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

Big Data Analytics using Machine Learning Algorithms

Word2vec - A CBOW and Skip-gram comparative study

Diploma Thesis

of

EVANGELOS F. STAMOS

Supervisor: Georgios Stamoulis
Professor

Volos, September 2021

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

Big Data Analytics using Machine Learning Algorithms

Word2vec - A CBOW and Skip-gram comparative study

Diploma Thesis

of

EVANGELOS F. STAMOS

Supervisor: Georgios Stamoulis
Professor
Department of Electrical and Computer Engineering
University of Thessaly

A thesis submitted in fulfillment of the requirements for the degree of Diploma in **Electrical and Computer Engineering**.

Approved by the three-members examination committee on 23th September 2021.

(Signature)	*(Signature)*	*(Signature)*
.....	.....	.....
Georgios Stamoulis Professor Department of Electrical and Computer Engineering University of Thessaly	Vassilios Plagianakos Professor Department of Computer Science and Biomedical Informatics University of Thessaly	Sotirios Tasoulis Assistant Professor Department of Computer Science and Biomedical Informatics University of Thessaly

Volos, September 2021

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

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism.

(Signature)

.....
Evangelos F. Stamos

23th September 2021

Περίληψη

Διπλωματική Εργασία

Ανάλυση δεδομένων μεγάλου όγκου
με χρήση αλγορίθμων Μηχανικής Μάθησης

Word2vec - Μία συγκριτική μελέτη των CBOW και Skipgram αρχιτεκτονικών

Στάμος Ευάγγελος

Το Word2vec είναι μια τεχνική για την επεξεργασία φυσικής γλώσσας που περιέχει δύο κύριες διαφορετικές αρχιτεκτονικές την CBOW και την Skipgram. Εκπαιδεύοντάς την σε ένα μεγάλο σώμα κειμένου χρησιμοποιώντας ένα μοντέλο νευρωνικού δικτύου, αντιπροσωπεύει κάθε λέξη με ένα διάνυσμα, όντας ικανή να ανιχνεύσει συνώνυμες λέξεις. Η σημασιολογική ομοιότητα των λέξεων εκφράζεται ως η το συνημιτονιακή ομοιότητα μεταξύ της διανυσματικής αναπαράστασης των λέξεων.

Αυτή η διπλωματική στοχεύει στην διεξαγωγή μίας συγκριτικής μελέτης των δύο διαφορετικών Word2vec αρχιτεκτονικών και στην αξιολόγηση της επίδοσης αυτών σε δεδομένα μεγάλου όγκου.

Λέξεις Κλειδιά

Αλγόριθμοι Μηχανικής Μάθησης, Αξιοποίηση δεδομένων μεγάλου όγκου, Δεδομένα μέγαλου όγκου, Διανυσματική αναπαράσταση λέξεωνν, Διαχείριση και Ανάλυση δεδομένων μεγάλου όγκου, Επεξεργασία φυσικής γλώσσας, Μηχανική Μάθηση, Νευρωνικά Δίκτυα

Abstract

Word2vec is a technique for Natural Language Processing containing two major different architectures continuous bag-of-words (CBOW) and Skipgram. Trained on a large corpus of text using neural network model, representing each word with a vector, it can detect synonymous words. Semantic similarity of words is expressed as the cosine similarity between the vector representation of words.

This diploma thesis aims to the conduct of a comparative study of two different Word2vec architectures and evaluate performance in context of big data input.

Keywords

Big Data, Continuous Bag Of Words, Comparative study, Natural Language Processing, Machine Learning, Machine Learning Algorithms, Neural Networks, Word2vec, Word embeddings, Skipgram

to my family

Acknowledgements

First and foremost, I would like to sincerely thank my supervisor and dean of the School of Engineering of the University of Thessaly Prof. Georgios Stamoulis for giving me the chance to work on this thesis, which is a topic that I desired to work on since the first years of my studies. I would also wish to express my gratitude to my advisors from Department of Computer Science and Biomedical Informatics of University of Thessaly Prof. Vassilios Plagianakos and Asst. Prof. Sotirios Tasoulis for their invaluable support and the motivation that encouraged me to dive into the inspiring field of data science and machine learning.

Furthermore, I am always grateful to my family for their continuous and unconditional love and support. They helped me become a strong independent person, always with the will to discuss, learn and explore.

Finally, I would like to thank my friends, especially all those fellow travellers in this long and many times stormy journey to the fountain of knowledge and achievement.

Volos, September 2021

Evangelos F. Stamos

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

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. Each Internet user generates huge amount of data either text, image or video, mainly through its searches, social media activity. Besides social data, another primary source is machine data which includes is defined as the information which is generated by the activity of computers, mobile phones, embedded systems and other networked devices such as sensors, satellites, road cameras and numerous more. With the wide spread of Internet of things and as data connectivity evolves into 5G, Wi-Fi capabilities expand, and smartphone users grow even larger in population networks, there is no doubt that the "big" in big data will grow even bigger. Society is now transformed into a multinational global community where each member is a continuous data provider.

Collecting data nowadays is not a difficult task, even if searching unique data types and information. The key is searching in the right resources and knowing exactly the type of data needed, but data alone is not enough. It takes an entire system of tools and technology to extract value from data, with many multibillion dollar tech companies in the industry creating state-of-the-art tools produced by their researchers. Academic researchers frame the leaders by suggesting innovative methods to achieve more effective data extraction. Machine Learning and Artificial Intelligence are the two main fields involved into this continuous effort of optimizing and creating new techniques for data analytics.

