Continuous Word Embedding Fusion via Spectral Decomposition

Tianfan Fu
Georgia Institute of Technology
Atlanta, GA, USA
tfu42@gatech.edu

Cheng Zhang
Microsoft Research Cambridge
Cambridge, CB1 2FB, UK
cheng.zhang@microsoft.com

Stephan Mandt
Los Angeles, CA, USA
stephan.mandt@gmail.com

Abstract

Word embeddings have become a mainstream tool in statistical natural language processing. Practitioners often use pre-trained word vectors, which were trained on large generic text corpora, and which are readily available on the web. However, pre-trained word vectors oftentimes lack important words from specific domains. It is therefore often desirable to extend the vocabulary and embed new words into a set of pre-trained word vectors. In this paper, we present an efficient method for including new words from a specialized corpus, containing new words, into pre-trained generic word embeddings. We build on the established view of word embeddings as matrix factorizations to present a spectral algorithm for this task. Experiments on several domain-specific corpora with specialized vocabularies demonstrate that our method is able to embed the new words efficiently into the original embedding space. Compared to competing methods, our method is faster, parameter-free, and deterministic.

1 Introduction

There has been a recent surge of neural word embedding models (Mikolov et al., 2013a,b). These models have been shown to perform well in a variety of NLP problems, such as word similarity and relational analogy tasks. Word embeddings play a crucial role in diverse fields such as computer vision (Hwang and Sigal, 2014), news classification (Kenter and De Rijke, 2015; Phung and De Vine, 2015), machine translation (Zou et al., 2013; Wu et al., 2014), and have been extended in various ways (Rudolph et al., 2016; Bamler and Mandt, 2017; Peters et al., 2018).

Instead of training word embeddings from scratch, practitioners often resort to high-quality, pre-trained word embeddings which can be downloaded from the web. These embeddings were trained on massive corpora, such as online news. The downside is that their vocabulary is often restricted, and by nature, is very generic. An open question remains of how to optimally include new words from highly specialized text corpora into an existing word embedding fit. Such transfer learning has several advantages. First, one saves the computational burden of learning high-quality word vectors from scratch. Second, one can already rely on the fact that the majority of word embedding vectors are semantically meaningful. Third, as we show in this paper, there are deterministic and parameter-free approaches that fulfill this goal, making the scheme robust and reproducible.

For a practical application, imagine that we are given a small corpus, such as a collection of scientific articles, and our goal is to include the associated vocabulary into a pre-trained set of word vectors which were learned on Google Books or Wikipedia. The scientific corpus contains both common words (e.g., "propose", "experiment") and domain-specific words, such as "submodular" and "sublinear". However, this specialized corpus can be safely assumed to be too small to train a word embedding model from scratch. Alternatively, we could merge the domain-specific corpus with a large generic corpus and train the entire word-embedding from scratch, but this would be computationally demanding and non-reproducible due to the non-convexity of the underlying optimization problem. In this paper, we show how to include the specialized vocabulary into the generic set of word vectors without having to re-train the model on the large vocabulary, simply relying on linear algebra.

A naive baseline method is to fix the pre-trained word vectors and only update the new ones. We found that this approach suffers from local optima and sensitive to hyper-parameters; therefore it is not reliable in practice. In contrast, our approach is not based on gradient descent and therefore more robust, deterministic, and parameter-free.

In this paper, we propose a Spectral Online Word Embedding (**SOWE**) algorithm for integrating new words into a pre-trained word embedding fit. Our approach is based on online matrix factorization. In more detail, our main contributions are as follows:

- We propose a Spectral Online Word Embedding (SOWE) method to include new words into a pre-trained set of word vectors. This approach does naturally not suffer from optimization problems, such as initialization, parameter tuning, and local optima.
- Our approach approximately reduces to an online matrix factorization problem. We provide a bound on the approximation error and show that this error does not scale with the vocabulary size (Theorem 1).
- The complexity of proposed method scales linearly with the size of vocabulary and quadratically with embedding dimension, making the approach feasible in large-scale applications.
- We evaluate our method on two domain specific corpora. Experimental results show that our method is able to embed new vocabulary faster than the baseline method while obtaining more meaningful embeddings. It is also parameter-free and deterministic, making the approach easily reproducible.

2 Related Work

Our paper departs from word embeddings learned via the skip-gram method, and shows how the vocabulary can be extended in an online fashion, using methods from linear algebra. As such, our approach relates to word embeddings, online learning, and the singular value decomposition (SVD).

Skip-Gram Model Our model builds on word embeddings trained via the skip-gram model with negative sampling (SGNS), proposed by Mikolov et al. (2013a,b). These papers proposed a scalable training algorithm based on negative sampling.

The model predicts a target word in the middle of a sentence based on its surrounding words (contexts). Each target word / context word is associated with a feature vector whose entries are latent, and are treated as parameters to be learned. This model is efficient to train via stochastic gradient descent; its resulting word vectors provide state-of-the-art results on various linguistic tasks (Liu et al., 2015; Pickhardt et al., 2014). The skip-gram model was influential both in the machine learning and related communities.

Levy and Goldberg (2014) showed that word2vec can be viewed as an implicit matrix factorization of the pointwise mutual information matrix (PMI) of word distributions. The authors present a closed-form solution based on a singular value decomposition of a sparse version of this matrix, termed SPPMI. In this paper, we extend on this view and present an efficient online learning algorithm based on a decomposition of SPPMI matrix which departs form pre-trained word embeddings.

Online Word Embeddings Kiros et al. (2015) propose adding new words into an existing embedding space using a projection method to warm-start learning the new words. The authors assumed that there already exist well-trained word vectors from a large underlying vocabulary, and present a method that projects word vectors from an old space to a new space, where the projection matrix is learned from known words.

Bojanowski et al. (2016) exploited character-level features. Concretely, they train a character-n-gram model to locate the new word vector near the existing word with similar root. Le and Mikolov (2014) introduced paragraph-level vectors (instead of word-level), a fixed-length feature representations for variable-length texts. When embedding new paragraphs, old paragraph vectors are frozen, and the new ones are updated. Furthermore, Luo et al. (2015) proposed an efficient online method to address the memory issue encountered when learning word embeddings based on nonnegative matrix factorization.

