Effects of Positivization on the Paragraph Vector Model

Aydın Gerek, Mehmet Can Yüney, Erencan Erkaya, Murat Can Ganiz
Department of Computer Engineering
Marmara University
Istanbul, Turkey
aydin.gerek@marmara.edu.tr, {mehmetyuney, erenerkaya}@marun.edu.tr, murat.ganiz@marmara.edu.tr

***Abstract*—Natural language processing (NLP) is an important field of Artificial Intelligence. One of the fundamental problems in NLP is to create vector (distributed) representations of words so that vectors of words that have similar meaning lie closer in space. One of the most popular algorithms for creating these representations are word embedding models such as word2vec and fastText. Similarly the paragraph vector model (doc2vec) is used to create distributed representations of documents while simultaneously creating distributed representations for the words in these documents. These models create a dense, and low dimensional (usually in the low hundreds) vector representations which may include negative values. In this study we focus on these negative values and introduce a family of regularization methods in which document, word and/or context vectors of the paragraph vector model are forced to have only positive components. We measure its effects on several tasks; text classification, semantic similarity, and analogy tasks. Although positivization greatly increases the sparsity of the word embeddings, and should be expected to result in a loss of information, our results show that there is almost no reduction in the performance of the regularized embeddings in these tasks. We also observe an increase in the classification accuracy in one case. We foresee that these approaches can be beneficial in machine learning systems which require non-negative vectors.**

***Index Terms*—artificial intelligence, natural language processing, word embeddings, regularization, text classification, analogy, semantic similarity.**

I. Introduction

A. Background

Most machine learning models require training data to be in the form of vectors. Therefore to learn from textual data we need ways to vectorize words, sentences or whole documents. This is one of the fundamental subjects of natural language processing: distributed representations. Note that we will refer to any tasks undertaken using distributed representations as downstream NLP tasks, meaning to say that they come after the vectorization of text, and their success depends on the quality of representations produced by the vectorization technique. One such task is text classification, in which a supervised or semi-supervised machine learning model is trained to classify textual documents into predetermined categories. Sentiment classification, e.g. classifying customer comments as positive or negative is an example of text classification. Another example is the classification of news texts into categories such as politics, sports, and economics. Another downstream NLP task is semantic similarity in which distributed representations of words are used to score similarity between them. One final example is that of the analogy task, in which questions of the form "Paris is to France as Ankara is to what?" are answered. These analogy questions are usually encoded as triplets of words and require a single word as an answer (Turkey is the answer in this case).

Older distributed representation techniques, called vector space models, represent text as sparse, high dimensional vectors. Best known among these are are TF-IDF [1] and PMI [2].

TF-IDF matrices are word by document matrices that assign the importance of each word to a given document. This is calculated by multiplying the term frequency (TF) of the word in the document (how many times it occurs in the document), with its inverse document frequency: a measure of how infrequently the word occurs in other documents. The formula is $tfidf(w,d) = tf(w,d)\ln\frac{|D|}{|D(w)|}$ where $tf(w,d)$ stands for the term frequency of word w in document d, $|D|$ stands for the total number of documents, and $|D(w)|$ stands for the number of documents in which the word w occurs.

On the other hand positive mutual information (PMI) matrices are word by word matrices which describe the association between pairs of words based on their co-occurance probabilities. By co-occurance we mean the number of times two words occur within a fixed distance (called the window size) of each other in the text. If we denote as f_{wc} the number of times in which the words w and c co-occur, then we can compute their PMI as follows:

$$P(w,c) = \frac{f_{wc}}{\sum_{i,j} f_{ij}}$$
$$P(w) = \frac{\sum_j f_{wj}}{\sum_{i,j} f_{ij}}$$
$$P(c) = \frac{\sum_i f_{ic}}{\sum_{i,j} f_{ij}}$$
$$PMI(w,c) = \ln\frac{P(w,c)}{P(w)P(c)}$$

One of the differences between them is that while the components of a TF-IDF vector are all non-negative, PMI vectors can have negative components. A variation on PMI is

PPMI (positive PMI) [3] in which PMI vectors are positivized. By positivization we mean that the negative components of the vector are replaced with zeroes.

$$pos(\boldsymbol{v})_i = max(0, v_i)$$

It was shown [4] that PPMI performs better than PMI for many NLP tasks such as semantic similarity or syntactic categorization.

Currently the most favored distributed representations are the embedding models popularized by word2vec [5]. Embeddings models are low dimensional dense representations often originating from neural language models [6]. As with PMI, embedding models produce vectors whose components might be negative.

