Optimizations of the Skip-Gram model with negative sampling

Milinaire Cédric

Chair of Data Science
University of Passau, Germany

1 April 2019

Overview of my thesis

- Word embeddings are vector representations of words
- Word embeddings are a powerful tool that facilitate NLP
- Skip Gram Model with negative sampling, is a simple and powerful algorithm (Mikolov et al.) [1]
- This work focused on optimizing the convergence time
- Techniques used:
  - Advanced optimizers
  - Input shuffling

Outline

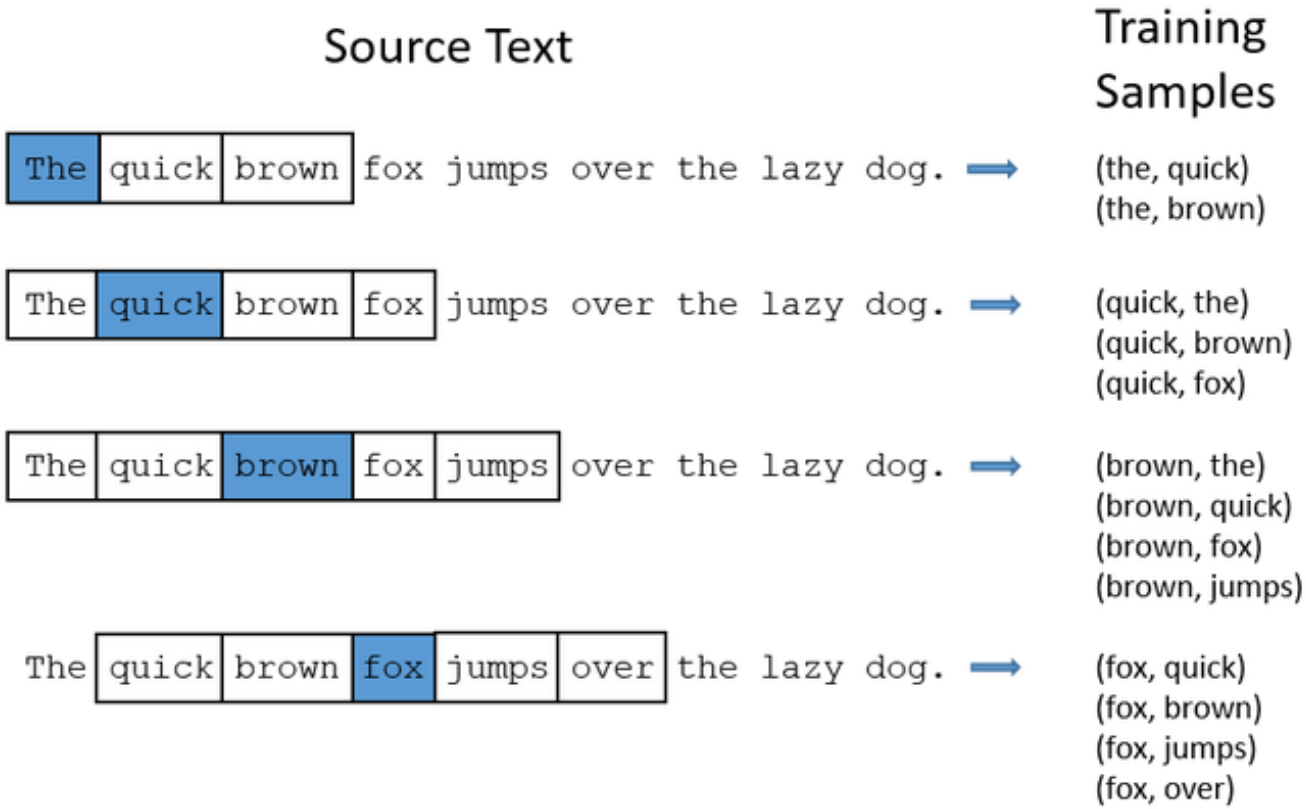
Background

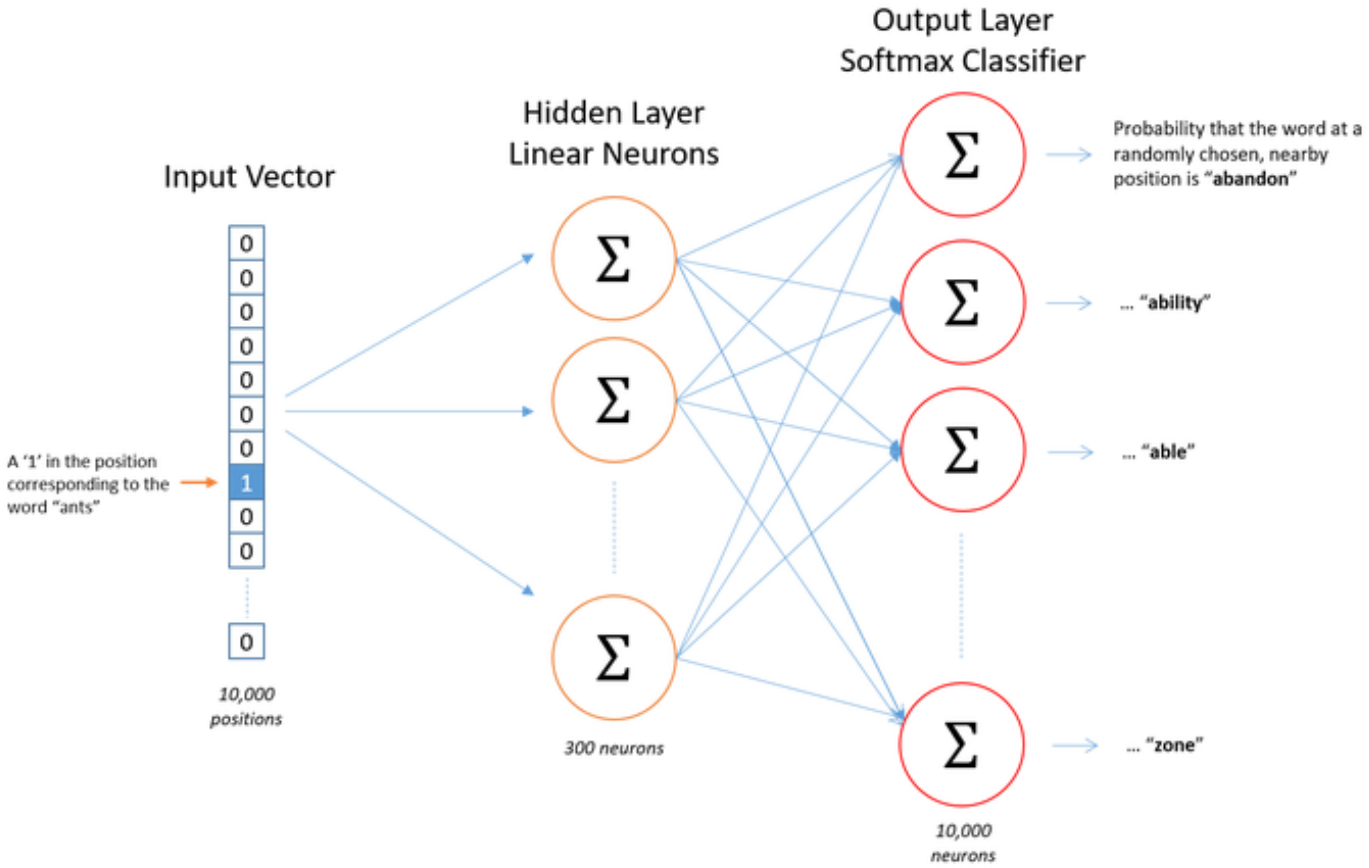
1. Skip Gram Model
2. Skip Gram with Negative Sampling (SGNS)

Background

Main idea: train a network on a "fake task" then use the weights as embedding.

- The fake task:
- Given a word w guess the context words.





