The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Spectral Word Embedding with Negative Sampling

Behrouz Haji Soleimani, Stan Matwin
Faculty of Computer Science, Dalhousie University
Institute for Big Data Analytics
6050 University Avenue, Halifax, NS, Canada
behrouz.hajisoleimani@dal.ca, st837183@dal.ca

Abstract

In this work, we investigate word embedding algorithms in the context of natural language processing. In particular, we examine the notion of "negative examples", the unobserved or insignificant word-context co-occurrences, in spectral methods. we provide a new formulation for the word embedding problem by proposing a new intuitive objective function that perfectly justifies the use of negative examples. In fact, our algorithm not only learns from the important word-context co-occurrences, but also it learns from the abundance of unobserved or insignificant co-occurrences to improve the distribution of words in the latent embedded space. We analyze the algorithm theoretically and provide an optimal solution for the problem using spectral analysis. We have trained various word embedding algorithms on articles of Wikipedia with 2.1 billion tokens and show that negative sampling can boost the quality of spectral methods. Our algorithm provides results as good as the state-of-the-art but in a much faster and efficient way.

Introduction

In recent years there has been an increasing interest in learning compact representations (i.e embeddings) for a set of input datapoints. In these approaches, input data is mapped to a low-dimensional latent space with the goal of preserving the geometrical properties of data with respect to some similarity measure in the input space. That is, similar datapoints in the input space should be mapped to nearby points in the latent embedded space. In the embedded space, each input datapoint is described with a dense d-dimensional continuous vector representing the coordinates of the datapoint in the latent space.

In natural language processing, embedding algorithms are particularly used to learn a vector space representation for words aiming to capture semantic similarities and syntactic relationships between words. The traditional way of treating individual words as unique symbols and representing documents by sparse word count vectors, known as Bag-of-Words (BOW) representation (Salton 1971), has strong limitations since it does not exploit countless semantic and syntactic relations encoded in the corpus. Moreover, it does not take into account the ordering of the words, therefore, two different sentences can have the same representation as long as they use the same words. N-gram models (Suen 1979; Brown et al. 1992; Fürnkranz 1998) tried to overcome these limitations by counting the occurrences of sequences of words rather than individual words ($1 \leq n \leq 5$). They consider the word order in a short context but suffer from the curse of dimensionality since the number of n-grams increases dramatically as n increases. They also do not capture the semantics of the words or more formally the similarities between the words (Le and Mikolov 2014).

In recent years, distributed word representations or word embedding algorithms have shown to be very effective in capturing certain aspects of similarity between words. Statistical language modeling methods take advantage of the fact that words that are temporally closer in a sentence are statistically more dependent, and therefore, model the probability of a word conditioned on the previous words in the sentence (Bengio et al. 2003). The idea of using a moving context window and assuming words in the same window are semantically similar is widely exploited (Mikolov et al. 2013b; 2013a; Pennington, Socher, and Manning 2014). GloVe calculates the global co-occurrence statistics first using a fixed-size context window, and then minimizes its least squares objective function using stochastic gradient descent which is essentially factorizing the log co-occurrence matrix (Pennington, Socher, and Manning 2014). Many neural-network based approaches have been proposed for learning distributed word representations (Bengio et al. 2006; Collobert and Weston 2008; Mikolov et al. 2013a; 2013b). Skip-Gram with Negative Sampling (SGNS) is still the state-of-the-art word embedding algorithm and is successfully applied in a variety of linguistic tasks (Mikolov et al. 2013a; 2013b).

Levy et al. in (Levy, Goldberg, and Dagan 2015) have studied the effect of various hyper-parameters in different embedding algorithms and showed that many of these parameters can be transferred to traditional methods (e.g SVD) to boost their performance. In fact, they showed that explicit matrix factorization methods can provide competitive results if used properly and there is no significant advantage over any of the algorithms. Levy and Goldberg in (Levy and Goldberg 2014) showed that SGNS is implicitly factorizing a Shifted Positive Point-wise Mutual Information (SPPMI) matrix and they argue that the shift parameter is

almost equivalent to the negative sampling parameter k in SGNS. However, their proposed alternative approach using SVD provides lower quality embedding than SGNS mainly for two reasons: first, the unweighted L_2 optimization in SVD which gives equal importance to frequent and infrequent pairs, and second, the shift cannot capture certain aspects of the parameter k in SGNS, that is higher k in SGNS results in using more data and better estimating the distribution of negative examples (Levy, Goldberg, and Dagan 2015). Therefore, SGNS remains superior to others with a small margin in most NLP applications.

