Learning Word Vectors with Linear Constraints: A Matrix Factorization Approach

Wenye Li^{1,2}, **Jiawei Zhang**^{1}, **Jianjun Zhou**^{2} and **Laizhong Cui**^{3}
^{1} The Chinese University of Hong Kong, Shenzhen, China
^{2} Shenzhen Research Institute of Big Data, Shenzhen, China
^{3} Shenzhen University, Shenzhen, China
wyli@cuhk.edu.cn, 216019001@link.cuhk.edu.cn, benz@sribd.cn, cuilz@szu.edu.cn

Abstract

Learning vector space representation of words, or word embedding, has attracted much recent research attention. With the objective of better capturing the semantic and syntactic information inherent in words, we propose two new embedding models based on the singular value decomposition of lexical co-occurrences of words. Different from previous work, our proposed models allow for injecting linear constraints when performing the decomposition, with which the desired semantic and syntactic information will be maintained in word vectors. Conceptually the models are flexible and convenient to encode prior knowledge about words. Computationally they can be easily solved by direct matrix factorization. Surprisingly simple yet effective, the proposed models have reported significantly improved performance in empirical word analogy and sentence classification evaluations, and demonstrated high potentials in practical applications.

1 Introduction

Vector space representation, or word embedding, refers to the technique of mapping words or phrases to vectors of real numbers. As a key step in natural languages processing (NLP), it has attracted much research attention since the 1970s. A popular approach is the one-hot representation, with which words are treated as atomic units and represented as sparse vectors [Salton *et al.*, 1975]. Although simple and successful in certain scenarios, the approach has evident limitations. There is no way to compare the similarity or relatedness of two words, which often restricts the method from even wider applications.

Very recently, people began to resort to machine learning-based approaches for word embedding, which tries to automatically obtain a semantically *meaningful* representation of words with the support from a large text corpus. Methods trying to generate this semantic mapping include matrix factorizations, neural networks, probabilistic models, and so on [Lund and Burgess, 1996; Bengio *et al.*, 2003; Hofmann, 1999]. A key result achieved in this area is the concept of distributed representation of words, which achieves state-of-the-art results in many practical applications [Bengio *et al.*, 2003; Mikolov *et al.*, 2013; Pennington *et al.*, 2014].

Despite the success, these distributed word embedding approaches often suffer from the vulnerability of not being able to distinguish word "genuine" similarity from "associative" similarity [Tversky, 1977]. In practice, this vulnerability could cause significant side effects to downstream applications if not properly handled, which brings a non-trivial technical challenge to these word embedding techniques.

To tackle the challenge, our work follows the machine learning framework. Instead of seeking a completely distributed representation of words, we aim to answer the following question. *Is it possible to integrate the prior knowledge into this learning process*? To achieve this goal, our work proposes two linearly constrained decomposition models for word embedding, the *additive* model and the *projection* model. Starting with a word co-occurrence matrix, the two models try to find a low rank approximation of the matrix while satisfying given linear constraints. These constraints are simple, flexible and could encode domain specific knowledge about words in a variety of forms, which helps to make up the vulnerability of the distributed word embedding approaches.

There are a number of reasons to consider integrating prior knowledge in embedding. Namely, it benefits real applications with better visualization, desired semantic relationship among word vectors, and the ability to differentiate word similarity from relatedness, etc. Empirically, the word vectors generated from the *additive* model bring significantly better results in a benchmark evaluation of word analogies. The word vectors generated from the *projection* model give evidently improved accuracies in sentiment analysis and related tasks.

Similarly to unconstrained models, the proposed models can be solved through direct matrix factorization, which is effective for moderately large problems. For even larger problems, approximation is possible through modern optimization techniques, which admit high execution efficiency in parallel and distributed computing platforms and thereby provide a practical solution to very large problems.

The paper is organized as follows. Firstly we introduce the related work in Section 2, and present our *additive* and *projection* models in Section 3. After reporting the empirical results in Section 4, we conclude the paper with discussion and future work in Section 5.