(Source: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/)

Softmax:

$$p(c|w) = \frac{exp(v_c^{'\mathsf{T}} v_w)}{\sum_{i=1}^{T} exp(v_i^{'\mathsf{T}} v_w)} \quad (1)$$

$v^{'}$is the output layer vector v is the input layer vector

Negative Sampling

- Distinguish data from noise $\Rightarrow$ reduce problem to a logistic regression.
- Guess k random samples
- For each pair (w, c) we get:

$$\arg\max_{\theta} \; log(\sigma(v_c^{'\mathsf{T}} v_w) + \sum_{k \in K} log(\sigma(-v_k^{'\mathsf{T}} v_w)) \quad (2)$$

- Uses SGD as an optimizer

State of the Art

- word2vec (Mikolov et al. 2013) [1]
- Parallelizing Word2Vec in Shared and Distributed Memory (Ji et al. 2016)[2]
- Acceleration of Word2vec Using GPUs (Seulki and Youngmin 2016) [3]
- Gensim (Řehůřek and Sojka) [4]

Research Questions:

Can the convergence time of the skip Gram Model be optimized by the use of:

- Advanced optimizers

and

- Input Shuffling

while at the same time maintaining it's accuracy?

Our Implementation

Main Idea:

- Create a large batch of training samples, i.e 2000 pairs
- Compute loss for each pair
- Use sum over all pairs as loss for batch

Implementation

1. Setting
   - Dataset
   - Network Architecture
2. Optimization Process

Dataset

- Text8 dataset
- First 30MB of clean text from wikipedia
- Vocabulary $\approx$ 250k word (small)
- Subsampling $\implies$ 50% decrease of data set size

Optimization process

Optimization techniques:

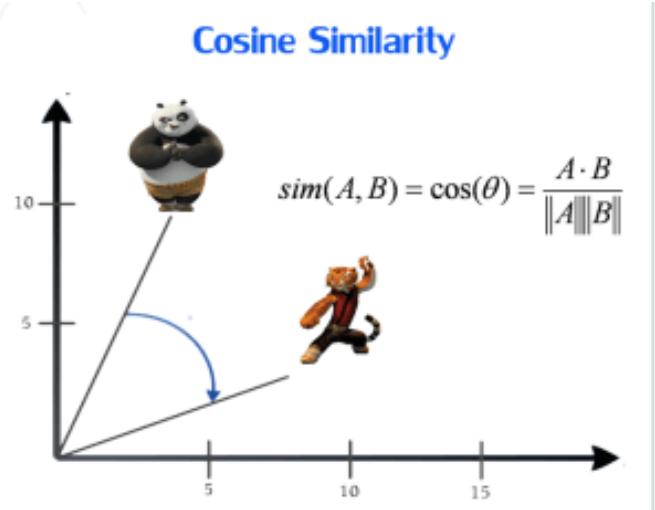
- Advanced Optimizers
  - Momentum
  - Nesterov accellerated Momentum
  - Adagrad
  - Adam
- Input Shuffling

Results

- Rating our work
  - Word similarity
  - Convergence time
- Results
- Discussion
  - Comparison to Gensim and other related work