Natural language processing (NLP) is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data [1]. In NLP, Word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.Word embeddings can be obtained using a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a much lower dimension. [2]

1.1 Thesis Statement and Contributions

This thesis focuses on exploring and experiment the behaviour of Word2vec algorithms (CBOW and Skip-gram), in Big text Data processing, in order to obtain useful knowledge on the time-varying application interaction on the architecture.

1.2 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2 provides background on the Word2vec algorithms, clarifying critical definitions and functionality features. We explain the two different architectures approaches we followed for performing our comparative study, along with each models' collecting data process and transformation procedures.

Chapter 3 presents in details the dataset selection and the methodology we following in building the corpus. We justify our dataset selection, analyzing corpus quality and characteristics.

Chapter 4 describes the methodology we following to train the two different models and explains the parameterization factors that can affect training. We present data for training procedure and we construe behaviour of each model extracting significant insight.

Chapter 5 explores on testing CBOW and Skipgram trained models. We design and present a comparative tool which we use to perform a multiple aspect study on how the two models behave on various word embeddings.

Finally, Chapter 6 concludes this thesis by discussing our key findings and by presenting some directions for future work.

Chapter 2

Word2vec

Word2vec is a technique for Natural Language Processing created in 2013 by a team of researchers led by Tomas Mikolov at Google over two papers. [3] [4] 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. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors. [5] Word2vec contains two different model architectures to produce a distributed representation of words: CBOW (continuous bag-of-words) or Skipgram (continuous skip-gram).

2.1 Applications

The Word2Vec model is used to extract the notion of relatedness across words or products such as semantic relatedness, synonym detection, concept categorization, selectional preferences, and analogy. [6] While Word2vec method is widely used in a large scale field of applications, it is highly interwoven with knowledge discovery and recommendation problems. [7]

2.1.1 Knowledge discovery

Word2Vec models can be trained over a large number of documents and find hidden relations among elements of those documents. In 2019, a paper published in Nature [8] that introduced a novel Word2Vec model to discover new chemical compound with specific properties. They trained a Word2Vec model on over three million abstracts from scientific papers published in a hundred years, across a thousand journals. The model helped them to extract relations and analogies among chemical compounds.

2.1.2 Recommender Systems

Word2vec models are not limited to textual data and they work for any data type that comes in a sequence such as users' click sessions, search history, and purchase history. You can use this data to build powerful recommender systems that can increase online

business profits by enhancing the CTR and conversions. You can create a fixed-sized vector representation for different items such as places or products that capture human-like relations and similarities between them. Airbnb trained Word2Vec models on user click and conversive (the final session in a booking procedure) sessions and use them to create business values.

2.1.3 Radiology

An extension of word vectors for creating a dense vector representation of unstructured radiology reports has been proposed by Banerjee et al.[9] One of the biggest challenges with Word2Vec is how to handle unknown or out-of-vocabulary (OOV) words and morphologically similar words. This can particularly be an issue in domains like medicine where synonyms and related words can be used depending on the preferred style of radiologist, and words may have been used infrequently in a large corpus. If the word2vec model has not encountered a particular word before, it will be forced to use a random vector, which is generally far from its ideal representation. [5]

2.1.4 Biological sequences (BioVectors)

An extension of word vectors for n-grams in biological sequences (e.g. DNA, RNA, and Proteins) for bioinformatics applications has been proposed by Asgari and Mofrad. [10] Named bio-vectors (BioVec) to refer to biological sequences in general with protein-vectors (ProtVec) for proteins (amino-acid sequences) and gene-vectors (GeneVec) for gene sequences, this representation can be widely used in applications of machine learning in proteomics and genomics. The results suggest that BioVectors can characterize biological sequences in terms of biochemical and biophysical interpretations of the underlying patterns. [5]

2.2 CBOW

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 a sentence. Different tasks are served better by different window sizes. [11]

2.3 Skip-gram

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. In a bijective function, given a single input word it predicts one output words for each word included in the vocabulary of dataset.

