Miguel Sequeira de Oliveira Bernardo

Bachelor Degree in Sciences of Engineering Physics

Construction of Geometries Based on Automatic Text Interpretation

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Engineering Physics

Advisers:	Professor Dr. Isabel Catarino, Assistant Professor at NOVA University of Lisbon
	Professor Dr. João Pires da Cruz, Partner at Closer Consulting, Invited Professor at University of Lisbon

Jury

Chair:	Professor Dr. Maria Raposo
Examiner:	Professor Dr. André Wemans
Supervisor:	Professor Dr. Joao Pires da Cruz

November, 2020

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Construction of Geometries
Based on Automatic Text Interpretation

"Quando a Educação não é libertadora, o sonho do oprimido é ser o opressor!"
- Paulo Freire

Acknowledgements

Throughout the making of this thesis, the help of a few people was essential to me and this work. Thus, I wish to express my deepest gratitude to all who took part and help me finish this chapter.

I would first like to express my gratitude and appreciation to my adviser, professor Isabel Catarino, whose passion and dedication shaped me and all students who had the opportunity of interacting with her. Your valuable guidance and patience throughout my studies were truly vital and your will to improve our school, although a Dantesque task, is inspiring. And for all this, thank you!

For the opportunity of embarking in this project and the incredible help through the thesis, I would like to say a personal thank you to my adviser João Cruz.

For the warm welcome, I want to thank all my colleagues from Closer. I would particularly like to single out Ali and Barroca for their guidance and patience during my mental blockades.

An extraordinary thankfulness goes to FCT/UNL's students union and all the people whom I worked with, for giving me the strength to grow throughout the three years I had the pleasure to work there.

I would like to say a special thank you to all my colleagues from the Integrated Master in Physics Engineering for all the companionship throughout these last five years. Especially an enormous acknowledgement to Ana Isabel Francisco, Ana Claudia Onofre e Eduardo Freitas for all the moments we have shared.

To all my friends and family, from the bottom of my heart, I want to thank you for accompanying me through this journey. Especially to Maria João, Beatriz and Sara, for the marvellous company during this pandemic, and Bruno, for forcing me to work hard to conclude this work on time.

I couldn't finish without thanking my amazing parents for all their sacrifices and patience that granted my sister and me all opportunities and privileges throughout our lives. Thank you so much!

Abstract

When dealing with expanding systems, like the universe or the economy, due to its constant expansion, the statistical error is remarkably high to allow an understanding of the behaviour of the system. Thus, arises the necessity to transform an expanding system into a more straightforward system to work with. In order to address the problems, a geometric word space was constructed based on automatic text interpretation. News articles and economic reports about the European Union were collected and, using Python scripts, were cleaned and used to train a Word2Vec model. The trained model created multi-dimensional word spaces from three periods of the last decades, a pre-2000 period, a 2000-2008 period and a post-2008 period. After the interpretation of the created word spaces, a difference in behaviour between the country names was noticed. All the European Union member states were getting closer to each other until 2008, but after that year there was an abrupt rupture in this trend and ever country drifted apart. This behaviour can be linked with the 2008 financial crisis, though more research is needed to confirm this behaviour and hopefully found other correlations connecting the word spaces behaviour and the real world. An improvement in the quantity and quality of the corpus will certainly improve the accuracy of the world space and enable a better understanding of the word spaces behaviour.

Keywords: Text interpretation, Word space geometry, Word2Vec, Data Science, Economy, Python.

Resumo

Ao trabalhar com sistemas em expansão, como o universo ou a economia, o erro estatístico, decorrente da sua dilatação, é demasiado alto para permitir a compreensão dos comportamentos do sistema. Daí surgiu a necessidade de transformar um sistema em expansão num sistema cuja compreensão fosse mais acessível. Para resolver este problema, foi criado um espaço geométrico constituído por palavras através da interpretação automática de texto. Foram recolhidas noticias e relatórios sobre a situação económica da União Europeia e, usando scripts escritos em Python, foram limpos e utilizados para treinar um modelo de Word2Vec. O modelo treinado de Word2Vec criou três espaços multidimensionais constituídos por palavras em períodos diferentes. Um dos espaços foi construído com dados anteriores a 2000, outro com dados entre 2000 e 2008 e por último um com dados posteriores a 2008. Após a interpretação dos espaços criados, foi evidente uma grande mudança de comportamento entre os objetos que representam os nomes dos países. Todos os nomes dos estados membros da União Europeia estavam a aproximar-se até ao ano de 2008, no entanto, após esse ano, este comportamento susteve-se abruptamente e todos os países se afastaram. Este comportamento poderá estar ligado com a crise financeira de 2008, no entanto é necessária mais investigação para confirmar este comportamento e encontrar mais correlações entre o espaço criado e o mundo real. Um aumento na quantidade e qualidade da coleção de textos irá certamente melhorar a precisão na construção do espaço e contribuir para uma melhor compreensão dos comportamentos dos espaços criados.