In this work, we provide a different perspective for looking at the negative samples, word pairs that never co-occur. Most embedding algorithms only use the word pairs that occur in the corpus and maximize the similarity of those word vectors based on how frequent they co-occur. This can result in *concentration effect*: word clusters from totally different topics can be placed somewhat close to each other. There are lots of possible word pairs that never co-occur in the corpus or they co-occur insignificantly, which we call them negative examples. We argue that minimizing the similarity of negative examples is also crucial in the quality of final embedding and results in better distribution of words in the latent space. We show how matrix factorization methods can benefit from the abundance of information (i.e. negative examples) which was disregarded previously since they were considered useless. We incorporate the notion of negative sampling in standard matrix factorization methods by randomly choosing a tiny fraction of zeros in the PMI matrix and assigning negative values to them or simply by not ignoring all negative PMI values. We formulate the problem as an optimization task by proposing an intuitive objective function which perfectly justifies the use of negative values in the PMI matrix. Our optimization has an optimal closed form solution and we make a theoretical connection between our solution and SVD.

Background

Notation

Let's assume we have a text corpus which is a collection of words $w \in V_W$ where V_W is the word vocabulary. The context of a word w_i is commonly defined as the words surrounding it in a window of size L, $w_{i-L}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+L}$. This results in a set of context words $c \in V_C$ with their corresponding context vocabulary V_C. The sizes of the vocabularies are typically in $10^5 - 10^6$ range.

For each word-context pair (w, c), we use $\#(w, c)$ to denote the number of times they co-occurred in the corpus. Similarly, we use $\#(w) = \sum_{i=1}^{|V_C|} \#(w, c_i)$ and $\#(c) = \sum_{i=1}^{|V_W|} \#(w_i, c)$ for denoting the number of times w and c occurred in the corpus, respectively.

By moving an L-sized context window over the corpus, a global co-occurrence matrix X of size $|V_W| \times |V_C|$ can be built where each matrix entry $x_{ij} = \#(w_i, c_j)$ denote the number of times w_i and c_j co-occurred. The co-occurrence matrix is a very sparse matrix with lots of zero entries since most possible word-context pairs never co-occur in the corpus.

Most word embedding algorithms embed all words and contexts in a d-dimensional space where each word $w \in V_W$ and each context $c \in V_C$ is represented with a vector $\vec{w} \in \mathbb{R}^d$ and $\vec{c} \in \mathbb{R}^d$. The set of all word vectors is commonly represented by a matrix W of size $|V_W| \times d$. Similarly, context vectors are the rows in a $|V_C| \times d$ matrix C.

Pointwise Mutual Information (PMI)

The co-occurrence matrix contains the global statistics of the corpus and is the primary source of information for most algorithms. However, not all co-occurrences are meaningful. Pointwise Mutual Information (PMI) is an information theoretic measure that can be used for finding collocations or associations between words (Church and Hanks 1990) and can detect significant versus insignificant co-occurrences to some extent. For any word-context pair (w, c), PMI is defined as the log ratio between their joint probability and product of their marginal probabilities:

$$PMI(w, c) = \log \frac{P(w, c)}{P(w)P(c)} \tag{1}$$

Based on the formulation, if two words co-occur more often than being independent then their PMI will be positive, and if they co-occur less frequent than being independent then their PMI will be negative. For instance, in the data we used for the paper, the words "right" and "align" have a high PMI of 10.38[1] which indicates a strong collocation, but words "school" and "species" have a low PMI of -6.88 which means they are not likely to happen in the same context.

Calculating PMI values for all word-context pairs gives us a $|V_W| \times |V_C|$ PMI matrix which we call it M^{PMI}. Most of entries in the co-occurrence matrix are zero $\#(w, c) = 0$, for which $PMI(w, c) = \log 0 = -\infty$. A common approach to handle the situation is to use M_0^{PMI} in which $PMI(w, c) = 0$ whenever $\#(w, c) = 0$. Another commonly accepted approach is to use Positive PMI (PPMI) matrix M^{PPMI} by replacing all the negative values with 0.

$$PPMI(w, c) = \max(PMI(w, c), 0) \tag{2}$$

In fact, a traditional approach to word embedding is to use explicit PPMI representation in which each word is described by its corresponding sparse row vector in the PPMI matrix M^{PPMI} and it is shown that it outperforms M_0^{PMI} on semantic similarity tasks (Bullinaria and Levy 2007).