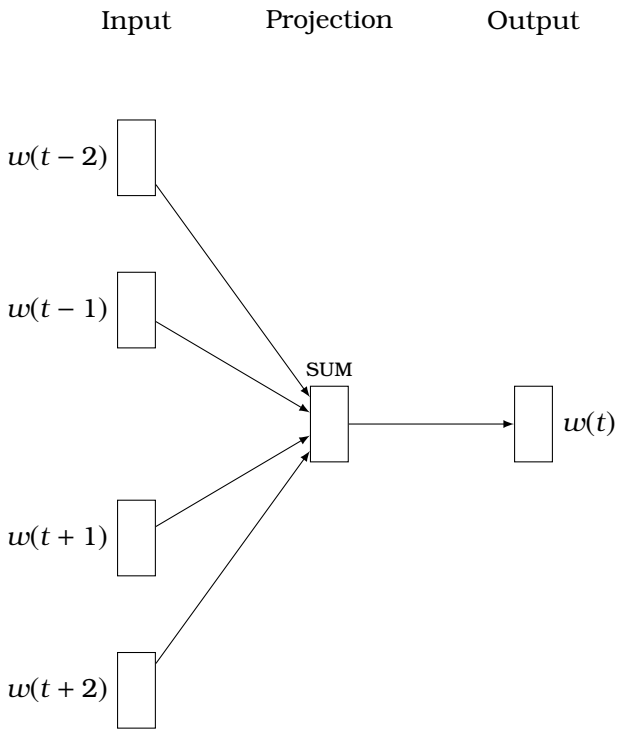

Figure 2.1. *CBOW's Architecture*

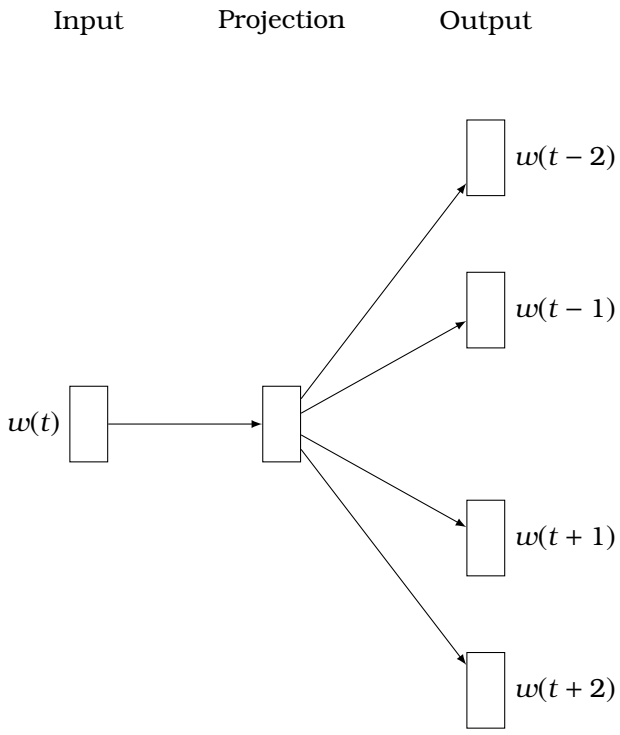

Figure 2.2. *Skipgram's Architecture*

Chapter 3

Dataset

In this thesis comparative study, we use a Wikipedia dump 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. Dataset provided by Wikipedia is in XML format and the structure is quite complex. A dump refers to a periodic snapshot of a database. Dataset includes a big number of English Wikipedia articles.

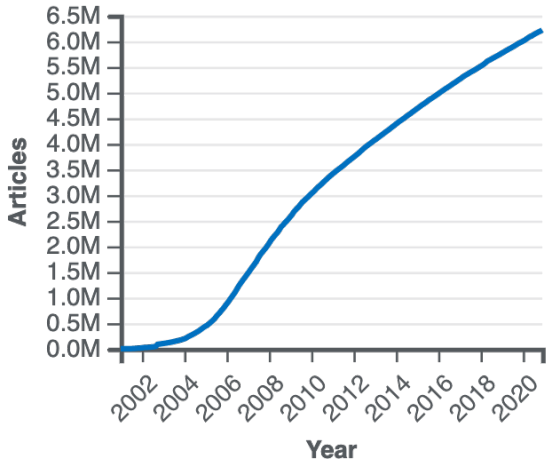

Figure 3.1. *Number of English Wikipedia articles*

3.1 Corpus build

Using gensim and corpora.wikicorpus we construct a corpus from a Wikipedia (or other MediaWiki-based) database dump. Final corpus created after processing is a single .txt file of 1.048.576.002 bytes (1.05 Gigabytes) containing more than 1.7 billion words from Wikipedia articles.

```
1 import sys
2 from gensim.corpora import WikiCorpus
3
4 def make_corpus(in_f, out_f):
5
```

```
  # Convert Wikipedia xml dump file to text corpus

  output = open(out_f, 'w')
  wiki = WikiCorpus(in_f)

  i = 0
  for text in wiki.get_texts():
    output.write(bytes(' '.join(text), 'utf-8').decode('utf-8') + '\n')
    i = i + 1
    if (i % 10000 == 0):
      print('Processed ' + str(i) + ' articles')
  output.close()
  print('Processing complete!')

if __name__ == '__main__':

  if len(sys.argv) != 3:
    print('Usage: python make_wiki_corpus.py <wikipedia_dump_file> <processed_text_file>')
    sys.exit(1)
  in_f = sys.argv[1]
  out_f = sys.argv[2]
  make_corpus(in_f, out_f)
```

Listing 3.1: Corpus build from Dataset

Chapter 4

Training

4.1 Parameterization

Results of word2vec training can be sensitive to parameterization. The following are some important parameters in word2vec training.