Among the various word2vec architectures the most successful one is the skip-gram negative sampling architecture (SGNS) [7, 8]. Like many other distributed representations SGNS is based upon the distributional hypothesis [9], which states that words that occur in same contexts have similar meanings. In this sense, when focusing on a specific word in a corpus the words near it are considered its context. Thus any two words which occur within a specified distance of each other form a positive word-context pair. This specified distance is called the window size and is a hyper-parameter of the model. A negative word-context pair, on the other hand, is a randomly picked pair of words. The probability of a two randomly selected words appearing close to each other on the corpus is low, therefore negative pairs represent pairs of words which do not co-occur.

SNGS trains two sets of embeddings called the word and context embeddings respectively. Each word in the training corpus thus has two vectors associated with it, a word vector, and a context vector. Given a positive pair (w, c), the training algorithm attempts to maximize the dot product $\boldsymbol{w} \cdot \boldsymbol{c'}$, where $\boldsymbol{w}$ is the word vector for the word w and $\boldsymbol{c'}$ is the context vector for the word c. Of course trivially setting all word and context vectors equal to each other would maximize this product, so simultaneously the algorithm attempts to minimize the same dot product over k negative word-context pairs, in this way trying to minimize the similarity of word-context pairs which do not co-occur in the corpus.

Closely related to word2vec is the paragraph vector model (PV) [10, 11], which trains both word vectors and document vectors. In this work we will focus on the distributed bag of words (PV-DBOW) architecture with negative sampling, which is an extension of the word2vec skip-gram architecture.

B. Motivation

In the previous subsection we noted that embedding models can contain negative components, and that they're similar to PMI in this manner. We've also noted that PPMI works better than PMI. Furthermore there's a more concrete connection between word2vec and PMI. In [12] it was shown that the word2vec negative skipgram model is, assuming that the embedding dimension is high enough, equivalent to a factorization of a shifted PMI matrix. In the same work it was also shown that factorizing a shifted PPMI matrix provides superior embeddings. It is then natural to wonder if positivizing embedding models improves their downstream NLP performance or at least not lower performance significantly so that we obtain positive vectors without loosing much information.

Furthermore there's also a question of interpretability in negative components. One method for measuring similarity between two vectors is cosine similarity which computes the cosine of the angle between the vectors.

$$\cos\theta = \frac{\boldsymbol{v} \cdot \boldsymbol{w}}{\|\boldsymbol{v}\|\|\boldsymbol{w}\|}$$

In the work of [4] multiple distributed representations and similarity measures have were evaluated against various NLP tasks and cosine similarity paired with PPMI vectors was found to perform best.

Of note is the fact that cosine similarity is non-negative for representations such as tf-idf and PPMI where the vector components are all non-negative. Where as for PMI and embedding models cosine similarity is between -1 and 1 and can be negative. Negative similarity is difficult to interpret. One might assume that two words whose similarity is close to -1 might be antonyms. However, due to antonyms often occurring in similar contexts, similarity of antonyms tends to be closer to 1 in all representations based on co-occurrence statistics. As such, a distributed representation in which cosine similarity between vectors is non-negative is preferable over one which isn't.

In this work we analyze the effects of positivization on the PV-DBOW model and the downstream NLP tasks of classification word similarity and analogy. For evaluating the classification task we will be using the paragraph vectors, and for the other two tasks we will use the word vectors trained by the PV-DBOW model.

C. Related Work

Improving performance of downstream NLP tasks by post-processing word vectors has been studied before. [13] center the mean of word vectors to this end. [14] detect and remove abnormal dimensions in GloVe [15] embeddings. [16] improve averaged word vectors as a sentence representation by removing their first principal components. In addition to centering the mean, [17] remove the top few principal components. Instead of removing them [18] regularize the variance of principal components.

II. Proposed Model

We look into various ways in which vectors may be positivized. We compare three different positivization schemes: the baseline, the post-processing and the soft-constraint regularization schemes.

In the baseline scheme no positivization occurs and a vanilla PV-DBOW model is trained.

In the post-processing scheme a baseline model is positivized after training by simply setting negative components of the resulting vectors to zero. In other words, we train our classification models on the positivized document embedding matrix $pos(D) = [max(d_{ij}, 0)]$ instead of D.