Online SVD We already discussed word embeddings via implicit matrix factorization (IMF) above. This method is based on a truncated SVD on a square matrix whose size is the vocabulary size. As this paper combines this idea with online learning, we review related word on online singular value decompositions.

Online SVD (incremental SVD) is a classical problem in numerical linear algebra (Datta, 2010), and is intensively used in recommendation systems (Sarwar et al., 2002; Brand, 2003) and subspace learning (Li, 2004). Online SVD only possesses an approximate solution. Recently some methods have been proposed to reduce the involved approximation error (Shamir, 2015; Allen-Zhu and Li, 2016) based on iterative learning. In this paper, we use the same online SVD method as in Sarwar et al. (2002), which owns a closed-form solution.

3 Method

We present our spectral word embedding method to efficiently insert new words from an extended vocabulary into pre-trained word embeddings, without having to re-train the model on the extended vocabulary. We first introduce some relevant background with respect to word embedding via implicit matrix factorization (Section 3.1) before presenting our method (Section 3.2) and theoretical consideration (Section 3.3).

Notation In this paper, the vocabulary of the existing pre-trained word embedding is called *base vocabulary*, whose size is m. After adding new words, we call the whole vocabulary the *extended vocabulary*; its size is $n \equiv m + m'$, where m' is the number of unique new words. We assume $m' \ll m$, so $\mathcal{O}(n) = \mathcal{O}(m)$. Furthermore, let d denote the embedding dimension. As will be explained below, let S_0, S_{full} denote the SPPMI matrices of the base and extended vocabularies, respectively. The subscript "full", thus, always refers to the extended vocabulary.

3.1 Background: Word Embedding via Implicit Matrix Factorization (IMF)

The basis of our approach is the skip-gram model with negative sampling (SGNS), also called word2vec (Mikolov et al., 2013a,b). Let D denote the set of all observed of word-context pairs. Furthermore, $\#(w,c)$ denotes the number of times the pair (w,c) appears in D, and $\vec{w}$ and $\vec{c}$ are the word and context embeddings.

The objective that SGNS minimizes is

$$L = \sum_{w \in V_w} \sum_{c \in V_c} \{\#(w,c) \log \sigma(\vec{w} \cdot \vec{c}) + k \cdot \mathbb{E}_{c_N \sim p_D} [\log \sigma(-\vec{w} \cdot \vec{c}_N)]\}. \tag{1}$$

In the limit of large d, Levy and Goldberg (2014) found the following closed-form solution to Eq 1: $x = \vec{w} \cdot \vec{c} = \log\left(\frac{\#(w,c) \cdot |D|}{\#(w)\,\#(c)}\right) - \log k$.

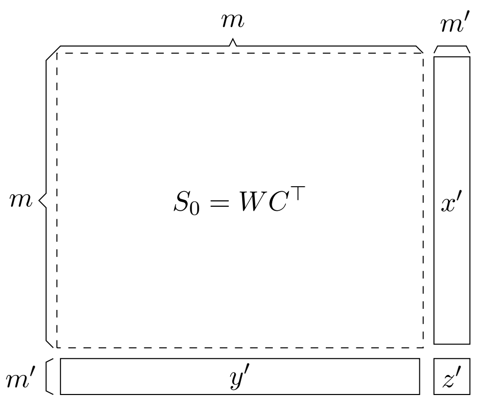


Figure 1: Block-structure of the Pointwise Mutual Information Matrix considered in this paper, and its block-structure for extended vocabulary. It corresponds to Equation (6).

The first term can be seen as an empirical estimate of Pointwise Mutual Information (PMI): $\text{PMI}(w,c) = \log\left(\frac{P(w,c)}{P(w)P(c)}\right)$. Thus, the matrix M that SGNS factorizes can be constructed from $M_{ij} = \text{PMI}(w_i, c_j) - \log k$. For computational convenience, Levy and Goldberg (2014) suggested a sparse and consistent alternative called Shifted Positive PMI (SPPMI):

$$\text{SPPMI}(w,c) = \max(\text{PMI}(w,c) - \log k, 0). \tag{2}$$

Levy and Goldberg (2014) showed that using such sparse representation, word and context embeddings could be efficiently obtained using a truncated singular value decomposition.

As will be explained in the next section, our approach builds on the intuition that word2vec implicitly factorizes the SPPMI matrix. Given pre-trained word and context vectors, we ask for an efficient way of extending the vocabulary and re-adjusting these vectors accordingly.

3.2 SOWE: Spectral Online Word Embedding

Our method takes advantages of the implicit matrix factorization method for efficiently embedding previously unseen words. Given a pre-trained word embedding, we firstly transform it to the SVD form. Using such form, under mild approximation, we can utilize efficient online SVD (Sarwar et al., 2002) to obtain the word embeddings for the extended vocabulary.

Figure 1 presents a sketch of the problem that we want to solve. We start from a $(m+m') \times (m+m')$ matrix in the extended vocabulary space, for which

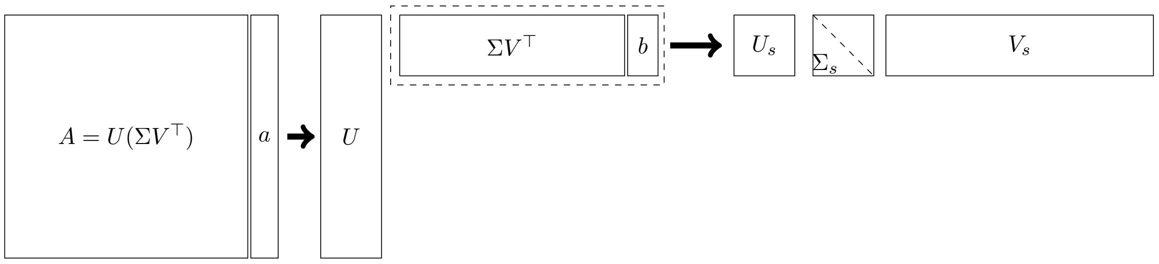


Figure 2: A sketch of the matrix manipulations carried out in online SVD (Algorithm 2). Given the truncated SVD of $A \approx U\Sigma V^\top$ and new columns a, we first compute b, the projection of a on U, i.e., $b = U^\top a$. Then we concatenate $Z = \Sigma V^\top$ and b via $V' = [Z, b]^\top$. Finally, we do rank-d truncated SVD on V', i.e, $[V_s, \Sigma_s, Us] = \text{tSVD}(V', d)$. Final result is $[UU_s, \Sigma_s, V_s] = \text{tSVD}([A, a], d)$.