A recognized shortcoming of PMI and consequently PPMI is their bias towards infrequent events (Turney and Pantel 2010). This happens when a rare context c co-occurs with a word w a few times (or even once) and this often results in a high PMI value since $P(c)$ in PMI's denominator is very small. However, explicit PPMI representation is a well-known approach in distributional-similarity models.

[1] Base 2 logarithms has been used here as well as all the experiments in the paper.

Singular Value Decomposition (SVD)

SVD is a matrix factorization technique in linear algebra which has a broad range of applications. It is widely used for image compression and also dimensionality reduction. SVD on the matrix $\mathbf{X}_{n\times p}$ gives a factorization of the form $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ in which $\mathbf{V}_{p\times p} = [\mathbf{v}_1\ \mathbf{v}_2\ ...\ \mathbf{v}_p]$ is an orthonormal basis for the row space of $\mathbf{X}$, $\mathbf{U}_{n\times n} = [\mathbf{u}_1\ \mathbf{u}_2\ ...\ \mathbf{u}_n]$ is an orthonormal basis for the column space of $\mathbf{X}$ and $\mathbf{\Sigma}$ is a diagonal matrix of singular values $\sigma_1, \sigma_2, ..., \sigma_p$. If $n > p$ then the corresponding $\sigma_{p+1}, ..., \sigma_n$ will be 0. The matrix $\mathbf{X}$ can be seen as a transformation matrix between the two bases $\mathbf{X}[\mathbf{v}_1\ \mathbf{v}_2\ ...\ \mathbf{v}_p] = [\sigma_1\mathbf{u}_1\ \sigma_2\mathbf{u}_2\ ...\ \sigma_n\mathbf{u}_n]$ or $\mathbf{X}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$.

Using this factorization one can just choose the largest singular values from $\mathbf{\Sigma}$ and eliminate the vectors in $\mathbf{U}$ and $\mathbf{V}$ corresponding to the smallest singular values to compress the data. In fact, the largest singular values and their corresponding columns in $\mathbf{U}$ and $\mathbf{V}$ can explain most of the information in the matrix, and hence we can reconstruct the original matrix even after removing the smallest singular values. If we just pick the r largest singular values $\mathbf{X}_r = \mathbf{U}_{n\times r}\mathbf{\Sigma}_{r\times r}\mathbf{V}^T_{r\times p}$, then $\mathbf{X}_r$ will be the best rank r approximation of $\mathbf{X}$.

SVD can be applied to either co-occurrence matrix or the PPMI matrix to obtain the embedding. The word embedding is taken as the first d columns of U usually weighted by the singular values or the square root of singular values.

Skip-Gram with Negative Sampling (SGNS)

Word2vec is the most popular and state-of-the-art method for training vector space representations for words. There are two types of training for the word2vec model: Continuous Bag-Of-Words (CBOW) and skip-gram. Word2vec scans the corpus and employs a moving window which defines the context of the words. The intuition of the method is that co-words that frequently appear in the same window have high semantic relatedness and hence, they should have similar word vectors. Both variants use a single layer neural network but their objective is different. In the CBOW model, the aim is to predict a word given its context while in the skip-gram the objective is to predict the context given the word itself. The error term is defined in such a way that maximizes the dot product of words and their context words' vectors. At the end of the optimization, weights of the network are considered as the word vectors.

Word2vec learns the words and context words' vectors separately. Updating the weights of the network for context words is computationally expensive and requires iterating over the entire vocabulary for each input instance. For this reason, they have proposed to use hierarchical Softmax and negative sampling methods as optimization tricks in order to improve the efficiency (Mikolov et al. 2013b). Negative sampling works better in practice, in which some of the words are randomly sampled from the vocabulary (i.e. negative samples) to be updated.

Global Vectors (GloVe)

The GloVe model scans the corpus to determine the vocabulary and then build the global co-occurrence table by moving a context window over the text and counting the co-occurrences of the words. The co-occurrence table contains the global statistics about words that appear together in a window and is the primary source of information for unsupervised learning of word vectors. They exploit the information encoded in this table and formulate the problem as a sum of squared error minimization between the dot product of word and context word vectors and the log of their co-occurrences. Another weighting function is also applied to each error term to make the importance of error terms proportional to their co-occurrence value. This way, more frequent co-words will have more weight in updating their vectors.