2 Related Work

2.1 Learning Word Vectors

Our work is closely related to the word embedding techniques recently studied in machine learning and NLP areas. These techniques usually start with a "term-term" matrix where each element gives the information about the number of times that two words co-occur in a context. The idea of using co-occurrence for word embedding is that words with similar semantics tend to occur in similar contexts, and therefore the statistics provides a natural basis for semantic representation. Convincing evidences and excellent empirical results support this argument well in the literature [Landauer and Dumais, 1997; Lund and Burgess, 1996; Mikolov *et al.*, 2013; Pennington *et al.*, 2014].

Based on the co-occurrence statistics, a natural idea is to factorize the matrix into a low-rank representation for vector representations of words. A number of models were developed along this line. The hyperspace analogue to language model [Lund and Burgess, 1996] directly uses the number of co-occurrences as the "term-term" matrix entry. The correlated occurrence analogues to lexical semantic model transforms the co-occurrence statistics through a correlation-based normalization. More recent work also suggests positive point-wise mutual information and Hellinger PCA for the transformation and reports improved results [Bullinaria and Levy, 2007; Lebret and Collobert, 2013].

Besides matrix factorization methods, another line of work resorts to neural networks to learn word vectors within local context windows. A neural network architecture for language and word modeling was proposed in [Bengio *et al.*, 2003], which has significant influence on subsequent research. For example, based on the idea of re-constructing linguistic contexts with large text corpus, the work of [Mikolov *et al.*, 2013] proposes the *skip-gram (SG)* model and the *continuous bag-of-words (CBOW)* model to learn word representations. Another work, the *Global Vectors (GloVe)* algorithm, performs on aggregated global term-term co-occurrence statistics from a corpus, and the resulting representation often exhibits interesting linear sub-structures of the word vector space [Pennington *et al.*, 2014].

It is worth mentioning that, as revealed in the work of [Levy and Goldberg, 2014], the two lines of work have a close relationship. The neural network based word embedding approaches can also be studied from a matrix factorization point of view. And therefore matrix factorization in fact provides a unified framework to investigate distributed word embedding techniques.

2.2 Encoding Prior Knowledge in Word Vectors

A key challenge to distributed word vector learning approaches is that the corpus-driven approaches are often unable to distinguish "genuine" similarity from "associative" similarity (relatedness) [Tversky, 1977]. Based on the distributional hypothesis, these approaches could generally learn embeddings that capture both similarity and relatedness reasonably well, but neither perfectly. As an example, antonyms are often interchangeable in context and thus have similar word embeddings even though they denote totally opposite meanings, which could often bring non-trivial side effects in applications [Kiela *et al.*, 2015].

As a remedy for the challenge, a natural choice is to encode the knowledge from existing knowledge-bases, such as the WordNet [Miller, 1995] and the paraphrase database [Ganitkevitch *et al.*, 2013], into word vectors. Wide lexical resources, such as synonym/antonym relations between words, are available in these knowledge-bases which provide a good start for us to improve downstream performances.

To integrate these manually annotated knowledge into word vectors, a number of *combinational* approaches were proposed which modify the distributional objective through a combination of context-based learning and knowledge-based learning. Specifically, the work of [Yu and Dredze, 2014] combines *CBOW* objective with a set of linguistic constraints describing a lexical relation. The work of [Kiela *et al.*, 2015] combines *SG* objective with similar linguistic constraints. The work of [Liu *et al.*, 2015] transforms linguistic knowledge into ordinal constraints, which results in a neural network architecture for optimization. The work of [Osborne *et al.*, 2015] encodes the knowledge in the form of canonical correlation analysis. All these approaches have achieved some success in real applications, with reasonable computational overhead.

Besides the combinational approaches, post-processing methods were designed as well, which optimize a cost function to bring semantically similar words closer to each other while not being far away from their initial word vectors. The methods in this category include the retro-fitting method [Faruqui *et al.*, 2014], the counter-fitting method [Mrkšić *et al.*, 2016], the PARAGRAM embedding method [Wieting *et al.*, 2015], etc. Simple and computationally efficient as they are, these methods have also exhibited good performance in real applications.

3 Models

3.1 Linear Operations on Word Vectors

In practice, different types of knowledge exist in different forms and need different models to encode them. This paper considers two types of linear operations to encode semantic relationship among word vectors: the additive operation and the projection operation. The two types of operations are common and powerful with various applications.

