A Generative Word Embedding Model and its Low Rank Positive Semidefinite Solution

Shaohua Li¹, Jun Zhu², Chunyan Miao¹

¹Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), Nanyang Technological University, Singapore
²Tsinghua University, P.R. China
lish0018@ntu.edu.sg, dcszj@tsinghua.edu.cn, ascymiao@ntu.edu.sg

Abstract

Most existing word embedding methods can be categorized into Neural Embedding Models and Matrix Factorization (MF)-based methods. However some models are opaque to probabilistic interpretation, and MF-based methods, typically solved using Singular Value Decomposition (SVD), may incur loss of corpus information. In addition, it is desirable to incorporate global latent factors, such as topics, sentiments or writing styles, into the word embedding model. Since generative models provide a principled way to incorporate latent factors, we propose a generative word embedding model, which is easy to interpret, and can serve as a basis of more sophisticated latent factor models. The model inference reduces to a low rank weighted positive semidefinite approximation problem. Its optimization is approached by eigendecomposition on a submatrix, followed by online blockwise regression, which is scalable and avoids the information loss in SVD. In experiments on 7 common benchmark datasets, our vectors are competitive to word2vec, and better than other MF-based methods.

1 Introduction

The task of word embedding is to model the distribution of a word and its context words using their corresponding vectors in a Euclidean space. Then by doing regression on the relevant statistics derived from a corpus, a set of vectors are recovered which best fit these statistics. These vectors, commonly referred to as the embeddings, capture semantic/syntactic regularities between the words.

The core of a word embedding method is the link function that connects the input — the embeddings, with the output — certain corpus statistics. Based on the link function, the objective function is developed. The reasonableness of the link function impacts the quality of the obtained embeddings, and different link functions are amenable to different optimization algorithms, with different scalability. Based on the forms of the link function and the optimization techniques, most methods can be divided into two classes: the traditional neural embedding models, and more recent low rank matrix factorization methods.

The neural embedding models use the softmax link function to model the conditional distribution of a word given its context (or vice versa) as a function of the embeddings. The normalizer in the softmax function brings intricacy to the optimization, which is usually tackled by gradient-based methods. The pioneering work was (Bengio et al., 2003). Later Mnih and Hinton (2007) propose three different link functions. However there are interaction matrices between the embeddings in all these models, which complicate and slow down the training, hindering them from being trained on huge corpora. Mikolov et al. (2013a) and Mikolov et al. (2013b) greatly simplify the conditional distribution, where the two embeddings interact directly. They implemented the well-known “word2vec”, which can be trained efficiently on huge corpora. The obtained embeddings show excellent performance on various tasks.

Low-Rank Matrix Factorization (MF in short) methods include various link functions and optimization methods. The link functions are usually not softmax functions. MF methods aim to reconstruct certain corpus statistics matrix by the product of two low rank factor matrices. The objective is usually to minimize the reconstruction error, optionally with other constraints. In this line of research, Levy and Goldberg (2014b) find that “word2vec” is essentially doing stochastic weighted factorization of the word-context pointwise mutual information (PMI) matrix. They then...
Pennington et al. (2014) propose a bilinear regression function of the conditional distribution, from which a weighted MF problem on the bigram log-frequency matrix is formulated. Gradient Descent is used to find the embeddings. Recently, based on the intuition that words can be organized in semantic hierarchies, Yogatama et al. (2015) add hierarchical sparse regularizers to the matrix reconstruction error. With similar techniques, Faruqui et al. (2015) reconstruct a set of pretrained embeddings using sparse vectors of greater dimensionality. Dhillon et al. (2015) apply Canonical Correlation Analysis (CCA) to the word matrix and the context matrix, and use the canonical correlation vectors between the two matrices as word embeddings. Stratos et al. (2014) and Stratos et al. (2015) assume a Brown language model, and prove that doing CCA on the bigram occurrences is equivalent to finding a transformed solution of the language model. Arora et al. (2015) assume there is a hidden discourse vector on a random walk, which determines the distribution of the current word. The slowly evolving discourse vector puts a constraint on the embeddings in a small text window. The maximum likelihood estimate of the embeddings within this text window approximately reduces to a squared norm objective.

There are two limitations in current word embedding methods. The first limitation is, all MF-based methods map words and their context words to two different sets of embeddings, and then employ Singular Value Decomposition (SVD) to obtain a low rank approximation of the word-context matrix $M$. As SVD factorizes $M^\top M$, some information in $M$ is lost, and the learned embeddings may not capture the most significant regularities in $M$. Appendix A gives a toy example on which SVD does not work properly.

The second limitation is, a generative model for documents parameterized by embeddings is absent in recent development. Although (Stratos et al., 2014; Stratos et al., 2015; Arora et al., 2015) are based on generative processes, the generative processes are only for deriving the local relationship between embeddings within a small text window, leaving the likelihood of a document undefined. In addition, the learning objectives of some models, e.g. (Mikolov et al., 2013b, Eq.1), even have no clear probabilistic interpretation. A generative word embedding model for documents is not only easier to interpret and analyze, but more importantly, provides a basis upon which document-level global latent factors, such as document topics (Wallach, 2006), sentiments (Lin and He, 2009), writing styles (Zhao et al., 2011b), can be incorporated in a principled manner, to better model the text distribution and extract relevant information.