Algorithm 1 Spectral Online Word Embedding (SOWE)

Input: Old word/context vectors W and C with $S_0 \equiv WC^\top$, co-occurrence matrices involving new vocabulary x', y', and z' (see Fig. 1).

Output: Word/context vectors $W_{\text{full}}, C_{\text{full}}$ for extended vocabulary.

```
1:  **SVD of $WC^\top$ via QR decomposition.**
2:  $U, R_1 \leftarrow \text{QR}(W)$  // $O(md^2)$
3:  $V, R_2 \leftarrow \text{QR}(C)$  // $O(md^2)$
4:  $U_0, \Sigma, V_0 \leftarrow \text{SVD}(R_1 R_2^\top)$ // $O(d^3)$
5:  $U \leftarrow UU_0$  // $O(md^2)$
6:  $V \leftarrow VV_0$  // $O(md^2)$
7:  **Horizontal and vertical Online SVD.**
8:  $U', \Sigma', V' \leftarrow \text{OSVD}([U\Sigma V^\top, x'])$
9:  $U_{\text{full}}, \Sigma_{\text{full}}, V_{\text{full}} \leftarrow \text{OSVD}(\begin{bmatrix} U'\Sigma'V'^\top \\ [y', z'] \end{bmatrix})$.
10: **Return new embeddings.**
11: $W_{\text{full}} \leftarrow U_{\text{full}} \cdot \sqrt{\Sigma_{\text{full}}}$,
12: $C_{\text{full}} \leftarrow V_{\text{full}} \cdot \sqrt{\Sigma_{\text{full}}}$
```

we seek an approximate factorization. We assume that we already have a factorization of the upper-left submatrix of size $m \times m$, implicitly obtained from word2vec or related word embedding algorithms. We seek an efficient linear algebra algorithm that, given the block-structure, results in a factorization of the whole matrix in linear time in m. This will be detailed below.

Overview The following steps summarize our overall proposed procedure.

- We start by assuming that our word embedding algorithm came from a factorization of the pointwise mutual information matrix of word frequencies. Thus, $S_0 \approx WC^\top$.
- In order to make use of efficient only SVD, we need to convert this matrix product into an SVD form. This can be done in $O(m)$ time and results in $WC^\top = U\Sigma V^\top$ (Section 3.2.1).
- Next, we need to estimate all elements in the extended pointwise mutual information matrix, see Figure 1. We first estimate the dominant block (section 3.2.2 (i)), and show that it can be approximated by our previously obtained SVD. We then estimate the remaining blocks (section 3.2.2 (ii)). For the latter, we need to estimate the frequencies of the new words relative to the old words.
- We are now in a position to efficiently compute a new SVD for the extended pointwise mutual information matrix, $U_{\text{full}}\Sigma_{\text{full}}V_{\text{full}}^\top$, using online SVD. The operational costs are still $O(m)$ ((iii) in Section 3.2.2).
- Finally we define our new embedding matrices as $W_{\text{full}} = U_{\text{full}}\sqrt{\Sigma_{\text{full}}}$ and $C_{\text{full}} = V_{\text{full}}\sqrt{\Sigma_{\text{full}}}$, which completes our algorithm.

These steps will be explained in more detail below.

3.2.1 SVD from Word-Context Vectors

The first step in our algorithm is to obtain a singular value decomposition (SVD) of the old vocabulary's approximate PMI matrix S_0. Our working hypothesis is that our pre-trained word and context embedding matrices W and C are already approximately factorizing this matrix,

$$S_0 \approx WC^\top. \tag{3}$$

Levy and Goldberg (2014) showed that this factorization is correct in the limit of a large enough embedding dimension d, but is only approximately true otherwise. In this paper, we will use Eq. 3 as a working hypothesis.

Computing an SVD from S_0 would usually be an operation that costs $O(m^2)$, thus would scale quadratically in the vocabulary size. In such factorization would be not practical, since m is typically of the order of hundred thousands. Instead, we show next that, given a low-rank factorization of S_0 in terms of W and C renders this cost *linear* in the vocabulary size, making such an approach practical. The following procedure corresponds to steps 1-6 in Algorithm 1.

A truncated SVD (tSVD) of S_0 with rank d can be obtained from QR decompositions (Golub and Loan, 1996) of W and C as follows:

$$U', R_1 = \text{QR}(W); \quad V', R_2 = \text{QR}(C)$$

This results in $S_0 = U' R_1 R_2^\top V'^\top$. In a second step, we apply an SVD to $R_1 R_2^\top$:

$$U'', \Sigma, V'' = \text{SVD}(R_1 R_2^\top).$$

The costs of this are small, as $R_{1,2}$ are $d \times d$ matrices. Since the composition of two orthogonal matrices is still orthogonal, we obtain the SVD of S_0 as $U = U'U''$, $V = V'V''$. Note that this transformation is exact since in our approximation, $S_0 = WC^\top$ was already of rank d. Thus, $WC^\top = U\Sigma V^\top$. The complexity of this operation is $O(md^2)$, which concludes the first step.

3.2.2 Utilizing Online SVD to Embed Extended Vocabulary

The next steps amout to adding new words to the old embeddings by adding rows and columns to the original SPPMI matrix, and efficiently factorizing it via online SVD.

Given a representation of the old block of the PMI matrix in terms of an SVD, our next task is to compute the new elements of this matrix that correspond to the extended vocabulary of m' words, with $m' \ll m$. We denote this matrix $S_{\text{full}} \in \mathbb{R}^{(m+m')\times(m+m')}$, and it has the following block structure (see also Fig. 1):

$$S_{\text{full}} \triangleq \begin{pmatrix} S' & x' \\ y'^\top & z' \end{pmatrix}. \tag{4}$$

In the following three steps, we describe how to estimate and efficiently factorize this matrix.

(i) Approximating the main block In a first step, we approximate the main block S' of the SPPMI matrix (Eq. 4). We show that to a first approximation, this is just the SPPMI matrix of the original vocabulary, hence $S' \approx S_0$.

