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

Shaohua Li[1], **Jun Zhu**[2], **Chunyan Miao**[1]
[1]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY),
Nanyang Technological University, Singapore
[2]Tsinghua 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

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1599–1609,
Lisbon, Portugal, 17-21 September 2015.

factorize this matrix directly as a new method. 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 $\boldsymbol{M}$. As SVD factorizes $\boldsymbol{M}^\top \boldsymbol{M}$, some information in $\boldsymbol{M}$ is lost, and the learned embeddings may not capture the most significant regularities in $\boldsymbol{M}$. Appendix A gives a toy example on which SVD does not work properly.

The second limitation is, a generative model for documents parametered 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 $\boldsymbol{S}, \boldsymbol{V}$ to denote a matrix or set, a lowercase bold letter as $\boldsymbol{v}_{w_i}$ 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 $\boldsymbol{S} = \{s_1, \cdots, s_W\}$ consists of all the words, where W is the vocabulary size. We further suppose $s_1, \cdots, s_W$ are sorted in decending 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 $d_i = (w_{i1}, \cdots, w_{iL_i}), w_{ij} \in \boldsymbol{S}$. A corpus is a collec-

Name	Description
$\boldsymbol{S}$	Vocabulary $\{s_1, \cdots, s_W\}$
$\boldsymbol{V}$	Embedding matrix $(\boldsymbol{v}_{s_1}, \cdots, \boldsymbol{v}_{s_W})$
$\boldsymbol{D}$	Corpus $\{d_1, \cdots, d_M\}$
$\boldsymbol{v}_{s_i}$	Embedding of word s_i
$a_{s_i s_j}$	Bigram residual for s_i, s_j
$\tilde{P}(s_i,s_j)$	Empirical probability of s_i, s_j in the corpus
$\boldsymbol{u}$	Unigram probability vector $(P(s_1),\cdots,P(s_W))$
$\boldsymbol{A}$	Residual matrix $(a_{s_i s_j})$
$\boldsymbol{B}$	Conditional probability matrix $\Big(P(s_j\|s_i)\Big)$
$\boldsymbol{G}$	PMI matrix $\Big(\text{PMI}(s_i, s_j)\Big)$
$\boldsymbol{H}$	Bigram empirical probability matrix $\Big(\tilde{P}(s_i, s_j)\Big)$

Table 1: Notation Table

tion of M documents $\boldsymbol{D} = \{d_1, \cdots, d_M\}$. In the vocabulary, each word s_i is mapped to a vector $\boldsymbol{v}_{s_i}$ in N-dimensional Euclidean space.

In a document, a sequence of words is referred to as a *text window*, denoted by $w_i, \cdots, w_{i+l}$, or $w_i{:}w_{i+l}$ 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:

$$P(s_i, s_j) = \exp\Big\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j}\Big\} P(s_i)P(s_j) \quad (1)$$

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 $\boldsymbol{v}_{s_i}, \boldsymbol{v}_{s_j}$. As $\exp\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}\} = \prod_l \exp\{v_{s_i,l} \cdot v_{s_j,l}\}$, 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, $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ captures linear interactions between s_i and s_j, the residual $a_{s_i s_j}$ captures nonlinear or noisy interactions. In applications, only $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ is of interest. Hence the bigger magnitude $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ is of 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, $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ imposes a symmetric part between $P(s_i, s_j)$ and $P(s_j, s_i)$.

(1) is equivalent to

$$P(s_j|s_i) = \exp\Big\{\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j} + \log P(s_j)\Big\}, \quad (2)$$

$$\log \frac{P(s_j|s_i)}{P(s_j)} = \boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j}. \quad (3)$$

(3) of all bigrams is represented in matrix form:

$$\boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{A} = \boldsymbol{G}, \quad (4)$$

where $\boldsymbol{G}$ 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 $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i} + a_{s_i s_j}$ to be a big negative number, making $\boldsymbol{v}_{s_i}$ 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 $\mathcal{N}(0, \frac{1}{2\mu_i}\boldsymbol{I})$ to $\boldsymbol{v}_{s_i}$:

$$P(\boldsymbol{v}_{s_i}) \sim \exp\{-\mu_i \|\boldsymbol{v}_{s_i}\|^2\},$$

where the hyperparameter μ_i increases as the frequency of s_i decreases.