Based on the above considerations, we propose to unify the embeddings of words and context words. Our link function factorizes into three parts: the interaction of two embeddings capturing linear correlations of two words, a residual capturing nonlinear or noisy correlations, and the unigram priors. To reduce overfitting, we put Gaussian priors on embeddings and residuals, and apply Jelinek-Mercer Smoothing to bigrams. Furthermore, to model the probability of a sequence of words, we assume that the contributions of more than one context word approximately add up. Thereby a generative model of documents is constructed, parameterized by embeddings and residuals. The learning objective is to maximize the corpus likelihood, which reduces to a weighted low-rank positive semidefinite (PSD) approximation problem of the PMI matrix. A Block Coordinate Descent algorithm is adopted to find an approximate solution. This algorithm is based on Eigendecomposition, which avoids information loss in SVD, but brings challenges to scalability. We then exploit the sparsity of the weight matrix and implement an efficient online blockwise regression algorithm. On seven benchmark datasets covering similarity and analogy tasks, our method achieves competitive and stable performance.

The source code of this method is provided at https://github.com/askerlee/topicvec.

2 Notations and Definitions

Throughout the paper, we always use a uppercase bold letter as $S, V$ to denote a matrix or set, a lowercase bold letter as $v_w$, to denote a vector, a normal uppercase letter as $N, W$ to denote a scalar constant, and a normal lowercase letter as $s_i, w_i$ to denote a scalar variable.

Suppose a vocabulary $S = \{s_1, \cdots, s_W \}$ consists of all the words, where $W$ is the vocabulary size. We further suppose $s_1, \cdots, s_W$ are sorted in descending order of the frequency, i.e. $s_1$ is most frequent, and $s_W$ is least frequent. A document $d_i$ is a sequence of words $d_i = (w_{i1}, \cdots, w_{iL_i}), w_{ij} \in S$. A corpus is a collec-
Table 1: Notation Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Vocabulary {s_1, \cdots, s_W}</td>
</tr>
<tr>
<td>V</td>
<td>Embedding matrix ( (v_{s_1}, \cdots, v_{s_W}) )</td>
</tr>
<tr>
<td>D</td>
<td>Corpus {d_1, \cdots, d_M}</td>
</tr>
<tr>
<td>(v_{s_i})</td>
<td>Embedding of word (s_i)</td>
</tr>
<tr>
<td>(a_{s_i,s_j})</td>
<td>Bigram residual for (s_i, s_j)</td>
</tr>
<tr>
<td>(P(s_i,s_j))</td>
<td>Empirical probability of (s_i, s_j) in the corpus</td>
</tr>
<tr>
<td>u</td>
<td>Unigram probability vector ( (P(s_1), \cdots, P(s_W)) )</td>
</tr>
<tr>
<td>A</td>
<td>Residual matrix ( (a_{s_i,s_j}) )</td>
</tr>
<tr>
<td>B</td>
<td>Conditional probability matrix ( P(s_j</td>
</tr>
<tr>
<td>G</td>
<td>PMI matrix ( {\text{PMI}(s_i, s_j)} )</td>
</tr>
<tr>
<td>H</td>
<td>Bigram empirical probability matrix ( {\tilde{P}(s_i, s_j)} )</td>
</tr>
</tbody>
</table>

In a document, a sequence of words is referred to as a text window, denoted by \(w_{i-1}, \cdots, w_i\), or \(w_i; w_{i+1}\) in shorthand. A text window of chosen size \(c\) before a word \(w_i\) defines the context of \(w_i\) as \(w_{i-c}, \cdots, w_{i-1}\). Here \(w_i\) is referred to as the focus word. Each context word \(w_{i-j}\) and the focus word \(w_i\) comprise a bigram \(w_{i-j}, w_i\).

The Pointwise Mutual Information between two words \(s_i, s_j\) is defined as
\[
\text{PMI}(s_i, s_j) = \log \frac{P(s_i, s_j)}{P(s_i)P(s_j)}.
\]

3 Link Function of Text

In this section, we formulate the probability of a sequence of words as a function of their embeddings. We start from the link function of bigrams, which is the building blocks of a long sequence. Then this link function is extended to a text window with \(c\) context words, as a first-order approximation of the actual probability.

3.1 Link Function of Bigrams

We generalize the link function of “word2vec” and “GloVe” to the following:
\[
P(s_i, s_j) = \exp \left\{ v_{s_j}^\top v_{s_i} + a_{s_i,s_j} \right\} P(s_i)P(s_j) \quad (1)
\]

The rationale for (1) originates from the idea of the Product of Experts in (Hinton, 2002). Suppose different types of semantic/syntactic regularities between \(s_i\) and \(s_j\) are encoded in different dimensions of \(v_{s_i}, v_{s_j}\). As \(\exp(v_{s_j}^\top v_{s_i}) = \prod_l \exp\{v_{s_j,l} \cdot v_{s_i,l}\}\), this means the effects of different regularities on the probability are combined by multiplying together. If \(s_i\) and \(s_j\) are independent, their joint probability should be \(P(s_i)P(s_j)\). In the presence of correlations, the actual joint probability \(P(s_i, s_j)\) would be a scaling of it. The scale factor reflects how much \(s_i\) and \(s_j\) are positively or negatively correlated. Within the scale factor, \(v_{s_j}^\top v_{s_i}\) captures linear interactions between \(s_i\) and \(s_j\), the residual \(a_{s_i,s_j}\) captures nonlinear or noisy interactions. In applications, only \(v_{s_j}^\top v_{s_i}\) is of interest. Hence the bigger magnitude \(v_{s_j}^\top v_{s_i}\) is relative to \(a_{s_i,s_j}\), the better.