The additive operation aims to encode the semantic similarity and difference of multiple words into word vectors. For example, a vector equation $w_{good} = 0.5 \times w_{excellent}$ denotes that *good* is half *excellent*; $w_{good} = -1 \times w_{bad}$ denotes that *good* is opposite to *bad* (additive inverse); $w_{very} + w_{good} = 0.9 \times w_{excellent}$ denotes that *very good* is quite similar to that of *excellent*.

The additive operation, or constraint, is especially useful for modeling word analogy. Word analogy operates on four words and probes the finer structure of the vector space by examining the distance or similarity between word vectors. For example, the analogy, *man is to woman as boy is to girl*, could be encoded by the vector equation $w_{man} - w_{woman} = w_{boy} - w_{girl}$.

As pointed out in [Bengio, 2009], ensuring word analogy in the vector space favors models that produce dimensions of semantics, and thereby captures the multi-clustering idea of the distributed representation.

The projection operation ensures that the word vectors lie in specific hyperplanes for a clear separation of words with different natures. For example in a practical application of sentiment analysis of movie reviews, *good* and *excellent* are often used in positive reviews and we may wish to place their word vectors in one hyperplane, which has no intersection with another hyperplane containing the word vectors of *bad* and *terrible* that typically appear in negative reviews. In our evaluation (ref. Section 4.3), NLP applications benefit much from this projection operation with significantly improved sentence classification accuracy.

Previously, the work of [Rothe and Schütze, 2016] models the polarity operation on exactly two words, i.e., $w_{good} = -1 \times w_{bad}$. The counter-fitting method [Mrkšić *et al.*, 2016] also injects constraints on two words. Comparatively our *additive* model generalizes the polarity operation and the counter-fitting method, and extends to linear relationship among an arbitrary number of words. Besides, our *projection* model is the first work that separates words with different semantics into parallel hyperplanes, as we are aware.

3.2 Additive Model

To implement an embedding technique that ensures additive constraints on word vectors, we propose the following model. For a text corpus with a vocabulary size of n, we construct an $n \times n$ symmetric matrix $D = \{d_{ij}\}$ with each element $d_{ij} = \log(n_{ij} + 1)$ where n_{ij} is the number of co-occurrences of the i-th word and the j-th word in the same context if $i \neq j$ and n_{ii} is the number of occurrences of the i-th word in the given corpus. The matrix D gives the pairwise co-occurrence information of all words. Our objective is to seek an $n \times r$ $(r \ll n)$ matrix W, each row being a word vector, to:

$$\min \left\|D - WW^T\right\|_F^2 \tag{1}$$

satisfying

$$AW = O_{m,r}$$

where $\|\bullet\|_F^2$ denotes the squared Frobenius norm of a matrix, A is an $m \times n$ $(m \ll n)$ matrix and $O_{m,r}$ is a zero matrix of size $m \times r$.

Comparing with classical matrix factorization-based embedding methods, the *additive* model exerts m linear constraints on each column of W in a mathematical form of $AW = O_{m,r}$.

The linearly constrained optimization problem in Equ. (1) can be solved via follows. Let an $n \times (n-m)$ matrix K be a basis for the null space of A which satisfies $K^T K = I_{n-m}$ where I_{n-m} is an identity matrix of size $(n-m) \times (n-m)$. Then for any W satisfying $AW = O$, it can be expressed as $W = KU$, where U is an $(n-m) \times r$ matrix.

Expand K to an orthonormal basis of R^n, $\tilde{K} = \begin{bmatrix} K & \bar{K} \end{bmatrix}$, which can be obtained through QR orthogonal-triangular decomposition of A^T. Considering that $\tilde{K}\tilde{K}^T = \tilde{K}^T\tilde{K} = I_n$, then we have

$$\left\|\tilde{K}^T \left(D - WW^T\right) \tilde{K}\right\|_F^2 = \left\|D - WW^T\right\|_F^2 \tag{2}$$

and

$$\tilde{K}^T \left(D - WW^T\right) \tilde{K} = \begin{bmatrix} K^T DK - UU^T & K^T D\bar{K} \\ \bar{K}^T DK & \bar{K}^T D\bar{K} \end{bmatrix} \tag{3}$$

The terms $K^T D\bar{K}, \bar{K}^T DK, \bar{K}^T D\bar{K}$ are all constant. Therefore minimizing the norm of $\tilde{K}^T \left(D - WW^T\right) \tilde{K}$ becomes equivalently the problem of seeking an $(n-m) \times r$ matrix U to minimize the norm of $K^T DK - UU^T$, which is an unconstrained optimization problem and can be easily solved by singular value decomposition or eigen-decomposition [Golub and Van Loan, 2012].

