GNEG: Graph-Based Negative Sampling for word2vec

Zheng Zhang
LIMSI, CNRS, Université Paris-Saclay, Orsay, France
& LRI, Univ. Paris-Sud, CNRS,
Université Paris-Saclay, Orsay, France
zheng.zhang@limsi.fr

Pierre Zweigenbaum
LIMSI, CNRS,
Université Paris-Saclay,
Orsay, France
pz@limsi.fr

Abstract

Negative sampling is an important component in word2vec for distributed word representation learning. We hypothesize that taking into account global, corpus-level information and generating a different noise distribution for each target word better satisfies the requirements of negative examples for each training word than the original frequency-based distribution. In this purpose we pre-compute word co-occurrence statistics from the corpus and apply to it network algorithms such as random walk. We test this hypothesis through a set of experiments whose results show that our approach boosts the word analogy task by about 5% and improves the performance on word similarity tasks by about 1% compared to the skip-gram negative sampling baseline.

1 Introduction

Negative sampling, as introduced by Mikolov et al. (2013b), is used as a standard component in both the CBOW and skip-gram models of word2vec. For practical reasons, instead of using a softmax function, earlier work explored different alternatives which approximate the softmax in a computationally efficient way. These alternative methods can be roughly divided into two categories: softmax-based approaches (hierarchical softmax (Morin and Bengio, 2005), differentiated softmax (Chen et al., 2015) and CNN-softmax (Kim et al., 2016)) and sampling-based approaches (importance sampling (Bengio et al., 2003), target sampling (Jean et al., 2014), noise contrastive estimation (Mnih and Teh, 2012) and negative sampling Mikolov et al. (2013b)). Generally speaking, among all these methods, negative sampling is the best choice for distributed word representation learning (Ruder, 2016).

Negative sampling replaces the softmax with binary classifiers. For instance, in the skip-gram model, word representations are learned by predicting a training word's surrounding words given this training word. When training, correct surrounding words provide positive examples in contrast to a set of sampled negative examples (noise). To find these negative examples, a noise distribution is empirically defined as the unigram distribution of the words to the $3/4^{th}$ power:

$$P_n(w) = U(w)^{\frac{3}{4}} / \sum_{i=1}^{|vocab|} U(w_i)^{\frac{3}{4}} \qquad (1)$$

Although this noise distribution is widely used and significantly improves the distributed word representation quality, we believe there is still room for improvement in the two following aspects: First, the unigram distribution only takes into account word frequency, and provides the same noise distribution when selecting negative examples for different target words. Labeau and Allauzen (2017) already showed that a context-dependent noise distribution could be a better solution to learn a language model. But they only use information on adjacent words. Second, unlike the positive target words, the meaning of negative examples remain unclear: For a training word, we do not know what a good noise distribution should be, while we do know what a good target word is (one of its surrounding words).

Our contributions: To address these two problems, we propose a new graph-based method to calculate noise distribution for negative sampling. Based on a word co-occurrence network, our noise distribution is targeted to training words. Besides, through our empirical exploration of the noise distribution, we get a better understanding of the

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 566–571
Melbourne, Australia, July 15 - 20, 2018.

meaning of 'negative' and of the characteristics of good noise distributions.

The rest of the paper is organized as follows: Section 2 defines the word co-occurrence network concepts and introduces our graph-based negative sampling approach. Section 3 shows the experimental settings and results, then discusses our understanding of the good noise distributions. Finally, Section 4 draws conclusions and mentions future work directions.

2 Graph-based Negative Sampling

We begin with the word co-occurrence network generation (Section 2.1). By comparing it with the word2vec models, we show the relation between the stochastic matrix of the word co-occurrence network and the distribution of the training word contexts in word2vec. We introduce three methods to generate noise distributions for negative sampling based on the word co-occurrence network:

- Directly using the training word context distribution extracted from the word co-occurrence network (Section 2.2)
- Calculating the difference between the original unigram distribution and the training word context distribution (Section 2.3)
- Performing t-step random walks on the word co-occurrence network (Section 2.4)

We finally insert our noise distribution into the word2vec negative sampling training (Sec. 2.5).