Note that we do not assume \(a_{s_i,s_j} = a_{s_j,s_i}\). This provides the flexibility \(P(s_i, s_j) \neq P(s_j, s_i)\), agreeing with the asymmetry of bigrams in natural languages. At the same time, \(v_{s_j}^\top v_{s_i}\) imposes a symmetric part between \(P(s_i, s_j)\) and \(P(s_j, s_i)\).

(1) is equivalent to
\[
P(s_j|s_i) = \exp \left\{ v_{s_j}^\top v_{s_i} + a_{s_i,s_j} + \log P(s_j) \right\}, \quad (2)
\]
\[
\log \frac{P(s_j|s_i)}{P(s_j)} = v_{s_j}^\top v_{s_i} + a_{s_i,s_j}. \quad (3)
\]

(3) of all bigrams is represented in matrix form:
\[
V^\top V + A = G,
\]
where \(G\) is the PMI matrix.

3.1.1 Gaussian Priors on Embeddings

When (1) is employed on the regression of empirical bigram probabilities, a practical issue arises: more and more bigrams have zero frequency as the constituting words become less frequent. A zero-frequency bigram does not necessarily imply negative correlation between the two constituting words; it could simply result from missing data. But in this case, even after smoothing, (1) will force \(v_{s_j}^\top v_{s_i} + a_{s_i,s_j}\) to be a big negative number, making \(v_{s_i}\) overly long. The increased magnitude of embeddings is a sign of overfitting.

To reduce overfitting of embeddings of infrequent words, we assign a Spherical Gaussian prior \(\mathcal{N}\left(0, \frac{1}{2\mu_i} I\right)\) to \(v_{s_i}\):
\[
P(v_{s_i}) \sim \exp\{-\mu_i \|v_{s_i}\|^2\},
\]
where the hyperparameter \(\mu_i\) increases as the frequency of \(s_i\) decreases.

3.1.2 Gaussian Priors on Residuals

We wish \(v_{s_j}^\top v_{s_i}\) in (1) captures as much correlation as possible. Thus the smaller \(a_{s_i,s_j}\) is, the better. In addition, the more frequent \(s_i, s_j\) is in the corpus, the less noise there is in their empirical distribution, and thus the residual \(a_{s_i,s_j}\) should be more heavily penalized.
To this end, we penalize the residual $a_{s_i,s_j}$ by $f(\tilde{P}(s_i, s_j))a_{s_i,s_j}^2$, where $f(\cdot)$ is a nonnegative monotonic transformation, referred to as the weighting function. Let $h_{ij}$ denote $\tilde{P}(s_i, s_j)$, then the total penalty of all residuals are the square of the weighted Frobenius norm of $A$:
\[
\sum_{s_i,s_j \in S} f(h_{ij})a_{s_i,s_j}^2 = \|A\|_f^2(H), \tag{5}
\]
By referring to “GloVe”, we use the following weighting function, and find it performs well:
\[
f(h_{ij}) = \begin{cases} \frac{\sqrt{h_{ij}}}{C_{\text{cut}}} & \sqrt{h_{ij}} < C_{\text{cut}}, i \neq j \\ 1 & \sqrt{h_{ij}} \geq C_{\text{cut}}, i \neq j \\ 0 & i = j \end{cases},
\]
where $C_{\text{cut}}$ is chosen to cut the most frequent 0.02% of the bigrams off at 1. When $s_i = s_j$, two identical words usually have much higher probability to collocate. Hence $\tilde{P}(s_i, s_j)$ does not reflect the true correlation of a word to itself, and should not put constraints to the embeddings. We eliminate their effects by setting $f(h_{ii})$ to 0.

If the domain of $A$ is the whole space $R^{W \times W}$, then this penalty is equivalent to a Gaussian prior $\mathcal{N}(0, \frac{1}{2f(h_{ij})})$ on each $a_{s_i,s_j}$. The variances of the Gaussians are determined by the bigram empirical probability matrix $H$.

3.1.3 Jelinek-Mercer Smoothing of Bigrams
As another measure to reduce the impact of missing data, we apply the commonly used Jelinek-Mercer Smoothing (Zhai and Lafferty, 2004) to smooth the empirical conditional probability $\tilde{P}(s_j | s_i)$ by the unigram probability $\bar{P}(s_j)$ as:
\[
\tilde{P}_{\text{smoothed}}(s_j | s_i) = (1 - \kappa) \tilde{P}(s_j | s_i) + \kappa \bar{P}(s_j). \tag{6}
\]
Accordingly, the smoothed bigram empirical joint probability is defined as
\[
\tilde{P}(s_i, s_j) = (1 - \kappa) \tilde{P}(s_i, s_j) + \kappa \bar{P}(s_i) \bar{P}(s_j). \tag{7}
\]
In practice, we find $\kappa = 0.02$ yields good results. When $\kappa \geq 0.04$, the obtained embeddings begin to degrade with $\kappa$, indicating that smoothing distorts the true bigram distributions.

3.2 Link Function of a Text Window
In the previous subsection, a regression link function of bigram probabilities is established. In this section, we adopt a first-order approximation based on Information Theory, and extend the link function to a longer sequence $w_0, \ldots, w_{c-1}, w_c$.

Decomposing a distribution conditioned on $n$ random variables as the conditional distributions on its subsets roots deeply in Information Theory. This is an intricate problem because there could be both (pointwise) redundant information and (pointwise) synergistic information among the conditioning variables (Williams and Beer, 2010). They are both functions of the PMI. Based on an analysis of the complementing roles of these two types of pointwise information, we assume they are approximately equal and cancel each other when computing the pointwise interaction information. See Appendix B for a detailed discussion.