Algorithm 2 Recap: Online SVD (OSVD)

```
Input: Rank-$d$ truncated SVD (tSVD) of $A \in \mathbb{R}^{m\times n}$: $U, \Sigma, V = \text{tSVD}(A, d)$, where $a \in \mathbb{R}^{m\times m'}, U \in \mathbb{R}^{m\times d}, V \in \mathbb{R}^{n\times d}, \Sigma \in \mathbb{R}^{d\times d}$, $\{d, m'\} \ll \{m, n\}$.
Output: Rank-$d$ truncated SVD of $[A, a]$:
 1: $U^*, \Sigma^*, V^* \leftarrow \text{tSVD}([A, a], d)$
 2: Compute projection of new matrix to $U$.
 3: $b \leftarrow U^\top a$.  // $O(ndm')$
 4: $Z \leftarrow \Sigma V^\top$  // $O(nd)$
 5: $V' \leftarrow [Z, b]^\top$.  // $O(nd^2)$
 6: Apply tSVD to projection.
 7: $V_s, \Sigma_s, Us \leftarrow \text{tSVD}(V', d)$.
 8: // $O((n + m')d^2) \approx O(nd^2)$
 9: Return the results.
10: $U^* \leftarrow UU_s, \Sigma^* \leftarrow \Sigma_s, V^* \leftarrow V_s$.  // $O(nd^2)$
```

To set up the SPPMI matrix, the following formula has to be applied to the observed co-occurrence counts $\#(w, c)$ between all word and context words in the extended vocabulary:

$$\max\left\{\log\left(\frac{\#(w,c)\ \cdot |D|}{\#(w)\ \#(c)}\right) - \log k, 0\right\} \tag{5}$$

Besides the co-occurrence counts, this also involves the absolute frequencies $\#(w)$ and $\#(c)$ of words and context vectors in the extended vocabulary, as well as the total number of counts $|D|$. Note that all these quantities enter only on a logarithmic scale.

The co-occurrence counts $\#(w, c)$ are the same for the SPPMI matrices of the original and full vocabularies. What differs slightly are the absolute counts $\#(w)$, $\#(c)$, and $|D|$ (these are slightly higher in the extended corpus). However, since we assumed that the original training corpus was much bigger than the corpus containing the new words, we can safely assume that the change in $\log \#(w)$, $\log \#(c)$, and $\log |D|$ is negligible (we will further specify and analyze this approximation in our section 3.3). Thus, $S' \approx S_0$. Furthermore, since we have shown in section 3.2.1 that $S_0 = U\Sigma V^\top$, this results in

$$S_{\text{full}} \approx \begin{pmatrix} U\Sigma V^\top & x' \\ y'^\top & z' \end{pmatrix}. \tag{6}$$

(ii) Adding rows and columns The matrices $x', y' \in \mathbb{R}^{m\times m'}$ in Eq. 6 are tall-and-skinny matrices that contain information about cross co-ocurrences between old and new words, and $z' \in \mathbb{R}^{m'\times m'}$ are the co-occurrences of new words in the new vocabulary. Next, we will describe how to estimate these quantities, taking into account that we don't have access to the original training corpus that was used to learn the word embeddings of the old vocabulary.

As follows, we focus on x' as an example (estimating y' works analogously). In this case, we observe the co-occurrence counts $\#(w, c)$ between words w from the base vocabulary in the context of context words c from the new vocabulary. To compute the SPPMI (2), we then apply Eq. 5 to all obtained counts. This results in x'. The remaining problem is that $\#(w)$ and $|D|$ are unknown to us, and some heuristics have to be found to circumvent this problem.

First, notice that $\frac{\#(w)}{|D|}$ corresponds to the word frequencies in the original corpus. Furthermore, we are only interested in $\log \frac{\#(w)}{|D|}$. The logarithm is less sensitive to the result of the estimation of this quantity.

When using pre-trained word embeddings, the embedding vectors are typically ranked according to their frequency. We estimated the word frequency based based on their frequencies on the smaller corpus. For words from the old vocabulary that are not present in the new corpus, we interpolated using an exponential model, taking their frequency rankings into account.

Another heuristic has to be found to approximate z', in which case $\#(w)$ and $\#(c)$ are available, but $|D|$ is unknown. Here, we assume that the new words are about as rare as the rarest words in the old vocabulary, setting $\#(w)/|D|$ to the frequency of the least frequent word in corpus. This specifies the extended SPPMI matrix. Next, we show how to efficiently re-factorize it.

(iii) Factorizing the Extended SPPMI Matrix Finally, the approximated SPPMI matrix is efficiently factorized using online SVD.

This can not be carried out in a single step, because the online SVD method sketched in Algorithm 2 only supports the addition *either* rows *or* columns (here presented for columns). Thus, we first concatenate $U\Sigma V^\top$ and x' horizontally and perform a rank d truncated SVD. In a second step, we concatenate the resulting singular value decomposition vertically with the concatenation of y' and z' to obtain a truncated SVD of the full SPPMI matrix. Algorithm 2 gives the details; for more details we refer to (Sarwar et al., 2002). This results in an approximate SVD for the full matrix. Word and context embeddings can be obtained trivially from the SVD.

Finally, let us discuss the complexity of the method. The online SVD subroutine dominates the complexity of our approach, as it scales as $O(nd^2)$. In all steps, the costs remain linear in the vocabulary size. This makes our approach scalable and convenient to use. In contrast, when carrying out an SVD from scratch to compute the word and context embeddings, we would have a quadratic scaling in the vocabulary size, which would be impractical.

3.3 Theoretical Analysis

In this section, we show that under certain assumptions, the difference between approximate SPMMI matrix S'_{full} (Equation 6) and SPMMI matrix S_{full} (Equation 4) is bounded. This justifies the previous assumption that we can substitute S'_{full} for S_{full}. Now we want to show theoretically that this is a reasonable approximation. First, we make some assumptions.

Assumption 1. *There exists a constant $c_1 > 0$ such that number of nonzero (nnz) entries in $m \times m$ SPPMI matrix can be upper-bounded by $c_1 m$, i.e., nnz(SPPMI) $\leq c_1 m$.*

Remark 1. *It is reasonable to assume that in Shifted Positive PPMI matrix, most of the words are only closely related to a small number of other words.*

Assumption 2. *Every entry in co-occurrence matrix can be bounded by $c_2 > 0$, i.e., $\#(w, c) \leq c_2$ for $\forall\ w, c$.*

This is always satisfied, since the number of observed co-occurrences is always bounded.