2.1 Word Co-occurrence Network and Stochastic Matrix

A word co-occurrence network is a graph of word interactions representing the co-occurrence of words in a corpus. An undirected edge can be created when two words co-occur within a sentence; these words are possibly non-adjacent, with a maximum distance defined by a parameter d_{max} (Ferrer-i-Cancho and Solé, 2001). Given two words w_u^i and w_v^j that co-occur within a sentence at positions i and j $(i, j \in \{1 \dots l\})$, we define the distance $d(w_u^i, w_v^j) = |j - i|$ and the co-occurrence of w_u and w_v at a distance δ as $cooc\,(\delta, w_u, w_v) = \left|\left\{\, (w_u^i, w_v^j) \,|\, d\,(w_u^i, w_v^j) = \delta \,\right\}\right|$. We define the weight $w(d_{max}, w_u, w_v)$ of an edge (w_u, w_v) as the total number of co-occurrences of w_u and w_v with distances $\delta \leq d_{max}$:

$$w\,(d_{max}, w_u, w_v) = \sum_{\delta=1}^{d_{max}} cooc(\delta, w_u, w_v).$$

An undirected weighted word co-occurrence network can also be represented as a symmetric adjacency matrix (Mihalcea and Radev, 2011), a square matrix A of dimension $|W| \times |W|$. In our case, W is the set of words used to generate the word co-occurrence network, and the matrix elements A_{uv} and A_{vu} have the same value as the weight of the edge $w(d_{max}, w_u, w_v)$. Then each row of the adjacency matrix A can be normalized (i.e., so that each row sums to 1), turning it into a right stochastic matrix S.

Negative sampling, in the skip-gram model, uniformly draws at random for each training word one of its surrounding words as the (positive) target word. This range is determined by the size of the training context c. In other words, a surrounding word w_s of the training word w_t must satisfy the following condition: $d(w_t^i, w_s^j) \leq c$.

For the same corpus, let us set d_{max} equal to c in word2vec and generate a word co-occurrence network of the whole corpus. Then element S_{uv} in the adjacency matrix extracted from the network represents the probability that word w_v be selected as the target word for training word w_u ($P_{bigram}(w_u, w_v)$ in Eq. 2). Row S_u thus shows the distribution of target words for training word w_u after training the whole corpus. Note that no matter how many training iterations are done over the corpus, this distribution will not change.

$$P_{bigram}(w_u, w_v) = \frac{\sum_{\delta=1}^{d_{max}} cooc(\delta, w_u, w_v)}{\sum_{i=1}^{|vocab|} \sum_{\delta=1}^{d_{max}} cooc(\delta, w_u, w_i)} = S_{uv} \tag{2}$$

Networks and matrices are interchangeable. The adjacency matrix of the word co-occurrence network can also be seen as the matrix of word-word co-occurrence counts calculated in a statistical way as in GloVe (Pennington et al., 2014). But unlike Glove, where the matrix is used for factorization, we use word co-occurrence statistics to generate a network, then use network algorithms.

2.2 Training-Word Context Distribution

As discussed in Section 2.1, the stochastic matrix S of the word co-occurrence network represents the context distribution of the training words. Here, we use S directly as one of the three types

of bases for noise distribution matrix calculation.

The idea behind this is to perform nonlinear logistic regression to differentiate the observed data from some artificially generated noise. This idea was introduced by Gutmann and Hyvärinen (2010) with the name Noise Contrastive Estimation (NCE). Negative sampling (NEG) can be considered as a simplified version of NCE that follows the same idea and uses the unigram distribution as the basis of the noise distribution. We attempt to improve this by replacing the unigram distribution with a bigram distribution (word co-occurrence, not necessarily contiguous) to make the noise distribution targeted to the training word (see Eq. 2).

