrence while the latter reflects similarity. But how is "gone" very selective for "went" if they do not cooccur that often? This is because "went" and "awry" are also very selective for each other (as they cooccur often). Hence, the representations for "gone", "went", and "awry" all end up near each other in terms of cosine distance.

8 Future Work

8.1 Typed Dependency-based Embeddings

As a next step, we will enhance our dependency-based embeddings by integrating relation types into context. However, instead of adopting the same approach as Levy and Goldberg (2014a), which involves separate covectors for each (context word, relation) pair, we train matrices (similar to R in the vector-only case) for each relation type. These matrices are meant to contextualize a word's covector with a relation type. For instance we can transform the covector for "red" with the matrix for "adjectival modifier" (*amod*). The transformed covector is now the word "red" searching for its head under the relation *amod*. We now take $\psi_{ijk} = \langle i \cdot R_k \cdot j \rangle := \langle i|R_k|j\rangle + b_{\langle i|} + b_{|j\rangle} + b_k$ for a triple (i, j, k), where i and j are words, and k is a relation type.

There remains the issue of determining which ϕ_{ijk} is appropriate to use, which links to how we sample from the data distribution and the model. A first attempt, which consisted of sampling k from a unigram distribution of relation types (for each word in the corpus, we look at the label of the arc to its head) and then sampling words i and j from their unigram distributions conditional on k, failed. Storing K (where K is the number of relation types) cooccurrence matrices was impractical.

Our next attempt will be to use a three-variable extension of PMI, which we denote by PMI*:

$$\text{PMI}^*(i, j, k) = \ln \frac{p_{ijk}}{p_i p_j p_k}, \tag{16}$$

where p_{ijk} is the probability of words i and j cooccurring with i as the dependent, j as the head, and k as the type of dependency relation that links them.

We follow the same derivation as in the original Hilbert-MLE and take the balanced sampling approach. We sample i, j, and k from their respective unigram distributions and obtain the following loss:

$$\mathcal{L} = \mathop{\mathbb{E}}_{i\sim p_i, j\sim p_j, k\sim p_k} \left[e^{\text{PMI}^*(i,j,k)} \langle i \cdot R_k \cdot j \rangle - e^{\langle i \cdot R_k \cdot j \rangle} \right] \tag{17}$$

Taking the partial derivative with respect to our inner product, we have:

$$\frac{\partial \mathcal{L}}{\partial \langle i \cdot R_k \cdot j \rangle} = (p_i p_j p_k)^{\frac{1}{\tau}} \left[e^{\text{PMI}^*(i,j,k)} - e^{\langle i \cdot R_k \cdot j \rangle} \right] \tag{18}$$

Other potential approaches that will be explored include using Gibbs sampling.

8.2 Application to Selectional Preference

Selectional preference is a natural language phenomenon where given a head and a dependency relation, there are preferences as to which words will plausibly be the dependent. As of its publication, recent SP-10K dataset (Zhang et al., 2019) is the largest available ground truth dataset for this task. It consists of 10,000 examples across 5 dependency relations (3 one-hop relations, and 2 two-hop relations) containing 2500 verbs, nouns and adjectives selected from the 5000 most common words in the Corpus of Contemporary American English. The three one-hop relations are *dobj* (verb-object), *nsubj* (verb-noun subject), and *amod* (noun-adjective). The two-hop relations are *dobj_amod* (verb-adjective modifying object) and *nsubj_amod* (verb-adjective modifying noun subject). An average score across at least 11 annotators per example mapped to the interval $[0, 10]$ is provided as a measure of the plausibility of each word pair under a given dependency relation. The goal is to learn a model that ranks the plausibility of pairs that correlates positively with the ground truth human annotation.

For example, say we are given the pair (eat, meal) with the *dobj* relation. We view this as asking the question "How plausible would it be for the noun 'meal' to be the object of the verb 'eat'?" Unsurprisingly the ground truth annotator score for this example in the dataset is 10.0, and thus a model is expected to give the pair a high score. Similarly, say we are given the pair (lift, heavy) with the *dobj_amod* relation. This asks the question "How plausible would it be for the adjective 'heavy' to modify the object of the verb 'lift'?". The ground truth plausibility score for the latter example is 9.17.

We plan on evaluating our dependency-based embeddings (both typed and untyped) on this task. Leveraging the covector-vector inner product, we expect the embeddings to perform well in the "one-hop" case. The "two-hop" case will be more challenging as the Hilbert embeddings are limited to dyadic covector-vector inner products. Directly computing the inner product between an adjective and a verb will not work as the adjective is not a dependent of the verb but rather a dependent of a dependent of the verb.