Assumption 3. *For $m \times m$ SPPMI matrix, there exists a constant $c_3 > 0$ such that the number of w and c occur in corpus D at least $c_3 m$ times, i.e., $\min\{c_i, w_i\} \geq c_3\sqrt{m}$ for $\forall\ i$.*

Now, we provide our main theoretical result.

Theorem 1. *Under Assumption 1, 2 and 3, the gap between S'_{full} and S_{full} in Frobenius norm can be bounded by a constant $c_4 = \frac{3\sqrt{c_1}c_2}{c_3}$, i.e., $\|S'_{full} - S_{full}\|_F \leq c_4$.*

This results implies that the difference between S'_{full} and S_{full} is bounded by a constant independent of the vocabulary size. Since in large-scale word embedding models the size of the vocabulary is typically 10^5 or even 10^6, the relative difference between these two matrices can be negligible. We

Dataset	Method	Loss for new	Loss for all	time
NIPS	FOUN	-1.47± 0.004e5	-1.7026±0.00004e7	103.8
	FOUN+anneal	-1.46±0.006e5	-1.7026±0.00006e7	103.3
	SOWE (ours)	-1.52e5	-1.7031e7	**47.1**
Economic News	FOUN	-8.64±0.007e4	-1.6907±0.000007e7	84.3
	FOUN+anneal	-8.63±0.01e4	-1.6907±0.00001e7	87.8
	SOWE (ours)	-8.65e4	-1.6904e7	**45.79**

Table 1: Performance on NIPS Abstract and Economic News. The unit of running time is second. We report the average value of 10 independent runs for FOUN and standard deviation in brackets only for "loss for new". For FOUN, "loss for all words" is the sum of "loss for new words" and "loss that are only related to old words"(which is a constant). So standard deviation of "loss for all" is equal to "loss for all".

provide the proof of the theorem in supplementary materials.

4 Experiment

In this section, we show empirical results where we compare our proposed SOWE with other continuous word embedding algorithms.We first present some generic settings of our experiments, followed by quantitative and qualitative baseline comparisons. Compared to the baselines, we find that our method is more efficient and finds more semantically meaningful embeddings. Our method takes less than one minute to insert about 1,000 domain-specific words (e.g., machine learning related words from NIPS abstracts) into a pre-trained embedding model with more than 180,000 words.

Experimental Setup Our approach departs from pre-trained word vectors from a generic training corpus. We downloaded publicly available pre-trained word embeddings and two small text corpora from specific domains. Our goal is to insert the domain-specific words that do not already appear in the original vocabulary efficiently into the embeddings.

First, we report on basic settings for our experiments. The pre-trained embedding model based on English Wikipedia is available online [1]. It contains 183,870 words. The embedding dimension is 300. The small, domain-specific corpora that we considered were the following ones: (1) "NIPS Abstracts": this data set contains abstracts of all the NIPS papers from 2014 to 2016. The data set contains 981 new words. (2) "Economy News": This data set contains news articles, containing 868 new words. These two corpora are much smaller than base corpus. For both the base corpus and the new corpus, the text was pre-processed using a window of 5 tokens on each side of the target word, where stop-words and non-textual elements were removed, and sentence splitting and tokenization were applied.

[1] http://u.cs.biu.ac.il/~yogo/data/syntemb/bow5.words.bz2

Baselines: FOUN and FOUN+annealing A natural idea for inserting new words into a pre-estisting word embedding fit is to fix the old word/context vectors, and only to update the new ones. This is our baseline method, referred in the following as "Fix Old, Update New" (**FOUN**). The approach uses the word2vec objective and employs stochastic gradient descent for training. We employ Robbins-Monro learning schedules (Robbins and Monro, 1951), setting the stepsize at the t-th step as $\epsilon_t = a(t+\gamma)^{-0.51}$. We used grid-search to find optimal parameters on all considered data sets, and found $k = 5$, $\gamma = 1e4$, $a = 1/10$ (for different tasks) to be optimal. Due to the involved randomness in the baseline approach, we conducted 10 independent trials using different random seeds for each results and reported the average results. For "FOUN+annealing" we used the same settings as in FOUN, but added random zero-mean Gaussian noise to the gradient. To this end, we employed a version of Stochastic Gradient Langevin Dynamics (Welling and Teh, 2011), where we scaled down the noise by a factor of 0.01.

Loss minimization and runtime We considered the word2vec loss on the extended vocabulary and evaluated the value of the loss function on the embedding vectors obtained form the different methods under consideration. The associated loss values and runtimes are reported in Table 1 for the "NIPS Abstracts", and for "Economy News". We found both approaches yield similar values of the loss function. (In our experiments below, however, we will show that our obtained word vectors seem to reflect the semantics of the original corpus better.) As a clear improvement, we found that our method

Method	Nearest Neighbor
FOUN	joyous haworth legionnaires kristiansand cade dingo gaozu mightywords budged freret pepco
FOUN+annealing	emmerich hotham totnes crescendo emt demesne family-friendly rutter khazars dijon isidro
SOWE (ours)	midcap ultralow jawboning cdw lennar sucres moviefone terest supercenters ious wci

Table 2: Nearest Neighbors of "**eurodollars**" in extended vocabulary.

Method	Nearest Neighbor
FOUN	mattias valmiki invalided mailman malkovich bufo khufu dijon propagating madman
FOUN + annealing	interrogative bouncers jf time-dependent invalidating bmo enameled subgroup brenda anal keller
SOWE (ours)	cannot nonsmooth cnns coreset svrg sublinear denoising lstm interpretability

Table 3: Nearest Neighbors of "**submodular**" in extended vocabulary. We measure the distance using cosine distance.

Method	EN-WS-353-SIM	EN-MTurk-771	EN-MEN-TR-3k	EN-RW-STANFORD	EN-WS-353-ALL
Ideal	69.94	57.45	64.43	43.26	65.14
FOUN	65.13	54.67	57.47	42.99	64.02
FOUN+anneal	64.99	54.89	**57.50**	42.73	64.01
SOWE (ours)	**66.48**	**55.49**	56.56	**42.92**	**64.82**

Table 4: Performance on word similarity task using text8 dataset. "ideal" means the matrix factorization method proposed in Levy and Goldberg (2014). FOUN and FOUN+annealing are the baseline that we are comparing with.

