Sparse Lifting of Dense Vectors: A Unified Approach to Word and Sentence Representations

Senyue Hao[1] and Wenye Li[1,2](✉)

[1] The Chinese University of Hong Kong, Shenzhen, China
116010062@link.cuhk.edu.cn, wyli@cuhk.edu.cn
[2] Shenzhen Research Institute of Big Data, Shenzhen, China

Abstract. As the first step in automated natural language processing, representing words and sentences is of central importance and has attracted significant research attention. Despite the successful results that have been achieved in the recent distributional dense and sparse vector representations, such vectors face nontrivial challenge in both memory and computational requirement in practical applications. In this paper, we designed a novel representation model that projects dense vectors into a higher dimensional space and favors a highly sparse and binary representation of vectors, while trying to maintain pairwise inner products between original vectors as much as possible. Our model can be relaxed as a symmetric non-negative matrix factorization problem which admits a fast yet effective solution. In a series of empirical evaluations, the proposed model reported consistent improvement in both accuracy and running speed in downstream applications and exhibited high potential in practical applications.

Keywords: Language representation · Bag of words · Sparse lifting

1 Introduction

With tremendous theoretical and practical values, the study of natural language processing (NLP) techniques has attracted significant research attention in computer science and artificial intelligence community for many years. A key step in NLP is to represent language and text elements (words, sentences, etc.) in a form that can be processed by computers. For this task, the classical vector space model, which treats the elements as vectors of identifiers, has been routinely applied in tremendous applications for decades [26].

As an implementation of the vector space model, the *one-hot* method encodes a word by a sparse vector with exactly one element being non-zero. Accordingly,

This work was partially supported by Shenzhen Fundamental Research Fund (JCYJ20170306141038939, KQJSCX20170728162302784), awarded to Wenye Li.

to represent a sentence, the *bag-of-words* scheme is naturally applied on *one-hot* vectors of words. Simple as it is, the representation scheme has reported good empirical results [10]. Besides, people have designed various representation methods, which try to encode each word into a low dimensional continuous space either as a dense vector or as a sparse one, such as the work of [22,23,25,32]. With these methods, a sentence vector is typically built as an average, or a concatenation, of the vectors of all sentence words. All these methods have achieved quite successful results in a variety of applications.

Despite the successful results reported, all existing encoding methods face practical challenges. Motivated by both the success and the limitation of the existing methods, we designed a novel word representation approach, called the *lifting* representation. Our method projects dense word vectors to a moderately higher dimensional space while sparsifying and binarizing the projected vectors. Intuitively, comparing with *one-hot* word vectors, our encoding dimension is lower yet with generally more than one non-zero elements in each vector. Comparing with other word representation schemes, our encoding dimension is typically higher while most elements are zero. In this way, the proposed approach has the potential to encode the semantics of the words without bringing about much computational burden.

Based on the proposed *lifting* word representation method, representing a sentence is straightforward and natural. A sentence vector can be obtained in the way of *bag-of-words* which sums up the *lifting* vectors of all sentence words. In this way, the similarity and difference of any two sentences can also be easily obtained by calculating their vector similarity or Euclidean distance.

2 Related Work

2.1 Word and Sentence Representations

With *one-hot* representation model, each word is encoded as a binary vector [26]. Only one vector element is 1 and all other elements are 0. The length of the vector equals to the number of words in the vocabulary. The position of the 1 element in a vector actually gives an index of a specific word in the vocabulary. Accordingly, a sentence can be represented easily by the bag of its words, disregarding the grammar and word order. Simple as it is, this *bag-of-words* representation is commonly used in document classification and information retrieval, and has reported quite successful empirical results [10].