In the soft-constraint regularization scheme we augment the cost function of the baseline model by adding a regularization term which penalizes negative values in embeddings. More formally:

$$J^* = J + \lambda\Omega(W, C, D)$$

where J is the cost function, J^* is the augment cost function, λ is the regularization hyperparameter, and W, C, D are the word, context and document embedding matrices respectively. While many combinations are possible to formulate the structure of Ω we elected the simplest one: $\Omega(W, C, D) = \delta_w R(W) + \delta_c R(C) + \delta_d R(D)$, where $\delta_w, \delta_c, \delta_d \in \{0, 1\}$ are indicator variables which determine whether a given embedding is considered for regularization. Here $R(E)$ is a function which penalizes negative entries in matrices. While there are eight possible combinations of δs to consider, due to time and hardware constraints we could run experiments for only four of them. These are the cases where

$$\begin{array}{lll} \delta_w = 0 & \delta_c = 0 & \delta_d = 1 \\ \delta_w = 1 & \delta_c = 0 & \delta_d = 0 \\ \delta_w = 1 & \delta_c = 1 & \delta_d = 0 \\ \delta_w = 1 & \delta_c = 1 & \delta_d = 1 \end{array}$$

Note that the case where $\delta_w = \delta_c = \delta_d = 0$ corresponds to the baseline scheme where no regularization is present.

For R we tried only two different penalty functions based on L^1 and L^2 norms:

$$R_1(E) = \sum_{e_{ij}<0} |e_{ij}| \ = \sum_{ij} |max(-e_{ij}, 0)|$$
$$R_2(E) = \sum_{e_{ij}<0} |e_{ij}|^2 = \sum_{ij} |max(-e_{ij}, 0)|^2$$

These two are the most common norms used in regularization.

III. Experimental Methodology

A. Approach

To train our doc2vec models we used the gensim library [19] with some modifications to their Cython source code. Since the gensim does not have automatic differentiation capabilities, we needed to compute the gradient of the regularization terms by hand and add these to the gradient descent step. Here we will only write the gradient formulas for $R_1(E)$ and $R_2(E)$ as the rest may be derived by substitution and the linearity of the derivative operator.

$$\left[\frac{\partial R_1}{\partial E}\right]_{ij} = \begin{cases} -1 & ; e_{ij} < 0 \\ 0 & ; e_{ij} \geq 0 \end{cases}$$
$$\left[\frac{\partial R_2}{\partial E}\right]_{ij} = \begin{cases} 2e_{ij} & ; e_{ij} < 0 \\ 0 & ; e_{ij} \geq 0 \end{cases}$$

B. Setup

We tested the accuracy of PV models for test classification, training them on the classification datasets themselves rather than training a common model on a common corpus and then inferring paragraph vectors for the datasets from this common model. The datasets we used for text classification are the Yelp Reviews Challenge Round 12 dataset [20], which consists of 5996996 documents in 5 classes, and the 20 newsgroups dataset [21] consisting of 18846 documents in 20 classes.

Recall that during our experiments we positivize not only paragraph vectors, but also word and context vectors. In addition to evaluating the paragraph vectors on the text classification task, we evaluate the word vectors on semantic similarity and analogy tasks. For semantic similarity task we used the Wordsim353 [22] dataset, which consists of 353 pairs of words assigned similarity scores between 1 and 10 by people. For the analogy task we used the Google analogy test set [7]. It contains 19544 question pairs.

IV. Results and Discussion

R	δ_w	δ_c	δ_d	Accuracy
-	0	0	0	49.58
Post	-	-	-	51.32
L^1	0	0	1	49.72
L^1	1	0	0	49.71
L^1	1	1	0	49.67
L^1	1	1	1	49.68
L^2	0	0	1	49.58
L^2	1	0	0	49.66
L^2	1	1	0	49.67
L^2	1	1	1	49.62

TABLE I
Classification results on the Yelp dataset. R=regularization type

R	δ_w	δ_c	δ_d	Accuracy
-	0	0	0	53.10
Post	-	-	-	50.86
L^1	0	0	1	53.00
L^1	1	0	0	53.10
L^1	1	1	0	53.25
L^1	1	1	1	53.23
L^2	0	0	1	53.06
L^2	1	0	0	53.03
L^2	1	1	0	53.21
L^2	1	1	1	52.94

TABLE II
Classification results on the 20newsgroups dataset. R=regularization type

Postprocessing appears to have increased classification accuracy in the Yelp reviews dataset by 1.74 percentage points, but decreased it by 2.24 percentage points in the 20 Newsgroups dataset. 20 Newsgroups consisting of about 20000 documents is a much smaller dataset than Yelp reviews which contains almost six million documents. For the larger datasets positivization postprocessing appears to have a positive effect on text classification. But of course this needs to be confirmed by further experiments which we aim to do in future work.

