Big Data Analytics
using Machine Learning Algorithms

Word2vec - A CBOW and Skip-gram comparative study

Evangelos Stamos

Department of Electrical and Computer Engineering
School of Engineering
University of Thessaly

Examination committee:

Georgios Stamoulis
Proffesor
Department of Electrical
and Computer Engineering

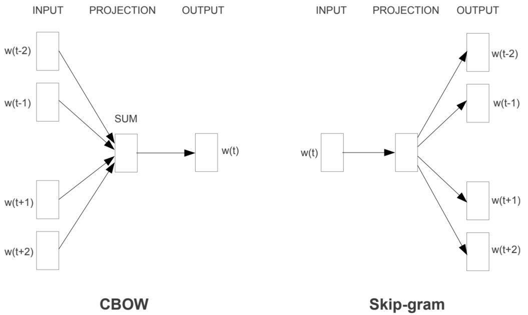
Vassilios Plagianakos
Proffesor
Department of Computer Science
and Biomedical Informatics

Sotirios Tasoulis
Assistant Professor
Department of Computer Science
and Biomedical Informatics

In the course of past decades data has become the world's most valuable asset. In a digital world where society lives on the internet, huge amount of information is globally available to everyone at anytime. NLP can leverage this data and extract valuable information.

[1]

The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector. [2] [3] [4]



- Knowledge Discovery : discovering new chemical compounds with specific properties [1], uncover novel relationships between diseases and disease-genes associations [1]
- Recommendations : user's search history, purchase history, places visits history, click sessions
- Extended in bioinformatics, radiology : creating a dense vector representation of unstructured radiology reports [2]

Dataset
Wikipedia dump

In this thesis comparative study, we use a Wikipedia dump [2] file as input. After evaluating numerous datasets, we select a Wikipedia dump as it is a distinguished, well maintained and carefully composed of vetted wikipedia articles. In addition, the large size corpus of text containing considerably word associations, is the most suitable input for our two models to train. Using gensim and corpora.wikicorpus [3] we construct a corpus from a Wikipedia (or other MediaWiki-based) database dump. Final corpus created after processing is a single .txt file [1] of 1.048.576.002 bytes (1.05 Gigabytes) containing more than 1.7 billion words from Wikipedia articles.

CBOW

Continuous Bag of Words model

The CBOW model predicts a target word based on it's neighboring words. The sum of the context vectors are used to predict the target word. The neighboring words taken into consideration is determined by a pre-defined window size surrounding the target word, which is the maximum distance between the current and predicted word within sentence. Different tasks are served better by different window sizes

Skipgram

The SkipGram model, predicts a word based on a neighboring word. To put it simply, given a word, it learns to predict another word in it's context.

Training I

Word2vec Parameterization

Results of word2vec training can be sensitive to parameterization. In our study, we used Gensim as it is the fastest library for training of vector embeddings. The following are some important parameters in word2vec training.

Gensim parameter	**Tensorflow parameter**	**Type**	**Details**
alpha	learning_rate	float	The initial learning rate
cbow_mean	-	boolean	0: use the sum of the context word vectors 1: use the mean, only applies when cbow is used
epochs	epochs	int	Number of iterations (epochs) over the corpus
hs	-	boolean	0: hierarchical softmax will be used for model training 1: if negative is non-zero, negative sampling will be used
min_count	min_count	int	Maximum distance between the current and predicted word within a sentence
negative	num_neg_samples	int	how many "noise words" should be drawn
sample	subsample	float	The threshold for configuring which higher-frequency words are randomly downsampled
sg	-	boolean	0: CBOW 1: Skipgram
vector_size	embedding_dim	int	Dimensionality of the word vectors
window	window_size	int	Maximum distance between the current and predicted word within a sentence

Training II

Total training time & effective words

Table: Total training time (sec)