Spectral Word Embedding with Negative Sampling (SENS)

Our approach at a glance builds a symmetric co-occurrence matrix, calculates the PMI matrix, applies a threshold on PMI matrix to remove only some of the negative PMI values, and finally factorizes the resulting matrix to find the embedding.

In the first step, we use a fixed-size context window and scan through the corpus to build the global co-occurrence statistics matrix just like many other methods such as GloVe. One of the key differences of our algorithm to others such as SGNS, is that we use a symmetric context in which we update X_{ij} and X_{ji} symmetrically whenever two words w_i and w_j appear in the same window. This will provide nice properties for the matrix in our optimization. Please note that from now on, we do not refer to context words as c since there is no distinction between words and contexts. Consequently, in our approach $|V_W| = |V_C| = n$ and hence, both X and M^{PMI} are symmetric matrices of dimension $n \times n$.

After calculating the global co-occurrence matrix X, we apply PMI on it to obtain M^{PMI}. Here, instead of using M^{PPMI}, we propose to apply a threshold α on the PMI table to obtain M^α where each entry in the matrix is calculated as follows:

$$m^\alpha_{ij} = \begin{cases} PMI(w_i, w_j) & PMI(w_i, w_j) > \alpha \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

We refer to the word pairs with $m^\alpha_{ij} > 0$ as positive examples and word pairs with $m^\alpha_{ij} < 0$ as negative examples. M^α is the final matrix that we factorize and the negative values in this matrix correspond to the negative examples that we employ in our model. Please note that by setting $\alpha = 0$ we get M^{PPMI} which does not exploit the negative examples. However, by choosing a negative threshold $\alpha < 0$, we keep a portion of negative values while disregarding the extreme cases. In fact, by adjusting this parameter α, we control the amount of negative examples to be used in the model.

It is noteworthy to mention that this thresholded PMI matrix, M^α, is different from what is being used in Levy and Goldberg's work (Levy and Goldberg 2014). In the shifted PPMI approach proposed in (Levy and Goldberg 2014), the PMI values are shifted by $-\log k$ and then all the negative

values are removed. This way, not only all the originally negative PMI values are removed, but also some of the small positive PMI values (which are less than $\log k$) are eliminated as well. On the contrary, our method does the opposite operation by encouraging to keep the negative values. In the following, we will see how this negative threshold α can provide a better approximation to the true PMI matrix.

Let us define our intuitive objective function that justifies the use of negative examples in the matrix.

$$\mathcal{J}(W) = \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^{\alpha}(\vec{w_i}.\vec{w_j}) \tag{4}$$

where $\vec{w_i}$ and $\vec{w_j}$ are the d-dimensional embedded word vectors for w_i and w_j in the vocabulary, and W is the $n \times d$ matrix of all word vectors. Here, we formulated the problem as a maximization task where $\mathcal{J}(W)$ has to be maximized.

In this maximization formulation, the algorithm will maximize the similarity of word vectors, $\vec{w_i}.\vec{w_j}$, whenever $m_{ij}^{\alpha} > 0$ (i.e. positive examples), and it will minimize the similarity of word vectors whenever $m_{ij}^{\alpha} < 0$ (i.e. negative examples). In other words, word pairs with a strong degree of association will be placed close to each other while the negative examples will be placed far apart in the latent embedded space. The use of negative examples is ignored in most embedding algorithm since zero or negative PMI values in M^{PMI} are typically ignored and considered useless as in the explicit PPMI (Bullinaria and Levy 2007) and factorizations methods on PPMI (Levy and Goldberg 2014). Here, we show that the appropriate use of negative examples can be beneficial to the spectral algorithms and factorization based embedding methods.

Please note that most of the entries in M^{α} are zero, and word pairs with $m_{ij}^{\alpha} = 0$ have no effect on our optimization. It is also noteworthy to mention that positive examples are indeed more informative than negative examples. This means that excessive use of negative examples can have destructive effects on the quality of final embedding since the algorithm will mostly focus on making negative examples far apart rather than making word vectors with strong association closer to each other. However, we show that appropriate use of negative examples improves the distribution of words in the embedded space and prevents the concentration effect.

Considering equation 4, we can rewrite our objective function as a trace optimization:

$$\begin{aligned}\mathcal{J}(W) &= \sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}^{\alpha}(\vec{w_i}.\vec{w_j}) = \sum_{i=1}^{n} \vec{w_i}.\Big(\sum_{j=1}^{n} m_{ij}^{\alpha}.(\vec{w_j})\Big) \\ &= Tr[W^T M^{\alpha} W] \end{aligned} \tag{5}$$

Then, our objective function has an optimal closed form solution as a result of the following two theorems.