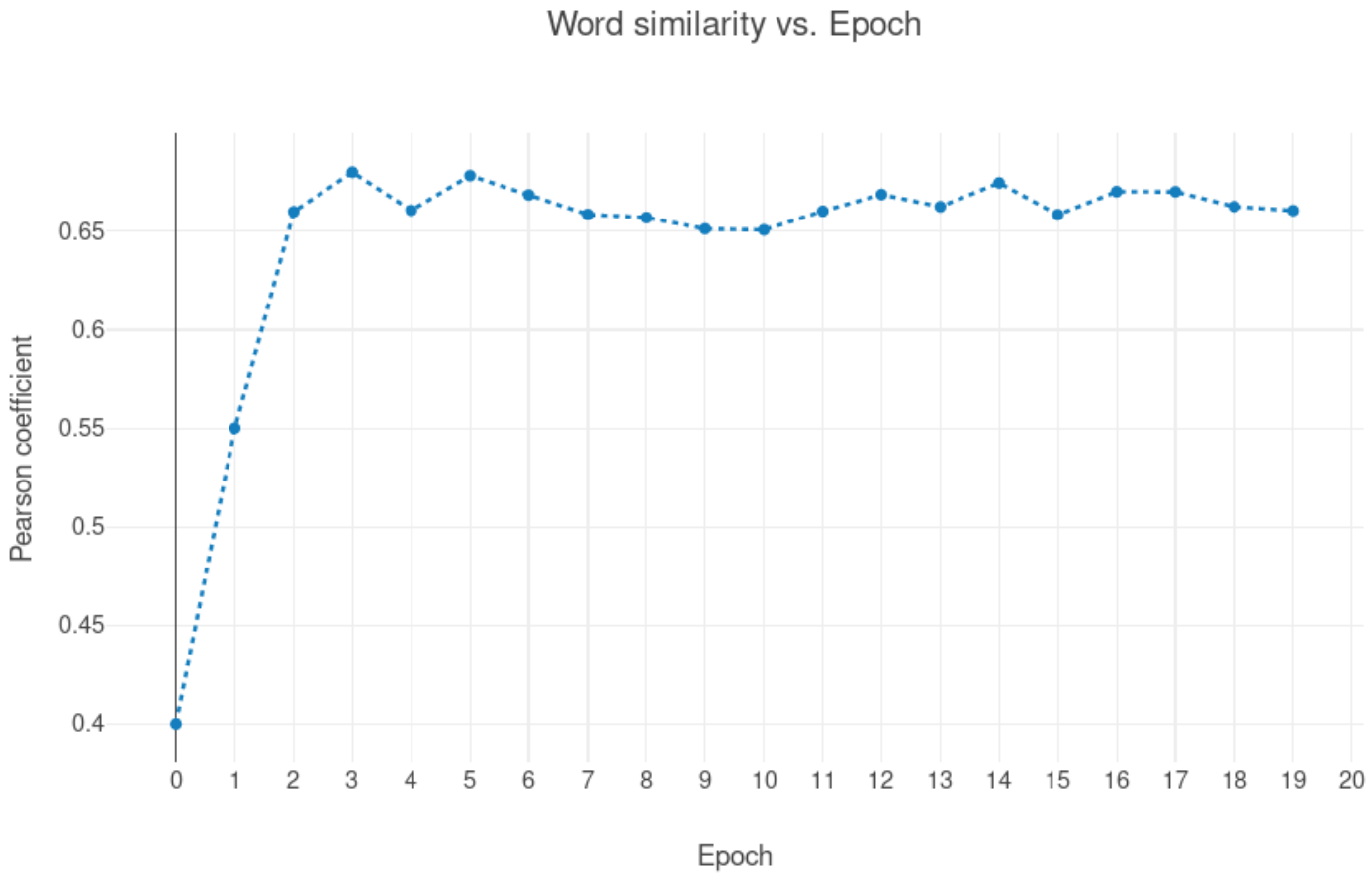
What is word similarity?

- Two word embeddings are close to each to other if their cosine distance is small.
- Pairs of word rated between 1 and 10 on their similarity,
- `['FBI', 'investigation', '8.31', 'Mars', 'scientist', '5.63']`
- We are going to rank our model on the corelation between the distance of the word pairs and the human score.