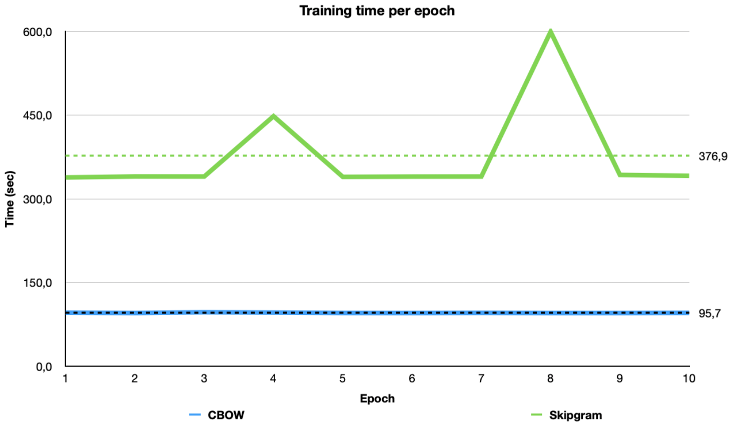
CBOW	**Skipgram**
956.5	3768.5

Table: Total effective words

CBOW	**Skipgram**
1327456338	1327454735

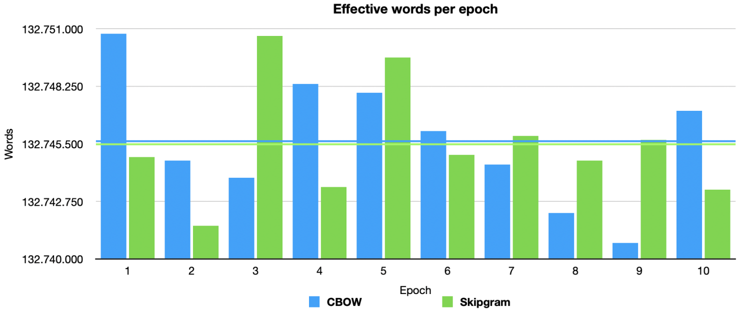
Training III

Time per epoch



Training IV

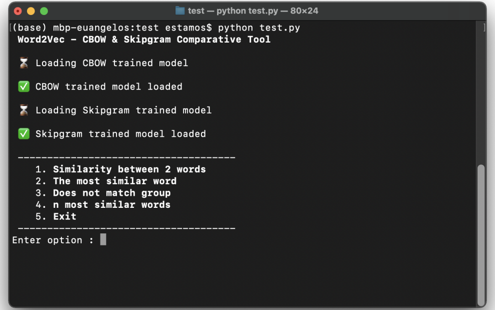
Effective words per epoch



Testing

Comparative tool design & study

Performing a comparative test study between two models presupposes the existence of a reliable, detailed and mutltifunctional tool. Thus, the development of such a tool was imposed as a part of this thesis. A simple, fast and flexible python script was written to simplify behavioural analysis of two models architectures.



Similarity between 2 words

Comparative tool case I

As illustrated above, we are able to compute and print for each algorithm the cosine similarity of 2 given words.

test — python test.py — 80×22

```
 -------------------------------------
Enter option : 1

Similarity between 2 words
------------------------------------
Enter two words separated with spaces : apple orange

The similarity between apple and orange is :

+--------+------------+
|   CBOW |   SKIPGRAM |
+========+============+
| 0.3206 |     0.5384 |
+--------+------------+
 -------------------------------------
    1. Similarity between 2 words
    2. The most similar word
    3. Does not match group
    4. n most similar words
    5. Exit
 -------------------------------------
Enter option :
```

The most similar word

Comparative tool case II.a

Another function of tool is finding the most similar word to a group of words given. In fact we search for the word with the highest cosine index similarity to the word. In test case a we search for the most similar word to 'triangle'.

test — python test.py — 89×20

```
-------------------------------------
Enter a word or words separated with spaces : triangle

The most similar word to  ['triangle']  :
```

CBOW	SKIPGRAM
quadrilateral	escribed
0.7934	0.7995

```
 -------------------------------------
    1. Similarity between 2 words
    2. The most similar word
    3. Does not match group
    4. n most similar words
    5. Exit
 -------------------------------------
Enter option :
```