3.1.2 Gaussian Priors on Residuals

We wish $\boldsymbol{v}_{s_j}^\top \boldsymbol{v}_{s_i}$ in (1) captures as much correlations between s_i and s_j 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 $\boldsymbol{A}$:

$$\sum_{s_i, s_j \in \boldsymbol{S}} f(h_{ij}) a_{s_i s_j}^2 = \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2. \quad (5)$$

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

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 $\tilde{P}(s_j)$ as:

$$\tilde{P}_{\text{smoothed}}(s_j|s_i) = (1-\kappa)\tilde{P}(s_j|s_i) + \kappa P(s_j). \quad (6)$$

Accordingly, the smoothed bigram empirical joint probability is defined as

$$\tilde{P}(s_i, s_j) = (1-\kappa)\tilde{P}(s_i, s_j) + \kappa P(s_i)P(s_j). \quad (7)$$

In practice, we find $\kappa = 0.02$ yields good results. When $\kappa \geq 0.04$, the obtained embeddings begin to degrade with κ, 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 $w_0, \cdots, w_{c-1}, w_c$.

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 $\text{PMI}(w_2; w_0, w_1) \approx \text{PMI}(w_2; w_0) + \text{PMI}(w_2; w_1)$:

$$\log \frac{P(w_0, w_1|w_2)}{P(w_0, w_1)} \approx \log \frac{P(w_0|w_2)}{P(w_0)} + \log \frac{P(w_1|w_2)}{P(w_1)}.$$

Plugging (1) and (3) into the above, we obtain

$$P(w_0, w_1, w_2) \approx \exp\left\{ \sum_{\substack{i,j=0 \\ i \neq j}}^{2} (\boldsymbol{v}_{w_i}^\top \boldsymbol{v}_{w_j} + a_{w_i w_j}) + \sum_{i=0}^{2} \log P(w_i) \right\}.$$

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 $c > 2$:

$$P(w_0, \cdots, w_c) \approx \exp\left\{ \sum_{\substack{i,j=0 \\ i \neq j}}^{c} (\boldsymbol{v}_{w_i}^\top \boldsymbol{v}_{w_j} + a_{w_i w_j}) + \sum_{i=0}^{c} \log P(w_i) \right\}.$$

From it derives the conditional distribution of w_c, given its context $w_0, \cdots, w_{c-1}$:

$$P(w_c \mid w_0 : w_{c-1}) = \frac{P(w_0, \cdots, w_c)}{P(w_0, \cdots, w_{c-1})} \approx P(w_c) \exp\left\{ \boldsymbol{v}_{w_c}^\top \sum_{i=0}^{c-1} \boldsymbol{v}_{w_i} + \sum_{i=0}^{c-1} a_{w_i w_c} \right\}. \quad (8)$$

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 $\boldsymbol{\mu} = (\mu_1, \cdots, \mu_W)$, the generative process of the whole corpus is:

1. For each word s_i, draw the embedding $\boldsymbol{v}_{s_i}$ from $\mathcal{N}(0, \frac{1}{2\mu_i}\boldsymbol{I})$;
2. For each bigram s_i, s_j, draw the residual $a_{s_i s_j}$ from $\mathcal{N}\left(0, \frac{1}{2f(h_{ij})}\right)$;
3. For each document d_i, for the j-th word, draw word w_{ij} from $\boldsymbol{S}$ with probability $P(w_{ij} \mid w_{i,j-c} : w_{i,j-1})$ defined by (8).

