WORD EMBEDDING INTERPRETATION USING CO-CLUSTERING

Zainab Albujasim[1], Diana Inkpen[2] and Yuhong Guo[3]

[1]School of Computer Science, Carleton University, ON, Canada
[2]School of Electrical Eng. and Computer Science,
University of Ottawa, ON, Canada
[3]School of Computer Science, Carleton University, ON, Canada

ABSTRACT

Word embedding is the foundation of modern language processing (NLP). In the last few decades, word representation has evolved remarkably resulting in an impressive performance in NLP downstream applications. Yet, word embedding's interpretability remains a challenge. In this paper, We propose a simple technique to interpret word embedding. Our method is based on post-processing technique to improve the quality of word embedding and reveal the hidden structure in these embeddings. We deploy Co-clustering method to reveal the hidden structure of word embedding and detect sub-matrices between word meaning and specific dimensions. Empirical evaluation on several benchmarks shows that our method achieves competitive results compared to original word embedding.

KEYWORDS

Word Embedding, Interpretation, Quantization, Post-processing.

1. INTRODUCTION

Over the last few years, pre-trained word embeddings have become the backbone of Natural Language Processing (NLP) applications. Several word embedding models have been proposed, such as Word2ve [1], GloVe [2], and FastText [3]. Word embedding is very interesting concept because it enables us to reason about the semantics and syntax of a text. It allows us to group similar words together and find pattern in words or texts. These word vectors are a dense representation built on unstructured text; each word vector encodes salient information about a word, such as the semantic and syntactic features of the word. However, these word vectors are very hard to interpret since they consist of continuous values, not discrete values.

Interpreting word embeddings has been investigated by many researchers, in an effort to understand the meaning of the dimensions of word embeddings.

Hanselowski and Gurevych [4] introduced a framework for decomposing word embeddings into smaller meaningful units, which we call sub-vectors. Zobnin [5] employed several rotation algorithms to improve the interpretability of word embeddings.

Fyshe, Talukdar, Murphy, and Mitchell [6] proposed joint non-negative sparse embedding (JNNSE), which incorporates both text-based similarity information for words and brain activity-based similarity information in order to improve the interpretability of word embedding. However, collecting neuroimaging data from multiple subjects remains a challenging task that takes a great deal of time and effort. Arora, Li, Liang, Ma, and Risteski [7] and Faruqui, Tsvetkov, Yo-gatama, Dyer, and Smith [8] proposed sparse coding techniques on dense word embeddings rather than learning interpretable word representations directly from co-occurrence matrices. Senel, Utlu, Yücesoy, Koc, and Cukur [9] used a statistical method to reveal the underlying structure of dense word embeddings. However, these methods do not interpret the meaning of specific values in the embedding matrix.

In this paper, we propose a simple technique to interpret word embedding. Our proposal is based on the analysis of positive and negative values in word embedding. As a way to remove complexity in word vector representation, we propose a simple quantization technique mapping positive values to 1 and negative values to 0. Next, we apply a post-processing technique introduced by Mu, Bhat, and Viswanath [10] to further improve the quality of binary representation of the word embeddings.

Our findings indicate that the sign of a word vector values carries important information about the meaning of a word. We deploy Co-clustering techniques to reveal the hidden structure of word embedding. Moreover, mapping word embeddings into a binary representation is an efficient way to store word embeddings. Word embeddings are huge files requiring a large capacity to store and time to transfer. The complexity of word embeddings is attained at the cost of the significant increase in computing and storage requirements. As a result, NLP applications on limited hardware are challenging. It is possible for our method to have further advantages and to be used as a method to compress word embeddings as well.

2. METHOD

We propose an unsupervised post-processing quantization technique for word embeddings that does not require any training. It can be utilized to interpret and reduce the complexity of the word vectors while keeping the salient information regarding the basic structure of word vectors. For quantization, we use a sign function to map the continuous values of the word embeddings to binary values, zero for the negative sign, and 1 for the positive sign.

We also investigate the effectiveness of word embedding quantization and suggest a method for improving the quantization of the word embedding by using a post-processing algorithm Mu, Bhat, and Viswanath [10].

The post-processing algorithm (PPA) is built based on the following observations: first, word vectors have a large common mean vector, and by removing the common means vector, by subtracting the mean from the embedding matrix, the representation becomes anisotropic (which is a property of a material which allows it to change or assume different directions). Second: most of the energy of the word embedding is constrained within a low dimensional subspace, such as 7 out of 300.

