Word2vec: What and Why

Jiaqi Mu
University of Illinois at Urbana-Champaign
jiaqimu2@illinois.edu

***Abstract*—Words have been studied for decades as the basic unit in natural language. Although words are traditionally modeled as atomic units, a real-valued representation can wield power in many application domains. The state-of-the-art in such real-valued representations is *word2vec*, known for its efficiency in handling large datasets and its ability to capture multiple degrees of similarities. In this report, we will present the training model of word2vec and summarize the evolution of word2vec, ranging from its canonical form to state-of-the-art implementations. We show that word2vec can be understood as a low-dimensional factorization of the so-called word-context matrix, whose cells are the pointwise mutual information (PMI) of the respective word and context pairs, shifted by a global constant. Following this insight, we explain the ability of word2vec of modeling similarity by a probabilistic interpretation of the "distributional hypothesis" from linguistics.**

***Keywords*—word, vector representations, multiple degrees of similarities.**

I. Introduction

Most applications in natural language processing (NLP) take words as the basic unit. Understanding the interaction of words in a corpus of documents is one of the most important research areas in NLP. The fundamental difficulty is the large cardinality of the vocabulary (for example, the size of the vocabulary in English is around 10^6). Learning marginal word distribution, i.e., unigram distribution, is a difficult task – the underlying distribution follows Zipf-law, where the majority of words appear only a few times in the corpus. If one would like to model the joint distribution of two words, i.e., bigram distribution, there are potentially 10^{12} free parameters, which are too many to infer from current datasets. Due to the grammatical and semantical structures in documents, neither unigram distribution nor bigram distribution is enough to model them. Higher order statistics, however, are almost impossible to infer.

An alternative solution to the cardinality problem that arises from representing words as atomic units is to represent them in real space by using an appropriate function (example: neural network) to model the interaction between words. Such an approach is particularly useful because similarities between words can now be captured by the distances between their representations, which is not possible when we take words as discrete items. Moreover, we would like to consider functions (in the neural network) that operate on these real-valued representations in a smooth fashion: that is, similar input word sequences should map to similar outputs. The primary challenge, then, lies in defining an appropriate mapping between words and their real-valued representations.

To address this challenge, different word embedding algorithms have been proposed in recent years. Neural network language models represent the word by its context words [1], [2], [3], [4], where the context word with respect to the current word is defined to be any word within a window ρ to the left and right of the current word, excluding the current word itself. For example, in the following piece of text where the current word is "genres" and $\rho = 3$,

> ... a series of many genres, including fantasy, drama, coming of age ...

"series", "of", "many", "including", "fantasy", and "drama" are the context words. The nonlinearity and non-convexity when constructing the representations leads to computationally-intensive training stages. To ameliorate the computational difficulties, [5] simplified the deep neural network models by using a single layer feedforward neural network architecture to allow an efficient training process. Due to this simplification, word vector representations can be benefit from a word being modeled by a larger window of contexts in sentences and being able to handle larger training corpus. This work has made a large impact on the NLP community and is popularly known as *word2vec*.

It is not clear if word2vec is the best representation, nor is it clear exactly what properties of words and sentences the representations should model. Different properties of the representations are useful in different applications. There is one property – similar words should have similar representations – is important in all tasks, since it captures the basic difference between continuous representations and discrete ones. What makes word2vec a successful algorithm is the property that the representations obtained from word2vec can capture multiple degrees of similarity, e.g., between words or between pairs of words, in an *unsupervised* fashion. The similarity between words is captured by the distance between the words' corresponding vectors. The similarity between pairs of words is captured by the distances between the pairs' corresponding difference vectors. For example, the closest vector to $vector(\text{``}king'') - vector(\text{``}man'') + vector(\text{``}woman'')$ is the vector of the word "queen".