Theorem 1 ***(Courant-Fischer Theorem)*** *Let A be a symmetric $n \times n$ matrix with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and corresponding eigenvectors $u_1, u_2, \ldots, u_n$, then:*

$$\lambda_1 = \min_{\|b\|=1} b^T A b = \min_{b \neq 0} \frac{b^T A b}{b^T b} = u_1^T A u_1, \tag{6}$$

$$\lambda_2 = \min_{\substack{\|b\|=1 \\ b \perp u_1}} b^T A b = \min_{\substack{b \neq 0 \\ b \perp u_1}} \frac{b^T A b}{b^T b} = u_2^T A u_2, \tag{7}$$

$$\vdots$$

$$\lambda_n = \lambda_{max} = \min_{\substack{\|b\|=1 \\ b \perp u_1 \perp \cdots \perp u_{n-1}}} b^T A b = \max_{\|b\|=1} b^T A b \tag{8}$$

$$= \max_{b \neq 0} \frac{b^T A b}{b^T b} = u_n^T A u_n \tag{9}$$

Proof of the Courant-Fischer theorem can be found in standard linear algebra textbooks (Dym 2013). The following theorem (Parlett 1998) is an immediate consequence of the Courant-Fischer theorem.

Theorem 2 *Given a symmetric matrix $A_{n \times n}$, and an arbitrary unitary matrix $B_{n \times d}$, then the trace of $B^T A B$ is maximized when B is an orthonormal basis for the eigenspace of A associated with its algebraically largest eigenvalues (Kokiopoulou, Chen, and Saad 2011). In particular, the eigenbasis itself is an optimal solution: If $U = [u_1, \ldots, u_d]$ is the set of eigenvectors associated with d largest eigenvalues $\lambda_1, \ldots, \lambda_d$, and $U^T U = I$, then:*

$$\max_{\substack{B \in \mathbb{R}^{n \times d} \\ B^T B = I}} Tr[B^T A B] = Tr[U^T A U] = \lambda_1 + \cdots + \lambda_d \tag{10}$$

Our thresholded PMI matrix, M^{α}, is symmetric and therefore, our optimization in equation 5 fits into theorem 2. Consequently, our optimization has an optimal closed form solution in which the word vectors matrix W is formed by the eigenvectors of M^{α} corresponding to its d algebraically largest eigenvalues. This formulation is simply a more accurate approximation of the true objective than SVD factorization of PPMI or shifted PPMI.

Connection to SVD and an alternative solution (SVD-NS)

Consider the singular value decomposition of the thresholded PMI matrix, $M^{\alpha} = U \Sigma U^T$. We know that U is the set of eigenvectors of $M^{\alpha} M^{\alpha^T}$ and V is the set of eigenvectors of $M^{\alpha^T} M^{\alpha}$. In case of symmetric input, $M^{\alpha} M^{\alpha^T} = M^{\alpha^T} M^{\alpha}$, and consequently, $U = V$ and the well-formed unique factorization of $U \Sigma U^T$ can be obtained.

Moreover, it is easy to show that eigenvectors of powers k of a matrix, A^k, are equivalent to the eigenvectors of the matrix itself, but, their eigenvalues will be taken to the power k.

$$Av = \lambda v \tag{11}$$

$$A^2 v = AAv = A(\lambda v) = \lambda(Av) = \lambda\lambda v = \lambda^2 v \tag{12}$$

Therefore, U which is the set of eigenvectors of $M^{\alpha}M^{\alpha T} = M^{\alpha 2}$ is equivalent to the eigenvectors of M^{α}, and Σ will be the absolute value (i.e. magnitude) of eigenvalues of M^{α} in decreasing order. As a consequence, one can apply SVD on the symmetric thresholded PMI matrix M^{α} and consider the first d columns of U corresponding to d largest singular values of the matrix as the final word embedding.

