Word Replaceability Through Word Vectors

Peter Taraba¹

Received: 9 April 2019 / Accepted: 14 April 2020 / Published online: 25 April 2020
© The Author(s) 2020

Abstract

There have been many numerical methods developed recently that try to capture the semantic meaning of words through word vectors. In this study, we present a new way to learn word vectors using only word co-appearances and their average distances. However, instead of claiming semantic or syntactic word representation, we lower our assertions and claim only that we learn word vectors, which express word's replaceability in sentences based on their Euclidean distances. Synonyms are a subgroup of words which can replace each other, and we will use them to show differences between training on words that appear close to each other in a local window and training that uses distances between words, which we use in this study. Using ConceptNet 5.5.0's synonyms, we show that word vectors trained on word distances create higher contrast in distributions of word similarities than was done with Glove, where only word appearances close to each other were engaged. We introduce a measure, which looks at intersection of histograms of word distances for synonyms and non-synonyms.

Keywords Word vectors · Word embeddings · Synonyms · Sentence structure · Glove · Histogram comparison

Introduction

Language modeling and word vector representation is an interesting field of study, not only because it has the ability to group similar words together, but the number of applications using these models is quite extensive. These include speech recognition [9], text correction [6], machine translation [10] and many other natural language processing (NLP) tasks [16].

One of the first vector space models (VSMs) was developed by [17] and his colleagues [18]. A good survey of VSMs can be found in [20] and in [8] using neural networks. Not surprisingly, due to success of neural networks there are a lot of neural language models—for example, [2, 3]. Current methods use a local context window and a logarithm of counts how many times words appear together in various lengths of these windows. Works [15] and [14] are examples. Authors of [11] claim that learning word vectors via language modeling produces syntactic rather than semantic focus, and in their work, they wish to learn semantic representation.

Similar works which try to find synonyms automatically but without use of word vectors include [12] and [13].

In this work, we use the distance between words in a training dataset to train word vectors on higher-quality texts (almost 120,000 of New York Times articles published between 1990 and 2010) instead of relying on a mix of higher- and lower-quality texts coming from the internet as it was done in Glove. A text quality study can be found in [1]. Rather than claiming semantic representation which would be a significant step forward, we claim only word replaceability through word vectors representation. Two words are replaceable if they have approximately the same relationship with other words in the vocabulary, and a relationship with other words is expressed through distances in sentences they appear in. To test our results, we use 64,000 synonyms from ConceptNet 5.5.0 [19] instead of using just 353 related words in [5] or 999 related words in [7]; hence, we verify our results statistically more robust way.

Outline The remainder of this article is organized as follows. “**Training Algorithm**” section describes how we break the text into shorter forms, how we convert the text data into two-word combinations, and how we describe the training of words vectors in comparison with Glove. In “**Results**” section, we compare results with Glove on distributions of synonyms and non-synonyms from ConceptNet 5.5.0, which has 64,000 synonyms in its English dictionary intersected

✉ Peter Taraba
taraba.peter@mail.com

¹ 251 Brandon Street Apt 131, San Jose, CA 95134, USA

with vocabulary in the training set, and we also show words which end up closest together. Finally, in “Conclusions” section we draw conclusions.

Training Algorithm

Similarly to the works of many others, we represent each word w_i in vocabulary V by vector v_i of size N . The main idea behind finding replaceable words is to describe their relationship with other words in vocabulary, and words, which should be found as replaceable, should end up after training in each other’s vicinities represented by word vectors with low Euclidean distance. One way to do this is to count how many times the words appear with another word in an individual context windows as it was done in Glove [15], for example, using the entire dataset. In Glove, the main optimization function uses the dot product of word vectors $v_i \cdot v_j = v_i^T v_j$

$$J_{\text{glove}} = \sum_{i,j} f(X_{i,j})(v_i^T \tilde{v}_j + b_i + \tilde{b}_j - \log X_{i,j})^2,$$

where $X_{i,j}$ are the counts of words combination appearances, f is weighing function, $\tilde{v}_j$ is a separate context word vector, and $b_i, \tilde{b}_j$ are biases. Hence, rather than using Euclidean distance to measure word similarities, it is better to use cosine similarity on trained vectors as the main optimization function and cosine similarity both use the dot product. In Results section, we show both Euclidean distance and cosine similarity histograms for Glove trained word vectors and new word vectors as this simple test shows significant shift.

In order to achieve improved results over Glove, we go a step further and instead of only using counts of word co-appearances in the same context windows, we also measure distances between words d_{i_k,j_k} , where d represents words w_{i_k} and w_{j_k} distance in a sub-sentence. As two words can appear many times together in different sub-sentences and those distances can differ, we specify k for these different occasions.