Palavras-chave: Interpretação de texto, Geometria de espaços formados por palavras, Word2Vec, Data Science, Economia, Python.

Contents

List of Figures

List of Tables

Acronyms

CBOW -	Continuous Bag Of Words
ECU -	European Central Bank
EU -	European Union
IILS -	International Institute for Labour Studies
IMF -	International Monetary Fund
LDA -	Latent Dirichlet Allocation
NLP -	Natural Language Processing
NLTK -	Natural Language Toolkit
NN -	Artificial Neural Networks
SNE -	Stochastic Neighbor Embedding
SG -	Skip-gram Model
SGNS -	Skip-gram Model With Negative Sampling
SVD -	Singular Value Decomposition
t-SNE -	t-Distributed Stochastic Neighbor Embedding

CHAPTER 1

Introduction

Having in mind disastrous events for the world economy that occurred in the past as the Great Depression in the '30s and the Great Recession in 2008, produced by reckless conduct and feeble knowledge of the performance of the economy [1], we can recognize the importance of a fair comprehension about the economy's behaviour and its effects. In some reflection studies regarding these two major crises, it is revealed that there is some correlation between them, like the rapid decline in global manufacturing in the subsequent months of the global peaks in industrial production [2].

In the work of Luc Laeven and Fabian Valencia [3], is proposed that in the midst of numerous causes of financial crises, the leading and most impactful of them are "combination of unsustainable macroeconomic policies (including large current-account deficits and unsustainable public debt), excessive credit booms, large capital inflows, and balance sheet fragilities, combined with policy paralysis due to a variety of political and economic constraints." Furthermore, these are exciting factors that they used to identify hundreds of systematic banking crises between the '70s and 2007.

In "Literature review of past crises" produced by the IILS [1], even though it is first described as extremely difficult to identify a financial crisis with certainty, several signs are pointed out that can signify their imminent occurrence. These can be the presence of widespread bank runs, bank failure or bank insolvencies, sudden stops in the inflow of foreign capital [4], credit booms [5] or significant "loss in the value of important classes of assets such as government debt [6], stocks and housing [4]. All these signs can be useful tools to understand and predict identical crises like the ones that happened before.

To complex problems similar to this, Data Science and Machine Learning are indispensable tools that transform a vast volume of data and, combining them with statistics and computer science, are able to predict events, recognize patterns and solve an unimaginable number of tasks. Also, the presence of these innovative tools is revolutionizing the way economists handle economic research. As Liran Einav and

Jonathan Levin concluded [7], "there is little doubt (. . .) that over the next decades, “big data” will change the landscape of economic policy and economic research”.

The most common way of understanding the economy’s stochastic behaviour is through the observation of direct data like fluctuations in the stock market, interest rates, in the country’s sovereign debt, and so on. This observation aids us to formulate models that seek to predict the changes in the economic environment, enhancing our knowledge of its behaviour.

However, a substantial portion of the developments in this environment is challenging to predict or model. Not only because of their stochastic essence but also due to the factors that generate the changes. The foundation of these factors is based not on direct data but on the perception that people have towards the market, making them extremely hard to predict, if not impossible. The intent of making as much profit possible in a short period, like in market speculation, is an illustration of one of these market changes attributed to a person’s behaviour. So is panic felt throughout the Great Recession, as P. Bacchetta and E. van Wincoop address [8], that grabbed companies and consumers and drove to a collapse in demand, which in turn drained the profits leading to a worsening of the crisis.