3.3 Projection Model

To implement an embedding technique that supports projection constraints on word vectors, we propose the following model. For an $n \times n$ co-occurrence matrix D, we seek an $n \times r$ matrix W to

$$\min \left\|D - WW^T\right\|_F^2 \tag{4}$$

satisfying

$$AW^T = B$$

where A is an $m \times r$ matrix and $B = \{b_{ij}\}$ is an $m \times n$ matrix.

Denote the m rows of A by $a_1, \cdots, a_m$ and the n rows of W by $w_1, \cdots, w_n$. For the j-th word, a constraint $a_i w_j^T = b_{ij}$ ensures that its word vector w_j appears in a specified hyperplane. And with all m constraints, w_j appears in the intersection of the m given hyperplanes.

To solve the optimization problem in Equ. (4), similarly we rewrite $W^T = NB + KU$ with $N = A^T \left(AA^T\right)^{-1}$ and matrix K is a basis for the null space of A. Then considering the fact that $N^T K = O_{r,r-m}$, $K^T N = O_{r-m,m}$ and $K^T K = I_{r-m}$, we have:

$$D - WW^T = \left(D - B^T N^T NB\right) - U^T U \tag{5}$$

Now the problem becomes again unconstrained. We seek an $(r-m) \times n$ matrix U to minimize the Frobenius norm of $\left(D - B^T N^T NB\right) - U^T U$. Similarly this can be solved by decomposing the matrix $D - B^T N^T NB$.

3.4 Scalability Issues

As remarked in previous studies [Mikolov *et al.*, 2013; Pennington *et al.*, 2014], a key challenge to matrix factorization-based embedding approaches lies in the computational issue. To fully decompose an $n \times n$ matrix through SVD, the best known algorithm has a time complexity of $O\left(n^3\right)$ [Golub and Van Loan, 2012] and a space complexity of $O\left(n^2\right)$, which is quite impractical for large problems.

Fortunately in our projection model, the matrix to be decomposed, $D - B^T N^T NB$, has special structures. D is usually sparse. B is an $m \times n$ matrix. N is an $r \times m$ matrix. Considering that m and r are two small numbers and far less than n, multiplying $B^T N^T NB$ with an $n \times 1$ column vector becomes very cheap. Therefore we are able to solve the problem very effectively by iteratively finding each column vector of U with the power method.

For the additive model, the matrix to be decomposed, $K^T DK$, is dense and its structure is hard to explore. We found that conventional matrix factorization approaches are still applicable and effective in many cases. On a mainstream workstation, we are able to decompose a moderately large matrix ($n = 50,000, r = 300$) within 30 minutes (ref. Section 4.4).

For even larger un-structured problems (say, $n > 100,000$), the computation and memory requirements grow quickly. It becomes infeasible to directly decompose such a large matrix. As a remedy, we can re-write our two models as an optimization problem of seeking two matrices X and Y to:

$$min_{X,Y} \left\|Z - XY^T\right\|_F^2, \; s.t. \; X = Y, \tag{6}$$

for a given matrix Z. The problem can be approximated by:

$$min_{X,Y} \left\|Z - XY^T\right\|_F^2 + \left\|\Gamma^T (X - Y)\right\|_F^2 + \frac{\rho}{2} \left\|X - Y\right\|_F^2 \tag{7}$$

where Γ denotes the matrix of dual variables, and ρ is a positive number. This becomes a standard problem that can be solved by the alternating direction method of multipliers (ADMM) [Boyd *et al.*, 2011], which executes efficiently in parallel or distributed computing platforms with high scalability. The detailed discussion of the ADMM method goes beyond the scope of the paper, and is therefore omitted.

4 Evaluation

The proposed models were evaluated on different NLP applications, including word visualization, word analogy and sentence classification. The results demonstrated high potentials in practical applications.