Following the above assumption, we have
\[
\log \frac{P(w_0, w_1 | w_2)}{P(w_0, w_1)} \approx \log \left( \frac{P(w_0 | w_2)}{P(w_0)} + \frac{P(w_1 | w_2)}{P(w_1)} \right).
\]
Plugging (1) and (3) into the above, we obtain
\[
P(w_0, w_1, w_2) 
\approx \exp \left\{ \frac{2}{i,j=0 \atop i \neq j} (\mathbf{v}_{w_i}^\top \mathbf{v}_{w_j} + a_{w_i,w_j}) + 2 \sum_{i=0}^c \log P(w_i) \right\}.
\]
We extend the above assumption to that the pointwise interaction information is still close to 0 within a longer text window. Accordingly the above equation extends to a context of size $c > 2$:
\[
P(w_0, \ldots, w_c) 
\approx \exp \left\{ \sum_{i,j=0 \atop i \neq j}^c (\mathbf{v}_{w_i}^\top \mathbf{v}_{w_j} + a_{w_i,w_j}) + \sum_{i=0}^c \log P(w_i) \right\}.
\]
From it derives the conditional distribution of $w_c$, given its context $w_0, \ldots, w_{c-1}$:
\[
P(w_c | w_0 : w_{c-1}) = \frac{P(w_0, \ldots, w_c)}{P(w_0, \ldots, w_{c-1})} 
\approx P(w_c) \exp \left\{ \mathbf{v}_{w_c}^\top \sum_{i=0}^{c-1} \mathbf{v}_{w_i} + \sum_{i=0}^{c-1} a_{w_i,w_c} \right\}. \tag{8}
\]

4 Generative Process and Likelihood
We proceed to assume the text is generated from a Markov chain of order $c$, i.e., a word only depends on words within its context of size $c$. Given the hyperparameter $\mu = (\mu_1, \ldots, \mu_W)$, the generative process of the whole corpus is:
1. For each word $s_i$, draw the embedding $\mathbf{v}_{s_i}$ from $\mathcal{N}(0, \frac{1}{2\mu_1} I)$;
2. For each bigram $s_i, s_j$, draw the residual $a_{s_i,s_j}$ from $\mathcal{N}(0, \frac{1}{2f(h_{ij})})$;
3. For each document $d_i$, for the $j$-th word, draw word $w_{ij}$ from $S$ with probability $P(w_{ij} | w_{i,j-c} : w_{i,j-1})$ defined by (8).
The above generative process for a document \(d\) is presented as a graphical model in Figure 1.

Based on this generative process, the probability of a document \(d\) can be derived as follows, given the embeddings and residuals \(V, A\):

\[
P(d_i | V, A) = \prod_{j=1}^{L_i} P(w_{ij}) \exp \left\{ v_{w_{ij}}^\top \sum_{k=j-c}^{j-1} v_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\}.
\]

The complete-data likelihood of the corpus is:

\[
p(D, V, A) = \prod_{i=1}^{W} \mathcal{N}(0, \frac{I}{2\mu_i}) \prod_{i,j=1}^{W} \mathcal{N} \left( 0, \frac{1}{2f(h_{ij})} \right) \prod_{i=1}^{M} p(d_i | V, A)
\]

\[
= \frac{1}{Z(H, \mu)} \exp \left\{ -\sum_{i,j=1}^{W,W} f(h_{ij}) a_{s_i{s_j}}^2 - \sum_{i=1}^{W} \mu_i \| v_{s_i} \|^2 \right\}
\]

\[
\times \prod_{i,j=1}^{M,L_i} P(w_{ij}) \exp \left\{ v_{w_{ij}}^\top \sum_{k=j-c}^{j-1} v_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\},
\]

where \(Z(H, \mu)\) is the normalizing constant.

Taking the logarithm of both sides of \(p(D, A, V)\) yields

\[
\log p(D, V, A)
\]

\[
= C_0 - \log Z(H, \mu) - \| A \|^2_{f(H)} - \sum_{i=1}^{W} \mu_i \| v_{s_i} \|^2
\]

\[
+ \sum_{i,j=1}^{M,L_i} \left\{ v_{w_{ij}}^\top \sum_{k=j-c}^{j-1} v_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\},
\]

where \(C_0 = \sum_{i,j=1}^{M,L_i} \log P(w_{ij})\) is constant.

5 Learning Algorithm

5.1 Learning Objective

The learning objective is to find the embeddings \(V\) that maximize the corpus log-likelihood (9).

Let \(x_{ij}\) denote the (smoothed) frequency of bigram \(s_i, s_j\) in the corpus. Then (9) is sorted as:

\[
\log p(D, V, A)
\]

\[
= C_0 - \log Z(H, \mu) - \| A \|^2_{f(H)} - \sum_{i=1}^{W} \mu_i \| v_{s_i} \|^2
\]

\[
+ \sum_{i,j=1}^{W,W} x_{ij} (v_{s_i}^\top v_{s_j} + a_{s_i{s_j}}).
\]

\[
= \left( \sum_{i,j=1}^{W,W} x_{ij} \right) \cdot \max \left\{ \tilde{P}_{\text{smoothed}}(s_i, s_j) \log P(s_i, s_j) \right\}
\]

As the corpus size increases, \(\sum_{i,j=1}^{W,W} x_{ij} (v_{s_i}^\top v_{s_j} + a_{s_i{s_j}})\) will dominate the parameter prior terms. Then we can ignore the prior terms when maximizing (10).