Table 4.1. *Word2vec Parameterization*

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

Table 4.2. *Total training time (sec)*

CBOW	Skipgram
956.5	3768.5

Table 4.3. *Total effective words*

CBOW	Skipgram
1327456338	1327454735

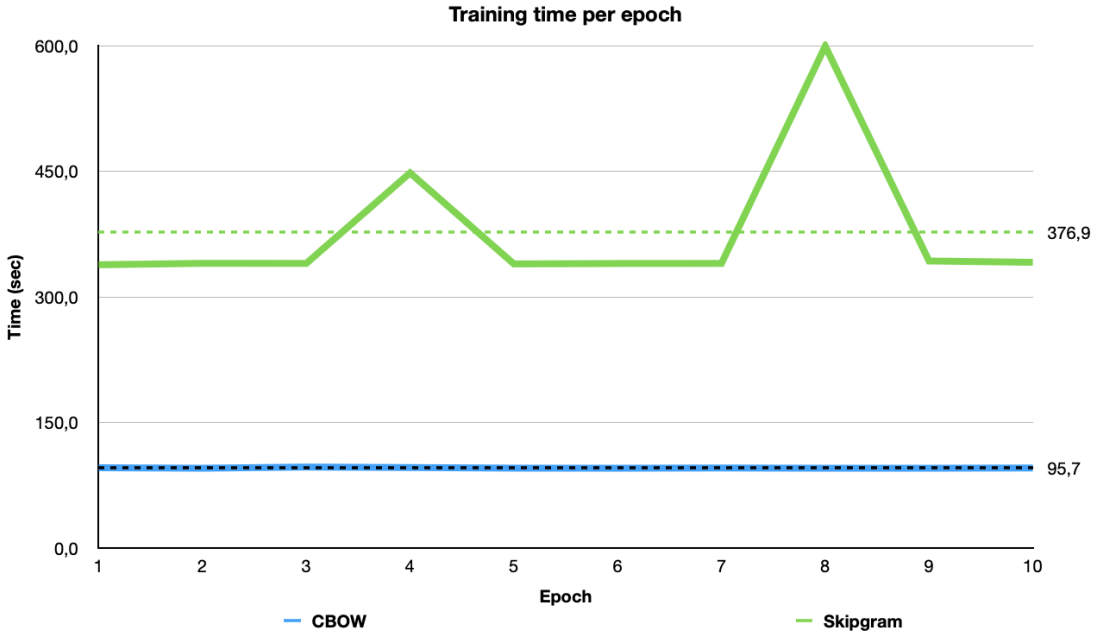

Figure 4.1. *Training time per epoch graph*

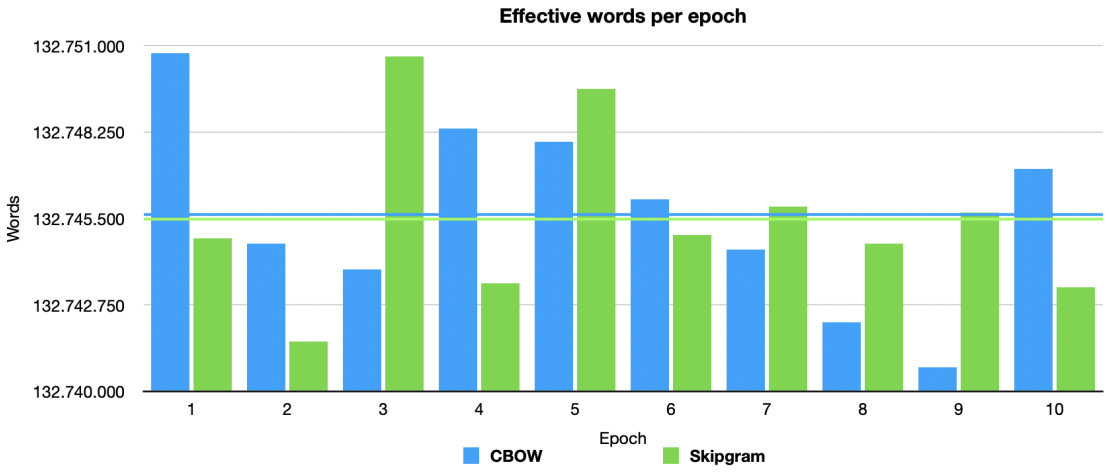

Figure 4.2. *Effective words per epoch graph*

Table 4.4. *Training time per epoch (sec)*

Epoch	CBOW	Skipgram
Average	95.65	376.85
1	95.9	338.3
2	95.3	340.0
3	96.7	339.9
4	96.1	448.0
5	95.4	339.3
6	95.3	339.8
7	95.6	339.9
8	95.3	599.3
9	95.3	342.8
10	95.6	341.2

Table 4.5. *Effective words per epoch*

Epoch	CBOW	Skipgram
Average	132745634	132745474
1	132750757	132744876
2	132744712	132741580
3	132743879	132750658
4	132748376	132743435
5	132747942	132749631
6	132746112	132744974
7	132744511	132745877
8	132742194	132744706
9	132740767	132745693
10	132747088	132743305

Chapter 5

Testing

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.

5.1 Models Comparative Tool

Models Comparative Tool was designed so it can load any model stored in gensim. It's easy to use, it has a simple, minimalistic user interface and visualizes Current version supports 4 different types of comparative functions, which are the main and most critical case to study. It has integrated various input validations check, such as checking if input words are included into vocabulary or if not then the use must enter a valid word.

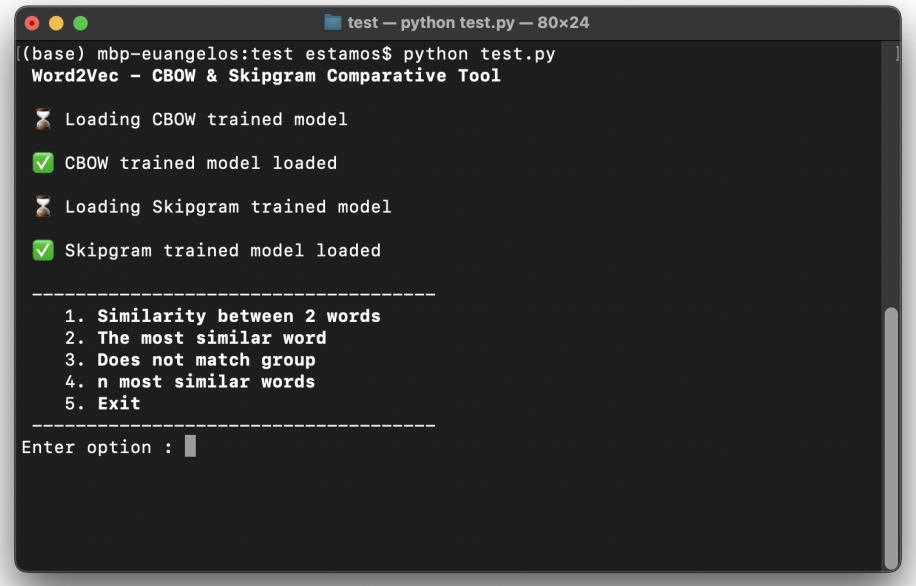

Figure 5.1. *Models Comparative Tool Menu*

5.1.1 Similarity of two words

First function of tool is finding the similarity of two words given. An example use in which we test the similarity of words 'apple' and 'orange' is illustrated in figure 5.2. Tool calculates the similarity for each model with the command above and then we pretty print them.

```
cbow.wv.similarity('apple', 'orange')       # Calculating similarity for CBOW model
skipgram.wv.similarity('apple', 'orange')   # Calculating similarity for Skipgram model
```