Compared to the word-frequency-based unigram distribution, the word co-occurrence based bigram distribution is sparser. With the unigram distribution, for any training word, all the other vocabulary words can be selected as noise words because of their non-zero frequency. In contrast, with the bigram distribution, some vocabulary words may never co-occur with a given training word, which makes them impossible to be selected for this training word. To check the influence of this zero co-occurrence case, we also provide a modified stochastic matrix S' smoothed by replacing all zeros in matrix S with the minimum non-zero value of their corresponding rows.

2.3 Difference Between the Unigram Distribution and the Training Words Contexts Distribution

Contrary to the hypothesis underlying the previous section, here we take into account the positive target words distribution in the training word context distribution. Starting from the 'white noise' unigram distribution, for each training word w_u, we subtract from it the corresponding context distribution of this training word. Elements in this new basis matrix $S_{difference\,u,v}$ of noise distribution are:

$$P_{difference}(w_u, w_v) = P_n(w_u) - S_{uv} \quad (3)$$

where w_v is one of the negative examples of w_u, P_n is the unigram distribution defined in Eq. 1 and S is the stochastic matrix we used in Section 2.2. For zeros and negative values in this matrix, we reset them to the minimum non-zero value of the corresponding row $P_{difference}(w_u)$.

2.4 Random Walks on the Word Co-occurrence Network

After generating the word co-occurrence network, we apply random walks (Aldous and Fill, 2002) to it to obtain yet another noise distribution matrix for negative sampling.

Let us define random walks on the co-occurrence network: Starting from an initial vertex w_u , at each step we can cross an edge attached to w_u that leads to another vertex, say w_v. For a weighted word co-occurrence network, we define the transition probability $P(w_u, w_v)$ from vertex w_u to vertex w_v as the ratio of the weight of the edge (w_u, w_v) over the sum of weights on all adjacency edges of vertex w_u. Using the adjacency matrix A and the right stochastic matrix S presented in Section 2.1, $P_{u,v}$ can be calculated by:

$$P(w_u, w_v) = A_{uv} / \sum_{i=1}^{|A_u|} A_{ui} = S_{uv}.$$

As we want to learn transition probabilities for all training words, we apply random walks on all vertices by making each training word an initial vertex of one t-step random walk at the same time.

The whole set of transition probabilities can be represented as a transition matrix, which is exactly the right stochastic matrix S of the word co-occurrence network in our case. We found that the self-loops (edges that start and end at the same vertex: the main diagonal of an adjacency matrix or a stochastic matrix) in matrix S represent the occurrence of a word in its own context, which may happen in repetitions. We hypothesize they constitute spurious events and therefore test the t-step random walk both on matrix S and its smoothed version R' in which the self-loops are removed. To see the effect of the self-loops, we perform the t-step random walk on both matrices S and R'.

Based on that, the elements of the t-step random walk transition matrix can be calculated by:

$$P_{random\text{-}walk}(w_u, w_v) = S^t_{uv} \text{ or } (R')^t_{uv} \quad (4)$$

Ferrer-i-Cancho and Solé (2001) showed that a word co-occurrence network is highly connected. For such networks, random walks converge to a steady state in just a few steps. Steady state means that no matter which vertex one starts with, the distribution of the destination vertex probabilities remains the same. In other words, all S^t columns will have the same value. So we set the maximum step number t_{max} to 4. We will use these t-step random walk transition matrices as the basis for

one of our noise distributions matrices for negative sampling.

2.5 Noise Distribution Matrix

Starting from the basic noise distribution matrix, we use the power function to adjust the distribution. Then we normalize all rows of this adjusted matrix to let each row sum to 1. Finally, we get:

$$P_n(w_u, w_v) = (B_{uv})^p / \sum_{i=1}^{|B_u|} (B_{ui})^p \qquad (5)$$

where B is the basic noise distribution calculated according to Eq. 2, 3 or 4 and p is the power.