As direct data is unable to provide us with the knowledge to predict these changes in the economy’s behaviour, a way to understand them is to look for data that can provide the perception of the people towards the economy. Hence, the use of text data gathered from financial newspapers and other fonts of information regarding the economy may be an interesting approach to understand people’s perceptions about this topic.

The work present in this document consists of developing a program using Data Science and Machine Learning techniques to answer the foregoing question. This program will be supplied with text data, in this case, text data from finance news articles and economic reports from the European Union and the International Monetary Fund gathered from distinctive periods of the last four decades. It will assemble a multi-dimensional space with all the different words present on the corpus. This constructed space will allow for the study of a geometry based on the relation between words along with a better understanding of the European economy’s behaviour. For instance, the expansion or contraction of the economy, its velocity, or the relations between each country.

CHAPTER 2

Theoretical Concepts

This chapter holds the theoretical concepts, methods and theories that laid the foundation for this work and to the construction of the model present in the next chapter.

2.1 Word Occurrence Frequency

To think about language modelling, it helps to follow the concepts of the 1916's book of Ferdinand de Saussure "Course in General Linguistics" [9], where is presented the idea that language is a system in which the linguistic entities, like words, are related to one another. Considering language as a series of relations within a multitude of words, the only question is how to access these relations.

As attempted previously by Mandelbrot in 1953 [10] and Simon in 1955 [11], this problem could be tackled through the word's distributions given by their frequency of occurrence on a given corpus using stochastic models. However, problems arise due to the nature of the data. The continuous increase in corpus size that consequently increases the error in the prediction of common words, along the intricate relations among all the words inside the corpus vocabulary make it impossible to see this problem statistically. In this subsection, we will present the explanation developed by J. Cruz in his doctoral thesis [12] to demonstrate this statistical problem that drives the necessity for the construction of a new geometry.

$$f(x) = 1 + x + ... \approx f(x) = 1 + x \tag{2.1}$$

J. Cruz assumed that the growth in the occurrence of a word close to another is proportional to a monotonous function that describes the number of times it as already occurred. This function can be described as a Taylor series in which the first two terms are the most significant, enabling us to simplify the function as seen in equation 2.1, where $x > 0$. As the appearance of new words is a multiplicative process, the growth of function $f(x)$ depends on itself.

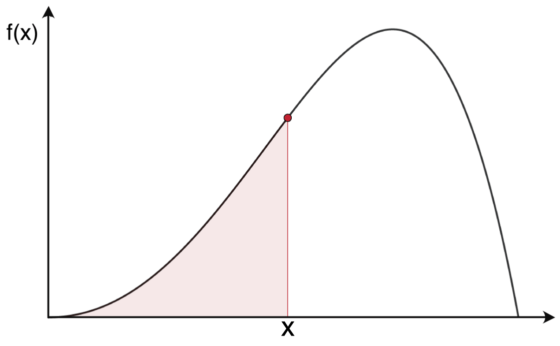


Figure 2.1: Depiction of a random function $f(x)$ and a red area regarding the $f(x)$ integral from 0 to x.

Considering all the universe of word occurrences as Λ.

$$\Lambda = \sum_i x_i \tag{2.2}$$

The variation of the universe is dependent on the variations of the objects that compose it. Thus, the relation can be regarded as

$$\frac{d\Lambda}{\Lambda} = \frac{dx}{x}. \tag{2.3}$$

However, this only occurs when all the objects in the universe are equal, growing at the same time and are not connected. In the case of this work, all the words are different, are growing at different rates and are connected. It is impossible to interact with only one object without disturbing the rest. In a universe like this, the variation of the universe is proportional, by a factor α, to the variation of an object, where α is a positive constant. Hence,

$$\frac{d\Lambda}{\Lambda} = \alpha\frac{dx}{x}. \tag{2.4}$$

Considering a random function of probability like the one in Figure 2.1, it is possible to say that the probability of occurrence of an event x_i smaller than x is equal to Z, the $f(x)$ integral from 0 to x, over the total area Λ.

$$P(x_i < x) = \frac{Z}{\Lambda} \tag{2.5}$$

Then is possible to calculate the derivative of the equation 2.5 with respect to the total area Λ.