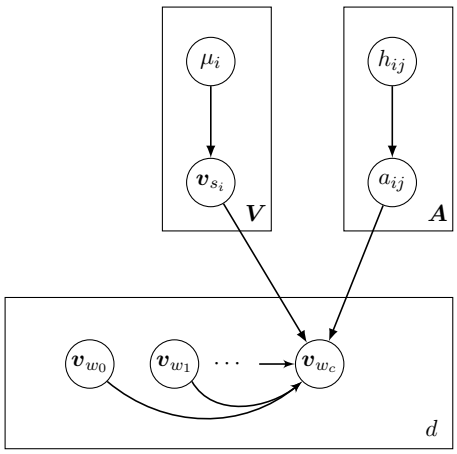

Figure 1: The Graphical Model of PSDVec

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_i can be derived as follows, given the embeddings and residuals $\boldsymbol{V}, \boldsymbol{A}$:

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

The complete-data likelihood of the corpus is:

$$\begin{aligned} & p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) \\ =& \prod_{i=1}^{W} \mathcal{N}(0, \frac{\boldsymbol{I}}{2\mu_i}) \prod_{i,j=1}^{W,W} \mathcal{N}\left(0, \frac{1}{2f(h_{ij})}\right) \prod_{i=1}^{M} p(d_i|\boldsymbol{V},\boldsymbol{A}) \\ =& \frac{1}{\mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu})} \exp\Big\{ -\sum_{i,j=1}^{W,W} f(h_{i,j}) a_{s_i s_j}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \Big\} \\ & \cdot \prod_{i,j=1}^{M,L_i} P(w_{ij}) \exp\left\{ \boldsymbol{v}_{w_{ij}}^\top \sum_{k=j-c}^{j-1} \boldsymbol{v}_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\}, \end{aligned}$$

where $\mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu})$ is the normalizing constant.

Taking the logarithm of both sides of $p(\boldsymbol{D},\boldsymbol{A},\boldsymbol{V})$ yields

$$\begin{aligned} & \log p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) \\ =& C_0 - \log \mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu}) - \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\ & + \sum_{i,j=1}^{M,L_i} \left\{ \boldsymbol{v}_{w_{ij}}^\top \sum_{k=j-c}^{j-1} \boldsymbol{v}_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik}w_{ij}} \right\}, \quad (9) \end{aligned}$$

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 $\boldsymbol{V}$ 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:

$$\begin{aligned} & \log p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) \\ =& C_0 - \log \mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu}) - \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\ & + \sum_{i,j=1}^{W,W} x_{ij} (\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}). \quad (10) \end{aligned}$$

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

$$\begin{aligned} & \max \sum x_{ij} (\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}) \\ =& \Big(\sum x_{ij} \Big) \cdot \max \sum \tilde{P}_{\text{smoothed}}(s_i, s_j) \log P(s_i, s_j). \end{aligned}$$

As both $\{\tilde{P}_{\text{smoothed}}(s_i, s_j)\}$ and $\{P(s_i, s_j)\}$ sum to 1, the above sum is maximized when $P(s_i, s_j) = \tilde{P}_{\text{smoothed}}(s_i, s_j)$.

The maximum likelihood estimator is then:

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

$$\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j} = \log \frac{\tilde{P}_{\text{smoothed}}(s_j|s_i)}{P(s_j)}. \quad (11)$$

Writing (11) in matrix form:

$$\begin{aligned} \boldsymbol{B}^* &= \left(\tilde{P}_{\text{smoothed}}(s_j|s_i) \right)_{s_i, s_j \in \boldsymbol{S}} \\ \boldsymbol{G}^* &= \log \boldsymbol{B}^* - \log \boldsymbol{u} \otimes (1 \cdots 1), \quad (12) \end{aligned}$$

where "$\otimes$" is the outer product.