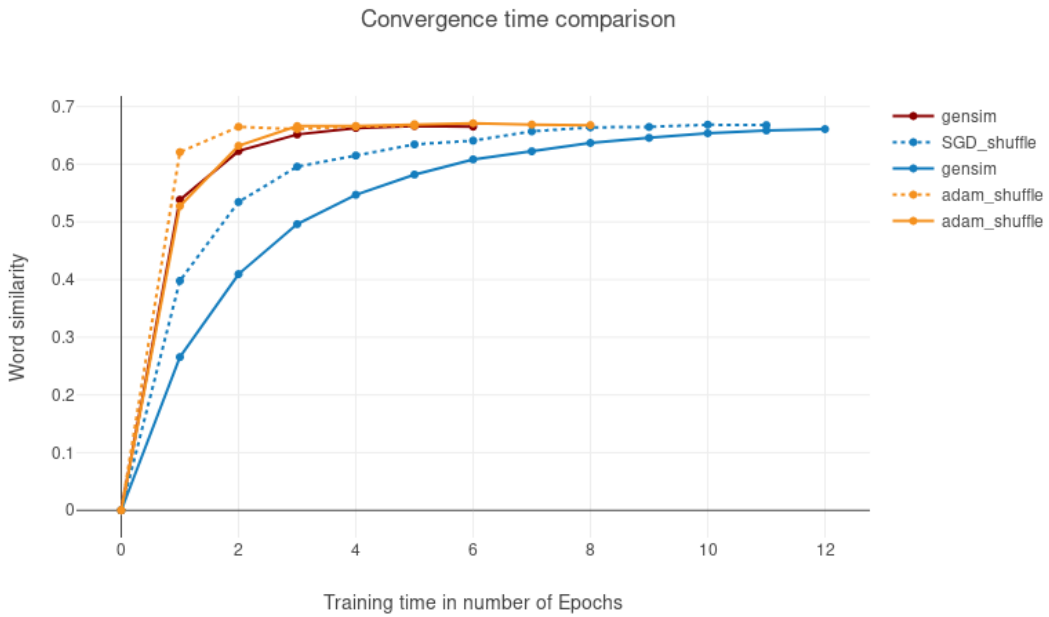
Convergence time

- Defined convergence time based on word similarity
- Early Stoppage if: $\rho - \rho_{prev} < 0.009 \vee \rho > 0.66$
- No more than 20 epochs.



Convergence time vs Gensim

Model	Convergence Time	Word Similarity
SGD	11	0.65
SGD w/shuffling	7	0.66
Adam	3	0.66
Adam w/ shuffling	**2**	0.66
Gensim	4	0.66



Discussion

Questions that arises from the Thesis

- Is the batched version hindering performance?
- Can the results be replicated on other datasets?
- Can the results be replicated on other tasks?

