GNEG: Graph-Based Negative Sampling for word2vec

Zheng Zhang[1,2] and Pierre Zweigenbaum[1]

[1]LIMSI, CNRS, Université Paris-Saclay

[2]LRI, Université Paris-Sud, CNRS, Université Paris-Saclay

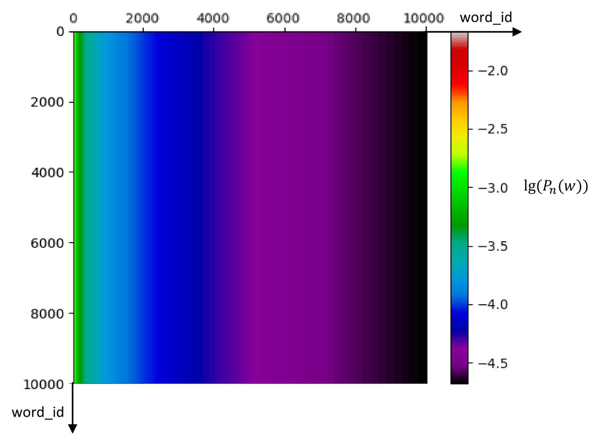
1. Motivation

- Negative Sampling (NEG) is an important component in word2vec:
  As an approximation to Noise Contrastive Estimation (NCE), NEG brings a significant speed-up and achieves very good performance on distributed word representation learning.

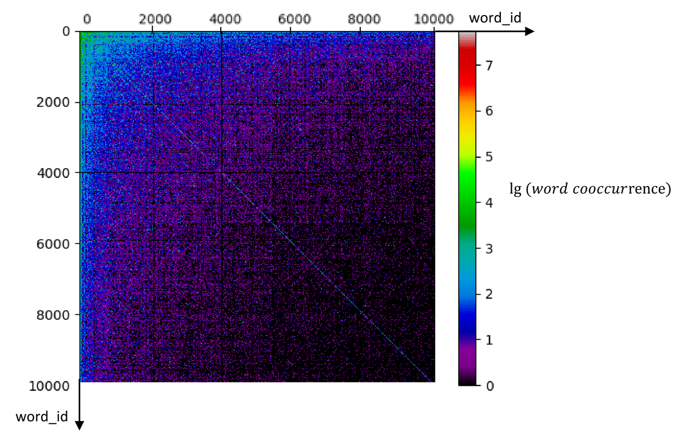
- But NEG is not targeted for training words, noise distribution is only based on the unigram distribution (word count):

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

- 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.



NEG (unigram) noise distribution

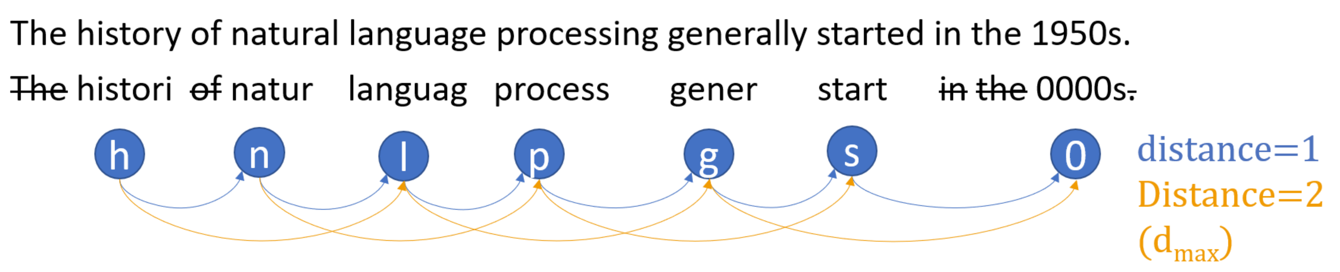


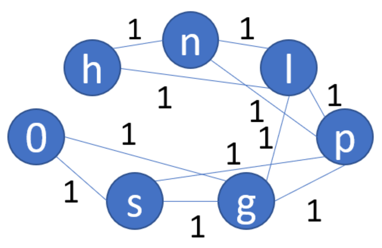
word co-occurrence (bi-gram) distribution

2. Graph-Based Negative Sampling

Build the graph-based negative sampling noise distribution in 3 steps!

Step 1: "Making the dough" - Generate an undirected weighted word co-occurrence network from the corpus and get the adjacency matrix A from it for the future use.





$weight = 1 \times number\ of \uparrow + 1 \times number\ of \uparrow$
undirected

Efficient Generation and Processing of Word Co-occurrence Networks Using corpus2graph
Zheng ZHANG, Ruiqing YIN, Pierre ZWEIGENBAUM, In Proceedings of NAACL 2018 Workshop on Graph-Based Algorithms for Natural Language Processing, New Orleans, US

Step 2: "Creating the toppings" - Three methods to generate basic noise distribution matrices on the word co-occurrence network.

Option 1 Directly using the training word context distribution A extracted from the word co-occurrence network.

- Zero co-occurrence case: Some vocabulary words may never co-occur with a given training word, which makes them impossible to be selected for this training word.
- Solution: Replacing all zeros in matrix with the minimum non-zero value of their corresponding rows.

Option 2 Calculating the difference between the original unigram distribution and the training word context distribution.