\[
\max \sum_{i,j=1}^{W,W} x_{ij} (v_{s_i}^\top v_{s_j} + a_{s_i{s_j}})
\]

\[
= \left( \sum_{i,j=1}^{W,W} x_{ij} \right) \cdot \max \left\{ \tilde{P}_{\text{smoothed}}(s_i, s_j) \log P(s_i, s_j) \right\}.
\]

The maximum likelihood estimator is then:

\[
P(s_j | s_i) = \tilde{P}_{\text{smoothed}}(s_j | s_i),
\]

\[
v_{s_j}^\top v_{s_j} + a_{s_i{s_j}} = \log \frac{\tilde{P}_{\text{smoothed}}(s_j | s_i)}{P(s_j | s_i)}.
\]

Writing (11) in matrix form:

\[
B^* = \left( \tilde{P}_{\text{smoothed}}(s_j | s_i) \right)_{s_i, s_j \in S}
\]

\[
G^* = \log B^* - \log u \odot (1 \cdots 1),
\]

where “\(\odot\)” is the outer product.

Now we fix the values of \(v_{s_j}^\top v_{s_j} + a_{s_i{s_j}}\) at the above optimal. The corpus likelihood becomes

\[
\log p(D, V, A) = C_1 - \| A \|^2_{f(H)} - \sum_{i=1}^{W} \mu_i \| v_{s_i} \|^2,
\]

subject to \( V^\top V + A = G^* \),

where \(C_1 = C_0 + \sum_{i,j=1}^{W} x_{ij} \log \tilde{P}_{\text{smoothed}}(s_i, s_j) - \log Z(H, \mu)\) is constant.

5.2 Learning \(V\) as Low Rank PSD Approximation

Once \(G^*\) has been estimated from the corpus using (12), we seek \(V\) that maximizes (13). This is to find the maximum a posteriori (MAP) estimates of \(V, A\) that satisfy \(V^\top V + A = G^*\). Applying this constraint to (13), we obtain
Algorithm 1 BCD algorithm for finding a unregularized rank-$N$ weighted PSD approximant.

**Input**: matrix $G^*$, weight matrix $W = f(H)$, iteration number $T$, rank $N$

Randomly initialize $X^{(0)}$

for $t = 1, \ldots, T$ do

$G_t = W \circ G^* + (1 - W) \circ X^{(t-1)}$

$X^{(t)} = \text{PSD\_Approximate}(G_t, N)$

end for

$\lambda, Q = \text{Eigen\_Decomposition}(X^{(T)})$

$V^* = \text{diag}(\lambda^2 (1:N)) \cdot Q^{\top}[1:N]$

**Output**: $V^*$

\[
\arg\max_V \log p(D, V, A) = \arg\min_V \|G^* - V^\top V\|_{f(H)} + \sum_{i=1}^{W} \mu_i \|v_{si}\|^2. \tag{14}
\]

Let $X = V^\top V$. Then $X$ is positive semidefinite of rank $N$. Finding $V$ that minimizes (14) is equivalent to finding a rank-$N$ weighted positive semidefinite approximant $X$ of $G^*$, subject to Tikhonov regularization. This problem does not admit an analytic solution, and can only be solved using local optimization methods.

First we consider a simpler case where all the words in the vocabulary are enough frequent, and thus Tikhonov regularization is unnecessary. In this case, we set $\mu_i = 0$, and (14) becomes an unregularized optimization problem. We adopt the Block Coordinate Descent (BCD) algorithm\(^1\) in (Srebro et al., 2003) to approach this problem. The original algorithm is to find a generic rank-$N$ matrix for a weighted approximation problem, and we tailor it by constraining the matrix within the positive semidefinite manifold.

We summarize our learning algorithm in Algorithm 1. Here “$\circ$” is the entry-wise product. We suppose the eigenvalues $\lambda$ returned by Eigen\_Decomposition($X$) are in descending order. $Q^{\top}[1:N]$ extracts the 1 to $N$ rows from $Q^{\top}$.

One key issue is how to initialize $X$. Srebro et al. (2003) suggest to set $X^{(0)} = G^*$, and point out that $X^{(0)} = 0$ is far from a local optimum, thus requires more iterations. However we find $G^*$ is also far from a local optimum, and this setting converges slowly too. Setting $X^{(0)} = G^* / 2$ usually yields a satisfactory solution in a few iterations.

The subroutine PSD\_Approximate() computes the unweighted nearest rank-$N$ PSD approximation, measured in F-norm (Higham, 1988).

5.3 Online Blockwise Regression of $V$

In Algorithm 1, the essential subroutine PSD\_Approximate() does eigendecomposition on $G_t$, which is dense due to the logarithm transformation. Eigendecomposition on a $W \times W$ dense matrix requires $O(W^3)$ space and $O(W^3)$ time, difficult to scale up to a large vocabulary. In addition, the majority of words in the vocabulary are infrequent, and Tikhonov regularization is unnecessary for them.

It is observed that, as words become less frequent, fewer and fewer words appear around them to form bigrams. Remind that the vocabulary $S = \{ s_1, \ldots, s_W \}$ are sorted in descending order of the frequency, hence the lower-right blocks of $H$ and $f(H)$ are very sparse, and cause these blocks in (14) to contribute much less penalty relative to other regions. Therefore these blocks could be ignored when doing regression, without sacrificing too much accuracy. This intuition leads to the following online blockwise regression.