After making this surprising observation, one would ask a natural question: "why can word2vec capture multiple degrees of similarity?" Even though this question largely remains satisfactorily not answered, several recent works provide insightful understanding of word2vec. The understanding of similarity between words comes from a linguistic hypothesis, i.e., the *distributional hypothesis* [6]: "a word is characterized

$$L_{\mathrm{SG}}^{\mathrm{NS}}(u,v)=\sum_{(w,c)}\#(w,c)\left(\log\sigma(u(w)^{\mathrm{T}}v(c))+k\mathbf{E}_{c'\sim p_C}\log\sigma\left(-u(w)^{\mathrm{T}}v(c')\right)\right). \tag{1}$$

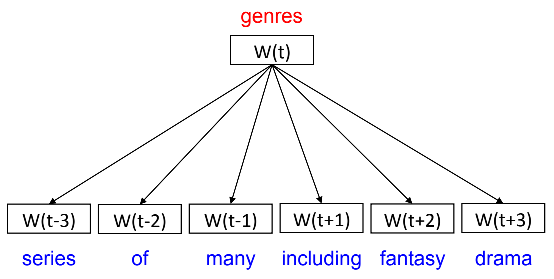

Fig. 1. The architectures in word2vec to predict the context words given the current one (SG).

by the company it keeps." According to this hypothesis, [7] interpreted the similarity between two words w_1 and w_2 as the similarity between the empirical conditional distributions of the context word given w_1 and w_2 respectively. In parallel, [8] showed that word2vec is implicitly doing a weighted low dimensional factorization on the cooccurrence statistics of the word and the words around it with some preprocessing and a careful choice of hyperparameters [8]. More recently, [9] showed that the distances between word vectors capture the similarity between the empirical distributions of their contexts via a generative model, and thus capture the similarities between words.

II. Algorithm

The skip-gram (SG) model of word2vec aims at efficiently predicting the context words given the current word. The architecture in SG is presented in in Figure 1, where $w(t)$ is the t-th word in the corpus. The goal of word2vec is to predict $w(t-\rho),...,w(t+\rho)$ (e.g. series, of, many, including, fantasy, drama) given the current word $w(t)$ (e.g. genres).

The complexity in prior work [1], [2], [3], [4] stemmed from nonlinear operations in the training methods' hidden layers. Word2vec addressed this by changing nonlinear operations to more efficient bilinear ones, while also training on larger datasets to compensate for the loss of nonlinearity. To allow efficient computation, word2vec further makes independence assumptions on the context words. The training objective for SG is to maximize the probability of the context words given the current one,

$$\begin{aligned} L_{\mathrm{SG}} &= \sum_t \log p(w_{t-\rho},...,w_{t-1},w_{t+1},...,w_{t+\rho}|w_t) \\ &= \sum_t \sum_{c\neq 0, c=-\rho}^{\rho} \log p_{C|W}(w_{t+c}|w_t). \end{aligned} \tag{2}$$

To avoid ambiguity, we use w to represent the current word and use c to represent its context word. The conditional distribution $p_{C|W}(c|w)$ in (2) is defined as,

$$p_{C|W}(c|w)=\frac{\exp(u(w)^{\mathrm{T}}v(c))}{\sum_{c'\in\mathcal{V}}\exp(u(w)^{\mathrm{T}}v(c'))}, \tag{3}$$

where u and v are the mapping from word to its word- and context-vector representations respectively, and $\mathcal{V}$ is the vocabulary as a list of words.

In (3), the summation over w (i.e., the vocabulary) is used to normalize the vectors; however, this summation is computationally intensive. To solve this issue, word2vec introduces the *negative sampling* (NS) mechanism to further reduce the computation complexity. Let $p(D=1|w,c)$ be the probability that the word/context pair (w,c) comes from the data, and $p(D=0|w,c)$ be the probability that it does not. The probability distribution is defined via a logistic regression,

$$\begin{aligned} p(D=1|w,c) &= \sigma\left(u(w)^{\mathrm{T}}v(c)\right), \\ p(D=0|w,c) &= \sigma\left(-u(w)^{\mathrm{T}}v(c)\right), \end{aligned}$$

where $\sigma(x)=1/(e^{-x}+1)$ is the sigmoid function. Let $\#(w,c)$ be the number of occurrence for word/context pair (w,c) in the training corpus. We define the marginal statistics by $\#(w)=\sum_{c\in\mathcal{V}}\#(w,c)$, $\#(c)=\sum_{w\in\mathcal{V}}\#(w,c)$ and $|D|=\sum_{w,c}\#(w,c)$. The training objectives for SG then turns into maximizing $p(D=1|w,c)$ for observed pair (w,c) and maximizing $p(D=0|w,c')$ for randomly generated pairs (w,c') (this pair is called a "negative" sample), where c' is randomly generated from an empirical unigram distribution p_C, which is defined as:

$$p_C(c)=\frac{\sum_w \#(w,c)}{\sum_{w,c'}\#(w,c')}=\frac{\#(c)}{|D|}.$$

The optimization objective for SG with NS is formulated in (1), where k is a hyperparameter controlling the number of negative samples. One can implement a parallel training algorithm using mini-batch asynchronous gradient descent with AdaGrad [10].