Now we fix the values of $\boldsymbol{v}_{s_i}^\top \boldsymbol{v}_{s_j} + a_{s_i s_j}$ at the above optimal. The corpus likelihood becomes

$$\begin{aligned} \log p(\boldsymbol{D},\boldsymbol{V},\boldsymbol{A}) =& C_1 - \|\boldsymbol{A}\|_{f(\boldsymbol{H})}^2 - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2, \\ & \text{subject to} \quad \boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{A} = \boldsymbol{G}^*, \quad (13) \end{aligned}$$

where $C_1 = C_0 + \sum x_{ij} \log \tilde{P}_{\text{smoothed}}(s_i, s_j) - \log \mathcal{Z}(\boldsymbol{H},\boldsymbol{\mu})$ is constant.

5.2 Learning V as Low Rank PSD Approximation

Once $\boldsymbol{G}^*$ has been estimated from the corpus using (12), we seek $\boldsymbol{V}$ that maximizes (13). This is to find the maximum a posteriori (MAP) estimates of $\boldsymbol{V}, \boldsymbol{A}$ that satisfy $\boldsymbol{V}^\top \boldsymbol{V} + \boldsymbol{A} = \boldsymbol{G}^*$. Applying this constraint to (13), we obtain

Algorithm 1 BCD algorithm for finding a unregularized rank-N weighted PSD approximant.

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

Randomly initialize $\boldsymbol{X}^{(0)}$
for $t = 1, \cdots, \mathcal{T}$ **do**
  $\boldsymbol{G}_t = \boldsymbol{W} \circ \boldsymbol{G}^* + (1 - \boldsymbol{W}) \circ \boldsymbol{X}^{(t-1)}$
  $\boldsymbol{X}^{(t)} = \text{PSD\_Approximate}(\boldsymbol{G}_t, N)$
end for
$\boldsymbol{\lambda}, \boldsymbol{Q} = \text{Eigen\_Decomposition}(\boldsymbol{X}^{(\mathcal{T})})$
$\boldsymbol{V}^* = \text{diag}(\boldsymbol{\lambda}^{\frac{1}{2}}[1{:}N]) \cdot \boldsymbol{Q}^\top[1{:}N]$

Output: $\boldsymbol{V}^*$

$$\arg\max_{\boldsymbol{V}} \log p(\boldsymbol{D}, \boldsymbol{V}, \boldsymbol{A})$$
$$= \arg\min_{\boldsymbol{V}} \|\boldsymbol{G}^* - \boldsymbol{V}^\top \boldsymbol{V}\|_{f(\boldsymbol{H})} + \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2. \quad (14)$$

Let $\boldsymbol{X} = \boldsymbol{V}^\top \boldsymbol{V}$. Then $\boldsymbol{X}$ is positive semidefinite of rank N. Finding $\boldsymbol{V}$ that minimizes (14) is equivalent to finding a rank-N *weighted positive semidefinite approximant* $\boldsymbol{X}$ of $\boldsymbol{G}^*$, 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 $\forall \mu_i = 0$, 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 $\boldsymbol{\lambda}$ returned by Eigen_Decomposition($\boldsymbol{X}$) are in descending order. $\boldsymbol{Q}^\top[1{:}N]$ extracts the 1 to N rows from $\boldsymbol{Q}^\top$.

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

[1]It is referred to as an Expectation-Maximization algorithm by the original authors, but we think this is a misnomer.

5.3 Online Blockwise Regression of $\boldsymbol{V}$

In Algorithm 1, the essential subroutine PSD_Approximate() does eigendecomposition on $\boldsymbol{G}_t$, which is dense due to the logarithm transformation. Eigendecomposition on a $W \times W$ dense matrix requires $O(W^2)$ 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 necessary 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 $\boldsymbol{S} = \{s_1, \cdots, s_W\}$ are sorted in decending order of the frequency, hence the lower-right blocks of $\boldsymbol{H}$ and $f(\boldsymbol{H})$ 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 $\boldsymbol{G}$ and $f(\boldsymbol{H})$. The embeddings of core words are learned approximately using Algorithm 1, on the top-left blocks of $\boldsymbol{G}$ and $f(\boldsymbol{H})$. 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 $\boldsymbol{G}$ and $f(\boldsymbol{H})$ 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 $\boldsymbol{S}$ into K consecutive groups $\boldsymbol{S}_1, \cdots, \boldsymbol{S}_k$. Take $K = 3$ as an example. The first group is core words;
2. Accordingly partition $\boldsymbol{G}$ into $K \times K$ blocks,