The most similar word

Comparative tool case II.b

While in test case II.b we search for the word with the most similar context to group of words ['square', 'cycle', 'triangle', 'shape']. We search for the highest indexed word to the given group. Logical expected output would be another shape or a property which most of given shapes have.

```
test — python test.py — 89×20
-------------------------------------
Enter a word or words separated with spaces : square cycle triangle shape

The most similar word to  ['square', 'cycle', 'triangle', 'shape']  :

+--------------+----------------+
| CBOW         | SKIPGRAM       |
+--------------+----------------+
| tetrahedron  | equidiagonal   |
+--------------+----------------+
| 0.7328       | 0.7367         |
+--------------+----------------+
 -------------------------------------
    1. Similarity between 2 words
    2. The most similar word
    3. Does not match group
    4. n most similar words
    5. Exit
 -------------------------------------
Enter option :
```

Does not match group

Comparative tool case III

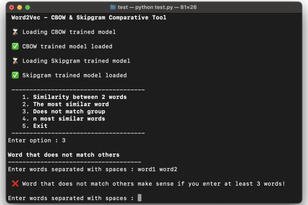
One of the most recent add on functionalities of comparative tool is the 'does not match group'. Given a group of words, must be at least three words, it detects the word which context does not match the others'. Algorithm implies the words which context has the lowest similarity with groups' others words.

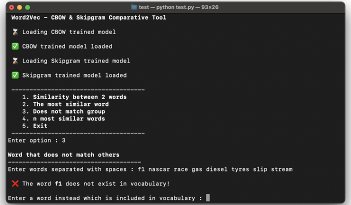
```
test — python test.py — 81×26
    1. Similarity between 2 words
    2. The most similar word
    3. Does not match group
    4. n most similar words
    5. Exit
 -------------------------------------
Enter option : 3

Word that does not match others
-------------------------------------
Enter words separated with spaces : thesis diploma degree airplane

Word that does not group  ['thesis', 'diploma', 'degree', 'airplane']  :
+----------+----------+
| CBOW     | SKIPGRAM |
+----------+----------+
| airplane | airplane |
+----------+----------+
 -------------------------------------
    1. Similarity between 2 words
    2. The most similar word
    3. Does not match group
    4. n most similar words
    5. Exit
 -------------------------------------
Enter option : 
```

Does not match group

Embedded checks and inspections procedures

Since this function makes sense for a group of words consisting of at least of 3 words, we have implemented multiple checks and inspection procedures all embedded in every function of script. More specific, each word given as input from user is checked if it's included in vocabulary both in CBOW's and Skipgram's vocab. If not, user has to enter a new word instead which is included in both vocabs.

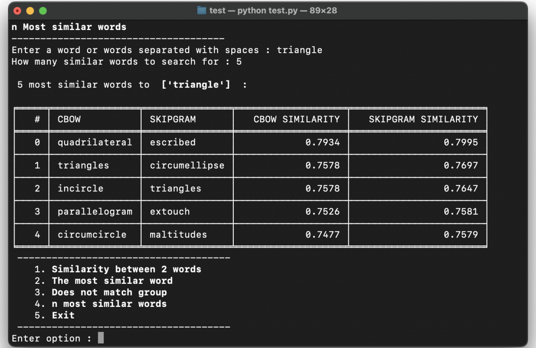




n most similar words

Comparative tool case IV.a

One of the most critical functions of our comparative tool is the n most similar words. Given a word / group of words we can detect the n most similar words along with their similarity. In fact we retrieve the n words with the higher cosine similarity. In test case a we search for the most similar word to 'triangle'.


test — python test.py — 89×28

n Most similar words

Enter a word or words separated with spaces : triangle
How many similar words to search for : 5

5 most similar words to **['triangle']** :

#	CBOW	SKIPGRAM	CBOW SIMILARITY	SKIPGRAM SIMILARITY
0	quadrilateral	escribed	0.7934	0.7995
1	triangles	circumellipse	0.7578	0.7697
2	incircle	triangles	0.7578	0.7647
3	parallelogram	extouch	0.7526	0.7581
4	circumcircle	maltitudes	0.7477	0.7579