Otherwise for most of the hyperparameter combinations there doesn't seem to be a major change in the classification accuracy. This is especially true for L^2 regularization. In the case of L^1 regularization, especially for the larger values of the regularization parameter, we note significant decreases in classification accuracy.

V. Conclusion

As noted in previous sections embedding models provide dense vector representations of words or documents that in turn used for several NLP tasks such as text classification, analogy, and semantic similarity. Motivated by the relationship between the embedding models and PPMI, and the performance of PPMI on several NLP tasks, we focus on the role of negative values in these dense vectors. We presented positivization of embedding vectors in paragraph vector model, a regularization method with multiple variants. By forcing elements of distributed representations to be positive it is meant to reduce the space in which representations are embedded, and also make them more interpretable.

Positivization does not appear to have a significant effect on text classification. Where it does have an effect, it decreases the accuracy. Similar results can be seen for semantic similarity and analogy tests.

While not ideal, the amount by which positivization reduces the performance of the embeddings is surprisingly small. This is especially the case for postprocessing where instead of applying a soft constraint during training we simply erase the negative components of the embeddings. By forcing document and word vectors into the positive orthant we are decreasing the volume of space in which embeddings appear by a factor of 2^d. While one might expect that the representational ability of the embeddings should significantly decrease as a result, we observe only a slight decrease in one dataset, and an actual increase in the other. We think that positivization can be beneficial in machine learning systems which require non-negative vectors.

VI. Future Work

Positivization via postprocessing appears to be a promising line of inquiry. We plan to experiment with a greater variety of datasets, embeddings, and downstream NLP tasks and thus explore its effects more fully.

We also plan to experiment with centering the mean. While the postprocessing aspects of this method have been fully explored [13, 17], it would be interesting to see if soft-regularization increased or decreased the beneficial effects of centering the mean.

Acknowledgment

This work is supported in part by The Scientific and Technological Research Council of Turkey (TÜBİTAK) grant number 116E047. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of the TÜBİTAK.

References

[1] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[2] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[3] Y. Niwa and Y. Nitta, "Co-occurrence vectors from corpora vs. distance vectors from dictionaries," in *Proceedings of the 15th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 1994, pp. 304–309.

[4] J. A. Bullinaria and J. P. Levy, "Extracting semantic representations from word co-occurrence statistics: A computational study," *Behavior research methods*, vol. 39, no. 3, pp. 510–526, 2007.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 3111–3119. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999792.2999959

[8] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *CoRR*, vol. abs/1402.3722, 2014. [Online]. Available: http://arxiv.org/abs/1402.3722

[9] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[10] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1188–II–1196. [Online]. Available: http://dl.acm.org/citation.cfm?id=3044805.3045025

[11] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv preprint arXiv:1507.07998*, 2015.

[12] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2177–2185. [Online]. Available: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf

[13] M. Sahlgren, A. C. Gyllensten, F. Espinoza, O. Hamfors,

J. Karlgren, F. Olsson, P. Persson, A. Viswanathan, and A. Holst, “The gavagai living lexicon,” in *Language Resources and Evaluation Conference*. ELRA, 2016.

[14] Y.-Y. Lee, H. Ke, H.-H. Huang, and H.-H. Chen, “Less is more: Filtering abnormal dimensions in glove,” in *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 71–72.

[15] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[16] S. Arora, Y. Liang, and T. Ma, “A simple but tough-to-beat baseline for sentence embeddings,” in *International Conference on Learning Representations*, 2016. [Online]. Available: https://openreview.net/forum?id=SyK00v5xx

[17] J. Mu and P. Viswanath, “All-but-the-top: Simple and effective postprocessing for word representations,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HkuGJ3kCb

[18] B. Wang, F. Chen, A. Wang, and C.-C. J. Kuo, “Post-processing of word representations via variance normalization and dynamic embedding,” *arXiv preprint arXiv:1808.06305*, 2018.

[19] R. Řehůřek and P. Sojka, “Software Framework for Topic Modelling with Large Corpora,” in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[20] Yelp, “Yelp dataset challenge round 12,” 2018, https://www.yelp.com/dataset/challenge.

[21] K. Lang, “Newsweeder: Learning to filter netnews,” in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[22] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. d. Wolfman, and E. Ruppin, “Placing search in context: The concept revisited,” *ACM Transactions on information systems*, vol. 20, no. 1, pp. 116–131, 2002.