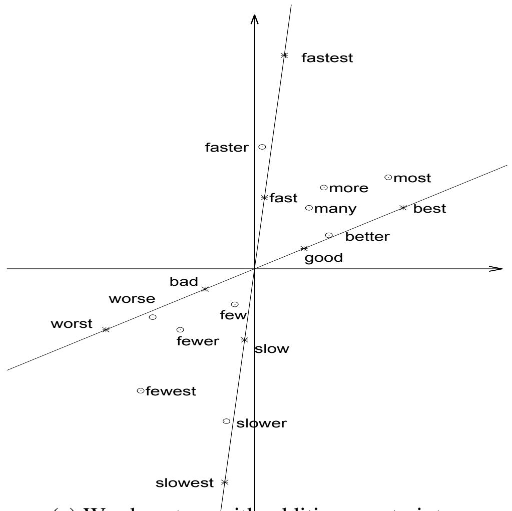
4.1 Word Visualization

Our first experiment is on word visualization. We chose a sub-matrix of the co-occurrence matrix generated from the Wikipedia dump with 4.2 billion sentences. With simple *additive* constraints: $w_{good} + w_{bad} = 0$, $w_{fast} + w_{slow} = 0$, $w_{good} = 0.33 \times w_{best}$, $w_{bad} = 0.33 \times w_{worst}$, $w_{fast} = 0.33 \times w_{fastest}$ and $w_{slow} = 0.33 \times w_{slowest}$, the *additive* model generated an embedding in 2-dimensional space. From the results shown in Fig. 1a, we can see that the constraints not only place the words *good/bad/best/worst/fast/slow/fastest/slowest* correctly, but also place the words *better/worse/few/fewer/fewest/...* by their semantic meanings.

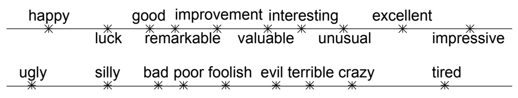
The second example is on ensuring constraints with the *projection* model, which requires a set of words be in two parallel hyperplanes (specified by $aw^T = 1$ and $aw^T = -1$ respectively where a is a row vector of random coefficients). From the visualization results in Fig. 1b, we can observe the correct placement of the words. The *positive* words *happy/luck/good/...* appear in one line, while the *negative* words *ugly/silly/bad/...* appear in a parallel line.

4.2 Word Analogy

Our next experiment is on the word analogy task, as described in the work of [Mikolov *et al.*, 2013]. Its objective is to answer questions of the type: *word a is to word b as word c is to word _?*. The benchmark set has $19,544$ such questions, dividing into a semantic category and a syntactic category. Questions



(a) Word vectors with additive constraints



(b) Word vectors with projection constraints

Figure 1: Visualization of words with additive constraints and projection constraints.

in the semantic category are mostly about people or places. Questions in the syntactic category are mostly about verb tenses or forms of adjectives.

For each question, each method uniquely identifies the missing word. An exact correspondence is counted as a correct match. The question is answered by finding the word whose representation w is closet to $w_b - w_a + w_c$ in the ℓ_2-distance (*SVD* and our model) or the cosine similarity (*SG, CBOW* and *GloVe*). The accuracy of a method is defined as the ratio between the number of correctly identified words and the total number of questions.

The experiment was carried out on the latest 2017 Wikipedia dump of English articles with 1 billion and 4.2 billion tokens respectively [1]. The dump is in XML format. After removing XML tags and special symbols, tokenizing and lowercasing each token [2], we were left with 27 character text (lowercase letters and spaces). We built a vocabulary of the $50,000$ most common words, and then constructed a matrix of co-occurrence statistics. When constructing the co-occurrence matrix, we used the context window size of 5.

We compared the results of our *additive* model against the results of *SG*, *CBOW* and *GloVe*. The results from the *SVD* decomposition of the co-occurrence matrix is also provided as a baseline. For our *additive* model, 1%, 2%, 3% of questions (with answers) were randomly chosen and used as the prior knowledge during training. For each selected question and its