Delete double occurrences

How can we improve the batched approach?

Problem:

Words appear more than once in a batch $\rightarrow$ performance loss

Solution:

Create batch of different sizes, each batch will hold at most one pair per context word

Delete double occurrences

Sentence = The fox jumps over the dog

Example Dictionnary:

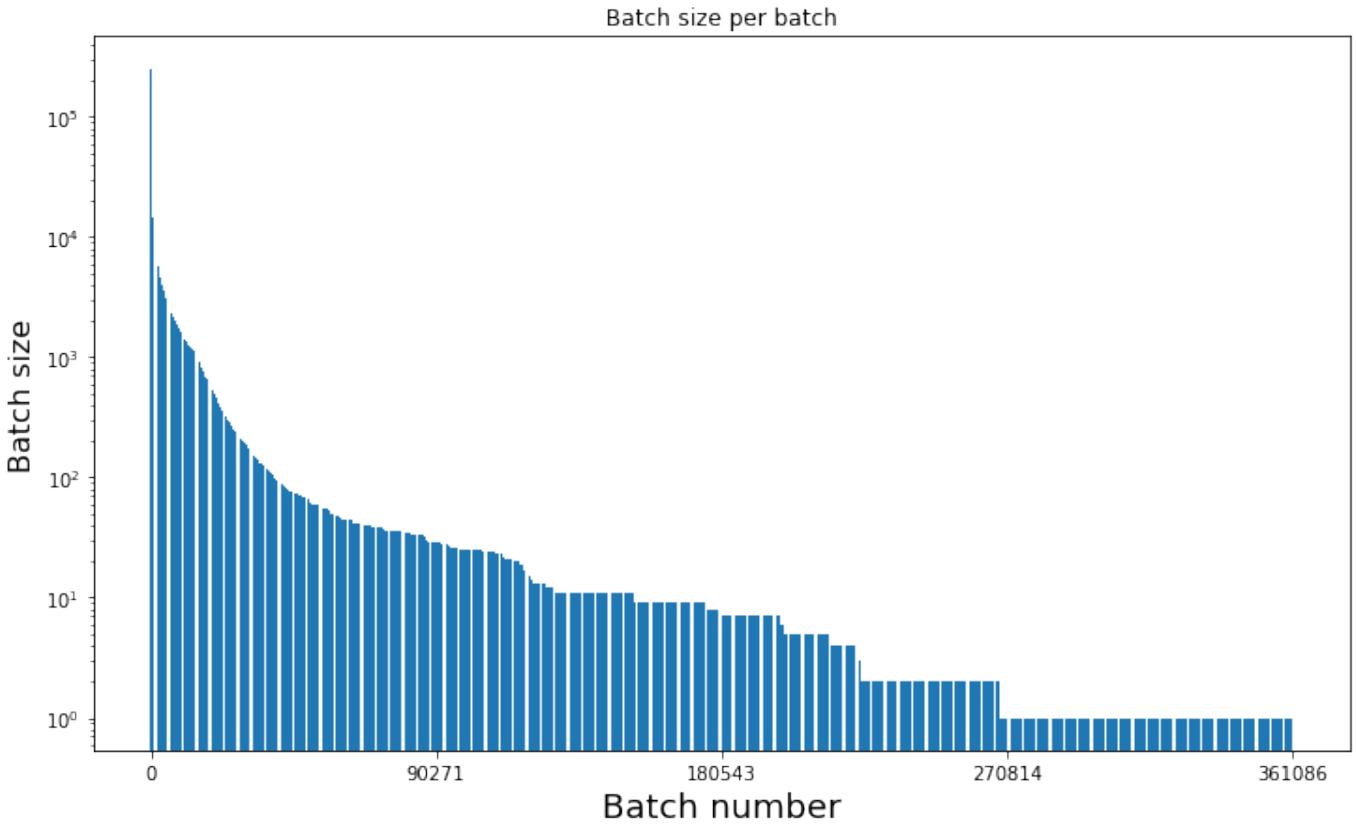
```
{'The': ['fox', 'jumps'],
 'fox': ['jumps', 'The', 'over'],
 'jumps': ['over', 'fox', 'the', 'The'],
 'over': ['the', 'jumps', 'dog', 'fox'],
 'the': ['dog', 'over', 'jumps'],
 'dog': ['the', 'over']})
```