Listing 5.1: Calculate similarity of two words

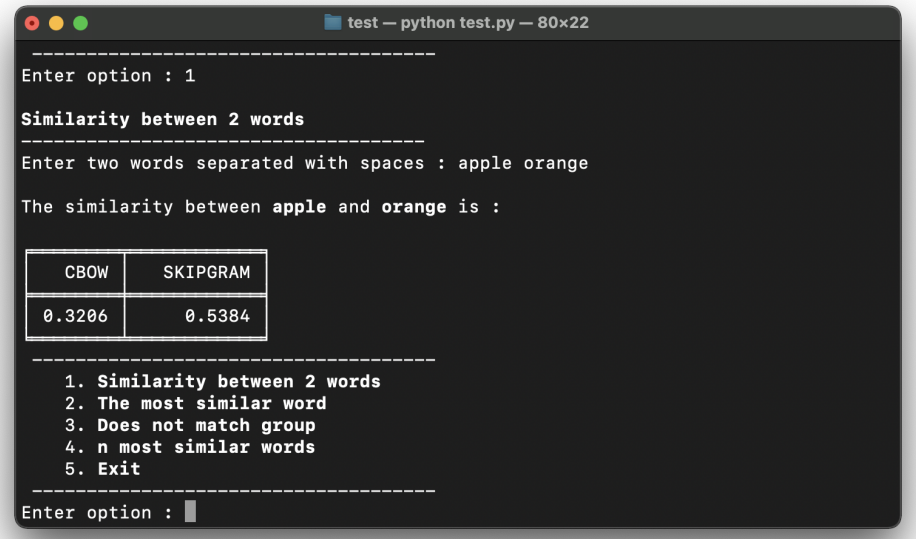

Figure 5.2. *Similarity between words 'apple' and 'orange'*

5.1.2 The most similar word

Another function of tool is finding the most similar word to a group of words given. An example use in which we search the most similar word of word 'triangle' is illustrated in figure 5.3, while in figure 5.4 is illustrated the most similar word of group of words ['square', 'cycle', 'triangle' and 'shape']. Tool calculates the similarity for each model with the command above and then we pretty print them.

```
# Calculating most similar word for input words string for CBOW model
cbow.wv.most_similar(positive = words, topn = 1)[0][0]
# Calculating most similar word for input words string for Skipgram model
skipgram.wv.most_similar(positive = word, topn = 1)[0][0]]
# Calculating similarity for CBOW model
round(cbow.wv.most_similar(positive = word, topn = 1)[0][1], PRECISION)
# Calculating similarity for Skipgram model
round(skipgram.wv.most_similar(positive = word, topn = 1)[0][1], PRECISION)
```

Listing 5.2: Calculate the most similar word

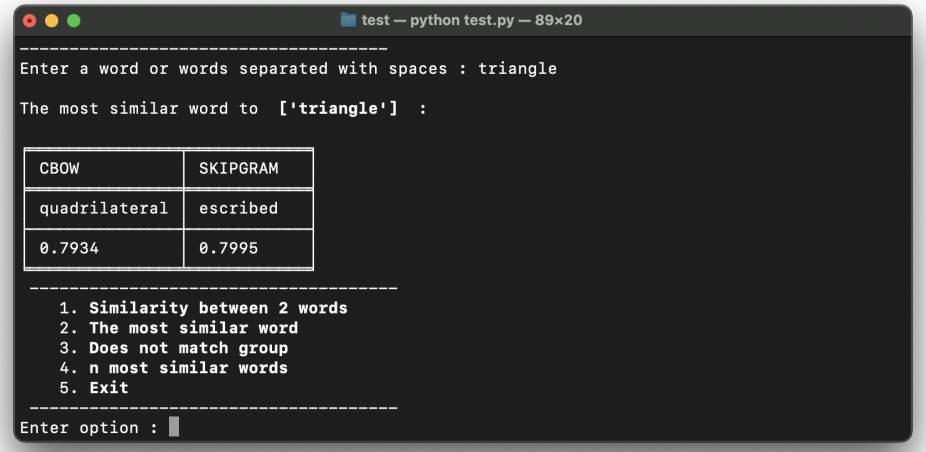

Figure 5.3. *The most similar word to word 'triangle'*

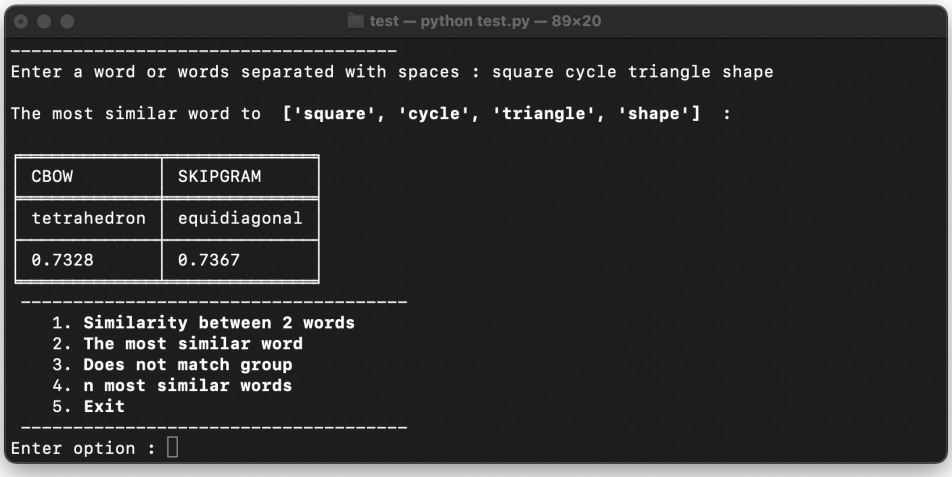

Figure 5.4. *The most similar word to group of words ['square', 'cycle', 'triangle' and 'shape']*