As a first step in obtaining the proposed post-processing embedding, we apply Principal Component Analysis Algorithm (PCA) on the original word embedding to extract the PCA components. Next, we apply the quantization technique to the original word embedding and normalize the word vectors using mean normalization. Finally, we remove the 7 top PCA components from the new word embedding. The number of top components is selected based on the original PPA algorithm.

To interpret the new post-processed word embedding, we deploy a co-clustering algorithm to find the relation between(sub-matrices) a semantic meaning of a word and a specific set of embedding dimensions. To analyze and understand the word vectors and their values, we examine the word embedding in terms of positive and negative signs. Based on a statistical analysis of positive and negative signs, it is apparent that most common English words have approximately 40\% to 60\% positive or negative values; these signs alternate in different dimensions depending on the words' contextual similarity. Also, based on the examination of the pre-trained models, we observe that numerical values in the word vectors of similar words fall within a small range to maximize similarity score. Therefore, applying quantization will not change a word vector's structure; it may reduce the accuracy, but the essential information is still encoded.

2.1. Quantization

Quantization converts an infinite set of continuous values to a small set of discrete values. Quantization has been employed to reduce the size and reduce storage and transfer costs. It reduces the very large size of the word embeddings and this is a benefit for many NLP applications with limited resources. We use a simple sign function to map positive values to 1 and negative values to zero. The function is defined as follows:

$$sgn(x) = \begin{cases} +1, & x > 0 \\ 0, & x < 0 \end{cases} \qquad (2.1)$$

2.2. The Proposed Post-Processing Algorithm

The algorithm was introduced by [10]. To improve the binary word embeddings, we employ post-processing algorithms (PPA) after quantization. %The new embedding is used to improve the accuracy of quantized word embedding.

Here is a more formal description of our method:

Algorthim1: Word embedding post-processing

```
Input: The Word Vector Matrix A and d: number of PCA Components of the original word vectors
Output:
1. Convert the word embedding to binary representation using the sign function (Quantization step)
2. Center the binary word vectors by removing the mean of the embedding matrix(Normalization step)
3. Process the converted word embedding by removing d components
4. $A = A - A'$
5. $v = v - \sum_{i=1}^{d}(u_i^T.v)u_i$ , $u_i$ is the $PCA(A)$ where $i = 1,2,..d$
6. end
```

2.3. Co-clustering Algorithm

The basic idea of co-clustering techniques Hochreiter, Bodenhofer, Heusel, Mayr, Mitterecker, Kasim, Khamiakova, Van Sanden, Lin, Talloen, et al. [11] is to organize the data to reveal its hidden structure. Co-clustering or Bi-clustering algorithms were adopted to solve problems in various domains such as text mining, bio-informatics, and marketing. Co-clustering techniques attempt to find bi-clusters with higher values than those in the corresponding rows and columns. There are different types of co-clustering algorithms, based on how they define the bi-clusters. For example, some algorithms cluster based on constant rows or constant columns, high or low values, sub-matrices with low variances, or correlated rows and columns. In this work, we employ the Spectral Co-clustering Algorithm Dhillon [12].

The reason to select the spectral co-clustering algorithm is to extract specific dimensions related to specific semantic meaning by assigning each row (word) and each column (dimension) to exactly one cluster. The algorithm tries to arrange sub-clusters as a diagonal structure.

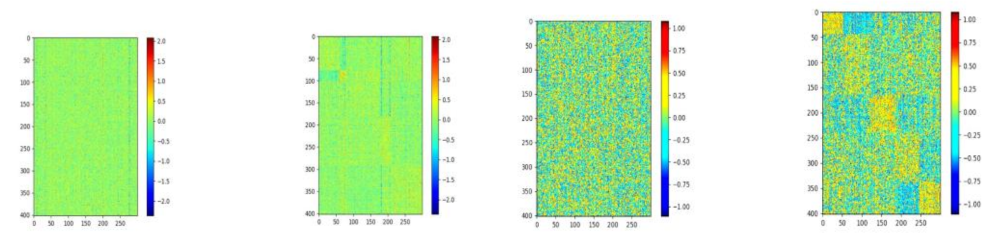
Spectral co-clustering was introduced originally to cluster documents and words simultaneously. The spectral co-clustering algorithm represents the data matrix as a bipartite graph where rows and columns denote the two sets of graph nodes in which each entry represents an edge between rows and columns, and tries to find the best dense sub-graphs between rows and columns of the data matrix.