Example Batches

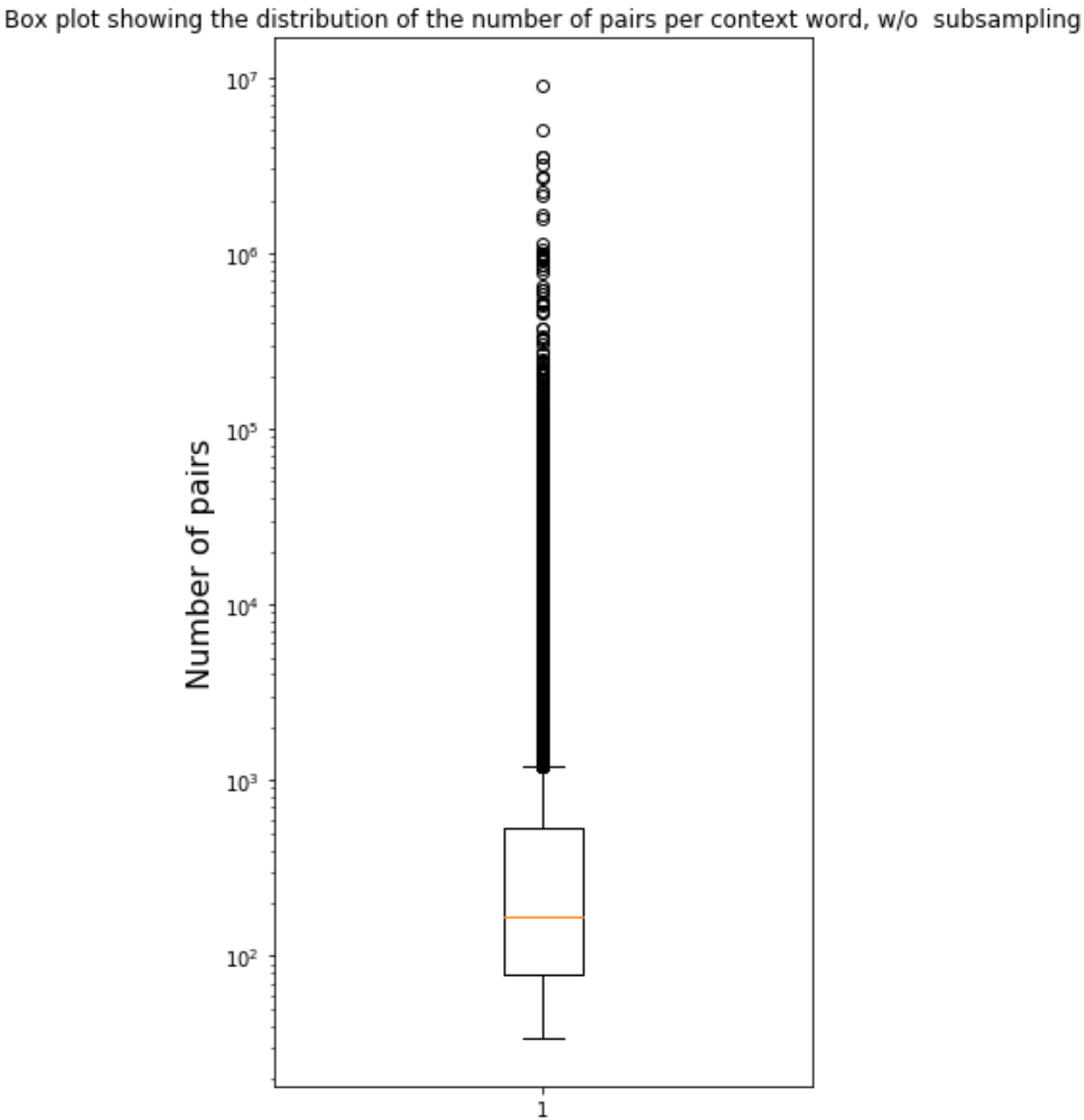
```
Batch 0:('The', 'fox'),('The', 'fox'),
        ('The', 'fox'),('fox', 'jumps'),
        ('fox', 'jumps'),('jumps', 'over'),
        ('jumps', 'over'),('over', 'the'),
        ('over', 'the'),('the', 'dog'),
        ('the', 'dog'),('dog', 'the'),('dog', 'the')
Batch 1:('The', 'jumps'),('The', 'jumps'),
        ('The', 'jumps'),('fox', 'The'),
        ('fox', 'The'),('jumps', 'fox'),
        ('jumps', 'fox'),('over', 'jumps'),
        ('over', 'jumps'),('the', 'over'),
        ('the', 'over'),('dog', 'over'),('dog', 'over')
Batch 2:('fox', 'over'),('fox', 'over'),
        ('fox', 'over'),('jumps', 'the'),
        ('jumps', 'the'),('over', 'dog'),
        ('over', 'dog'),('the', 'jumps'),('the', 'jumps')
Batch 3:('jumps', 'The'),('jumps', 'The'),
        ('jumps', 'The'),('over', 'fox'),('over', 'fox')
```

Delete double occurrences

Problem of the Solution:
Average Batch Size = 200, i.e training takes too long



Distribution of Words



Results of the Distribution

- A few words are responsible for the majority of pairs.
- They almost have the same context words
- Idea: delete outliers from dataset

First Results

Model	Convergence Time	Word Similarity
SGD	11	0.65
SGD w/shuffling	7	0.66
Adam	3	0.66
Adam w/ shuffling	2	0.66
Gensim	4	0.66
Adam w/o outliers	**1**	0.66

Future Work

- Creating the perfect batch
- Analyze the deletion of outliers on other (bigger) datasets.
- Confirm all results on other task

Conclusion

- Skip Gram Model powerful yet simple tool to create word embedings
- Advanced optimizers especially Adagrad and Adam improve convergence time
- Improved convergence time, while maintaining accuracy
- Deletion of outliers is a promising aspect
- Further work includes more testing on different datasets and tasks

References


Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff, 2013, *Distributed representations of words and phrases and their compositionality*


Ji, Shihao and Satish, Nadathur and Li, Sheng and Dubey, Pradeep, 2016, *Parallelizing word2vec in shared and distributed memory*


Bae, Seulki and Yi, Youngmin, 2016, *Acceleration of Word2vec Using GPUs*


Radim Řehůřek and Petr Sojka, 2010, *Software Framework for Topic Modelling with Large Corpora*