We should mention that this alternative solution is equivalent to the originally proposed solution only if the d largest singular values correspond to the d algebraically largest eigenvalues. More formally, if $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues in decreasing order, then the condition is satisfied if $\nexists \lambda_k < 0 \; such \; that \; |\lambda_k| \geq \lambda_d$. This is a guaranteed case when M^{α} is Positive Semi-Definite (PSD) since all the eigenvalues will be positive. Nevertheless, in case of non-PSD, M^{α} can be easily converted into a PSD matrix since it is already symmetric. This can be done by making it a diagonally dominant matrix in that each diagonal element is greater than or equal to the sum of all non-diagonal elements on that row, $m^{\alpha}_{ii} \geq \sum_{j,j\neq i} m^{\alpha}_{ij}$. In fact, by adjusting only the diagonal elements of M^{α} which are the strength of association of words with themselves, we can make it a PSD matrix. This process should not have drastic effects on the quality of embedding since it does not change the strength of pairwise word relations. However, in our experiments, we have used the alternative SVD solution on the original M^{α} without converting it to PSD and still achieved promising results. We refer to this alternative solution as SVD-NS. The source code of our algorithm is available at: https://github.com/behrouzhs/svdns.

Experiments

Data and Vocabulary

For the training of models, we have used English Wikipedia dump of March 05, 2016. After stripping the HTML tags and extracting the clean text of Wikipedia articles, we removed all the special characters and punctuation from the text. Then we converted the text into lowercase and tokenized it. The data have 2.1 billion tokens resulting in 8.9 million unique words as vocabulary. However, most of the vocabulary are infrequent words or they belong to named entities such as names of people, places, and organizations.

We have applied a filtering on the vocabulary to target the words in English dictionary as well as to reduce the size of vocabulary to a reasonable range. First, we filtered out all the words that appeared in the data less than 5 times. This is a common threshold that is used in other algorithms as well (e.g GloVe). We also removed all the words that contained non-English alphabet (e.g. names of people). Afterwards, we used WordNet database to identify words that exist in the English dictionary. WordNet is an ontology defining the relationship between the words and has an internal categorization of words into verbs, nouns, adjectives, and adverbs. We have used all the words that appeared in WordNet database as the English vocabulary. We also kept all the words with more than 3000 occurrences which do not necessarily exist in the English dictionary. After these filtering steps, the 8.9M unique words in Wikipedia were filtered and resulted in 163188 words.

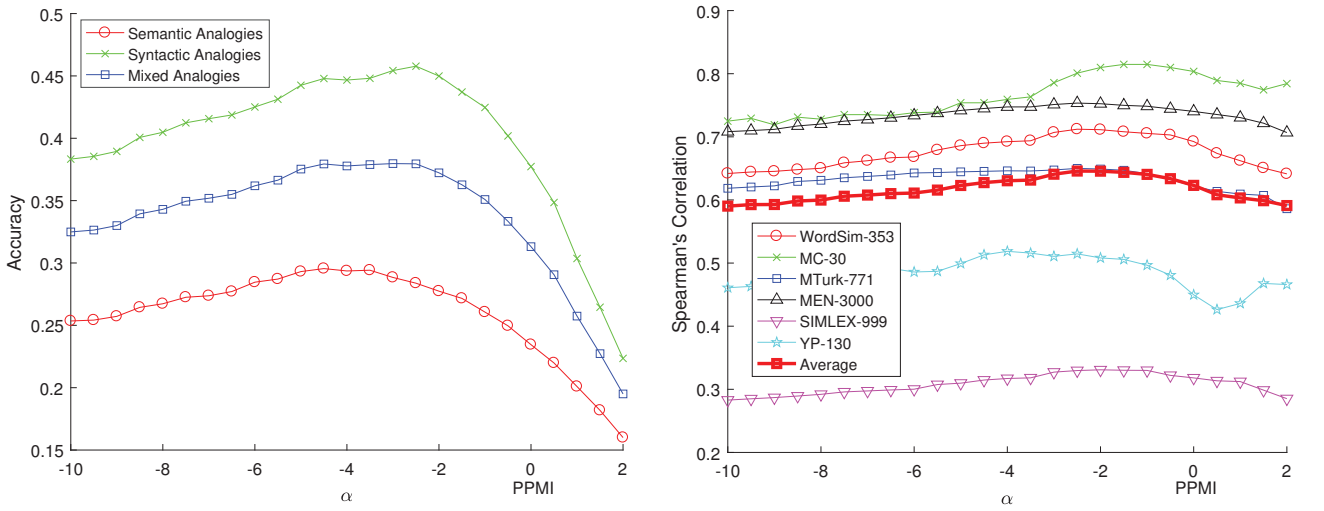
In our experiments, all the algorithms were trained on the exact same preprocessed input text data. We have also used the exact same vocabulary for all the algorithms. This will ensure a fair comparison as the size of the training data has a great influence on the quality of embeddings. The dimensionality of embeddings is 100 in all our experiments.