in this example as $\left(\begin{array}{c|cc} \boldsymbol{G}_{11} & \boldsymbol{G}_{12} & \boldsymbol{G}_{13} \\ \hline \boldsymbol{G}_{21} & \boldsymbol{G}_{22} & \boldsymbol{G}_{23} \\ \boldsymbol{G}_{31} & \boldsymbol{G}_{32} & \boldsymbol{G}_{33} \end{array}\right)$. Partition $f(\boldsymbol{H})$,$\boldsymbol{A}$ in the same way. $\boldsymbol{G}_{11}, f(\boldsymbol{H})_{11}, \boldsymbol{A}_{11}$ correspond to core bigrams. Partition $\boldsymbol{V}$ into $\underbrace{(\boldsymbol{V}_1}_{\boldsymbol{S}_1} \mid \underbrace{\boldsymbol{V}_2}_{\boldsymbol{S}_2} \; \underbrace{\boldsymbol{V}_3)}_{\boldsymbol{S}_3}$;

3. Solve $\boldsymbol{V}_1^\top \boldsymbol{V}_1 + \boldsymbol{A}_{11} = \boldsymbol{G}_{11}$ using Algorithm 1, and obtain core embeddings $\boldsymbol{V}_1^*$;
4. Set $\boldsymbol{V}_1 = \boldsymbol{V}_1^*$, and find $\boldsymbol{V}_2^*$ 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):

$$\begin{aligned} &\arg\min_{\boldsymbol{V}_2} \|\boldsymbol{G}_{12} - \boldsymbol{V}_1^\top \boldsymbol{V}_2\|^2_{f(\boldsymbol{H})_{12}} \\ &\quad + \|\boldsymbol{G}_{21} - \boldsymbol{V}_2^\top \boldsymbol{V}_1\|^2_{f(\boldsymbol{H})_{21}} + \sum_{s_i \in \boldsymbol{S}_2} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\ =&\arg\min_{\boldsymbol{V}_2} \|\overline{\boldsymbol{G}}_{12} - \boldsymbol{V}_1^\top \boldsymbol{V}_2\|^2_{\bar{f}(\boldsymbol{H})_{12}} + \sum_{s_i \in \boldsymbol{S}_2} \mu_i \|\boldsymbol{v}_{s_i}\|^2, \end{aligned}$$

   where $\bar{f}(\boldsymbol{H})_{12} = f(\boldsymbol{H})_{12} + f(\boldsymbol{H})_{21}^\top$; $\overline{\boldsymbol{G}}_{12} = \left(\boldsymbol{G}_{12} \circ f(\boldsymbol{H})_{12} + \boldsymbol{G}_{21}^\top \circ f(\boldsymbol{H})_{21}^\top\right) / \left(f(\boldsymbol{H})_{12} + f(\boldsymbol{H})_{21}^\top\right)$ is the weighted average of $\boldsymbol{G}_{12}$ and $\boldsymbol{G}_{21}^\top$, "$\circ$" and "/" are element-wise product and division, respectively. The columns in $\boldsymbol{V}_2$ are independent, thus for each $\boldsymbol{v}_{s_i}$, it is a separate weighted ridge regression problem, whose solution is (Holland, 1973): $\boldsymbol{v}_{s_i}^* = (\boldsymbol{V}_1^\top \mathrm{diag}(\bar{\boldsymbol{f}}_i) \boldsymbol{V}_1 + \mu_i \boldsymbol{I})^{-1} \boldsymbol{V}_1^\top \mathrm{diag}(\bar{\boldsymbol{f}}_i) \bar{\boldsymbol{g}}_i$, where $\bar{\boldsymbol{f}}_i$ and $\bar{\boldsymbol{g}}_i$ are columns corresponding to s_i in $\bar{f}(\boldsymbol{H})_{12}$ and $\overline{\boldsymbol{G}}_{12}$, respectively;
5. For any other set of noncore words $\boldsymbol{S}_k$, find $\boldsymbol{V}_k^*$ that minimizes the total penalty of the $1k$-th and $k1$-th blocks, ignoring all other kj-th and jk-th blocks;
6. Combine all subsets of embeddings to form $\boldsymbol{V}^*$. Here $\boldsymbol{V}^* = (\boldsymbol{V}_1^*, \boldsymbol{V}_2^*, \boldsymbol{V}_3^*)$.