As a first step in training, we do pre-processing of a dataset and break all sentences into sub-sentences separating them by commas, dots, and semicolons in order to better capture short-distance relationships. In Glove, the authors used word window, which was moving continuously in texts and did not separate the sentences. Next we take all two-word co-appearances in these sub-sentences and use their distances, as it is done in Fig. 1, for optimizing word vectors.

Optimization function can be described as follows:

$$J_1 = \sum_{k=1}^O (\|v_{i_k} - v_{j_k}\| - d_{i_k,j_k})^2,$$

Fig. 1 Using distances between word co-appearances for optimizing word vectors

where O is the number of all training two-word occurrences (the same tuple of two different words can occur more than once). As this would create too large of a training dataset (O can be rather high), we simplify it by uniting parts of the sum by the same word tuples, and hence, we get

$$J_1 = \sum_{i,j}^M a_0(i,j) \|v_i - v_j\|^2 + a_1(i,j) \|v_i - v_j\| + a_2(i,j),$$

where $M < O$ is the number of all word co-occurrences (every tuple of two different words appears only once), $a_0(i,j)$ is the number of occurrences of a two-word tuple (w_i, w_j) , $a_1(i,j) = -2 \sum_k d_{i_k,j_k}$ represents sum of distances between the same two words multiplied by -2 , and $a_2(i,j)$ is the sum of the squared distances of the same two words. If the same word appears twice in the same sentence, we do not include such sample in training, as the Euclidean distance between the same word vector will always be 0 irrespective of optimization function, and such sample would have no influence on the final result anyway. By doing this, all we have to remember for training the word vectors is $a_0(i,j)$ and $a_1(i,j)$ for word tuples (w_i, w_j) as $a_2(i,j)$ does not have to be remembered since it represents only an offset of our optimization function and has no influence when computing partial derivatives for vector optimization and the optimization function becomes

$$J = \sum_{i,j}^M a_0(i,j) \|v_i - v_j\|^2 + a_1(i,j) \|v_i - v_j\|.$$

As we already mentioned, while Glove used only $a_0(i,j)$ information (how many times word tuples appeared— $X_{i,j}$), in our optimization we also use extra information $a_1(i,j)$.

This optimization is equivalent to optimizing by remembering the count of two-word appearances and the average of their distances as the only thing that would differ would be the constant part of the optimization function, which we cut off anyway as it does not have influence on partial derivatives. The optimization function could be written also as

$$\begin{aligned} J_2 &= \sum_{i,j}^M a_0(i,j) (\|v_i - v_j\| - \bar{d}_{ij})^2 \\ &= \sum_{i,j}^M \left(a_0(i,j) \|v_i - v_j\|^2 - 2a_0(i,j) \bar{d}_{ij} \|v_i - v_j\| + a_0(i,j) \bar{d}_{ij}^2 \right) \end{aligned}$$

After removing the constant part of it, it becomes

$$\begin{aligned} J_3 &= \sum_{i,j}^M \left(a_0(i,j) \|v_i - v_j\|^2 - 2a_0(i,j) \bar{d}_{ij} \|v_i - v_j\| \right) \\ &= \sum_{i,j}^M \left(a_0(i,j) \|v_i - v_j\|^2 - 2a_0(i,j) \frac{\sum_k d_{ik,jk}}{a_0(i,j)} \|v_i - v_j\| \right) \\ &= \sum_{i,j}^M \left(a_0(i,j) \|v_i - v_j\|^2 + a_1(i,j) \|v_i - v_j\| \right) = J \end{aligned}$$

Hence, optimizing J , J_1 , J_2 , and J_3 yields the same partial derivatives of word vector values, as the only thing which differs is the constant offset for various versions of the optimization function.

As a side note, in our optimization function, $a_0(i,j)$ can be thought of as a weight, because the more samples of the same word tuple we have, the more robust estimation of distance average we get and hence we want to give higher importance to tuples which appear more often. Also, quite important is to mention that in our optimization we do not train any biases and more importantly we do not train two vectors for each word as it was done in Glove (one is used as the main vector and one as a context vector).

The training can be summarized to the following steps:

- Divide text to sentences by splitting texts using commas, dots, and semicolons
- Find word tuples, which appear in same sentence and count how many times tuples appeared and their average distance (or sum of the distances)
- Use optimization technique (we used gradient descent method using constant step size), which optimizes function J to find optimal values of vectors v_i

Time complexity of computing partial derivatives for vector's values in every epoch of gradient descent is $\mathcal{O}(MN)$, where M is number of word tuples, which appeared at least once and N is chosen vector size. We used NVIDIA's

GPU GeForce GTX 1080 Ti and C++ CUDA programming language to speed up the training process instead of using C++ and multi-threaded computations on CPU as it was done in Glove.

Results

In this section, we compare Glove and the results of our optimization function by using ConceptNet's synonyms in "Histogram Comparison" and look at the words that appear closest together in "Closest Words".