[1]https://dumps.wikipedia.org/

[2]http://mattmahoney.net/dc/textdata

Models			Analogy		
Name	Dim	Size	SEM	SYN	Overall
SVD	100	1.0B	23.4	31.3	27.4
GloVe	100	1.0B	46.3	38.4	42.3
CBOW	100	1.0B	34.8	37.0	36.0
SG	100	1.0B	49.7	39.4	44.1
Additive(1%)	100	1.0B	74.2	66.3	70.2
Additive(2%)	100	1.0B	89.2	91.3	90.3
Additive(3%)	100	1.0B	100.0	99.7	99.9
SVD	200	1.0B	23.4	31.2	27.3
GloVe	200	1.0B	58.7	49.5	54.0
CBOW	200	1.0B	40.8	42.9	42.0
SG	200	1.0B	63.5	47.0	54.5
Additive(1%)	200	1.0B	74.2	66.3	70.2
Additive(2%)	200	1.0B	89.3	91.3	90.3
Additive(3%)	200	1.0B	100.0	99.8	99.9
SVD	200	4.2B	38.3	46.6	42.5
GloVe	200	4.2B	71.1	60.5	65.7
CBOW	200	4.2B	52.9	55.3	54.1
SG	200	4.2B	70.8	63.3	67.0
Additive(1%)	200	4.2B	74.2	66.3	70.2
Additive(2%)	200	4.2B	88.9	90.7	89.8
Additive(3%)	200	4.2B	100.0	99.9	99.9
SVD	500	4.2B	38.5	46.6	42.6
GloVe	500	4.2B	77.1	65.5	71.2
CBOW	500	4.2B	55.3	60.4	57.9
SG	500	4.2B	67.1	67.9	67.5
Additive(1%)	500	4.2B	74.2	66.3	70.2
Additive(2%)	500	4.2B	88.9	90.8	89.9
Additive(3%)	500	4.2B	100.0	99.9	99.9
Retrofitting	-	-	-	-	49.9
SPPMI	-	-	-	-	65.5

Table 1: Comparison of semantic/syntactic (denoted by SEM/SYN) accuracies on the word analogy task (shown in percentage).

answer, *word a is to word b as word c is to word d*, a constraint in the form of $w_a - w_b = w_c - w_d$ was added as a constraint to the optimization problem.

The results are shown in Table 1. From the results, we can see that *SG, CBOW* and *GloVe* reported better results than a naïve implementation of the *SVD* approach. On the other hand, with 1% questions and answers as the prior knowledge for training, our model reported evidently better results, which confirms the benefits of encoding prior knowledge in word vectors. When we increased the portion of training questions to 2%, the improvement is even more significant and the accuracy is nearly 100% with 3% questions.

In addition to *SVD, SG, CBOW* and *GloVe* methods, the results of the *retrofitting* method from [Faruqui *et al.*, 2014] and the *SPPMI* method from [Levy and Goldberg, 2014] are also listed in the table[3]. It can be seen that our results also outperformed these results significantly.

[3]These results are taken as the best ones from their authors' published results among all parameter settings, which may over-estimate the methods' performance.

4.3 Sentence Classification

Besides the visualization and the word analogy tasks, we also evaluated our *projection* model's performance on practical sentence classification applications with *convolutional neural networks (CNN)* [Kim, 2014]. The *CNN* model achieves good classification accuracy across a variety of text classification tasks and has hence become a standard baseline for classification.

Three applications were investigated in the evaluation using the *movie review* dataset, the *subjectivity* dataset [Pang and Lee, 2005] and the *20-newsgroups* dataset [Lang, 1995]. The *movie review* dataset is a collection of $5,331$ positive and $5,331$ negative movie review sentences. The *subjectivity* dataset is a collection of $5,000$ subjective and $5,000$ objective pre-processed sentences. The *20-newsgroups* dataset is a collection of about $20,000$ messages evenly distributed in 20 groups. In our experiments, 5 groups (comp.*) were used as the positive samples and 4 groups (rec.*) were used as the negative samples.

Under the supervised learning setting, our objective in the experiment is to build a classifier based on the training data that is able to predict whether a future movie review is positive or negative, to predict whether a sentence is subjective or objective, and to predict the category a short message belongs to. To train the classifier, we used a publicly-available *CNN* implementation[4] on Google's Tensorflow platform. As suggested in [Kim, 2014], the training process was done through stochastic gradient descent over shuffled mini-batches based on the Adadelta update rule [Zeiler, 2012].