5.1.3 Does not match group

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.

```
# Note that in variable words we have already passed and split with space the group of words
# Calculating and pretty-printing the word that does not match group of words
print(tabulate([["CBOW","SKIPGRAM"],
    [cbow.wv.doesnt_match(words), skipgram.wv.doesnt_match(words)]],
    headers="firstrow", tablefmt="fancy_grid"))
```

Listing 5.3: Calculate the word the does not match group of words

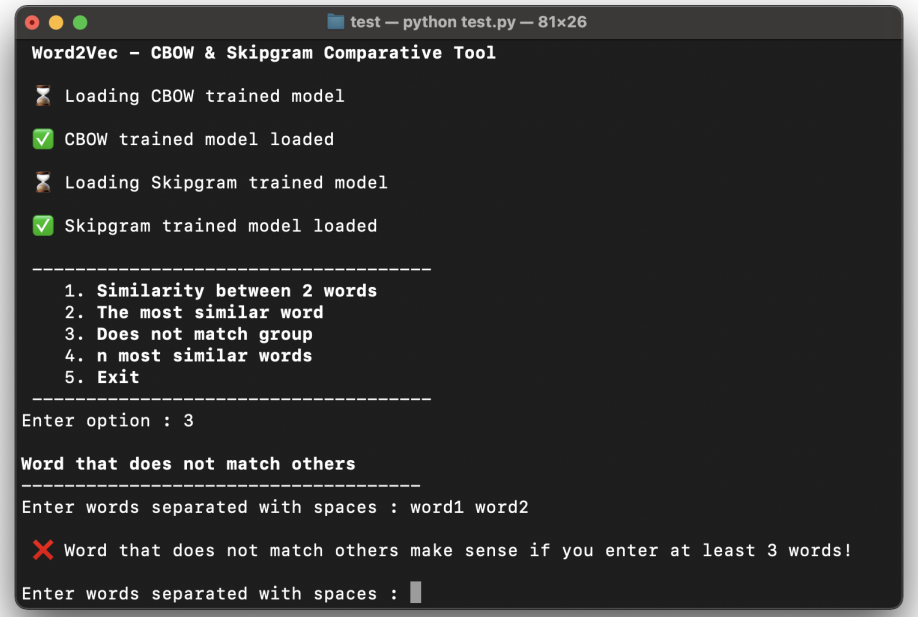

Figure 5.5. *Does not match error message for 2 words input*

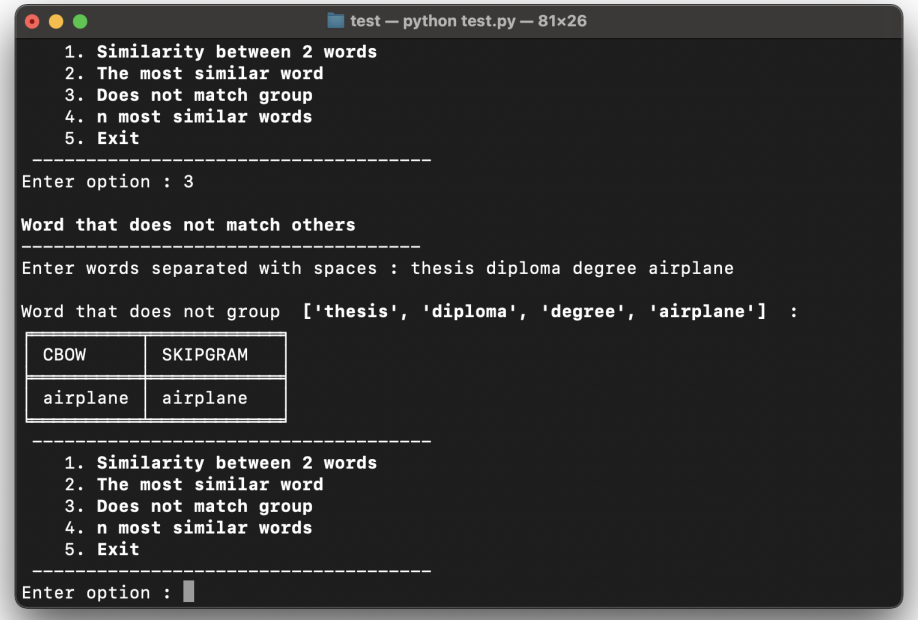

Figure 5.6. *Does not match word for group of words ['thesis', 'diploma', 'degree', 'airplane']*

5.1.4 n most similar words

```
1 # Calculating n most similar word for input words string for CBOW     for i in range(0, n):
2     table.append([cbow.wv.most_similar(words, topn = n)[i][0],
```