III. Theoretical Analysis

It is the ability to capture multiple degrees of similarity, e.g., between words or between pairs of words, in an *unsupervised* fashion that makes makes word2vec a successful algorithm. We will theoretically justify this ability in the remaining of this section.

Let n be the size of the vocabulary and let d be the dimension of the vector representations. Word2vec represents both words and their contexts in a dense low dimension space in $\mathbf{R}^d$ by the mappings u and v defined in Section II. We embed u and v to a word- and context-matrix $U\in\mathbf{R}^{n\times d}$ and $V\in\mathbf{R}^{n\times d}$ respectively, where the i-th row of each matrix is the corresponding vector representation for the i-th word in the vocabulary. The sufficient statistics in the training objective in (1) are the inner product between $u(w)$ and $v(c)$. As a result, the sufficient statistics are the product of two matrices, i.e., $UV^{\mathrm{T}} \in \mathbf{R}^{n\times n}$. We use w and c to denote the indices of w and c in the vocabulary if there is no ambiguity. Let $M = UV^{\mathrm{T}}$, one can rewrite the objective defined in (1) as,

$$\begin{aligned} L^{\text{NS}}_{\text{SG}}(M) =& \sum_{w,c} \#(w,c)(\log\sigma(M_{wc}) + k\mathbf{E}\log\sigma(-M_{w'c})) \\ =& \sum_{w,c} \#(w,c)\log\sigma(M_{wc}) \\ &+ \sum_{c} \#(c)k\sum_{w'}\frac{\#(w')}{|D|}\log\sigma(-M_{w'c}) \\ =& \sum_{w,c} \#(w,c)\log\sigma(M_{wc}) \\ &+ \sum_{w,c} \#(w,c)k\frac{\#(c)\#(w)}{|D|\#(w,c)}\log\sigma(-M_{wc}), \end{aligned}$$

and rewrite the optimization as,

$$\hat{M} = \arg\max_{\text{rank}(M)\le d} L^{\text{NS}}_{\text{SG}}(M). \qquad (4)$$

To study the matrix $\hat{M}$, we make the first assumption:

Assumption 1. The dimension d is large enough so that we can ignore the rank constraint.

Therefore each elements in M are independent of the others. This objective is then decomposed into n^2 objectives, whose variable is M_{wc} for each (w,c) pair.

$$L^{\text{NS}}_{\text{SG}}(M) = \sum_{w,c}\#(w,c)L_{wc}(M_{wc}),$$

where L_{wc} is defined over each (w,c) pair as,

$$L_{wc}(m) = \left(\log\sigma(m) + k\frac{\#(c)\#(w)}{|D|\#(w,c)}\log\sigma(-m)\right).$$

One can obtain the maximizing $\hat{M}$ by optimizing the objectives independently, i.e.,

$$\begin{aligned} \hat{M}_{wc} &= \arg\max_{m\in R}\left(\log\sigma(m) + k\frac{\#(c)\#(w)}{|D|\#(w,c)}\log\sigma(-m)\right) \\ &= \log\left(\frac{|D|\#(w,c)}{\#(c)\#(w)}\right) - \log k. \qquad (5) \end{aligned}$$

Observing the fact that $\log\left(\frac{|D|\#(w,c)}{\#(c)\#(w)}\right) = \log\left(\frac{\frac{\#(w,c)}{|D|}}{\frac{\#(c)}{|D|}\frac{\#(w)}{|D|}}\right) = \log\left(\frac{p_{W,C}(w,c)}{p_W(w)p_C(c)}\right)$ is the PMI between w and c, one can interpret SG as a low-dimensional matrix factorization on the shifted version of the empirical PMI matrix of word/context pair.