In the experiment, we hope to test the influence of different word representations on the classification accuracies. Specifically, we compared with three representation schemes as illustrated in [Kim, 2014] [5]: **1. rand**: All words are randomly initialized and then modified during training. **2. static**: the pre-trained vectors from *word2vec (CBOW)*, *GloVe* and our *projection* model. These vectors are kept static during training. **3. nonstatic**: the pre-trained vectors from *word2vec (CBOW)*, *GloVe* and our *projection* model. These word vectors can be fine-tuned during the training process.

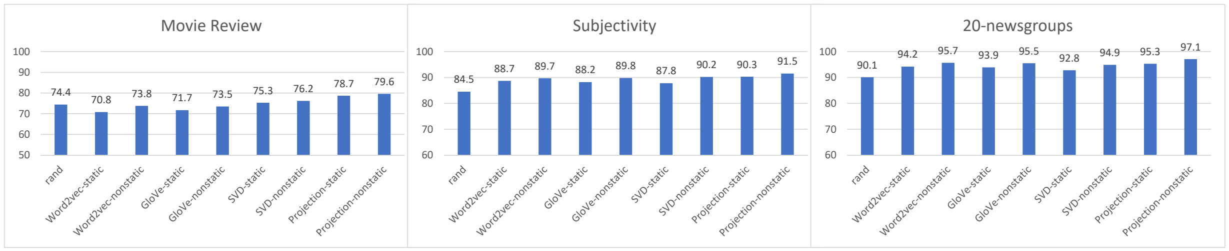
To obtain word vectors with the *projection* model, all words are projected into three parallel hyperplanes. Given each training sentence set, the words that appear only in positive/subjective/comp.* sentences were projected into the hyperplane of $aw^T = 1$ where a is a 300-dimensional random row vector; the words that appear only in negative/objective/rec.* sentences were projected into the hyperplane of $aw^T = -1$; all other words were projected into the hyperplane of $aw^T = 0$.

In the experiment, we used a similar experimental and hyperparameter setting as described in [Kim, 2014]. Ten-fold cross validation was used and the average accuracy in predicting positive/negative reviews and subjective/objective sentences was collected as a record of the performance of the trained classifiers.

The results are shown in Fig. 2. The horizontal axis lists the embedding methods under different experimental schemes

[4]https://github.com/dennybritz/cnn-text-classification-tf

[5]https://github.com/yoonkim/CNN_sentence



(a) Movie Review (b) Subjectivity (c) 20-newsgroups

Figure 2: Sentence classification accuracies on different datasets.

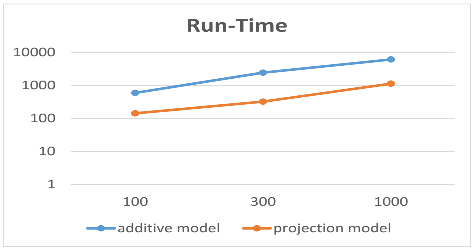


Figure 3: Run-time required by the *additive* and the *projection* models. (I/O time not counted)

(rand, static and nonstatic). The vertical axis shows the *CNN* classification accuracy in percentage. From the results we can see that different embeddings indeed have significant influences on the classification accuracies. Randomly generated word vectors performed remarkably the worst. Comparatively, our *projection* model reported evidently improved classification accuracy for both *static* and *non-static* schemes on all datasets. The improvement on the *Movie Review* dataset is especially evident, which further verifies the benefit brought from the projection of words based on prior knowledge.

4.4 Run-time

A practical concern about matrix factorization-based embedding techniques is about its computational complexity. To investigate this issue, we specially recorded the run-time of the two models when decomposing a $50,000 \times 50,000$ model with $100/300/1000$ dimensions on a mainstream workstation with 36 CPU cores.

The results are shown in Fig. 3. From the results, we can see that, without considering the input/output time, the computations of the *additive* model with 300 dimensions were finished within 30 minutes. Comparatively, by exploring the sparse structure of the co-occurrence matrix, the *projection* model runs much faster and is about five or more times quicker than the *additive* model, which is in fact quite comparable to the speed of *word2vec* and *GloVe* reported in the literature [Mikolov *et al.*, 2013; Pennington *et al.*, 2014].