The distributional hypothesis [21] attracted much attention in designing word embedding methods. Starting from a summary statistics of how often a word co-occurs with its neighbor words in a large text corpus, it is possible to project the count-statistics down to a small and dense vector for each word [6,14], etc. More recent development focuses on the prediction of a word from its neighbors in terms of dense embedding vectors. Influential work includes the classical neural language model, [2], the *word2vec* (*Skip-gram* and *CBOW*) models [22], and the *GloVe* algorithm [25]. With such dense word vectors, a sentence is often processed as a concatenation of all sentence words [8,11]. However, measuring the similarity or difference between dense sentence representations is non-trivial. Specialized distance measures, such as the *WMD* distance [13], were designed, at the cost of significantly increased computation. Another approach is to represent a sentence by the weighted average of all its word vectors, and then use the average vectors to calculate the sentence similarities [1].

Recent research investigated the possibility of sparse representations. Some work starts from the co-occurrence statistics of words, including the NNSE method [23], the FOREST method [32], and the sparse CBOW model [28]. Some other work starts from the dense word vectors, including the work of [7] and [27]. In practice, such sparse word representations have been successfully applied and achieved quite good empirical results [29]. Similarly to the dense word vectors, to represent a sentence with these sparse vectors, people can resort to either the concatenation-based approach or the average-based approach.

2.2 Dimension Expansion

Biological studies revealed strong evidence of dimension expansion for pattern recognition. By simulating the fruit fly's odor detection procedure, a novel *fly* algorithm reported excellent performance in practical similarity retrieval applications [4]. Subsequent work along this line [17,18] designed a *sparse lifting* model for dimension expansion, which is more directly related to our work. The input vectors are lifted to sparse binary vectors in a higher-dimensional space with the objective of keeping the pairwise inner product between data points as much as possible. Then the feature values are replaced by their high energy concentration locations which are further encoded in the sparse binary representation. The model reported quite good results in a number of machine learning applications [20].

3 Model

3.1 Sparse Lifting of Dense Vectors

Our work leverages recent studies on dense word representations. It starts from a word representation matrix $X \in \mathcal{R}^{N\times d}$ from either the *word2vec* or the *GloVe* representation, with which each row gives a dense vector representation of a word in a vocabulary and has been zero-centered.

Motivated by the idea of sparse lifting, we seek a matrix $Z \in \{0,1\}^{N\times d'}$ which keeps the pairwise inner products between the row elements of X as much as possible while satisfying the requirement that $\sum Z_{ij} = Nk$ where k is the average number of non-zero elements in each row vector of Z.

The binary constraint on the desired matrix makes the problem hard to solve. To provide a feasible solution, we resort to the following model by relaxing the binary constraint and seeking a matrix $Y \in \mathcal{R}^{N\times d'}$ to minimize the difference between XX^T and YY^T in the Frobenius norm:

$$\min_{Y} \frac{1}{2} \left\| XX^T - YY^T \right\|_F^2, \tag{1}$$

subject to the element-wise constraint:

$$Y \geq 0. \tag{2}$$

This is a symmetric non-negative matrix factorization model [5]. In practice, the non-negativity constraint on each element of Y in Eq. (2) implicitly provides some level of sparsity on the solution Y^* [15]. When the solution Y^* is available, we can recover the desired matrix Z of *sparse-lifting* word vectors, or *lifting* vectors for short, trivially by setting $Z_{ij} = 1$ if Y^*_{ij} is among the topmost Nk elements of Y^*, and setting $Z_{ij} = 0$ otherwise.

The *lifting* word representation can be easily extended to represent a sentence, in a way that is much similar to that of *bag-of-words*. It leads to an attribute-value representation of sentences by representing each sentence roughly as a sum of the vectors of all its words [31]. With the *lifting* sentence representation, measuring the similarity or difference between two sentences becomes straightforward and trivial, which can be done just by calculating the inner product value or the Euclidean distance between the two sentence vectors.

3.2 Algorithm

The optimization model formulated in Eq. (1) subject to the constraint in Eq. (2) is a *symmetric non-negative matrix factorization* problem [5,15]. Different computational approaches are possible to tackle the problem [16]. We resort to a simple relaxation approach:

$$\min_{W,H\geq 0} \left\| XX^T - WH^T \right\|_F^2 + \alpha \left\| W - H \right\|_F^2 . \tag{3}$$

Here we seek two matrices W, H of size $N \times d'$, and $\alpha > 0$ is a scalar parameter for the trade-off between the approximation error and the difference of W and H. With the relaxation, we force the separation of the unknown matrix Y by associating it with two different matrices W and H. Given a sufficiently large value of α, the matrix difference dominates the objective value and the solutions of W and H will tend to be close enough so that the word vectors will not be affected whether W or H are used as the result of Y.