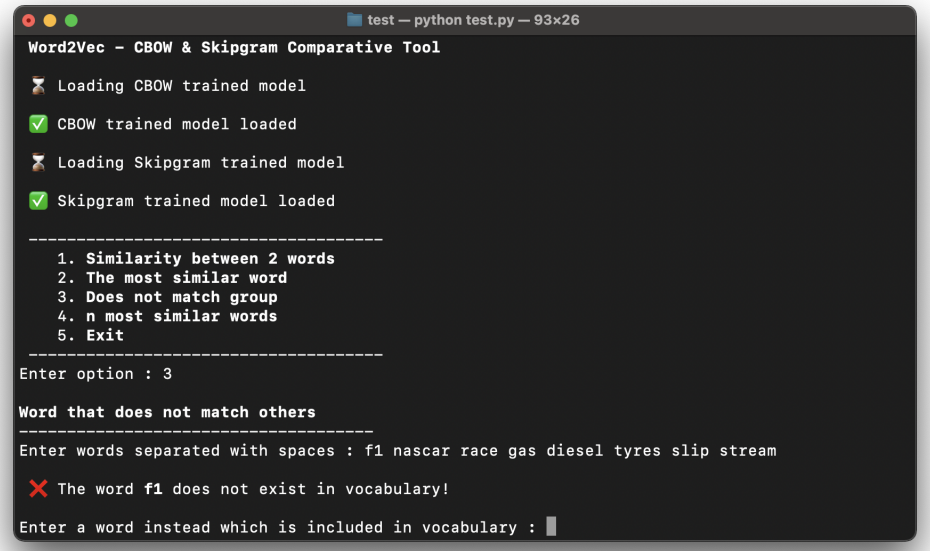

Figure 5.7. *Does not match word error message since word 'f1' is not included into vocabulary*

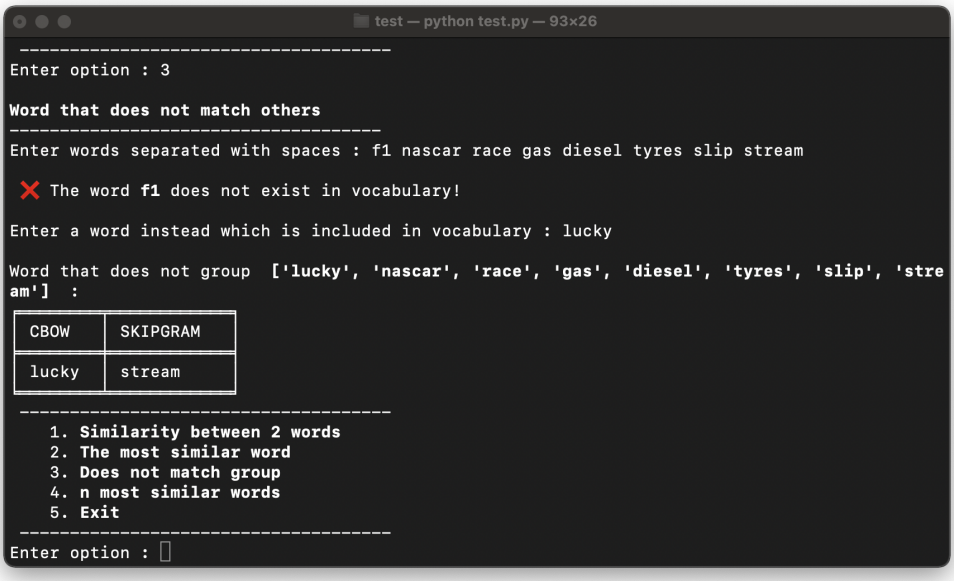

Figure 5.8. *Does not match word for group of words ['lucky', 'nascar', 'race', 'gas', 'diesel', 'tyres', 'slip', 'stream']*

```
3                 skipgram.wv.most_similar(words, topn = n)[i][0],
4                 cbow.wv.most_similar(words, topn = n)[i][1],
5                 skipgram.wv.most_similar(words, topn =n)[i][1]])
```