Histogram Comparison

We show for both Glove and our word vectors that there is a shift in the histogram between words which are defined as synonyms in ConceptNet and words which are not defined as synonyms in ConceptNet. There are approximately 64,000 of synonyms in the English language in ConceptNet, and for non-synonyms, we choose random 64,000 word tuples, which are not defined as synonyms in ConceptNet. For Glove, this is shown in Fig. 2.

Not surprisingly, there is a bigger shift for Glove looking at cosine similarity, as during the training, the dot product was used in the optimization function, which is closer to sinus similarity than Euclidean distance.

The advantage of using our optimization function is that instead of one cluster (peaks of two histograms close together) which can be seen for Glove, we can see two clusters of words (peaks of histograms far from each other), one representing replaceable words and one representing words that cannot replace each other. The majority of synonyms from ConceptNet end up in the cluster which represents replaceable words. As we already mentioned, as our optimization function optimizes on Euclidean distances, it is not surprising that cosine similarity does not show much of a shift in Fig. 3 (Top), while the Euclidean distance histogram does in Fig. 3 (Bottom).

To see how well different algorithms perform on separating synonyms and non-synonyms, we measure the percentage of the histogram's area intersection for synonyms distance histogram and non-synonyms distance histogram. The lower the number, the better separation of synonyms from non-synonyms there is (Table 1). Using our optimization, we achieved a better score 14.59 % in comparison with Glove which achieves 39.17 %. Using only the dot product for vectors instead of cosine similarity, which is normalized dot product, would only worsen Glove's results; hence, we omit these results.

Fig. 2 Results for GloVe: [top] histogram of word similarities expressed by their cosine similarity (x-axis); [bottom] histogram of word similarities expressed by their Euclidean distances (x-axis). Blue is the histogram of randomly selected pairs of words which are not defined as synonyms in ConceptNet, while green is the histogram of synonyms in ConceptNet

Closest Words

In Table 2, we show the closest words which came out of training of our optimization function to explain that rather than claiming semantic focus of word vectors, we use term replaceability instead. The most replaceable words found are different shortcuts of months (top of the table), followed by words expressing different sexes (middle of table). The bottom of the table shows other examples—names of people are easily replaceable, as well as days in a week, different time intervals and different counts. Some of the words which are used together such as Los Angeles appear at the top as well.

We did the same analysis for GloVe as well in Table 3 and what can be seen there is that different days of weeks

Fig. 3 Results for our optimization function for size of vectors $N = 150$ after 31k iterations: [top] histogram of word similarities expressed by their cosine similarity (x-axis); [bottom] histogram of word similarities expressed by their Euclidean distances (x-axis). Blue is the histogram of randomly selected pairs of words which are not defined as synonyms in ConceptNet, while green is the histogram of synonyms in ConceptNet

and different names of months come closest as well (using cosine similarity), but also a lot of words which do not have a lot of meaning but probably appeared in the training data due to different training dataset (Wikipedia, blogs, etc., versus only New York Times articles) and different ways of processing it (not excluding numbers, etc.).

Conclusions

We presented a new way to construct the vector space model, which finds replaceable words, and showed that in comparison with GloVe, it creates two clusters and distinguishes more between words that can be replaced and words that cannot. To achieve that, we used not only

Table 1 Percentage of histograms area intersection between synonyms scores and non-synonyms scores

Algorithm	ConceptNet	
	64k synonyms	
	CS (%)	ED (%)
Glove 6B $N = 300$	39.17	48.50
Our alg. $N = 50$ NI = 5k	92.64	19.82
Our alg. $N = 50$ NI = 31k	77.82	15.60
Our alg. $N = 150$ NI = 5k	89.91	18.88
Our alg. $N = 150$ NI = 31k	70.88	14.65
Our alg. $N = 300$ NI = 31k	67.13	14.59

Glove achieves 39.17% intersection using cosine similarity (CS), while our algorithm achieves 14.59% using Euclidean distance (ED). The lower the number, the better the separation of synonyms and non-synonyms there is. N is size of a word vector and NI is number of iterations using gradient descent optimization. ConceptNet has 64,000 synonyms, which match vocabulary in word vectors

Table 2 Closest word tuple examples using training of our optimization function for word vector size 50 after 30,000 iterations

w_1	w_2	$\ v_1 - v_2\ $
Oct	Dec	0.500262
Oct	Feb	0.51383
Nov	Dec	0.514184
Aug	Feb	0.516395
Nov	Jan	0.520554
Spokesman	Spokeswoman	0.592484
Stepdaughter	Stepson	0.500262
Daughter	Son	1.02896
Sons	Daughters	1.13604
Mr	Ms	1.15107
William	Robert	0.966206
Thursday	Wednesday	0.994217
Months	Weeks	1.09893
Los	Angeles	1.11654
Hundred	Thousand	1.35937

Score closer to 0 means better replaceability, and it can go up to 80 (see histogram in Fig. 3 [Bottom])

word tuple counts (number of appearances) as it was done in various previous approaches, but we used additional information as well—distances between words in sub-sentences. We prefer claiming only word replaceability focus in a new word vector space, which we showed on cluster of synonyms in histogram, rather than claiming semantic representation on only few selected words or smaller tests.