The basic idea is to select a small set (e.g. 30,000) of the most frequent words as the core words, and partition the remaining noncore words into sets of moderate sizes. Bigrams consisting of two core words are referred to as core bigrams, which correspond to the top-left blocks of $G$ and $f(H)$. The embeddings of core words are learned approximately using Algorithm 1, on the top-left blocks of $G$ and $f(H)$. Then we fix the embeddings of core words, and find the embeddings of each set of noncore words in turn. After ignoring the lower-right regions of $G$ and $f(H)$ which correspond to bigrams of two noncore words, the quadratic terms of noncore embeddings are ignored. Consequently, finding these embeddings becomes a weighted ridge regression problem, which can be solved efficiently in closed-form. Finally we combine all embeddings to get the embeddings of the whole vocabulary. The details are as follows:

1. Partition $S$ into $K$ consecutive groups $S_1, \ldots, S_K$. Take $K = 3$ as an example. The first group is core words;
2. Accordingly partition $G$ into $K \times K$ blocks,
in this example as \[
\begin{pmatrix}
G_{11} & G_{12} & G_{13} \\
G_{21} & G_{22} & G_{23} \\
G_{31} & G_{32} & G_{33}
\end{pmatrix}.
\]
Partition \( f(H),A \) in the same way. \( G_{11}, f(H)_{11}, A_{11} \) correspond to core bigrams. Partition \( V \) into \( \begin{pmatrix} V_1 & V_2 & V_3 \end{pmatrix} s_1 s_2 s_3 \);

3. Solve \( V_1^T V_1 + A_{11} = G_{11} \) using Algorithm 1, and obtain core embeddings \( V_1^* \);

4. Set \( V_1 = V_1^* \), and find \( V_2 \) that minimizes the total penalty of the 12-th and 21-th blocks of residuals (the 22-th block is ignored due to its high sparsity):

\[
\min_{V_2} \|G_{12} - V_1^T V_2\|^2_{f(H)_{12}} + \|G_{21} - V_2^T V_1\|^2_{f(H)_{21}} + \sum_{s_i \in S_2} \mu_i \|v_{s_i}\|^2
\]

where \( \bar{f}(H)_{12} = f(H)_{12} + f(H)_{21}; \)

\( \bar{G}_{12} = (G_{12} \circ f(H)_{12} + G_{21} \circ f(H)_{21}) \)

\(/(f(H)_{12} + f(H)_{21}) \) is the weighted average of \( G_{12} \) and \( G_{21} \), \( \cdot \circ \). “/” and “\( \cdot \) are element-wise product and division, respectively. The columns in \( V_2 \) are independent, thus for each \( v_{s_i} \), it is a separate weighted ridge regression problem, whose solution is (Holland, 1973):

\[
v_{s_i} = (V_1^T \text{diag}(\bar{f}_i) V_1 + \mu_i I)^{-1} V_1^T \text{diag}(\bar{f}_i) \hat{g}_i,
\]

where \( \bar{f}_i \) and \( \hat{g}_i \) are columns corresponding to \( s_i \) in \( \bar{f}(H)_{12} \) and \( \bar{G}_{12} \), respectively;

5. For any other set of noncore words \( S_k \), find \( V_k^* \) that minimizes the total penalty of the \( k \)-th and \( k \)-th blocks, ignoring all other \( k \)-th and \( j \)-th blocks;

6. Combine all subsets of embeddings to form \( V^* \). Here \( V^* = (V_1^*, V_2^*, V_3^*) \).

6 Experimental Results

We trained our model along with a few state-of-the-art competitors on Wikipedia, and evaluated the embeddings on 7 common benchmark sets.

6.1 Experimental Setup

Our own method is referred to as PSD. The competitors include:

- (Mikolov et al., 2013b): word2vec\(^2\), or SGNS in some literature;
- (Levy and Goldberg, 2014b): the PPMI matrix without dimension reduction, and SVD of PPMI matrix, both yielded by hyperwords;
- (Pennington et al., 2014): GloVe\(^3\);
- (Stratos et al., 2015): Singular\(^4\), which does SVD-based CCA on the weighted bigram frequency matrix;
- (Faruqui et al., 2015): Sparse\(^5\), which learns new sparse embeddings in a higher dimensional space from pretrained embeddings.

All models were trained on the English Wikipedia snapshot in March 2015. After removing non-textual elements and non-English words, 2.04 billion words were left. We used the default hyperparameters in Hyperwords when training PPMI and SVD. Word2vec, GloVe and Singular were trained with their own default hyperparameters.

The embedding sets PSD-Reg-180K and PSD-Unreg-180K were trained using our online block-wise regression. Both sets contain the embeddings of the most frequent 180,000 words, based on 25,000 core words. PSD-Unreg-180K was trained with all \( \mu_i = 0 \), i.e. disabling Tikhonov regularization. PSD-Reg-180K was trained with regularization as the sparsity increases. To contrast with the batch learning performance, the performance of PSD-25K is listed, which contains the core embeddings only. PSD-25K took advantages that it contains much less false candidate words, and some test tuples (generally harder ones) were not evaluated due to missing words, thus its scores are not comparable to others.

Sparse was trained with PSD-180K-reg as the input embeddings, with default hyperparameters.

The benchmark sets are almost identical to those in (Levy et al., 2015), except that (Luong et al., 2013)’s Rare Words is not included, as many rare words are cut off at the frequency 100, making more than 1/3 of test pairs invalid.