The key to solving the problem in Eq. (3) is by solving the following two *non-negative least squares* (NLS) sub-problems [12]:

$$\min_{W\geq 0} \left\| \begin{bmatrix} H \\ \sqrt{\alpha} I_{d'} \end{bmatrix} W^T - \begin{bmatrix} XX^T \\ \sqrt{\alpha} H^T \end{bmatrix} \right\|_F^2 , \text{ and } \min_{H\geq 0} \left\| \begin{bmatrix} W \\ \sqrt{\alpha} I_{d'} \end{bmatrix} H^T - \begin{bmatrix} XX^T \\ \sqrt{\alpha} W^T \end{bmatrix} \right\|_F^2 ,$$

where $I_{d'}$ is the $d' \times d'$ identity matrix. Solving the sub-problems in the two equations in an iterative way will lead to a stationary point solution, as long as an optimal solution is returned for every NLS sub-problem encountered.

4 Evaluations

4.1 General Settings

To evaluate the performance of the proposed word representation method, we carried out a series of evaluations. Our *lifting* vectors were generated from the dense word vectors released by the authors of *word2vec* and *GloVe*.

- *CBOW*: 300-dimensional word2vec vectors[1].
- *GloVe*: 300-dimensional word vectors[2].

We trained the *lifting* vectors with the $50,000$ most frequent words out of the *CBOW* and *GloVe* word vectors respectively. The expanded dimension of the trained vectors were set to $d' = 1,000$. After training, on average 20 elements of each vector were set non-zero, i.e. the hash length $k = 20$. The results reported in this paper are just based on this setting. Besides, we have also varied different combinations of the parameters within the range of $d' = 1,000/2,000/5,000$ and $k = 10/20$. The evaluation results are quite similar and are therefore omitted.

In the evaluation, six benchmark datasets were used.

- CUSTREV: A set of $3,774$ customers' positive or negative reviews [9].
- MPQA: A set of $10,606$ articles with two opinion polarities [30].
- RT-POLARITY: A set of $10,662$ movies' positive or negative reviews [24].
- STSA-binary: An extension of RT-POLARITY with $8,741$ sentences [24].
- TREC: A set of $5,692$ TREC questions with six question types [19].

In addition to the *one-hot*, *CBOW* and *GloVe* representations, our *lifting* vectors were compared with the following representations:

- *FOREST*: 52-dimensional word vectors[3].
- *NNSE*: 300-dimensional word vectors[4].
- *OVERCOMPLETE*: $1,000$-dimensional sparse overcomplete word vectors[5].

For *one-hot* and *lifting* methods, each sentence vector is represented as *bag-of-words*. For other word vectors, we represent each sentence as an average of fifty word vectors with zero-padding [1]. A concatenation-based representation which treats each sentence vector as a concatenation of *CBOW* vectors was also included in the experiment, combined with the Word Mover Distance (*WMD*) to measure sentence similarities [13].

[1] https://code.google.com/archive/p/word2vec/.
[2] https://nlp.stanford.edu/projects/glove/.
[3] http://www.cs.cmu.edu/~ark/dyogatam/wordvecs/.
[4] http://www.cs.cmu.edu/~bmurphy/NNSE/.
[5] https://github.com/mfaruqui/sparse-coding/.

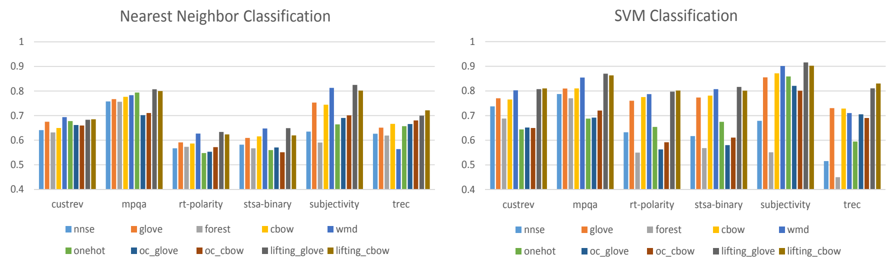

Fig. 1. Sentiment classification accuracies.