When performing word2vec training with negative sampling, for each training word, we use the corresponding row in our noise distribution matrix to replace the original unigram noise distribution for the selection of noise candidates.

3 Experiments and Results

3.1 Set-up and Evaluation Methods

We use the skip-gram negative sampling model with window size 5, vocabulary size 10000, vector dimension size 200, number of iterations 5 and negative examples 5 to compute baseline word embeddings. Our three types of graph-based skip-gram negative sampling models share the parameters of the baseline. In addition to these common parameters, they have their own parameters: the maximum distance d_{max} for co-occurrence networks generation, a Boolean *replace_zeros* to control whether or not to replace zeros with the minimum non-zero values, a Boolean *no_self_loops* to control whether or not to remove the self-loops, the number of random walk steps t (Eq. 4) and the power p (Eq. 5).

All four models are trained on an English Wikipedia dump from April 2017 of three sizes: about 19M tokens, about 94M tokens (both are for detailed analyses and non-common parameters grid search in each of the three graph-based models) and around 2.19 billion tokens (for four models comparison). During corpus preprocessing, we use CoreNLP (Manning et al., 2014) for sentence segmentation and word tokenization, then convert tokens to lowercase, replace all expressions with numbers by 0 and replace rare tokens with *UNK*s.

We perform a grid search on the ∼19M tokens corpus, with $d_{max} \in \{2, \dots 10\}$, $t \in \{1, \dots 4\}$, $p \in \{-2, -1, 0.01, 0.25, 0.75, 1, 2\}$ and $True, False$ for the two Boolean parameters. We retain the best parameters obtained by this grid search and perform a tighter grid search around them on the ∼94M tokens corpus. Then based on the two grid search results, we select the final parameters for the entire Wikipedia dump test. We evaluate the resulting word embeddings on word similarity tasks using WordSim-353 (Finkelstein et al., 2001) and SimLex-999 (Hill et al., 2014) (correlation with humans), and on the word analogy task of Mikolov et al. (2013a) (% correct). Therefore, we use the correlation coefficients between model similarity judgments and human similarity judgments for WordSim-353 and SimLex-999 tasks and the accuracy of the model prediction with gold standard for the word analogy task (the metrics in Table 1) as objective functions for these parameter tuning processes.

3.2 Results

The best grid search parameters are shown in Table 2, final evaluation results on the entire English Wikipedia in Table 1. The results show that graph-based negative sampling boosts the word analogy task by about 5% and improves word similarity by about 1%.

As vocabulary size is set to 10000, not all data in evaluation datasets is used. We report here the sizes of the datasets and of the subsets that contained no unknown word, that we used for evaluation: WordSim-353: 353; 261; SImLex-999: 999; 679; Word Analogy: 19544; 6032. We also computed the statistical significance of the differences between our models and the baseline model. Both word similarity tasks use correlation coefficients, so we computed Steiger's Z tests (Steiger, 1980) between the correlation coefficients of each of our models (bigram distribution, difference distribution and random walk distribution) versus the word2vec skip-gram negative sampling baseline. For WordSim-353, differences are significant (*2-tailed p* < 0.05) for difference distribution and random walk distribution for both Pearson and Spearman correlation coefficients; differences are not significant for bigram distribution. For SimLex-999, no difference is significant (all *2-tailed p* > 0.05). The word analogy task uses accuracy, we tested statistical significance of the differences by approximate randomization (Yeh, 2000). Based on 10000 shuffles, we confirmed that all differences between the accuracies of our

	WordSim-353		SimLex-999		Word Analogy		
	Pearson	Spearman	Pearson	Spearman	Semantic	Syntactic	Total
baseline word2vec	66.12%	69.60%	37.36%	36.33%	73.00%	70.67%	71.37%
bigram distr. (Eq. 2)	66.10%	69.77%	**38.05%**	**37.18%**	77.36%†	75.55%†	76.09%†
difference distr. (Eq. 3)	**67.71%**†	**71.51%**†	37.65%	36.58%	77.14%†	**75.98%**†	**76.33%**†
random walk (Eq. 4)	66.94%†	70.70%†	37.73%	36.74%	**77.75%**†	74.86%†	75.73%†