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 traind with all $\mu_i = 0$, i.e. disabling Tikhonov regularization. PSD-Reg-180K was trained with $\mu_i = \begin{cases} 2 & i \in [25001, 80000] \\ 4 & i \in [80001, 130000] \\ 8 & i \in [130001, 180000] \end{cases}$, i.e. increased 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] http://nlp.stanford.edu/projects/glove/

[4] https://github.com/karlstratos/singular

[5] https://github.com/mfaruqui/sparse-coding

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

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 $\boldsymbol{M}$ is the bigram matrix of interest. SVD embeddings are derived from the low rank approximation of $\boldsymbol{M}^\top \boldsymbol{M}$, 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:

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

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 $\boldsymbol{M}^{(1)}$ and $\boldsymbol{M}^{(2)}$ yield identical SVD embeddings $\boldsymbol{V} = \begin{pmatrix} 0.45 & 0.89 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ (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 $\boldsymbol{V}$) point at the same direction, suggesting they are positively correlated. However as $\boldsymbol{M}^{(2)}_{1,2} = \boldsymbol{M}^{(2)}_{2,1} = -1.6 < 0$, they are actually negatively correlated in the second corpus. This inconsistency is because the principal eigenvalue of $\boldsymbol{M}^{(2)}$ is negative, and yet the corresponding singular value/vector is kept.

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

Appendix B Information Theory

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

$$I(y; x_i) = E_{P(x_i,y)}[\log \frac{P(y|x_i)}{P(y)}]$$

$$\text{Rdn}(y; x_1, x_2) = E_{P(y)}\left[\min_{x_1,x_2} E_{P(x_i|y)}[\log \frac{P(y|x_i)}{P(y)}]\right]$$

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

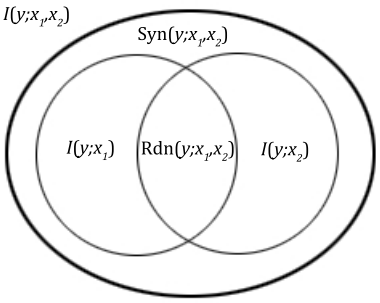

Figure 2: Different types of information among 3 random variables y, x_1, x_2. $I(y; x_1, x_2)$ is the mutual information between y and (x_1, x_2). $\text{Rdn}(y; x_1, x_2)$ and $\text{Syn}(y; x_1, x_2)$ are the redundant information and synergistic information between x_1, x_2, conditioning y, respectively.

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):

$$\begin{aligned}
&\text{Int}(x_1, x_2, y) \\
=&\text{Syn}(y; x_1, x_2) - \text{Rdn}(y; x_1, x_2) \\
=&I(y; x_1, x_2) - I(y; x_1) - I(y; x_2) \\
=&E_{P(x_1,x_2,y)}[\log \frac{P(x_1)P(x_2)P(y)P(x_1, x_2, y)}{P(x_1, x_2)P(x_1, y)P(x_2, y)}]
\end{aligned}$$

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 $\text{PInt}(x_1, x_2, y) = \text{PMI}(y; x_1, x_2) - \text{PMI}(y; x_1) - \text{PMI}(y; x_2) = \log \frac{P(x_1)P(x_2)P(y)P(x_1,x_2,y)}{P(x_1,x_2)P(x_1,y)P(x_2,y)}$. 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 PInt. Given this, 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 $\text{PMI}(w_2; w_0, w_1) = \text{PMI}(w_2; w_0) + \text{PMI}(w_2; w_1)$.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on discourse spaces: a new generative language model with applications to semantic word embeddings. *ArXiv e-prints*, arXiv:1502.03520 [cs.LG].