Method	capital-common-countries	nationality-adjective	family.txt	Syntactic
Ideal	49.60%	50.55%	55.23%	30.49%
FOUN	45.73%	50.63%	55.01%	**30.30**%
FOUN + anneal	44.70%	49.97%	**55.03%**	30.26%
SOWE (ours)	**46.43%**	**50.71%**	50.43%	29.38%

Table 5: Performance on word analogy task using text8 dataset. "ideal" means learning the word embedding with the full extended vocabulary from scratch. Here, the matrix factorization method proposed in Levy and Goldberg (2014) is used on the S_{full}. FOUN and FOUN+annealing are the baseline that we are comparing with.

Method	split number	capital-common-countries	nationality-adjective	family.txt	Syntactic
Ideal		64.74%	64.49%	62.32%	39.18%
FOUN		64.45±0.25 %	62.43±0.38%	61.29±0.44%	38.73±0.58 %
FOUN + anneal	20	64.26±0.38%	64.10±0.47 %	61.73±1.83 %	38.94±0.77%
SOWE (ours)		64.26±0.79%	64.22±0.47%	60.09±0.32%	38.20±0.72%
FOUN		59.80±1.27 %	60.93±1.76%	59.23±1.88%	38.47±1.23 %
FOUN + anneal	10	58.84±1.82%	59.93±1.90 %	59.77±2.00%	38.19±1.47%
SOWE (ours)		59.89±3.85%	60.86±1.02%	58.84±0.85%	38.00±1.83%
FOUN		51.88±4.03 %	52.10±2.94%	51.83±3.11%	33.33±2.09 %
FOUN + anneal	5	52.20±6.03%	52.08±3.96 %	54.01±3.88 %	33.84±2.11%
SOWE (ours)		51.14±%3.74	49.50±2.83%	50.84±1.44%	31.82±2.73 %

Table 6: Performance on word analogy task using existing embedding results and text8 dataset with different folds of splits. In the row "Ideal", we use the well-trained word embedding results downloaded from Internet. FOUN and FOUN+annealing are the baseline that we are comparing with.

is faster than the baseline, yielding a factor of 8 times speedup. We consider the baseline method to be converged when the loss value of the current epoch is close (smaller than a threshold) to that of the previous epoch, where the threshold is 1e2 for NIPS abstract and Economics News.

Qualitative Nearest Neighbor Test To test whether the learned embedding vectors are semantically meaningful, we chose some words from the new vocabulary and reported their nearest neighbors in the extended vocabulary. We expect the nearest neighbors to have a close semantic meanings. We chose cosine similarity as a means to measure distance between words.

We chose the words "eurodollars" as a query for "Economic News" and "submodular" for "NIPS Abstracts". The results are reported in Table 2 and 3. In the case of economics, we see that our algorithm recovered meaningful words such as "midcap" and "ultralow". The baseline methods failed to return meaningful results with respect to the query. One possible reason is that the baseline's underlying optimization algorithm got trapped in a poor local optimum. In the case of NIPS abstracts (Table 3), our SOWE method results in words such as "nonsmooth" and "coreset" which are highly related to the query "submodular", while the FOUN-based methods fail. Our approach thus outperforms the baseline in providing meaningful relationships between the pre-trained word vectors and the newly embedded ones. More examples are provided in the Appendix.

Evaluations on NLP Tasks Additionally, we evaluated the proposed method on some downstream NLP tasks, such as pairwise word similarity. To this end, we used datasets that contain word pairs associated with human-assigned similarity scores [2]. The word vectors are evaluated by ranking the pairs according to their cosine similarities, and measuring the correlation (Spearmans ρ) with the human ratings.

We excluded the word pairs of the similarity test from the original vocabulary and trained word2vec with the associated reduced vocabulary on several corpora. We then added the test words using the three competing methods (SOWE, FOUN, and FOUN+anneal). The fourth algorithm "Ideal" amounts to evaluating the test on the generic pre-trained word embeddings from the web. The results on the word similarity task are shown in Table 4. Our method obtains the best performance on four out of five word similarity tasks.

Word analogy tests consists of questions of the form "a is to a* as b is to b*", where b* must be completed (Mikolov et al., 2013b). We performed such word analogy tests; our results are reported in Table 5. We observe that SOWE outperform FOUN-based methods in two out of four cases.

Word Analogy Analysis with Varying Number of Folds We further split the corpus into folds to evaluate the word analogy task, where we varied the size of the folds. Here, we choose the most frequent 20,000 words in text8 and then split the vocabulary into $k \in \{5, 10, 20\}$ folds. All folds but one were considered as base vocabulary, and one fold was considered as new vocabulary. We used implicit matrix factorization on all but one fold, and added the last fold's vocabulary using the different methods under comparison (FOUN, FOUN+anneal and SOWE). We repeated this procedure k times and report means and standard deviations in Table 6. Our method achieves results comparable with the baselines.

5 Conclusion

We proposed a deterministic spectral algorithm for inserting new words into a pre-trained word embedding model. The approach is based on a small corpus (containing the new words) and the pre-trained word embedding vectors. Under well-specified assumptions, this vocabulary extension can be formulated as an online matrix factorization problem. This scheme scales linearly with the original vocabulary size, and quadratically with the embedding dimensions. Compared to baselines that involve optimizing the original word2vec loss with the old word vectors fixed, our method is parameter-free, does not suffer from optimization problems such as local optima, and as such easier to handle. We further provided an analysis on the involved approximation error and showed that it is bounded. While we found only slight improvements over the baseline methods in terms of quality of the word vectors, more work needs to be done to explore tradeoffs of the involved approaches.

[2]We used code and data from https://github.com/k-kawakami/embedding-evaluation

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2016. LazySVD: Even Faster SVD Decomposition Yet Without Agonizing Pain. In *Proceedings of the 30th Conference on Neural Information Processing Systems*, NIPS '16. Full version available at http://arxiv.org/abs/1607.03463.

Robert Bamler and Stephan Mandt. 2017. Dynamic word embeddings. In *International Conference on Machine Learning*, pages 380–389.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Matthew Brand. 2003. Fast online svd revisions for lightweight recommender systems. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 37–46. SIAM.

Biswa Nath Datta. 2010. *Numerical linear algebra and applications*. SIAM.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix computations (3. ed.)*. Johns Hopkins University Press.

Sung Ju Hwang and Leonid Sigal. 2014. A unified semantic embedding: Relating taxonomies and attributes. *Neural Information Processing Systems*, pages 271–279.