Word Similarity There are 5 datasets: WordSim Similarity (WS Sim) and WordSim Relatedness (WS Rel) (Zesch et al., 2008; Agirre et al., 2009), partitioned from WordSim353 (Finkelstein et al., 2002); Bruni et al. (2012)’s MEN dataset;

\(^2\)https://code.google.com/p/word2vec/
\(^3\)https://nlp.stanford.edu/projects/glove/
\(^4\)https://github.com/karlstatos/singular
\(^5\)https://github.com/mfaruqui/sparse-coding
Similarity Tasks Analogy Tasks

<table>
<thead>
<tr>
<th>Method</th>
<th>WS Sim</th>
<th>WS Rel</th>
<th>MEN</th>
<th>Turk</th>
<th>SimLex</th>
<th>Google</th>
<th>MSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>word2vec</td>
<td>0.742</td>
<td>0.543</td>
<td>0.731</td>
<td>0.663</td>
<td>0.395</td>
<td>0.734/0.742</td>
<td>0.650/0.674</td>
</tr>
<tr>
<td>PPMI</td>
<td>0.735</td>
<td>0.678</td>
<td>0.717</td>
<td>0.659</td>
<td>0.308</td>
<td>0.476/0.524</td>
<td>0.183/0.217</td>
</tr>
<tr>
<td>SVD</td>
<td>0.687</td>
<td>0.608</td>
<td>0.711</td>
<td>0.524</td>
<td>0.270</td>
<td>0.230/0.240</td>
<td>0.123/0.113</td>
</tr>
<tr>
<td>GloVe</td>
<td>0.759</td>
<td>0.630</td>
<td>0.756</td>
<td>0.641</td>
<td>0.362</td>
<td>0.535/0.544</td>
<td>0.408/0.435</td>
</tr>
<tr>
<td>Singular</td>
<td>0.763</td>
<td>0.684</td>
<td>0.747</td>
<td>0.581</td>
<td>0.345</td>
<td>0.440/0.508</td>
<td>0.364/0.399</td>
</tr>
<tr>
<td>Sparse</td>
<td>0.739</td>
<td>0.585</td>
<td>0.725</td>
<td>0.625</td>
<td>0.355</td>
<td>0.240/0.282</td>
<td>0.253/0.274</td>
</tr>
<tr>
<td>PSD-Reg-180K</td>
<td>0.792</td>
<td>0.679</td>
<td>0.764</td>
<td>0.676</td>
<td>0.398</td>
<td>0.602/0.623</td>
<td>0.465/0.507</td>
</tr>
<tr>
<td>PSD-Unreg-180K</td>
<td>0.786</td>
<td>0.663</td>
<td>0.753</td>
<td>0.675</td>
<td>0.372</td>
<td>0.566/0.598</td>
<td>0.424/0.468</td>
</tr>
<tr>
<td>PSD-25K</td>
<td>0.801</td>
<td>0.676</td>
<td>0.765</td>
<td>0.678</td>
<td>0.393</td>
<td>0.671/0.695</td>
<td>0.533/0.586</td>
</tr>
</tbody>
</table>

Table 2: Performance of each method across different tasks.

Radinsky et al. (2011)’s Mechanical Turk dataset; and (Hill et al., 2014)’s SimLex-999 dataset. The embeddings were evaluated by the Spearman’s rank correlation with the human ratings.

Word Analogy: The two datasets are MSR’s analogy dataset (Mikolov et al., 2013c), with 8000 questions, and Google’s analogy dataset (Mikolov et al., 2013a), with 19544 questions. After filtering questions involving out-of-vocabulary words, i.e. words that appear less than 100 times in the corpus, 7054 instances in MSR and 19364 instances in Google were left. The analogy questions were answered using 3CosAdd as well as 3CosMul proposed by Levy and Goldberg (2014a).

6.2 Results

Table 2 shows the results on all tasks. Word2vec significantly outperformed other methods on analogy tasks. PPMI and SVD performed much worse on analogy tasks than reported in (Levy et al., 2015), probably due to sub-optimal hyperparameters. This suggests their performance is unstable. The new embeddings yielded by Sparse systematically degraded compared to the old embeddings, contradicting the claim in (Faruqui et al., 2015).

Our method PSD-Reg-180K performed well consistently, and is best in 4 similarity tasks. It performed worse than word2vec on analogy tasks, but still better than other MF-based methods. By comparing to PSD-Unreg-180K, we see Tikhonov regularization brings 1-4% performance boost across tasks. In addition, on similarity tasks, online blockwise regression only degrades slightly compared to batch factorization. Their performance gaps on analogy tasks were wider, but this might be explained by the fact that some hard cases were not counted in PSD-25K’s evaluation, due to its limited vocabulary.

7 Conclusions and Future Work

In this paper, inspired by the link functions in previous works, with the support from Information Theory, we propose a new link function of a text window, parameterized by the embeddings of words and the residuals of bigrams. Based on the link function, we establish a generative model of documents. The learning objective is to find a set of embeddings maximizing their posterior likelihood given the corpus. This objective is reduced to weighted low-rank positive-semidefinite approximation, subject to Tikhonov regularization. Then we adopt a Block Coordinate Descent algorithm, jointly with an online blockwise regression algorithm to find an approximate solution. On seven benchmark sets, the learned embeddings show competitive and stable performance.

In the future work, we will incorporate global latent factors into this generative model, such as topics, sentiments, or writing styles, and develop more elaborate models of documents. Through learning such latent factors, important summary information of documents would be acquired, which are useful in various applications.

Acknowledgments

We thank Omer Levy, Thomas Mach, Peilin Zhao, Mingkui Tan, Zhiqiang Xu and Chunlin Wu for their helpful discussions and insights. This research is supported by the National Research Foundation, Prime Minister’s Office, Singapore under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.
Appendix A Possible Trap in SVD

Suppose $M$ is the bigram matrix of interest. SVD embeddings are derived from the low rank approximation of $M^T M$, by keeping the largest singular values/vectors. When some of these singular values correspond to negative eigenvalues, undesirable correlations might be captured. The following is an example of approximating a PMI matrix.