3. Experimental Results

In this section, we evaluate the proposed technique to interpret the word embedding, and then we evaluate our quantization of the word embeddings and how effective it is in several standard NLP tasks such as semantic similarity, classification, and clustering tasks.

3.1. Co-clustering Algorithm

In this experiment, we employ the Spectral Co-clustering algorithm to interpret and reveal the hidden structure of word embeddings. We utilize the Concept Categorization BLISS [13] with 200 words divided into 21 categories, and the Almuhareb-Poesio (AP) [14] which contains 402 concepts with 21 categories. For the purpose of the experiment, we combine some of the clusters in datasets, such as different types of animals, plants, and tools into more general clusters since these categories have very strong similar semantics. For example in the Bliss dataset, there are 5 types of animals (water-animal, bird, ground-mammal, insect, and amphibian). These categories are grouped into one cluster of animals. Likewise, we grouped different kinds of plants and tools into one cluster. We reduce the number of categories from 21 to 5 for the AP dataset and from 27 to 5 clusters for Bliss dataset. Additionally, our experiment shows that 5 clusters are the ideal number to get distinctive diagonal blocks for co-clustering (rows and columns) of word embedding for the two datasets. The best performance was selected after running spectral co-clustering 10 times. We use purity measures to evaluate the quality of the clustering. We observe that GloVe-binary-ppa and gives higher purity compared to the original GloVe model. Table 1 shows the performance of our approach with the three main embeddings models Glove, Word2vec and FastText.



(a) AP dataset

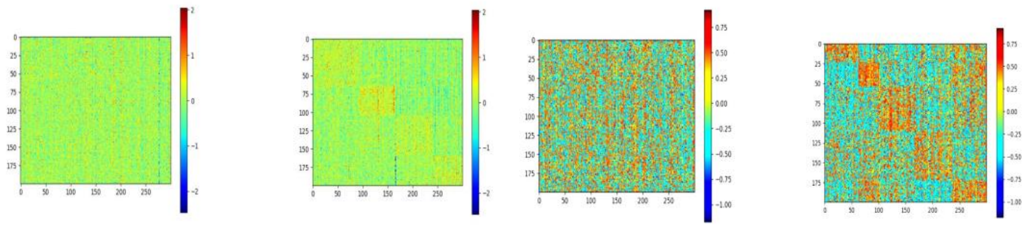
before Co-clustering

(b) AP dataset after Co-clustering (GloVe Original embedding

(c) AP dataset before Co-clustering

(d)AP dataset after Co-clustering (Glove-binary-ppa embedding Co-clustering

Figure 1. Co-clustering results for AP dataset using the original GloVe and Glove and GloVe-binary-ppa



(a) Bliss dataset

before Co-lustering

(b) Bliss dataset after Co-clustering (GloVe original embedding)

(c) Bliss dataset before Co-clustering (GloVe-binary-ppa embedding)

d) Bliss dataset after Co-clustering (GloVe-binary-ppa embedding

Figure 2. Co-clustering results for Bliss dataset using the original GloVe and Glove and GloVe-binary-ppa

Table 1. Purity measure of the Spectral co-clustering technique for Ap and Bliss dataset with three main embedding models GloVe, Word2vec and FastText

Model	**AP**	**Bliss**
GloVe-original	75.34	66.80
GloVe-binary	76.65	68.09
Glove-binary-ppa	**76.95**	**68.95**
Word2vec-original	40.28	64.14
Word2vec-binary	**41.91**	**71.80**
Word2vec-binary-ppa	39.95	63.30
FastText-original	**79.01**	76.54
FastText-binary	78.40	76.14
FastText-binary-ppa	75.89	**81.15**

We use the GloVe model for visualization of Co-clustering results as it produces more stable embeddings across different datasets Figures 1 and 2 show a comparison between the spectral coclustering result on the original GloVe model and on our Binary-GloVe for the AP dataset and bliss datasets. We observe that the GloVe-binary-ppa model produces better diagonal blocks, and dense sub-matrices can be detected easily, while with the original GloVe is it hard to visualize or detect any diagonal blocks in both datasets. Each sub-matrices and blocks represent specific semantic meanings and their strongly associated dimensions.

Table 2. Performance of binary and binary-ppa techniques with three word embeddings models across different datasets on semantic similarity task.