1. **Similarity between 2 words**
2. **The most similar word**
3. **Does not match group**
4. **n most similar words**
5. **Exit**

Enter option :

While in test case II.b we search for the word with the most similar context to group of words ['square', 'cycle', 'triangle', 'shape']. Logical expected output would be shapes or a properties which most of given shapes have.

test — python test.py — 89×28

```
n Most similar words
-------------------------------------
Enter a word or words separated with spaces : square cycle triangle shape
How many similar words to search for : 5

 5 most similar words to  ['square', 'cycle', 'triangle', 'shape']  :
```

#	CBOW	SKIPGRAM	CBOW SIMILARITY	SKIPGRAM SIMILARITY
0	tetrahedron	equidiagonal	0.7328	0.7367
1	diagonal	circular	0.7197	0.7212
2	triangular	tangental	0.7061	0.7177
3	rectangle	duocylinder	0.7032	0.7172
4	hexagon	triangular	0.6973	0.7170

```
 -------------------------------------
    1. Similarity between 2 words
    2. The most similar word
    3. Does not match group
    4. n most similar words
    5. Exit
 -------------------------------------
Enter option :
```

Conclusions

Key findings and future work

- Skipgram's training time significantly bigger than CBOW's, justifiable considering the different approach that each architecture implements (In our case study $Skipgram_{tt} \simeq 4 \cdot CBOW_{tt}$)
- Skipgram's behaviour seems to be a hit or miss. In some case studies predicts a neighboring word, while in others a semantically related or commutable word. Based on this observation, for query expansion and synonyms applications CBOW seems as a better fit.
- One very interesting extension of this thesis, would be the behavioural study of word2vec in phrases, and to a relatively new approach in which word2vec technique it is not trained on 1D word embeddings but on multi dimensional data properties, depending on application. [3]

Dataset References

Github — Releases
Dataset — .tar.gz
Accessed: 25-April-2021.

Wikimedia Downloads
Wikipedia Dumps
Accessed: 14-August-2021.

Corpus from a Wikipedia dump
Gensim corpora.wikicorpus
Accessed: 14-August-2021.

References I

- Big data and open data: what's what and why does it matter?
  Open Technologies Alliance (GFOSS)
  Accessed: 11-June-2021.
- Word2vec
  Wikipedia
  Accessed: 24-April-2021.
- Efficient Estimation of Word Representations in Vector Space
  Tomas Mikolov and Kai Chen and Greg Corrado and Jeffrey Dean.
  arXiv: 1301.3781
  Accessed: 11-June-2021
- Distributed Representations of Words and Phrases and their Compositionality.
  Tomas Mikolov and Ilya Sutskever and Kai Chen and Greg Corrado and Jeffrey Dean.
  arXiv: 1310.4546
  Accessed: 11-June-2021

References II

Unsupervised word embeddings capture latent knowledge from materials science literature
Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A. Persson, Gerbrand Ceder & Anubhav Jain
2016
Accessed: 16-August-2021.

Large-Scale Discovery of Disease-Disease and Disease-Gene Associations
Gligorijevic, D., Stojanovic, J., Djuric, N. et al.
2017
Accessed: 16-August-2021.

Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics
Ehsaneddin Asgari,Mohammad R. K. Mofrad
2015
Accessed: 16-August-2021.

Radiology Report Annotation using Intelligent Word Embeddings: Applied to Multi-institutional Chest CT Cohort
Imon Banerjee, Matthew C. Chen, Matthew P. Lungren, and Daniel L. Rubina
2017
Accessed: 16-August-2021.

Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba
Jizhe Wang and Pipei Huang and Huan Zhao and Zhibo Zhang and Binqiang Zhao and Dik Lun Lee.
arXiv: 1803.02349 Accessed: 31-August-2021.

Thank you for listening!

Evangelos Stamos

estamos@e-ce.uth.gr

https://github.com/estamos/word2vec-thesis