$$\frac{dP}{d\Lambda} = -\frac{1}{\Lambda^2}Z + \frac{1}{\Lambda}\frac{dZ}{d\Lambda} \tag{2.6}$$

$$\Leftrightarrow \frac{dP}{d\Lambda} = -\frac{P}{\Lambda} + \frac{1}{\Lambda}\frac{dZ}{d\Lambda} \tag{2.7}$$

Knowing that $\frac{dZ}{d\Lambda}$ can not be bigger than 1, because if that happens the area Z would become larger than the total area Λ, which is not possible. Either cannot $\frac{dZ}{d\Lambda}$ be smaller than 1 because if that happens the opposite occurs, the area Λ will grow so much that the area Z disappears. Hence, $\frac{dZ}{d\Lambda}$ must be equal to one [12].

$$\frac{dP}{d\Lambda} = -\frac{P}{\Lambda} + \frac{1}{\Lambda} = \frac{1-P}{\Lambda} \tag{2.8}$$

$$\Leftrightarrow -\frac{d(1-P)}{1-P} = \frac{d\Lambda}{\Lambda} \tag{2.9}$$

$$\Leftrightarrow \frac{d(1-P)}{1-P} = -\alpha\frac{dx}{x} \tag{2.10}$$

It is helpful to introduce here the property seen beneath in equation 2.11.

$$\frac{d(\log(x))}{dx} = \frac{1}{x} \tag{2.11}$$

With this property, it is possible to change both sides of the equation 2.10 with logarithms.

$$d\log(1-P) = d\log\left(\frac{1}{x^\alpha}\right) \tag{2.12}$$

Now solving both sides of the equation individually as simple integrals from x_0 to x, we obtain equation 2.13 and 2.14.

$$\int_{x_0}^{x} d\log(1-P(x))dx = \log(1-P(x)) - \log(1-P(x_0)) = \log(1-P(x)) \tag{2.13}$$

$$\int_{x_0}^{x} d\log\left(\frac{1}{x^\alpha}\right) = \log\left(\frac{1}{x^\alpha}\right) - \log\left(\frac{1}{x_0^\alpha}\right) = \log\left(\frac{x_0^\alpha}{x^\alpha}\right) \tag{2.14}$$

Substituting 2.13 and 2.14 in 2.12, we obtain 2.15, where both logarithms disappear plus the probability P(x) can be isolated.

$$\log(1-P(x)) = \log\left(\frac{x_0^\alpha}{x^\alpha}\right) \tag{2.15}$$

$$\Leftrightarrow 1-P(x) = \left(\frac{x_0}{x}\right)^\alpha \tag{2.16}$$

$$\Leftrightarrow P(x) = 1-\left(\frac{x_0}{x}\right)^\alpha \tag{2.17}$$

With the purpose of obtaining the probability density $\rho(x)$, it is necessary to calculate the derivative of $P(x)$ in equation 2.17 with respect to x.

$$\rho(x) = \frac{dP(x)}{dx} = \frac{d}{dx}\left(1-\frac{x_0^\alpha}{x^\alpha}\right) \tag{2.18}$$

$$\rho(x) = \alpha \frac{x_0^{\alpha}}{x^{(1+\alpha)}} \tag{2.19}$$

Hence, from 2.19, we have:

$$\rho(x) \propto x^{-(1+\alpha)}. \tag{2.20}$$

As seen in 2.4, the universe's growth is related to the growth objects that compose it by a factor alpha.

If two different objects in the universe are growing independently, the growth of the universe will be the same as in both objects. Hence, in this case, alpha is equal to one [12].

$$\rho(x) \propto x^{-2} \tag{2.21}$$

In the case of the two objects being connected and the growth of both is related, each event occurs twice, i.e., the growth of the universe is equal to double the growth of one of the objects. Hence, alpha is equal to two.

$$\rho(x) \propto x^{-3} \tag{2.22}$$

For the purpose of understanding the error's behaviour in the system, it is easy to start by calculating the mean of squares, whose formula is given by:

$$< x^2 >= \int_0^{\infty} x^2 \rho(x) dx. \tag{2.23}$$

Since $\rho(x)$ is defined in the equation 2.21 and 2.22 in the cases of α being 2 and 3 respectively, if substituted in 2.23 we observe that in both equations 2.24 and 2.25, the mean of squares tends to infinity, whatever the value of alpha.

$$\int_0^{\infty} x^2 x^{-2} dx = \int_0^{\infty} dx = x \Big|_0^{\infty} = \infty \tag{2.24}$$

$$\int_0^{\infty} x^2 x^{-3} dx = \int_0^{\infty} \frac{1}{x} dx = \ln x \Big|_0^{\infty} = \infty \tag{2.25}$$

An infinite $< x^2 >$ means that the statistical error will always become too large as the volume of data increases. In the case of this work, as the number of words increases the bigger the error becomes. Thus, as it is impossible to work in the statistical domain due to the sheer amount of data, arises the need for the construction and transformation of a new geometry space formed by words.