Listing 5.4: Calculate n most similar words

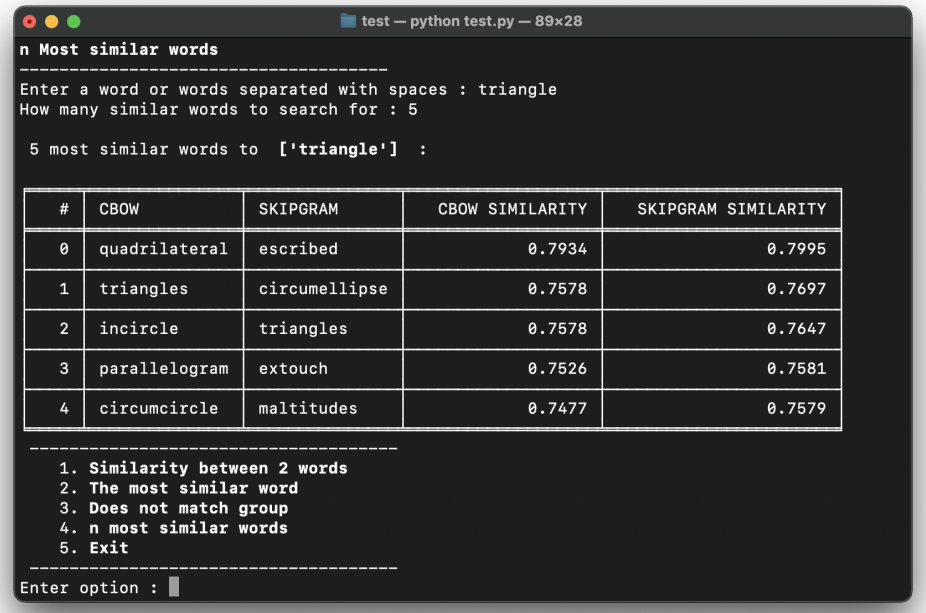

Figure 5.9. *5 most similar word to word 'triangle'*

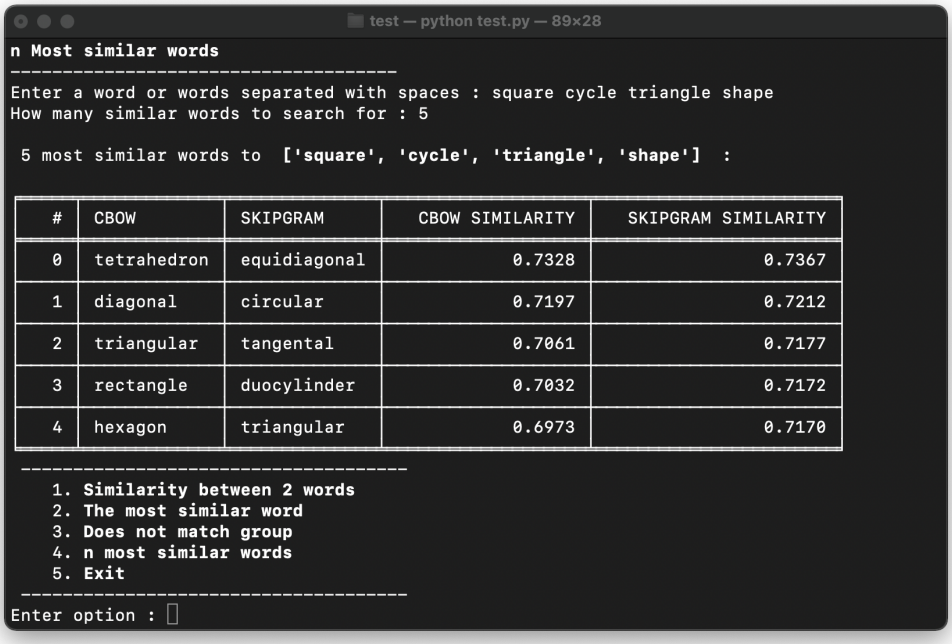

Figure 5.10. *5 most similar word to words 'square', 'cycle', 'triangle' and 'shape'*

Chapter 6

Conclusions and future work

6.1 Conclusions

Summarizing and analysing our comparative study's major points we can extract various critical conclusions. First of all, it is evident that there is a fair difference in training time of two architectures. In our case study, CBOW's training took ~16 mins, while Skipgram's training time lasted ~1 hr and 3 mins. Skipgram's training time is 4 times CBOW as expressed in equation 6.1. This disparity in training times of two compared architectures is totally justifiable, considering the different approach that each architecture have. Skipgram performs a significantly higher training time since for each word included in the vocabulary of dataset it learns to predict another word in it's context.

$$Skipgram_{trainingtime} \simeq 4 \cdot CBOW_{trainingtime} \tag{6.1}$$

Skipgram's behaviour seems to be a success 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.

6.2 Future work

In this thesis a big step was taken in terms of creation of a architectures comparative tool and evaluation of standard word2vec algorithms CBOW and Skipgram. However, there are still some steps that need to be taken in the implementation and the evaluation stage before it becomes a global way of performing a fully comparative study of two or more different architectures. First, we need to make a more extensive evaluation of the algorithms. We need to include more algorithm's parameters and make observations to which parameter or parameters each algorithm is more sensitive. Also in future experiments, the number of data processed could be remarkable larger premising a much more capable processing power.

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 which may include properties such as color, brand, size, weight depending on application. [12]

The code of this thesis, along with dataset and logs data can be found on Github.

Bibliography

[1] *Wikipedia Natural language processing*. https://en.wikipedia.org/wiki/Natural_language_processing. Accessed: 24-April-2021.

[2] *Wikipedia Word embedding*. https://en.wikipedia.org/wiki/Word_embedding. Accessed: 24-April-2021.

[3] Tomas Mikolov, Kai Chen, Greg Corrado και Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*, 2013.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado και Jeffrey Dean. *Distributed Representations of Words and Phrases and their Compositionality*, 2013.

[5] *Wikipedia Word2vec*. https://en.wikipedia.org/wiki/Word2vec. Accessed: 24-April-2021.

[6] Pedram Ataee. *Word2Vec Models are Simple Yet Revolutionary*. https://towardsdatascience.com/word2vec-models-are-simple-yet-revolutionary-de1fef544b87, 2020. Accessed: 14-August-2021.

[7] Djuric N. et al. Gligorijevic D., Stojanovic J. *Large-Scale Discovery of Disease-Disease and Disease-Gene Associations*. https://doi.org/10.1038/srep32404, 2016. Accessed: 16-August-2021.

[8] Weston L. et al. Tshitoyan V., Dagdelen J. *Unsupervised word embeddings capture latent knowledge from materials science literature*. https://doi.org/10.1038/s41586-019-1335-8, 2020. Accessed: 16-August-2021.

[9] Rubin DL Banerjee I Chen MC, Lungren MP. *Radiology report annotation using intelligent word embeddings: Applied to multi-institutional chest CT cohort*. https://doi.org/10.1016/j.jbi.2017.11.012, 2018. Accessed: 16-August-2021.

[10] Mohammad R. K. Mofrad Ehsaneddin Asgari. *Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics*. https://doi.org/10.1371/journal.pone.0141287, 2015. Accessed: 16-August-2021.

[11] Kavita Ganesan. *Word2Vec: A Comparison Between CBOW, SkipGram & SkipGramSI*. https://kavita-ganesan.com/comparison-between-cbow-skipgram-subword, 2020. Accessed: 01-May-2021.

[12] Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao και Dik Lun Lee. *Billion-scale Commodity Embedding for E-commerce Recommendation in Alibaba*, 2018.

List of Abbreviations

CBOW	Continuous Bag of Words
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network