5 Conclusion

Word embedding through machine learning has attracted much research attention recently. Following the line of matrix factorization-based approaches, our work explicitly models the semantic relationship among the words with the usage of linear constraints when decomposing lexical co-occurrence statistics, and results in two models that are simple, yet powerful to encode domain specific knowledge into word vectors. The effectiveness of the approach was empirically verified in various applications, as in the work of [Xu *et al.*, 2009].

More generally, a strong motivation of our work is the consideration of linear operations in word vector spaces. As pointed out in a seminal paper of object vectorization in the context of image processing [Beymer and Poggio, 1996], "Vectorization allows the creation of a flexible model for a class of objects as the linear combination of prototype images and their affine deformations". Our work just provides a trial along this line, trying to exhibit linear operations in a semantic word space, thereby making word vectors behave more *like* being in a mathematical vector space.

Despite the results obtained, a number of challenges remain for future study. One challenge is the computational complexity. Based on matrix factorization, the two proposed models operate well on moderately large applications. But when the word vocabulary grows even larger (e.g. $> 100,000$), the computational requirement often becomes prohibitive. To tackle the difficulty, more effective approximations, such as *ADMM* modeling, need to be investigated.

Another challenge seems more interesting. Besides the *additive* constraint and the *projection* constraint studied in this paper, are there any other constraints that are both practically useful and easy to solve? Unfortunately, our current answer is a bit pessimistic. We found the constraints other than the two forms always seem to incur nontrivial computational issues. Yet this remains an open problem.

Acknowledgments

This work is supported by Shenzhen Fundamental Research Fund under Grant No. JCYJ20170306141038939 and Grant No. KQJSCX20170728162302784, awarded to Wenye Li.

References

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Bengio, 2009] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2:1–127, 2009.

[Beymer and Poggio, 1996] David Beymer and Tomaso Poggio. Image representations for visual learning. *Science*, 272:1905–1909, 1996.

[Boyd *et al.*, 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

[Bullinaria and Levy, 2007] John Bullinaria and Joseph Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526, 2007.

[Faruqui *et al.*, 2014] Manaal Faruqui, Jesse Dodge, Sujay Jauhar, Chris Dyer, Eduard Hovy, and Noah Smith. Retrofitting word vectors to semantic lexicons. *arXiv:1411.4166*, 2014.

[Ganitkevitch *et al.*, 2013] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, 2013.

[Golub and Van Loan, 2012] Gene Golub and Charles Van Loan. *Matrix Computations*. John Hopkins University Press, 2012.

[Hofmann, 1999] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[Kiela *et al.*, 2015] Douwe Kiela, Felix Hill, and Stephen Clark. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, 2015.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv:1408.5882*, 2014.

[Landauer and Dumais, 1997] Thomas Landauer and Susan Dumais. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 1997.

[Lang, 1995] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

[Lebret and Collobert, 2013] Rémi Lebret and Ronan Collobert. Word emdeddings through Hellinger PCA. *arXiv:1312.5542*, 2013.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[Liu *et al.*, 2015] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1501–1511, 2015.

[Lund and Burgess, 1996] Kevin Lund and Curt Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28:203–208, 1996.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

[Miller, 1995] George Miller. Wordnet: a lexical database for English. *Communications of the ACM*, 38:39–41, 1995.

[Mrkšić *et al.*, 2016] Nikola Mrkšić, Diarmuid Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints. *arXiv:1603.00892*, 2016.

[Osborne *et al.*, 2015] Dominique Osborne, Shashi Narayan, and Shay Cohen. Encoding prior knowledge with eigenword embeddings. *arXiv:1509.01007*, 2015.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124, 2005.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[Rothe and Schütze, 2016] Sascha Rothe and Hinrich Schütze. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 512–517, 2016.

[Salton *et al.*, 1975] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:613–620, 1975.

[Tversky, 1977] Amos Tversky. Features of similarity. *Psychological Review*, 84:327, 1977.

[Wieting *et al.*, 2015] John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. From paraphrase database to compositional paraphrase model and back. *arXiv:1506.03487*, 2015.

[Xu *et al.*, 2009] Linli Xu, Wenye Li, and Dale Schuurmans. Fast normalized cut with linear constraints. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2866–2873, 2009.

[Yu and Dredze, 2014] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 545–550, 2014.

[Zeiler, 2012] Matthew Zeiler. Adadelta: an adaptive learning rate method. *arXiv:1212.5701*, 2012.