Evaluation method

For the evaluation of algorithms, we have used two well-known tasks of word similarity and word analogy. For the word similarity task, there exist several datasets containing word pairs with their corresponding human-assigned similarity score. These datasets are commonly used to evaluate the quality of word embeddings and to see how good they have modeled the word similarities in the embedding. The quantitative values of scores are not important here, but rather the ranking of different word pairs is the main focus. Therefore, Pearson's rank-order correlation is always used in word similarity task. In this task we have used 8 different dataset including *WordSim353* (WS-M) (Finkelstein et al. 2001), *WordSim Similarity* (WS-S) and *WordSim Relatedness* (WS-R), MEN (Bruni et al. 2012), *Mechanical Turk* (MTurk), SimLex (Hill, Reichart, and Korhonen 2016), MC, and YP (Yang and Powers 2006). For the analogy task, we have used Google's analogy dataset (Mikolov et al. 2013a) which contains 19544 questions of the form "a is to a^* as b is to b^*". Given three of the elements, the algorithm has to predict the missing element. About half of questions are semantic (e.g. "*brother* is to *sister* as *son* is to *daughter*") and the other half are syntactic questions (e.g. "*cold* is to *colder* as *short* is to *shorter*").

Analysis of the amount of negative examples

Let us first analyze the effect of volume of negative examples in the model. For this reason, we have run SVD-NS algorithm with various PMI thresholds ranging in $[-10, +2]$ and evaluate each case on analogy and word similarity tasks. Figure 1 illustrates the results of this experiment. Figures 1a and 1b show the accuracy of our algorithm on word analogy and word similarity tasks, respectively. Please note that the special case of $\alpha = 0$ is equivalent to the SVD factorization of PPMI matrix (Levy, Goldberg, and Dagan 2015). As we can see from both figures, using negative examples can boost the performance of factorization methods dramatically and yields a better quality embedding. In both analogy and word similarity tasks, the peak performance and accuracy has achieved when $\alpha = -2.5$ and in that case the number of positive examples is 1.54 times the number of negative examples. As we discussed before, excessive use of negative examples will have harmful effects since it will shift the focus of the algorithm to less important information. For instance, as we can observe from the figure 1 and it was previously shown in (Bullinaria and Levy 2007), using the entire PMI matrix M_0^{PMI} and factorizing it using SVD yields worse results than factorizing M^{PPMI}.



(a) Word analogy task

(b) Word similarity task

Figure 1: Accuracy of SVD-NS algorithm on (a) word analogy (b) word similarity tasks with respect to the amount of negative samples.

Comparison with other algorithms

Table 1 compares 12 algorithms on 8 word similarity datasets. The numbers in the table are Pearson's correlation between the rankings provided by the algorithms and the rankings of the human-scoring.

SVD, SVD-L, and SVD-S are the factorizations of co-occurrence, log co-occurrence, and square root of co-occurrence matrices, respectively. This type of factorization is popularized in NLP through Latent Semantic Analysis (Deerwester et al. 1990). PPMI-SVD-US, PPMI-SVD-Us, and PPMI-SVD-U are all SVD factorizations on PPMI matrix but they have different singular vector weighting scheme in that, they weight the singular vectors by singular values, square root of singular values, and no weighting, respectively. SPPMI-SVD-Us and SPPMI-SVD-U are the SVD factorizations on shifted PPMI matrix (Levy and Goldberg 2014) where the shift is $\log 5$. GloVe is trained with its recommended parameter setting (i.e. $x_{max} = 100$), CBOW and SGNS are trained with negative sampling set to 5. Our proposed algorithm, SVD-NS, is trained with $\alpha = -2.5$.

As we can see from table 1, our algorithm provides the best results in 6 out of 8 datasets. SGNS is the best on only WordSim Similarity dataset, and PPMI-SVD-U is the best on MC dataset and on the rest of the datasets our method outperforms others by a fair margin. We have to mention that in analogy task SGNS provides better results than SVD-NS. However, our main goal is to boost matrix factorization methods with the use of negative examples. We have parallelized SVD-NS algorithm using OpenMP and it is one order of magnitude faster than multi-threaded SGNS.