2.2 Word Embedding

Since it is ineffective to understand the relations between words by their frequency of occurrence in a statistical manner, it is necessary to introduce a different way of interpreting text. Word Embeddings come into the forefront of this discussion as techniques of word representation that take into account their intrinsic meaning, i.e., the closer the connotation of these words is, the closer these words will be in the geometrical space created.

Mikalov et al. [13] established two new and revolutionary methods that seem to resonate with Saussure's hypothesis [9]. The Mikalov et al. [13] methods of Natural Language Processing (NLP) are unsupervised neural networks that, using word embedding techniques, interpret the word's relations within a given data and assign to each word a specific vector in a vector space. These vectors create a multi-dimensions space that represents the complexities of the human language.

2.2.1 Artificial Neural Networks

A common feature among the majority of NLP methods such as Word2Vec is their foundation upon Artificial Neural Networks (NN). NN are structures composed of simple computational units called *neurons* that are connected via *links*, creating network-like structured inspired by the connections of our brains, thus similar names. An example of an NN architecture can be seen in Figure 2.2, where the input is received in the layer z, where it is interpreted and processed before passing on to the following layers. These ideas were initially proposed by Rosenblatt F. in his early work in the '50s. [14]

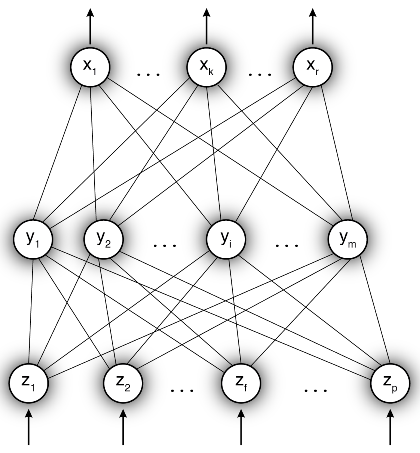


Figure 2.2: Example of a NN's architecture.

The basis of NN falls under the idea that information is received from the outside world by the neuron, in which it is interpreted and processed before producing a result that is sent to another neuron, repeating the process until there is a final output with a result. We can simply conceive a neuron as shown in Figure 2.3, where the inputs are received, interpreted and transformed in a single output. There are several inputs (x_i) arriving at the neuron, having each one of them a weight ($w_{(i,j)}$) associated with it, i.e., a real value that expresses their importance to the process.

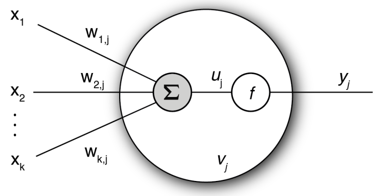


Figure 2.3: Schematic of a neuron.

Inside the neuron, an operation occurs that can be represented as u_j, the sum of all the r inputs, having in mind their relative weight, as shown in equation 2.26.

$$u_j = \sum_{i=1}^{r} w_{i,j} x_i \tag{2.26}$$

A sigmoid function $f(u_i - \tau_i)$ is then used to limit the amplitude of the output y_i of the neuron as it shows below in equation 2.27.

$$y_i = f(u_j - \tau_j) = \begin{cases} 0 & if \quad u_j \leq \tau_j, \\ 1 & if \quad u_j > \tau_j. \end{cases} \tag{2.27}$$

Being τ_j the threshold of the specific neuron. Several sigmoid functions can be used accordingly but choosing a sigmoid function is an important step, as different functions have different complexities that influence the processing power required to run the model. Depending on the purpose and the complexity, there can be several layers of neurons between the input and output layers in the NN's architecture, and these are called hidden layers.