To measure the multiple degrees of similarity between words by the distances between their corresponding vectors, we make another assumption:

Assumption 2. For any vector $x \in \mathbf{R}^d$,

$$\|x\|^2 \approx \alpha x^{\mathrm{T}}\mathbf{E}_c(v(c)v(c)^{\mathrm{T}})x = \alpha\mathbf{E}_c(x^T v(c))^2, \qquad (6)$$

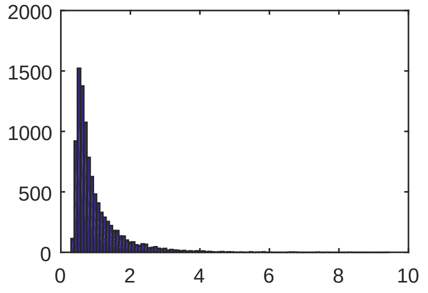

Fig. 2. Isotropy property of vector representations v. the histogram of $\mathbf{E}_c(x^T v(c))^2/\|x\|^2$ for 10,000 random vector x. The x-axis is normalized by the mean of values.

where α is a universal constant.

This assumption can be validated empirically: Figure 2 shows the histogram of $\mathbf{E}_c(x^T v(c))^2/\|x\|^2$ for 10,000 random vectors x. The x-axis is normalized by the mean of their values. It can be observed that most samples satisfy the property (6).

A. Word Similarity

The definition of word similarity comes from the distributional hypothesis [6]: "a word is characterized by the company it keeps." In a probabilistic view, if two words w_1 and w_2 are closely related, then for most context words c,

$$p_{C|W}(c|w_1) \approx p_{C|W}(c|w_2),$$

and if they are remotely related, for most context words c,

$$p_{C|W}(c|w_1) \neq p_{C|W}(c|w_2).$$

In the setting of word2vec, for any two words w_1, w_2 and any context word c, one can compute the differences of the conditional probabilities using (5),

$$u(w_1)^{\mathrm{T}}v(c) - u(w_2)^{\mathrm{T}}v(c) = \log\left(\frac{p_{C|W}(c|w_1)}{p_{C|W}(c|w_2)}\right). \qquad (7)$$

Under the Assumption 2, one has,

$$\|u(w_1) - u(w_2)\|^2 \approx \alpha\mathbf{E}_c\log\left(\frac{p_{C|W}(c|w_1)}{p_{C|W}(c|w_2)}\right)^2,$$

which indicates that the distances between the vectors measure the similarities between the corresponding words. To visualize the clustering property, we show two-dimensional PCA projections of the 300-dim SG vectors of some sample words in Figure 3, where the vectors of the similar words are close to each other.

B. Word Analogy

Other than the similarity of words, the similarity of word pairs can also be captured by word2vec. Word analogy is a task to evaluate the measure of word pair similarity: given four words w_a, w_b, w_c, and w_d where the relation between w_a and w_b are the same as w_c and w_d, one is asked to predict w_d given the other three words. For example, given three words

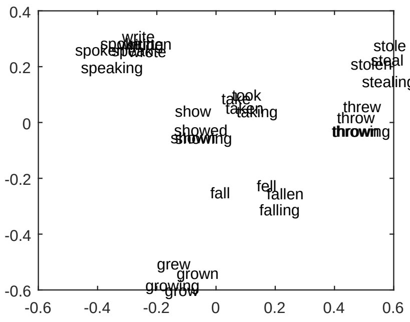

Fig. 3. Two dimensional PCA projection of the 300-dim SG vectors of some sample words, where the verbs that share the same infinitive clus. The figure illustrates the ability of the SG model to measure the similarities between words by the distances between the corresponding vectors.

"man", "king" and "woman", the answer should be "queen" since "queen" is the word that is similar to "king" in the same sense as "man" is to "woman". [7] and [8] justified the answer in a probabilistic fashion: most context words c satisfies,

$$\frac{p_{C|W}(c|king)}{p_{C|W}(c|man)} \approx \frac{p_{C|W}(c|queen)}{p_{C|W}(c|woman)}.$$

and therefore "queen" is the solution to the following optimization problem,

$$\min_w \mathbf{E}_c \left(\log\left(\frac{p_{C|W}(c|king)}{p_{C|W}(c|man)}\right) - \log\left(\frac{p_{C|W}(c|w)}{p_{C|W}(c|woman)}\right)\right)^2.$$