Table 1: Evaluation results on WordSim-353, SimLex-999 and the word analogy task for the plain word2vec model and our three graph-based noise distributions on the entire English Wikipedia dump. A dagger† marks a statistically significant difference to the baseline word2vec.

distribution	d_{max}	p	others
bigram	3	0.25	*replace_zeros=T*
difference	3	0.01	
random walk	5	0.25	$t = 2$, *no_self_loops=T*

Table 2: Best parameters

models and the accuracy of word2vec skip-gram are statistically significant ($p < 0.0001$).

The time complexity when using our modified negative sampling distribution is similar to that of the original skip-gram negative sampling except that the distribution from which negative examples are sampled is different for each token. We pre-compute this distribution off-line for each token so that the added complexity is proportional to the size of the vocabulary. Specifically, pre-computing the co-occurrences and graphs using corpus2graph (Zhang et al., 2018) takes about 2.5 hours on top of 8 hours for word2vec alone on the entire Wikipedia corpus using 50 logical cores on a server with 4 Intel Xeon E5-4620 processors : the extra cost is not excessive.

Let us take a closer look at each graph-based model. First, the word context distribution based model: we find that all else being equal, replacing zero values gives better performance. We believe a reason may be that for a training word, all the other words should have a probability to be selected as negative examples—the job of noise distributions is to assign these probabilities. We note that for SimLex-999, all combinations of parameters in our grid search outperform the baseline. But unfortunately the differences are not significant. Initially it sounds contradictory that directly using the word context distribution S as the noise distribution: a higher probability denotes that the word w_v is more likely to be the target word of w_u (i.e., positive example). So we tried assigning different negative powers to adjust the distribution S (Section 2.5) so as to make lower co-occurrence frequencies lead to higher probability of being selected as negative examples. But all these perform poorly for all three tasks.

Second, the difference model: the word analogy task results show a strong dependency on power p: the lower the power p, the higher the performance.

Third, the random-walk model: we observe that all top 5 combinations of parameters in the grid search do random walks after removing self-loops.

4 Conclusion

We presented in this paper three graph-based negative sampling models for word2vec. Experiments show that word embeddings trained by using these models can bring an improvement to the word analogy task and to the word similarity task.

We found that pre-computing graph information extracted from word co-occurrence networks is useful to learn word representations. Possible extensions would be to test whether using this information to select target words (positive examples) could improve training quality, and whether using it to reorder training words could improve training efficiency.

References

David Aldous and James Allen Fill. 2002. Reversible markov chains and random walks on graphs. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, pages 1–9.

Welin Chen, David Grangier, and Michael Auli. 2015. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*.

Ramon Ferrer-i-Cancho and Richard V. Solé. 2001. The small world of human language. *Proceedings of*

the Royal Society of London B: Biological Sciences, 268(1482):2261–2265.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *CoRR*, abs/1408.3456.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*, pages 2741–2749.

Matthieu Labeau and Alexandre Allauzen. 2017. An experimental analysis of noise-contrastive estimation: the noise distribution matters. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 15–20.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Rada F. Mihalcea and Dragomir R. Radev. 2011. *Graph-based Natural Language Processing and Information Retrieval*, 1st edition. Cambridge University Press, New York, NY, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Sebastian Ruder. 2016. On word embeddings - part 2: Approximating the softmax. `http://ruder.io/word-embeddings-softmax`. Last accessed 11 May 2018.

James H Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological bulletin*, 87(2):245.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zheng Zhang, Ruiqing Yin, and Pierre Zweigenbaum. 2018. Efficient generation and processing of word co-occurrence networks using corpus2graph. In *Proceedings of TextGraphs-12: the Workshop on Graph-based Methods for Natural Language Processing*. Association for Computational Linguistics.