2.2.2 Word2Vec

Word2Vec is a method used to create and perfect the vector representation that gives form to the word embeddings. There are two different approaches to Word2Vec, in which both receive an extensive text corpus. In Continuous Bag Of Words (CBOW), the concept of this approach revolves around the prediction of a word given a context, and in Skip-gram, the prediction focus on the context surrounding a specific word [15]. Can be seen in Figure 2.4 a schematic representing the idea of these two different approaches to Word2Vec.

Mikolov et al. [16] identified an interesting feature, in these multiple dimensions spaces where the words are mapped as a function of their relations, words can be located using simple arithmetics, as they put it "To find a word that is similar to *small* in the same sense as *biggest* is similar to *big*, we can simply compute vector $X = vector("biggest") - vector("big") + vector("small")$. Then, we search

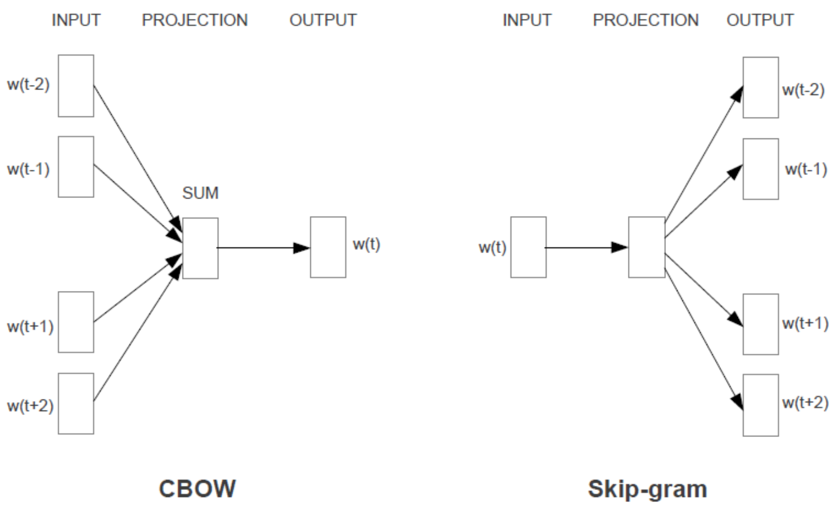


Figure 2.4: Schematic of the two approaches to Word2Vec. [13]

in the vector space for the word closest to X measured by *cosine distance*[1] and (...) when the word vectors are well trained, it is possible to find the correct answer (word smallest) using this method" [16]. This reveals to be an impressive feature of Word2Vec that offers an interesting manner of interacting with the output data.

2.2.3 The Skip-gram Model

The goal of the $Skip - gram$ (SG) is to predict the context of a given word and, behind this model, it is the objective of maximizing the average logarithmic probability.

$$L_{SG}(S) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t) \tag{2.28}$$

Where $S = \{w_1, w_2, ..., w_i\}$ is the sequence of the words trained, T is the total number of trained words and c is the length of the training context, i.e., the number of words that make the interval of context. The conditioned probability $p(w_{t+j}|w_t)$ can be written using the $softmax$ function, a function that converts the number it receives into a sum of probabilities.

$$p(w_O|w_I) = \frac{exp(v_{w_O}^{,\top} v_{w_I})}{\sum_{w=1}^{W} exp(v_w^{,\top} v_{w_I})} \tag{2.29}$$

Where W is the number of all the words in the vocabulary trained, v_n and $v_n^,$, are the vectors representing w in the input and output respectively, $^\top$ is the vector transpose, making $v_w^,\top$ the transpose of the input vector [16].

[1]Cosine distance or Cosine similarity is a similarity measurement described in subsection 2.3.

2.2.4 Hierarchical Softmax

Both the Skip-gram Model and the CBOW Model, though revolutionary in relation to the existing ones, had a huge problem, the duration of their computation time, which put them behind in the moment of choice. To solve this dilemma, as it is explained in Mnih et al. [17], the vocabulary is structured in a tree-like arrangement that gives more frequent words a smaller *distance* from the root to their respective position in the tree. Mikolov et al. [16] goes one step further defining $p(w|w_I)$, he considers $n(w, j)$ to be the j^{th} node of the path from the root to the word in question, $L(w)$ the length of this path, $ch(n)$ an arbitrary fixer child of n, $\sigma(x) = 1/(1 + exp(-x))$ and $[\![x]\!]$ to be 1 if $[\![0]\!]$ is true and -1 otherwise.