- For zeros and negative values in the matrix, we reset them to the minimum non-zero value of the corresponding rows.

Option 3 Performing t-step random walks on the word co-occurrence network.

- Using the t-step random walk transition matrix as the final noise distribution matrix
- Two versions: with/without self-loops

Step 3: "Baking" - Based on the previous results, use the power function to adjust the distribution and then normalize all rows of the adjusted matrix to get the final noise distribution.

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

3. Experiments

Corpora

- 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 graph-based negative sampling models share the parameters of the baseline.

- 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.

Evaluation Datasets

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).

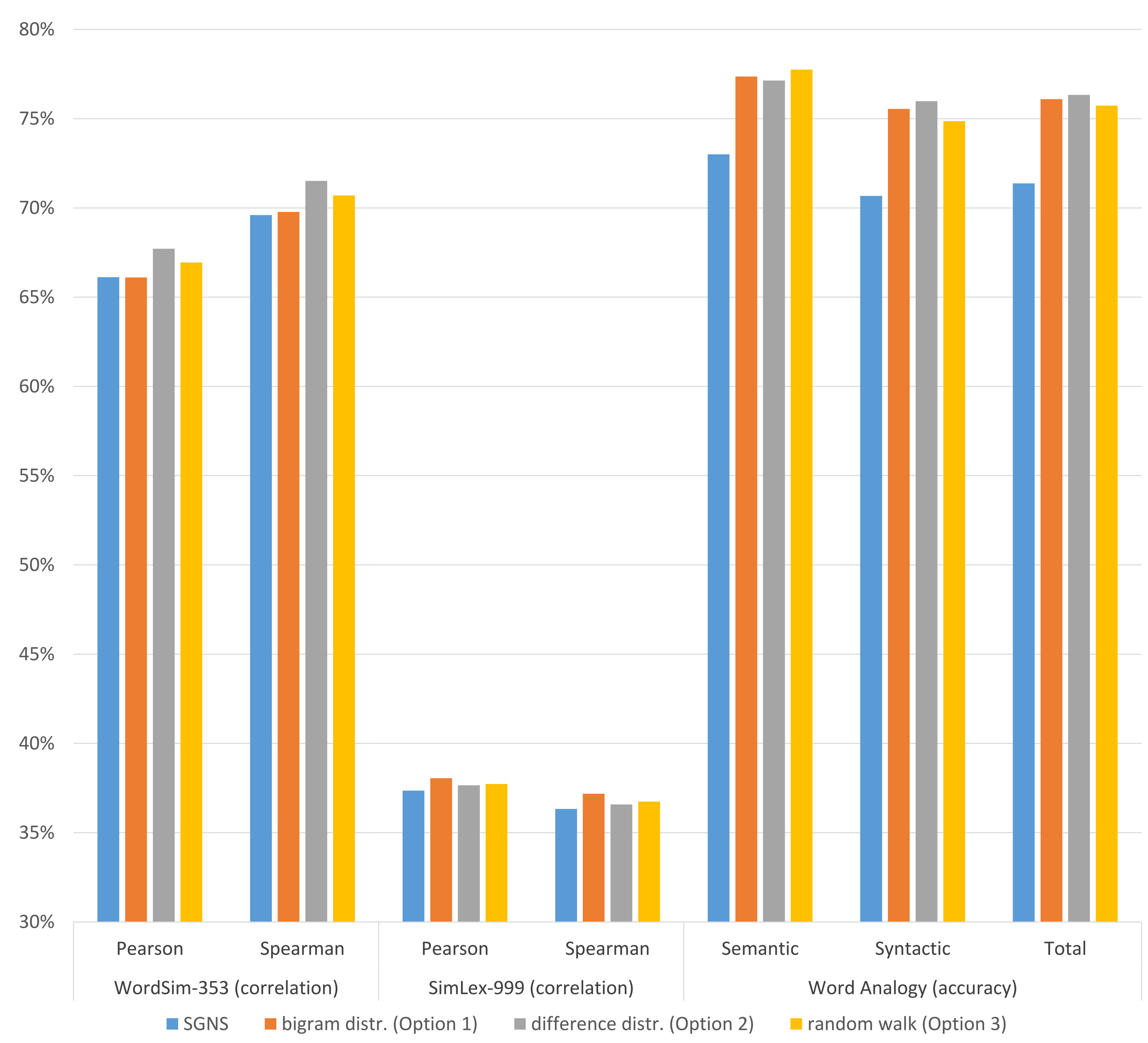
- on the word analogy task (Mikolov et al., 2013a) (% correct).

Statistical Significance

- Steiger's Z tests (Steiger, 1980) for WordSim-353 and SimLex-999

- Approximate randomization (Yeh, 2000) for the word analogy task

4. Results



Best parameters

distribution	d_{max}	p	*others*
bigram	3	0.25	$replace\_zeros = T$
difference	3	0.01	
Random walk	5	0.25	$t = 2, no\_self\_loops = T$

Training time

8 + 2.5 hours*
word2vec corpus2graph

*Trained on the entire Wikipedia corpus using 50 logical cores on a server with 4 Intel Xeon E5-4620 processors.

5. Future work

- Graph-based context words selection
- Graph-based training words reordering for word2vec
- Word co-occurrence matrix factorization for distributed word representation learning