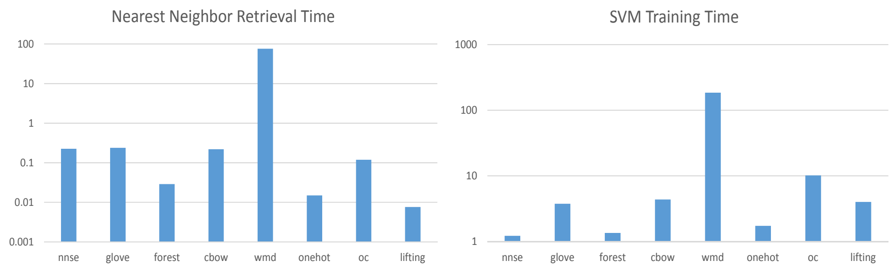

Fig. 2. Running time of NN classification and SVM classification on CUSTREV dataset.

4.2 Sentiment Analysis

We investigated the performance of the sparse-lifting vectors on sentiment analysis applications on the six benchmark datasets. On each dataset, we trained a classifier with different representations of sentences by the *nearest neighbors* (NN) algorithm and the *support vector machines* (SVM) algorithm respectively. For the SVM classification, we used the Gaussian radial basis function kernel, with the default kernel and cost parameters [3].

The comparison was made against with a number of representation schemes, including *NNSE*, *GloVe*, *FOREST*, *CBOW*, and *OVERCOMPLETE* (denoted by *oc_cbow* and *oc_glove* in the figure) representations with Euclidean distance measure. The classification accuracies are depicted in Fig. 1. Each result in the table is an average accuracy of 10-fold cross validations.

The *one-hot/bag-of-words* representation reported acceptable results when working with the SVM algorithm; but its results on NN classification were not as good. This result is consistent with the previous studies [10]. The *CBOW* and concatenation-based sentence vectors, when being combined with *WMD*, reported quite good accuracies. The performances of the averaged sentence representation with *Glove/CBOW/NNSE/FOREST* vectors, however, seemed not satisfactory, only slightly better than a random guess on some NN classification tasks.

Our proposed representations (denoted by *lifting_glove* and *lifting_word2vec* respectively) brought improved results in the experiments. Compared with *CBOW* and concatenation-based sentence vectors with the *WMD* measure, our proposed representations reported comparable (if not better) classification accuracies on most datasets; while on the TREC dataset which has six categories, both of our representations reported much better results.

4.3 Running Speed

We recorded the query time of the NN classifier with 90% of samples used in training and the rest 10% used in testing on CUSTREV dataset, which needs to compute the distances between each pair of testing and training samples. The experiment was performed in a computer server with 44 CPU cores. From Fig. 2, we can see that with highly sparse representations, the query time of the *lifting* representation and of the *one-hot/BOW* representation reported significantly superior results over other methods.

We recorded the training time of an SVM classifier with the libSVM package [3] with 90% of samples on CUSTREV dataset used as the training set. With a Gaussian kernel and the default setting of parameters, it took less than 10 seconds to train an SVM classifier with the *lifting* vectors and *bag-of-words* sentence representation. Similar results were found on SVM training with the *Glove*, *CBOW* and *OVERCOMPLETE* vectors and averaged sentence representations. All these results are tens of times faster than training with the *CBOW* vectors and concatenation-based sentence representation in *WMD* distance.

5 Conclusion

Our work designed a novel sparse lifting word representation method which projects given dense word vectors into a higher dimensional space while ensuring the sparsity and binarization of the projected vectors. Comparing with existing popular word and sentence vector representations, our proposed sparse-lifting representation has been shown to be an appropriate representation for distance-based learning tasks and has reported significantly improved results in sentiment analysis tasks. The improvement provides us with high confidence to apply the method in wider practical applications.