Tom Kenter and Maarten De Rijke. 2015. Short text similarity with word embeddings. pages 1411–1420.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

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

Yongmin Li. 2004. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2015. Learning context-sensitive word embeddings with neural tensor skip-gram model. pages 1284–1290.

Hongyin Luo, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2015. Online learning of interpretable word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1687–1692.

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.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Viet Phung and Lance De Vine. 2015. A study on the use of word embeddings and pagerank for vietnamese text summarization. page 7.

Rene Pickhardt, Thomas Gottron, Martin Korner, Paul Georg Wagner, Till Speicher, and Steffen Staab. 2014. A generalized language model as the combination of skipped n-grams and modified kneser ney smoothing. *meeting of the association for computational linguistics*, pages 1145–1154.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407.

Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. 2016. Exponential family embeddings. In *Advances in Neural Information Processing Systems*, pages 478–486.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer.

Ohad Shamir. 2015. A stochastic pca and svd algorithm with an exponential convergence rate. *international conference on machine learning*, pages 144–152.

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.

Haiyang Wu, Daxiang Dong, Xiaoguang Hu, Dianhai Yu, Wei He, Hua Wu, Haifeng Wang, and Ting Liu. 2014. Improve statistical machine translation with context-sensitive bilingual semantic embedding model. pages 142–146.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. pages 1393–1398.

A Appendix

A.1 Proof of Theorem 1

Proof. First, we define some notations. We denote co-occurrence matrix $\#(w, c)$ for S_0 (before adding the new word) and $\#(w', c')$ for S_1 (after adding the new word). We have $|D| = \sum \#(w)$ and $|D'| = \sum \#(w')$. According to Equation (4) and (6), we have

$$\|S_1 - S_1'\|_F = \|S_0 - R'\|_F. \tag{7}$$

We focus on the element-wise difference between S_0 and R', both of them are sparse matrix. For the (i, j)-th element (which is nonzero), we have

$$\begin{aligned}
&(S_0)_{ij} - (R')_{ij} \\
=&\log \#(w_i, c_j) + \log |D| - \log \#(w) - \log \#(c) \\
&- \Big(\log \#(w_i, c_j) + \log |D'| \\
&- \log \#(w') - \log \#(c') \Big) \\
=&\log |D| - \log |D'| - (\log \#(w) - \log \#(w')) \\
&- ((\log \#(c) - \log \#(c')))
\end{aligned} \tag{8}$$

Now we try to upperbound $\log \#(w') - \log \#(w)$ as

$$\begin{aligned}
&\log \#(w') - \log \#(w) \\
=&\log(1 + \frac{\#(w') - \#(w)}{\#(w)}) \\
\leq&\frac{\#(w') - \#(w)}{\#(w)} \\
\leq&\frac{c_2}{c_3\sqrt{m}},
\end{aligned} \tag{9}$$

where the first inequality follows from the fact that $\log(1 + x) \leq x$ for all x and the second inequality uses Assumption 2 and 3. Similarly, we have

$$\begin{aligned}
\log \#(c') - \log \#(c) \leq \frac{c_2}{c_3\sqrt{m}}, \\
\log |D'| - \log |D| \leq \frac{c_2}{c_3\sqrt{m}}.
\end{aligned} \tag{10}$$

Combining the above two equations, we have

$$|(S_0)_{ij} - (R')_{ij}| \leq \frac{3c_2}{c_3\sqrt{m}}.$$

Summing over all nonzero (i, j) pair, we obtain

$$\begin{aligned}
&\|S_1 - S_1'\|_F = \|S_0 - R'\|_F \\
\leq&\sqrt{\max\{\text{nnz}(S_0), \text{nnz}(R')\}\frac{9c_2^2}{c_3^2 m}} \\
\leq&\sqrt{\max\{\text{nnz}(S_0), \text{nnz}(S1)\}\frac{9c_2^2}{c_3^2 m}} \\
\leq&\frac{3\sqrt{c_1}c_2}{c_3}
\end{aligned} \tag{11}$$

Proved.

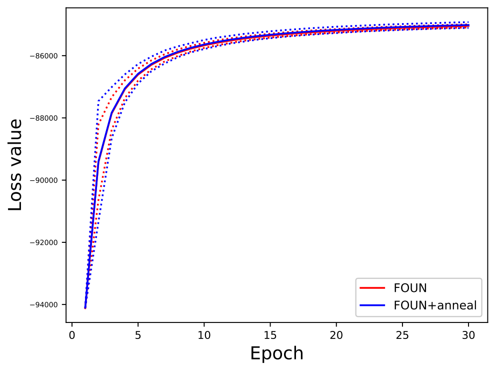
□

A.2 Nearest Neighbor results

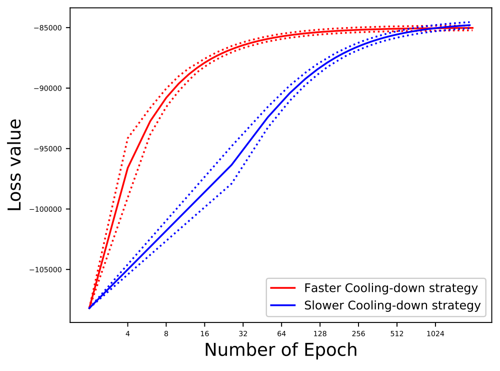
To explore more about our method, we choose some typical domain-specific words and observe their nearest neighbors (in both extended and base vocabulary) after embedding. We choose “ADMM” and “autoencoders” for “NIPS Abstract”. The results are reported in Table 7, 8, 9 and 10. For Economic News, we choose “GDPs” and “reflation”. The results are reported in Table 11, 12, 13 and 14.