Table 3 Closest word tuple examples using Glove with word vectors of size 300

w_1	w_2	$\frac{v_1^T v_2}{\ v_1\ \ v_2\ }$
Tuesday	Wednesday	0.99158
Tuesday	Thursday	0.987584
Tuesday	Monday	0.986374
Wednesday	Thursday	0.985771
Monday	Thursday	0.983714
October	November	0.970944
December	October	0.969877
December	February	0.968549
January	February	0.965211
January	December	0.96206
Sandberger	<unk>	1
Piyanart	Srivalo	0.999873
Hahrd	Shroh	0.990815
gph04bb	str94	0.988308
Kuala	Lumpur	0.985766

Score closer to 1 means better similarity (see histogram in Fig. 2 [Top])

Future work should focus on using even more information from the training dataset to even better capture similarities between words and their replaceability or semantic focus as well as including results on real tasks as it is done in other publications such as [4].

Acknowledgements The authors would like to thank anonymous reviewers for providing feedback which led to significant improvement of this paper.

Compliance with Ethical Standards

Conflict of Interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Antiqueira L, Nunes M, Nunes OOM Jr, Costa L da F. Strong correlations between text quality and complex networks features. *Physica A Stat Mech Appl*. 2007;373:811–20.

2. Bengio Y, Ducharme R, Vincent P, Janvin C. A neural probabilistic language model. *J Mach Learn Res.* 2003;3(null):1137–55.
3. Collobert R, Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning. In: *Proceedings of the 25th international conference on machine learning, ICML '08*, pp 160–167. New York, NY, USA, 2008. Association for Computing Machinery.
4. Devlin J, Chang M.-W, Lee K, Toutanova K Bert: Pre-training of deep bidirectional transformers for language understanding. In: *NAACL-HLT*, 2019.
5. Finkelstein L, Gabrilovich E, Matias Y, Rivlin E, Solan Z, Wolfman G, Ruppin E. Placing search in context: the concept revisited. In: *Proceedings of the 10th international conference on world wide web, WWW '01*, pp 406–414. New York, NY, USA, 2001. Association for Computing Machinery.
6. Guo J, Sainath T.N, Weiss R.J. A spelling correction model for end-to-end speech recognition. In: *ICASSP 2019—2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp 5651–5655, 2019.
7. Hill F, Reichart R, Korhonen A. Simlex-999: evaluating semantic models with (genuine) similarity estimation. *Comput Linguist.* 2014;41:665–95.
8. Khattak FK, Jeblee S, Pou-Prom C, Abdalla M, Meaney C, Rudzicz F. A survey of word embeddings for clinical text. *J Biomed Inf X.* 2019;4:100057.
9. Kuhn R, De Mori R. A cache-based natural language model for speech recognition. *IEEE Trans Pattern Anal Mach Intell.* 1990;12(6):570–83.
10. Lembersky G, Ordan N, Wintner S. Language models for machine translation: original vs. translated texts. *Comput Linguist.* 2012;38(4):799–825.
11. Maas A.L, Ng A.Y. A probabilistic model for semantic word vectors. In: *Workshop on deep learning and unsupervised feature learning, NIPS*, 2010;10.
12. McCarthy D, Navigli R. Semeval-2007 task 10: English lexical substitution task. In: *Proceedings of the 4th international workshop on semantic evaluations, SemEval '07*, pp. 48–53, USA, 2007. Association for Computational Linguistics.
13. Melamud O, Levy O, Dagan I. A simple word embedding model for lexical substitution. In *VS@HLT-NAACL*, 2015.
14. Mikolov T, Chen K, Corrado G.S, Dean J. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
15. Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.
16. Peters M.E, Ammar W, Bhagavatula C, Power R. Semi-supervised sequence tagging with bidirectional language models. In: *ACL*, 2017.
17. Salton G. The SMART retrieval system-experiments in automatic document processing. Englewood Cliffs: Prentice-Hall Inc; 1971.
18. Salton G, Wong A, Yang CS. A vector space model for automatic indexing. *Commun ACM.* 1975;18(11):613–20.
19. Speer R, Havasi C. Representing general relational knowledge in ConceptNet 5. In: *Proceedings of the eighth international conference on language resources and evaluation (LREC'12)*, pp. 3679–3686, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA).
20. Turney PD, Pantel P. From frequency to meaning: vector space models of semantics. *J Artif Int Res.* 2010;37(1):141–88.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.