A vocabulary consists of 3 words $s_1, s_2, s_3$. Two corpora derive two PMI matrices:

$M^{(1)} = \begin{pmatrix} 1.4 & 0.8 & 0 \\ 0 & 2.6 & 0 \\ 0 & 0 & 0 \end{pmatrix}$, $M^{(2)} = \begin{pmatrix} 0.2 & -1.6 & 0 \\ -1.6 & -2.2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$.

They have identical left singular matrix and singular values $(3, 2, 1)$, but their eigenvalues are $(3, 2, 1)$ and $(-3, 2, 1)$, respectively.

In a rank-2 approximation, the largest two singular values/vectors are kept, and $M^{(1)}$ and $M^{(2)}$ yield identical SVD embeddings $V = \begin{pmatrix} 0.45 & 0.89 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (the rows may be scaled depending on the algorithm, without affecting the validity of the following conclusion). The embeddings of $s_1$ and $s_2$ (columns 1 and 2 of $V$) point at the same direction, suggesting they are positively correlated.

However as $M^{(2)} = M^{(2)}$ is negative, and yet the corresponding singular value/vector is kept.

When using eigendecomposition, the largest two positive eigenvalues/eigenvectors are kept. $M^{(1)}$ yields the same embeddings $V$. $M^{(2)}$ yields $V^{(2)} = \begin{pmatrix} -0.89 & 0.45 & 0 \\ 0 & 0 & 1.41 \end{pmatrix}$, which correctly preserves the negative correlation between $s_1, s_2$.

Appendix B Information Theory

Reduced information refers to the reduced uncertainty by knowing the value of any one of the conditioning variables (hence redundant). Synergistic information is the reduced uncertainty ascribed to knowing all the values of conditioning variables, that cannot be reduced by knowing the value of any variable alone (hence synergistic).

The mutual information $I(y; x_1)$ and the redundant information $\text{Rdn}(y; x_1, x_2)$ are defined as:

$I(y; x_1) = E_{P(x,y)}[\log \frac{P(y|x_1)}{P(y)}]$ \hspace{1cm} $\text{Rdn}(y; x_1, x_2) = E_{P(y)} \left[ \min_{x_1, x_2} E_{P(x_1,y)}[\log \frac{P(y|x_1)}{P(y)}] \right]$.

The synergistic information $\text{Syn}(y; x_1, x_2)$ is defined as the PI-function in (Williams and Beer, 2010), skipped here.

Figure 2: Different types of information among 3 random variables $y, x_1, x_2$. $I(y; x_1, x_2)$ is the mutual information between $y$ and $(x_1, x_2)$. $\text{Rdn}(y; x_1, x_2)$ and $\text{Syn}(y; x_1, x_2)$ are the redundant information and synergistic information between $x_1, x_2$, conditioning $y$, respectively.

The interaction information $\text{Int}(x_1, x_2, y)$ measures the relative strength of $\text{Rdn}(y; x_1, x_2)$ and $\text{Syn}(y; x_1, x_2)$ (Timme et al., 2014):

$\text{Int}(x_1, x_2, y) = \text{Syn}(y; x_1, x_2) - \text{Rdn}(y; x_1, x_2)$

$= I(y; x_1, x_2) - I(y; x_1) - I(y; x_2)$

$= \text{E}_{P(x_1,x_2,y)}[\log \frac{P(x_1)P(x_2)P(y)P(x_1,x_2,y)}{P(x_1,x_2)P(x_1,y)P(x_2,y)}]$.

Figure 2 shows the relationship of different information among 3 random variables $y, x_1, x_2$ (based on Fig.1 in (Williams and Beer, 2010)).

PMI is the pointwise counterpart of mutual information $I$. Similarly, all the above concepts have their pointwise counterparts, obtained by dropping the expectation operator. Specifically, the pointwise interaction information is defined as $\text{PInt}(x_1, x_2, y) = \text{PMI}(y; x_1, x_2) - \text{PMI}(y; x_1) - \text{PMI}(y; x_2)$

$= \text{log} \frac{P(x_1)P(x_2)P(y)P(x_1,x_2,y)}{P(x_1,x_2)P(x_1,y)P(x_2,y)}$.

If we know $\text{PInt}(x_1, x_2, y)$, we can recover $\text{PMI}(y; x_1, x_2)$ from the mutual information over the variable subsets, and then recover the joint distribution $P(x_1,x_2,y)$.

As the pointwise redundant information $\text{PRdn}(y; x_1, x_2)$ and the pointwise synergistic information $\text{PSyn}(y; x_1, x_2)$ are both higher-order interaction terms, their magnitudes are usually much smaller than the PMI terms. We assume they are approximately equal, and thus cancel each other when computing $\text{PInt}$. Given this, $\text{PInt}$ is always 0. In the case of three words $w_0, w_1, w_2$, $\text{PInt}(w_0, w_1, w_2) = 0$ leads to $\text{PMI}(w_2; w_0, w_1) = \text{PMI}(w_2; w_0) + \text{PMI}(w_2; w_1)$. 

1607
References


Yan Yan, Mingkui Tan, Ivor Tsang, Yi Yang, Chengqi Zhang, and Qinfeng Shi. 2015. Scalable maximum margin matrix factorization by active riemannian subspace search. In *Proceedings of IJCAI 2015*.