A.3 Learning curve of Baseline method

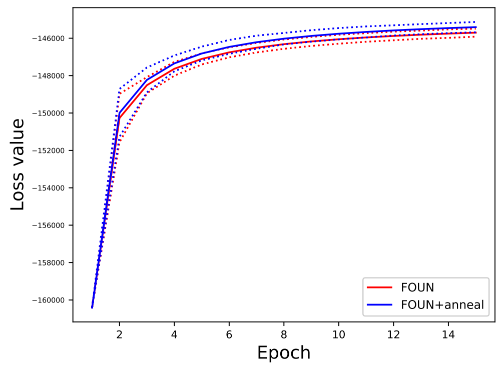
In this section, we provide the learning curve about FOUN and its annealing version in Figure 3. For t-th step, the step size is set to $\epsilon_t = a(t + k)^{-\alpha}$, satisfying Robbins-Monro condition. we compare these two method on several settings and show the case where $a = 1e1$ here. We compare the difference of SGD and Simulated Annealing(SA) on different learning rate for economic NEWS dataset. For each setting we run 10 independent times given the same initial condition and report the average results (solid line) and their 95% confidence interval (dashed line). Red line is the learning curve for SGD while blue line is the curve for SA. For SA method, we shrink the noise by 100 times. We find that the learning curve of simulated annealing has larger variance compared with SGD, validating the fact that the problem is non-convex. Furthermore, we also compare different cooling-down strategies (faster and slower). we find that cooling down slower would produce better results. The result also validates the fact the problem is highly non-convex.



(a) Economic News: optimal learning curve for FOUN and "FOUN+anneal": we find that "FOUN+anneal" outperforms FOUN slightly.



(b) Economic News: comparison of different cooling down strategy. For ease, we show the logorithm of epoches. $\alpha = 0.51$ for faster strategy. For slower one, we use two different annealing strategy for stepsize and noise. Slower strategy would cause a slightly better results than faster one.



(c) NIPS: optimal learning curve for FOUN and "FOUN+anneal": we find that "FOUN+anneal" outperforms FOUN slightly.

Figure 3: Learning curve about FOUN and FOUN+anneal. For t-th step, the step size is set to $\epsilon_t = a(t+k)^{-\alpha}$, satisfying Robbins-Monro condition. We compare these two method on several settings and show the case where $a = 5e0$ and $k = 1e3$. When loss value of the current epoch (denoted L_t) is close (smaller than a threshold, $1e2$) to that of previous epoch (L_{t-1}), we claim that the convergence condition is met. It is the near-optimal. We compare the difference of SGD and Simulated Annealing(SA) on different learning rate for economic NEWS dataset. For each setting we run 10 independent times given the same initial condition and report the average results (solid line) and their 95% confidence interval (dashed line). Red line is the learning curve for SGD while blue line is the curve for SA. For SA method, we shrink the noise by 100 times. We find that the learning curve of simulated annealing has larger variance compared with SGD, validating the fact that the problem is non-convex. Furthermore, we also compare different cooling-down strategies (faster and slower). we find that cooling down slower would produce better results. The result also validates the fact the problem is highly nonconvex.

Method	Nearest Neighbor
FOUN	brandreth blanshard wier bainton lockington goodacre estey stoddard eaton gillham golde champney sackheim ritson chapman pinchbeck moncure furr ellerton stookey
SOWE	thresholding nonlinearly drmm bhinflexible nonsmooth svrg biclustering learnable kernelized deconvolutional coreset vqa laplacians hlinear alexnet lstms

Table 7: Nearest Neighbors of "**ADMM**" in extended vocabulary.

Method	Nearest Neighbor
FOUN	brandreth blanshard wier bainton lockington goodacre estey stoddard eaton gillham golde champney sackheim ritson chapman pinchbeck moncure furr ellerton stookey
SOWE	pagerank throughput scalability latency efficiency maneuverability signal-to-noise conductivity reliability behaved glycemic torque bioavailability dof bandwidth

Table 8: Nearest Neighbors of "**ADMM**" in base vocabulary.

Method	Nearest Neighbor
FOUN	kahala sisa maco meridiana telefonica rassa ctia oyj aquiles balabac looc 3com credito disa sunsilk persil irm ipg ganoderma toco teliasonera dmo netapp cooperativa
SOWE	reparameterization kernelized softmax nonstationary nonsmooth coreset ckjinto avector learnable atakes nmbatch sgrrld regularizing tinflexible coevolving drmm

Table 9: Nearest Neighbors of "**autoencoders**" in extended vocabulary.

Method	Nearest Neighbor
FOUN	kahala sisa maco meridiana telefonica rassa ctia oyj aquiles balabac looc 3com credito disa sunsilk persil irm ipg ganoderma toco teliasonera dmo netapp cooperativa caja
SOWE	conclusion mindset probability next realization isomorphicalgebra technique stripped-down locality graph probabilistic nsongs extent idea mentality scaled-down latest formalization dystopian heuristics algorithm

Table 10: Nearest Neighbors of "**autoencoders**" in base vocabulary.

Method	Nearest Neighbor
FOUN	kadai papaver gaura bhang dioscorea boletes zanthoxylum tetraploid jawboning ascomycetes tremella basidiomycetes cordyceps kalaripayattu flavonoid shakha basidiomycota
SOWE	retrenching laggards downsizings nonfinancial strassel itemize rapcan countertrade overinvestment napodano cordant questech gobbling cyclicals mightywords

Table 11: Nearest Neighbors of "**GDPs**" in extended vocabulary.

Method	Nearest Neighbor
FOUN	kadai papaver gaura bhang dioscorea boletes zanthoxylum tetraploid ascomycetes tremella basidiomycetes cordyceps kalaripayattu flavonoid shakha basidiomycota
SOWE	initials copyrights signatory merits offspring attest contents ashamed signatories drillia remarry notable confided divest prefixed schilpp pseudonyms extent restates pronounces

Table 12: Nearest Neighbors of "**GDPs**" in base vocabulary.

Method	Nearest Neighbor
FOUN	kacher gohde overstocked formspring gibraltarpedia ruess doggystyle mick badfinger sheodred seeyou dreamhost smosh weebl duddy sahaj jtrainor collegehumor monahan
ours	firming shakeout sprinkel tion countertrade sluggishness dovish tiie orszag pessimists tlie jawboning recessionary bankshares flation refunding pretax junkins redemptions

Table 13: Nearest Neighbors of "**reflation**" in extended vocabulary.

Method	Nearest Neighbor
FOUN	kacher gohde formspring gibraltarpedia ruess doggystyle mick badfinger sheodred seeyou dreamhost smosh weebl duddy sahaj jtrainor collegehumor monahan dionyseus
SOWE	globalisation overpopulation warming soros globalization newsround carbon-neutral anata autarky reaganomics plunk splat clurman imbalance anti-semitism

Table 14: Nearest Neighbors of "**reflation**" in base vocabulary.