Conclusion

In this paper, we analyzed the use of negative examples in word embedding context both theoretically and empirically. We proposed an intuitive objective function for the word embedding problem and provided an optimal solution for it using matrix factorization techniques. Our thresholded PMI matrix is simply a better approximation of the word associations. And our objective function is a more accurate approximation of the true objective than SVD factorization of PPMI or shifted PPMI. Our algorithm removes the use of many hyper-parameters in other algorithms such as eigenvalue weighting and dealing with word and context vectors separately. In fact, our optimal solution is achieved using the eigenvectors themselves and no additional weighting is needed. Moreover, it builds, maintains, and exploits symmetric co-occurrence and PMI matrices, and consequently, its word and context vectors happen to be the same in our algorithm. The only parameter in our method is α which controls the amount of negative examples to be used in the algorithm. As a rule of thumb, one should not have negative examples more than $\frac{2}{3}$ of positive ones.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Bengio, Y.; Schwenk, H.; Senécal, J.-S.; Morin, F.; and Gauvain, J.-L. 2006. *Neural Probabilistic Language Models*. Berlin, Heidelberg: Springer Berlin Heidelberg. 137–186.

Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D.; and Lai, J. C. 1992. Class-based n-gram models of natural language. *Computational linguistics* 18(4):467–479.

Bruni, E.; Boleda, G.; Baroni, M.; and Tran, N.-K. 2012. Distributional semantics in technicolor. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 136–145. Association for Computational Linguistics.

Bullinaria, J. A., and Levy, J. P. 2007. Extracting semantic

Table 1: Evaluation of different word embedding algorithms on 8 word similarity datasets.

	WS-S 203	WS-R 252	WS-M 353	MC 30	MEN 3000	MTurk 771	SimLex 999	YP 130
SVD	0.4913	0.1890	0.3624	0.4096	0.2722	0.1600	0.0881	0.122
SVD-L	0.2705	0.1091	0.1866	0.1301	0.2321	0.2650	0.0882	0.0878
SVD-S	0.6245	0.3235	0.4854	0.6483	0.5356	0.4124	0.212	0.2445
PPMI-SVD-US	0.6865	0.4893	0.5968	0.7257	0.6823	0.5713	0.2683	0.3710
PPMI-SVD-Us	0.7189	0.6007	0.6674	0.7749	0.7302	0.6155	0.2976	0.4490
PPMI-SVD-U	0.7201	0.6387	0.6925	**0.8035**	0.7402	0.6217	0.3180	0.4496
SPPMI-SVD-Us	0.6469	0.5416	0.6046	0.7405	0.7075	0.5735	0.2728	0.4472
SPPMI-SVD-U	0.6412	0.5813	0.6238	0.7566	0.7000	0.5766	0.2764	0.4574
GloVe	0.6743	0.5534	0.5991	0.6649	0.7040	0.6227	0.3154	0.4872
CBOW	0.7457	0.5853	0.6714	0.7427	0.7079	0.6132	0.3274	0.3403
SGNS	**0.7587**	0.6519	0.7096	0.7946	0.7306	0.6441	0.3236	0.4679
SVD-NS	0.7519	**0.6537**	**0.7120**	0.8008	**0.7534**	**0.6499**	**0.3297**	**0.5145**

representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.

Church, K. W., and Hanks, P. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16(1):22–29.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167. ACM.

Deerwester, S.; Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6):391.

Dym, H. 2013. *Linear algebra in action*, volume 78. American Mathematical Soc.

Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; and Ruppin, E. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, 406–414. ACM.

Fürnkranz, J. 1998. A study using n-gram features for text categorization. *Austrian Research Institute for Artifical Intelligence* 3(1998):1–10.

Hill, F.; Reichart, R.; and Korhonen, A. 2016. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Kokiopoulou, E.; Chen, J.; and Saad, Y. 2011. Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications* 18(3):565–602.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, 1188–1196.

Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, 2177–2185.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 3111–3119.

Parlett, B. N. 1998. *The symmetric eigenvalue problem*. SIAM.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–1543.

Salton, G. 1971. The smart retrieval system–experiments in automatic document processing.

Suen, C. Y. 1979. N-gram statistics for natural language understanding and text processing. *IEEE transactions on pattern analysis and machine intelligence* (2):164–172.

Turney, P. D., and Pantel, P. 2010. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research* 37:141–188.

Yang, D., and Powers, D. M. 2006. *Verb similarity on the taxonomy of WordNet*. Masaryk University.