From (6) and (7), one can readily compute the ratios using their vector representations, i.e.,

$$\begin{aligned}
&\mathbf{E}_c \left(\log\left(\frac{p_{C|W}(c|w_b)}{p_{C|W}(c|w_a)}\right) - \log\left(\frac{p_{C|W}(c|w_d)}{p_{C|W}(c|w_c)}\right)\right)^2 \\
&\quad = \mathbf{E}_c \left((u(w_b) - u(w_a))^{\mathrm{T}} v(c) - (u(w_d) - u(w_c))^{\mathrm{T}} v(c)\right)^2 \\
&\quad = \mathbf{E}_c \left(((u(w_b) - u(w_a)) - (u(w_d) - u(w_c)))^{\mathrm{T}} v(c)\right)^2 \\
&\quad \approx \|(u(w_b) - u(w_a)) - (u(w_d) - u(w_c))\|^2/\alpha.
\end{aligned}$$

As a result, one can solve word analogy questions by a simple arithemic operation on the vectors, i.e.,

$$\hat{w}_d = \min_{w_d} \|(u(w_b) - u(w_a)) - (u(w_d) - u(w_c))\|^2.$$

The observation above indicates that the differences of vector pairs capture the relations between word pairs. One can use this property to learn word pairs that share the common relation in an unsupervised fashion. Table I lists the examples of the word pair relations using word2vec representations [5]. Figure 4 shows the difference vectors of a semantic relation (countries and their capitals) and a syntactic relation (verbs, their comparatives and their superlatives).

IV. Experiments

To empirically justify the ability of word2vec to capture multiple degrees of similarity, we perform two tasks – word similarity and word analogy.

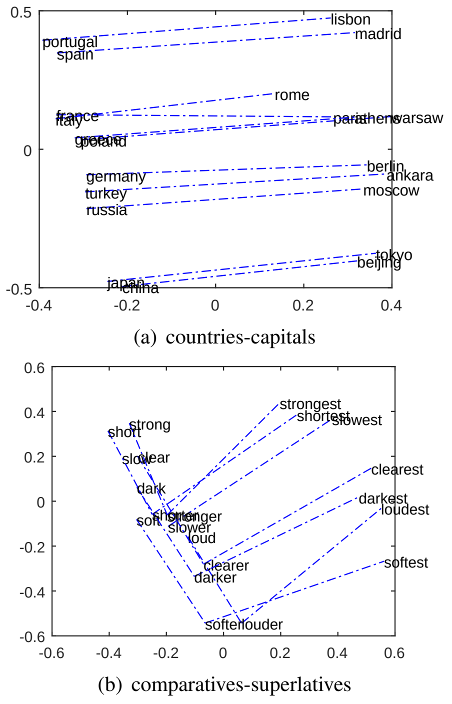

Fig. 4. Two dimensional PCA projection of the 300-dim SG vectors of semantic relations and syntactic relations. This figure illustrate the ability of SG vectors to automatically learn the implicit relations between words.

TABLE I
Examples of the word pairs generated from the vector representations that share the common relationship.

Relationship	Example 1	Example 2
france : paris	italy : rome	japan : tokyo
big : bigger	small : larger	cold : colder
miami : florida	baltimore : maryland	dallas : texas
einstein : scientist	messi : midfielder	mozart : violinist
sarkozy : france	berlusconi : italy	merkel : germany
copper : cu	zinc : zn	gold : au
berlusconi : silvio	sarkozy : nicholas	putin : medvedev
microsoft : windows	google : android	ibm : linux
microsoft : ballmer	google : yahoo	ibm : mcnealy
japan : sushi	germany : bratwurst	france : tapas

A. Parameters and Baselines

We train word2vec representations on the Wikipedia dump in August 2013 with 1.6 billion tokens (an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing). We use the preprocessing script from Matt Mahoney's website[1] to remove non-textual elements, split sentences, tokenize and lowercase the vocabulary. We build a vocabulary of words that occur more than 100 times in the training corpus. We set the context window size to be 10, use 10 negative samples and set the dimension of vectors to be 500.