Model	**MC**	**MTurk-287**	**WS-353-SIM**	**STAN-FORD**	**VERB-143**	**YP-130**	**MEN-TR**	**SIM-LEX**	**RG-65**
GloVe-300D -original	70.26	**63.32**	60.54	41.18	30.51	56.13	73.75	37.05	76.62
GloVe-binary	72.39	61.10	65.22	39.39	36.89	61.79	72.92	39.81	74.59
GloVe-binary-ppa	**75.55**	60.85	**66.92**	**42.36**	**36.72**	**60.73**	**72.46**	**40.59**	**76.29**
FastText-300-original	83.63	70.50	81.02	52.31	46.27	51.87	79.06	45.00	84.51
FastText_binary	68.90	59.66	72.67	53.04	48.96	**63.22**	74.57	**44.11**	69.47
FastText_binary_ppa	71.46	63.46	75.13	**52.96**	**48.95**	62.37	76.26	41.56	72.13
Word2vec-300D-orig	**78.80**	**68.40**	**77.71**	**53.42**	**49.73**	55.90	**77.08**	44.20	**74.98**
Word2vec-binary	71.39	57.84	68.51	45.23	29.19	**56.29**	70.04	41.76	64.89
Word2vec-binary-ppa	76.04	59.42	70.85	47.83	34.82	54.95	73.59	**43.29**	68.12

We use the GloVe model for visualization of Co-clustering results as it produces more stable embeddings across different datasets Figures 1 and 2 show a comparison between the spectral coclustering result on the original GloVe model and on our Binary-GloVe for the AP dataset and bliss datasets. We observe that the GloVe-binary-ppa model produces better diagonal blocks, and dense sub-matrices can be detected easily, while with the original GloVe is it hard to visualize or detect any diagonal blocks in both datasets. Each sub-matrices and blocks represent specific semantic meanings and their strongly associated dimensions

Table 3. Classification results (accuracy on the test data) on several classification datasets with original embeddings and the quantized embedding (binary representation), and the proposed method with quantized representation with Post-Processing Algorithm (ppa). We highlight the result within 1% of original embedding accuracy.

Model	**MR**	**CR**	**MPQA**	**SUBJ**	**Series-STS-B**	**SST-FG**	**TREC**	**SICK-E**	**MRPC**
GloVe-300D-orginal	75.22	75.82	86.35	91.04	78.20	40.77	66.60	77.19	72.46
GloVe-binary	74.81	77.09	84.92	90.36	76.22	41.49	84.80	77.21	71.94
GloVe-binary-ppa	**75.42**	**77.14**	**86.31**	**90.87**	76.88	**41.99**	**66.60**	**76.84**	**72.23**
Word2vec-300D-orig.	76.97	79.34	**88.29**	90.46	**81.11**	40.79	82.60	77.92	**72.35**
Word2vec-binary	67.95	68.19	86.88	89.00	78.31	40.00	81.20	76.5	69.39

								0	
Word2vec-binary-ppa	**75.50**	**76.00**	86.61	**89.25**	77.32	42.62	**81.80**	**78.22**	70.72
FastText-300D-orgig.	**77.65**	**80.48**	87.78	92.10	**82.15**	**44.30**	**84.40**	79.06	**74.38**
FastText-binary	66.97	71.68	84.31	88.16	77.76	39.41	77.20	76.80	71.48
FastText-binary-ppa	74.52	76.50	**85.51**	**90.41**	76.61	40.45	82.60	**77.76**	71.94

3.2. Word Similarity Benchmarks

We use the standard word similarity benchmarks described in [15]. The datasets cover a wide range of similarity tasks in various domains. They have been widely used to evaluate word similarity measures. The performance of semantic similarity of 9 benchmarks shows that the GloVe model quantization effectively produces better similarities than the original GloVe model in binary format and binary-ppa model. The GloVe-binary-ppa model improves the accuracy across 8 out of 9 data sets. This shows that the quantized GLoVe model is an effective technique for semantic similarity. The quantized Word2vec model brings improvements only on 3 datasets out of the 9 and similarly with FastText. Based on our experiment result we can conclude our quantization techniques are more effective with count-based embeddings such as GLoVe than conventional embedding techniques such as Word2vec Model. Table 2 shows the results for semantic similarity. For GloVe, our proposed quantization method works very well interim of semantic similarity task, while for Word2vec and FastText, it provides reasonable results.

3.3. Classification Similarity Benchmarks

To perform the classification task, we utilized the SentEval toolkit [16]. SentEval is a standard evaluation toolkit for classification and semantic analysis. It covers a wide range of domains. In SentEval, the sentences are represented by the mean of their word embeddings. Logistic regression and multilayer perceptron are used as the main classifiers. Table 3 clearly shows that using quantized word embeddings, the performance is comparatively good compare to original models for word2vec and FastText, and the performance is better than the original model in the case GloVe model.