References

1. Arora, S., Liang, Y., Ma, T.: A simple but tough-to-beat baseline for sentence embeddings. In: ICLR 2017 (2017)
2. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**, 1137–1155 (2003)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 27 (2011)

4. Dasgupta, S., Stevens, C., Navlakha, S.: A neural algorithm for a fundamental computing problem. Science **358**(6364), 793–796 (2017)
5. Ding, C., He, X., Simon, H.: On the equivalence of nonnegative matrix factorization and spectral clustering. In: SIAM SDM 2005, pp. 606–610 (2005)
6. Dumais, S.: Latent semantic analysis. Ann. Rev. Inf. Sci. Technol. **38**(1), 188–230 (2004)
7. Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., Smith, N.: Sparse overcomplete word vector representations. arXiv:1506.02004 (2015)
8. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.: Convolutional sequence to sequence learning. arXiv:1705.03122 (2017)
9. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: ACM SIGKDD 2004, pp. 168–177 (2004)
10. Joachims, T.: Text categorization with support vector machines: learning with many relevant features. In: ECML 1998, pp. 137–142 (1998)
11. Kim, Y.: Convolutional neural networks for sentence classification. arXiv:1408.5882 (2014)
12. Kuang, D., Yun, S., Park, H.: Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering. J. Global Optim. **62**(3), 545–574 (2015)
13. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: ICML 2015, pp. 957–966 (2015)
14. Lebret, R., Collobert, R.: Word emdeddings through hellinger PCA. arXiv:1312.5542 (2013)
15. Lee, D., Seung, H.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788 (1999)
16. Lee, D., Seung, H.: Algorithms for non-negative matrix factorization. In: NIPS 2001, pp. 556–562 (2001)
17. Li, W.: Modeling winner-take-all competition in sparse binary projections. In: ECML-PKDD 2020 (2020)
18. Li, W., Mao, J., Zhang, Y., Cui, S.: Fast similarity search via optimal sparse lifting. In: NeurIPS 2018, pp. 176–184 (2018)
19. Li, X., Roth, D.: Learning question classifiers. In: COLING 2002, pp. 1–7 (2002)
20. Ma, C., Gu, C., Li, W., Cui, S.: Large-scale image retrieval with sparse binary projections. In: ACM SIGIR 2020, pp. 1817–1820 (2020)
21. McDonald, S., Ramscar, M.: Testing the distributional hypothesis: the influence of context on judgements of semantic similarity. In: CogSci 2001 (2001)
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)
23. Murphy, B., Talukdar, P., Mitchell, T.: Learning effective and interpretable semantic models using non-negative sparse embedding. In: COLING 2012, pp. 1933–1950 (2012)
24. Pang, B., Lee, L.: Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In: ACL 2005, pp. 115–124 (2005)
25. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: EMNLP 2014, pp. 1532–1543 (2014)
26. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill Inc., New York (1986)
27. Subramanian, A., Pruthi, D., Jhamtani, H., Berg-Kirkpatrick, T., Hovy, E.: Spine: Sparse interpretable neural embeddings. In: AAAI 2018, pp. 4921–4928 (2018)
28. Sun, F., Guo, J., Lan, Y., Xu, J., Cheng, X.: Sparse word embeddings using l1 regularized online learning. In: IJCAI 2016, pp. 2915–2921 (2016)

29. Turney, P.: Leveraging term banks for answering complex questions: a case for sparse vectors. arXiv:1704.03543 (2017)
30. Wiebe, J., Wilson, T., Cardie, C.: Annotating expressions of opinions and emotions in language. Lang. Resour. Eval. **39**(2–3), 165–210 (2005)
31. Yang, J., Jiang, Y., Hauptmann, A., Ngo, C.: Evaluating bag-of-visual-words representations in scene classification. In: ACM SIGMM MIR 2007, pp. 197–206 (2007)
32. Yogatama, D., Faruqui, M., Dyer, C., Smith, N.: Learning word representations with hierarchical sparse coding. In: ICML 2015, pp. 87–96 (2015)