9 Conclusion

Continuing the work of the authors of Hilbert-MLE, we build on the central notion of bilinear parametrization (ψ_{ij}) of PMI (ϕ_{ij}). We test the effect that changes in ψ_{ij} and ϕ_{ij} have on the resulting pre-trained word embeddings. In particular, we find that the use of asymmetric context windows changes ϕ_{ij} in such a way that $\phi_{ij} \neq \phi_{ji}$, and that this is reflected in the interactions between covectors and vectors in the resulting embeddings. We see this with both positionally asymmetric embeddings and dependency-based embeddings. The former embeddings allow us to score the plausibility of words appearing in a certain order, while the latter allow us to score the plausibility of a directed dependency relation. Furthermore, we place emphasis on the importance of training dual representations for each word, as we find that vector-only embeddings lag behind in our evaluations. The interaction between vector-only embeddings conflates cooccurrence and similarity, which are two separate notions. The use of covectors along with vectors allows for distinction between the two. In future work, we will explore adding relation types to dependency-based embeddings as well as applying the latter embeddings to a selectional preference task.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Strakova, Marius Pasca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. pages 19–27, 01 2009. doi: 10.3115/1620754.1620758.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *CoRR*, abs/1502.03520, 2015. URL `http://arxiv.org/abs/1502.03520`.

Simon Baker, Roi Reichart, and Anna Korhonen. An unsupervised model for instance level subcategorization acquisition. In *Proceedings of the 2014 Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 278–289, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1034. URL https://www.aclweb.org/anthology/D14-1034.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. English web treebank. *Linguistic Data Consortium, Philadelphia, PA*, 2012.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P12-1015.

Jose Camacho-Collados, Mohammad Taher Pilehvar, Nigel Collier, and Roberto Navigli. SemEval-2017 task 2: Multilingual and cross-lingual semantic word similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 15–26, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2002. URL https://www.aclweb.org/anthology/S17-2002.

Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602, Sydney, Australia, July 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W06-1670.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. *CoRR*, abs/1705.02364, 2017. URL http://arxiv.org/abs/1705.02364.

Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies manual. 01 2008.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. volume 20, pages 406–414, 01 2001. doi: 10.1145/503104.503110.

Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-2002. URL https://www.aclweb.org/anthology/N16-2002.

Dave Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword third edition. *Linguistic Data Consortium*, 2007.

ZS Harris. Distributional structure. word, 10 (2-3): 146–162. reprinted in fodor, j. a and katz, jj (eds.), readings in the philosophy of language, 1954.

Felix Hill, Roi Reichart, and Anna Korhonen. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, De-

cember 2015. doi: 10.1162/COLI_a_00237. URL https://www.aclweb.org/anthology/J15-4004.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015. URL http://arxiv.org/abs/1508.01991.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. URL http://arxiv.org/abs/1607.01759.

Dan Jurafsky and James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 3rd Edition draft*. 2019. URL https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

Kian Kenyon-Dean, Andre Cianflone, Lucas Page-Caccia, Guillaume Rabusseau, Jackie Chi Kit Cheung, and Doina Precup. Clustering-oriented representation learning with attractive-repulsive loss. *CoRR*, abs/1812.07627, 2018. URL http://arxiv.org/abs/1812.07627.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL*, 2014a.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Advances in Neural Information Processing Systems*, 3:2177–2185, 01 2014b.

Pierre Lison and Andrey Kutuzov. Redefining context windows for word embedding models: An experimental study. *arXiv preprint arXiv:1704.05781*, 2017.

Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-3512.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. pages 142–150, 01 2011.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. pages 1–12, 01 2013a.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013b. URL http://arxiv.org/abs/1310.4546.

George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. A semantic concordance. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*, 1993. URL https://www.aclweb.org/anthology/H93-1061.

Edward Newell, Kian Kenyon-Dean, and Jackie C.K. Cheung. Low rank word embedders and the tempered bilinear pmi model. *Unpublished*, 2019.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL `https://www.aclweb.org/anthology/L16-1262`.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL `https://www.aclweb.org/anthology/D14-1162`.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *WWW*, 2011.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. Swivel: Improving embeddings by noticing what's missing. *CoRR*, abs/1602.02215, 2016. URL `http://arxiv.org/abs/1602.02215`.

Dongqiang Yang and David Powers. Verb similarity on the taxonomy of wordnet. 10 2006.

Hongming Zhang, Hantian Ding, and Yangqiu Song. Sp-10k: A large-scale evaluation set for selectional preference acquisition. *arXiv preprint arXiv:1906.02123*, 2019.