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 136–145. Association for Computational Linguistics.

Scott C. Deerwester, Susan T Dumais, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *J. Am. Soc. Inf. Sci.*

Paramveer Dhillon, Dean P Foster, and Lyle H Ungar. 2011. Multi-view learning of word embeddings via cca. In *Proceedings of Advances in Neural Information Processing Systems*, pages 199–207.

Paramveer S Dhillon, Dean P Foster, and Lyle H Ungar. 2015. Eigenwords: Spectral word embeddings. *The Journal of Machine Learning Research*.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of ACL 2015*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: The concept revisited. *ACM Trans. Inf. Syst.*, 20(1):116–131, January.

Amir Globerson, Gal Chechik, Fernando Pereira, and Naftali Tishby. 2007. Euclidean embedding of co-occurrence data. *Journal of Machine Learning Research*, vol. 8 (2007):2265–2295, Oct.

Nicholas J. Higham. 1988. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra and its Applications*, 103(0):103 – 118.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456.

Geoffrey Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.

Paul W. Holland. 1973. Weighted Ridge Regression: Combining Ridge and Robust Regression Methods. NBER Working Papers 0011, National Bureau of Economic Research, Inc, September.

Daniel Hsu, Sham M Kakade, and Tong Zhang. 2012. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of CoNLL-2014*, page 171.

Omer Levy and Yoav Goldberg. 2014b. Neural word embeddings as implicit matrix factorization. In *Proceedings of NIPS 2014*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and Knowledge Management*, pages 375–384. ACM.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

Thomas Mach. 2012. *Eigenvalue Algorithms for Symmetric Hierarchical Matrices*. Dissertation, Chemnitz University of Technology.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR 2013*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of HLT-NAACL 2013*, pages 746–751.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine learning*, pages 641–648. ACM.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. 2011. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 337–346, New York, NY, USA. ACM.

Nathan Srebro, Tommi Jaakkola, et al. 2003. Weighted low-rank approximations. In *Proceedings of ICML 2003*, volume 3, pages 720–727.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel Hsu. 2014. A spectral algorithm for learning class-based n-gram models of natural language. In *Proceedings of the Association for Uncertainty in Artificial Intelligence*.

Karl Stratos, Michael Collins, and Daniel Hsu. 2015. Model-based word embeddings from decompositions of count matrices. In *Proceedings of ACL 2015*.

Mingkui Tan, Ivor W. Tsang, Li Wang, Bart Vandereycken, and Sinno Jialin Pan. 2014. Riemannian pursuit for big matrix recovery. In *Proceedings of ICML 2014*, pages 1539–1547.

Nicholas Timme, Wesley Alford, Benjamin Flecker, and John M Beggs. 2014. Synergy, redundancy, and multivariate information measures: an experimentalist's perspective. *Journal of Computational Neuroscience*, 36(2):119–140.

Hanna M Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM.

Paul L Williams and Randall D Beer. 2010. Non-negative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*.

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*.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proceedings of ICML 2015*.

Torsten Zesch, Christof Müller, and Iryna Gurevych. 2008. Using wiktionary for computing semantic relatedness. In *Proceedings of AAAI 2008*, volume 8, pages 861–866.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.

Peilin Zhao, Steven CH Hoi, and Rong Jin. 2011a. Double updating online learning. *The Journal of Machine Learning Research*, 12:1587–1615.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011b. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval (Proceedings of the 33rd Annual European Conference on Information Retrieval Research)*, pages 338–349. Springer.