4. CONCLUSION

This paper presented a method based on spectral co-clustering for word embedding interpretability and for understanding the relationship between particular semantic meaning and dimensions. We also introduced a post-processing quantization technique for word embeddings, a simple and effective method in terms of cost and interpretability. It allows to reduce the storage needs and reduces the complexity of the word vectors. Our result show that this quantization techniques work well for semantic similarity, concept categorization and classification for Count-based embeddings models such as GloVe. Our work is a step forward towards understanding and interpreting the dimensions of word embedding models. We aim to design more interpretable embedding models in the future.

REFERENCES

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed representations of words and phrases and their compositionality”. In: Advances in neural information processing systems. 2013, pp. 3111–3119.

[2] J. Pennington, R. Socher, and C. D. Manning. “Glove: Global vectors for word representation”. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014, pp. 1532–1543.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. “Enriching word vectors with subwordinformation”.In: Transactions of the Association for Computational Linguistics 5 (2017), pp. 135–146.

[4] A. Hanselowski and I. Gurevych. “Analyzing Structures in the Semantic Vector Space: A Framework for Decomposing Word Embeddings”. In: arXiv preprint arXiv:1912.10434 (2019).

[5] A. Zobnin. “Rotations and interpretability of word embeddings: The case of the Russian language”. InInternational Conference on Analysis of Images, Social Networks and Texts. Springer. 2017, pp. 116–128.

[6] A. Fyshe, P. P. Talukdar, B. Murphy, and T. M. Mitchell. “Interpretable semantic vectors from a joint model of brain-and text-based meaning”. In: Proceedings of the conference. Association for Computational Linguistics. Meeting. Vol. 2014. NIH Public Access. 2014, p. 489.

[7] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. “Linear algebraic structure of word senses, with applications to polysemy”. In: Transactions of the Association for Computational Linguistics 6 (2018), pp. 483–495.

[8] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. Smith. “Sparse overcomplete word vector representations”. In: arXiv preprint arXiv:1506.02004 (2015).

[9] L. K. ̧Senel, I. Utlu, V. Yücesoy, A. Koc, and T. Cukur. “Semantic structure and interpretability of word embeddings”. In: IEEE/ACM Transactions on Audio, Speech, and Language Processing 26.10 (2018), pp. 1769–1779.

[10] J. Mu, S. Bhat, and P. Viswanath. “All-but-the-top: Simple and effective postprocessing for word representations”. In: arXiv preprint arXiv:1702.01417 (2017).

[11] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, T. Khamiakova, S. Van Sanden, D. Lin, W. Talloen, et al. “FABIA: factor analysis for bicluster acquisition”. In: Bioinformatics 26.12 (2010), pp. 1520–1527.

[12] I. S. Dhillon. “Co-clustering documents and words using bipartite spectral graph partitioning”. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. 2001, pp. 269–274.

[13] M. Baroni and A. Lenci. “How we BLESSed distributional semantic evaluation”. In: Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics. 2011, pp. 1–10.

[14] M. Poesio and A. Almuhareb. “Identifying concept attributes using a classifier”. In: Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition. 2005, pp. 18–27.

[15] M. Faruqui and C. Dyer. “Improving Vector Space Word Representations Using Multilingual Correlation”. In: Proceedings of EACL 2014. Apr. 2014, pp. 462–471.

[16] A. Conneau and D. Kiela. “Senteval: An evaluation toolkit for universal sentence representations”. In:arXiv preprint arXiv:1803.05449 (2018).

AUTHORS

Zainab Abujasim is a PhD. Candidate in the School of Computer Science at Carleton University. She holds a master degree in Computer Science from the University of Kentucky. Her primary research is in natural language processing with focus on word embeddings.

Diana Inkpen is a Professor at the University of Ottawa, in the School of Electrical Engineering and Computer Science. She obtained her Ph.D. from the University of Toronto, Department of Computer Science. She has a M.Sc. and B.Eng. degree in Computer Science and Engineering from the Technical University of Cluj-Napoca, Romania. Her research is in applications of Natural Language Processing and Text Mining.

Yuhong Guo is a Professor in the School of Computer Science at Carleton University. She holds a Canada Research Chair in Artificial Intelligence (CIFAR AI). Her primary area of research is machine learning, with an emphasis on learning useful data representations and learning accurate classification models under various circumstances. Her application areas are in natural language processing, computer vision and medical science.