We compare with two baselines, which are singular value decomposition (SVD) approaches motivated by (5). SVD-SPMI computes the SVD of the shifted PMI matrix to obtain

[1]http://mattmahoney.net/dc/textdata.html

TABLE II
SPEARMAN RAN CORRELATION ρ(X100) BETWEEN THE HUMAN SCORES AND THE ALGORITHM SCORES ON WORD SIMILARITY TASKS.

model	WordSim Similarity	WordSim Relatedness	Mechanical Turk	Rare Words
SG	**79.4**	**70.0**	**67.8**	28.1
SVD-SPMI	76.6	68.1	62.8	**31.2**
SVD-SPPMI	73.5	**70.0**	66.3	23.5

TABLE III
ACCURACY(X100) ON WORD ANALOGY TASKS EVALUATING USING GOOGLE'S DATASET AND MSR'S DATASET.

model	Google	MSR
SG	**69.4**	**52.0**
SVD-SPMI	52.6	35.6
SVD-SPPMI	53.2	24.9

the vector representations, and SVD-SPPMI computes the SVD of the modified version – the positive shifted PMI matrix [8]. The context window size and the dimension of vectors in SVD-SPMI and SVD-SPPMI are the same as those in word2vec.

B. Evaluation methods

We empirically validate that word2vec can capture multiple degrees of similarities on two different tasks – word similarity and word analogy.

a) Word similarity: We evaluate the performance on word similarity task of word2vec on various datasets including: the *WordSim353* from [11] (this dataset is partitioned into two datasets: *WordSim Similarity* and *WordSim Relatedness* [12]), the *Mechanical Turk* from [13] and the *Rare Words* from [14]. All datasets have word pairs, each of which has a human-assigned similarity score. The word vectors are evaluated by measuring the Spearman's rank correlation coefficients between the distances between word vectors and the human ratings.

b) Word analogy: We evaluate the performance on word analogy task of word2vec on MSR dataset and Google dataset. In MSR's analogy dataset [15] there are 8,000 morpho-syntactic analogy questions. In Google's dataset [5], there are 19,544 questions, half of which are of the same kind as in MSR and the other half are semantic questions, such as capital/cities. After filtering questions containing the out-of-vocabulary words, there are 7,118 and 19,258 remaining questions in MSR's dataset and Google's dataset, respectively.

C. Results

We present results on the word similarity task and on the word analogy task in Table II and Table III, respectively. The results indicate that SG is better at capturing the similarities between words and pairs of words than the straight forward SVD approaches. Even though it is not clear where the performance improvement comes from, we guess it could potentially come from the weights on the low dimension factorization – the more frequent $\#(w, c)$ is, the closer $u(w)^{\mathrm{T}} v(c)$ is to the optimal value $\hat{M}_{wc}$.

V. CONCLUSION

In this report, we present word2vec, an algorithm to embed words in real space, and study the quality of the representations on word similarity and word analogy tasks. We observe that the representations, even though they are trained using a simple model architecture, are able to capture the similarity between words and word pairs in an unsupervised fashion. We show some theoretical justifications for this property by interpreting the distributional hypothesis from linguistics in a probabilistic view, and empirically validate the property that the multiple degrees of similarity between words are captured by the distances between the corresponding vectors.

REFERENCES

[1] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," in *Innovations in Machine Learning*. Springer, 2006, pp. 137–186.

[2] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 641–648.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[4] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng, "Improving word representations via global context and multiple word prototypes," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics, 2012, pp. 873–882.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[6] J. R. Firth, *Papers in linguistics, 1934-1951*. Oxford University Press, 1957.

[7] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[8] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems*, 2014, pp. 2177–2185.

[9] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski, "Rand-walk: A latent variable model approach to word embeddings," *arXiv preprint arXiv:1502.03520*, 2015.

[10] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[11] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 406–414.

[12] T. Zesch, C. Müller, and I. Gurevych, "Using wiktionary for computing semantic relatedness." in *AAAI*, vol. 8, 2008, pp. 861–866.

[13] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch, "A word at a time: computing word relatedness using temporal semantic analysis," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 337–346.

[14] T. Luong, R. Socher, and C. D. Manning, "Better word representations with recursive neural networks for morphology." in *CoNLL*. Citeseer, 2013, pp. 104–113.

[15] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations." in *HLT-NAACL*, 2013, pp. 746–751.