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([\![n(w, j+1) = ch(n(w, j))]\!].v'_{n(w,j)}{}^{\top} v_{w_I}) \tag{2.30}$$

This apparently simple change in the architecture of the model has a dramatic effect in its computation time, outperforming non-hierarchical models by at least two orders of magnitude. [16]

2.2.5 Negative Sampling

Negative Sampling is an additional technique used by Mikolov et al. [16] to improve the Skip-gram model. It works by adding to each word of the vocabulary a small group of words that are unlikely to be present in the phrase in which that word is present. These lists of words are referred as Skip-gram model with negative sampling (SGNS) and are defined as follows:

$$L_{SGNS}(S) = \log \sigma(v'_{w_O}{}^{\top} v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)}[\log \sigma(-v'_{w_i}{}^{\top} v_{w_I})] \tag{2.31}$$

As it only adds a small number of negative samples in each word, it does not increase the time of computation and improves the accuracy of this model.

2.2.6 Dealing with Frequent Words

The occurrence of very frequent words in large text corpora creates a discrepancy in the value of the information of the vocabulary, i.e., the value of a word that appears several times in a corpus should be less than the value of the words that appear less often. Mikolov et al. [16] counteracted this discrepancy by subsampling frequent words using the following formula:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{2.32}$$

Where $f(w_i)$ is the frequency of the given word in the vocabulary and t is a chosen threshold. This subsampling has proven to be a good addition to the model as it improved his learning rate and accuracy.

2.3 Cosine Similarity

When working with multi-dimensional spaces, measuring the Euclidean distance is not always a reliable way to perceive the similarity between objects. When the multi-dimensional space is formed by dimensions that represent words, similarly to the space created for this work, the use of the Euclidean distance can be misleading. If two similar words, with different usage frequencies, are distant in a euclidean manner, they are both similarly oriented in the word space. Thus, making the usage of the cosine similarity an advantage, because it measures the difference in orientation.

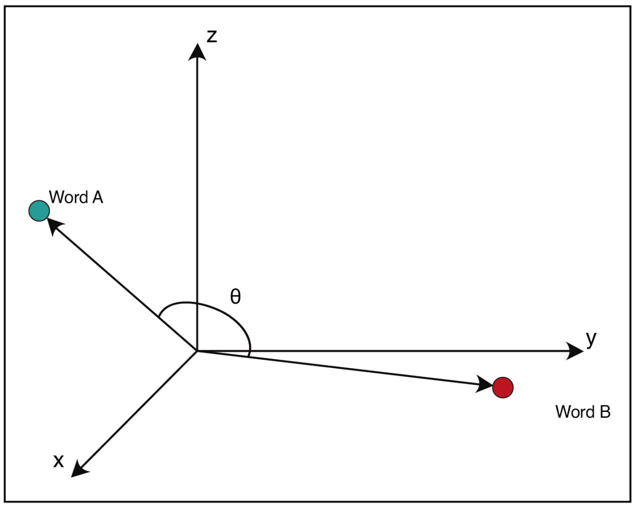


Figure 2.5: Representation of the angle θ between two objects in a word space.

Cosine similarity between two objects, similarly to the word vectors on Figure 2.5, is defined by the cosine of the angle between them. The formula that describes the cosine similarity is displayed below.

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\| \vec{a} \| \| \vec{b} \|} \tag{2.33}$$

The results of equation (2.34) range between [-1,1]. However, in the case of this work, the cosine similarity is employed in a positive space, shaping the outcome of the cosine similarity between [0,1].

2.4 Nonlinear Dimensionality Reduction

When working with high-dimensional data like word embeddings, the data visualization is remarkably complicated due to our inability to perceive more than three dimensions. Many tried to tackle this problem throughout the years by creating methods that aid in studying these high-dimensional data. One of the paths is through the reduction of dimensions of the data.

2.4.1 Stochastic Neighbor Embedding

G. Hinton and S. Roweis introduced, in their 2002 paper [18], Stochastic Neighbor Embedding (SNE) as a nonlinear dimensionality reduction technique. It was a groundbreaking technique that significantly improved the study of high-dimensional data. G. Hinton and S. Roweis propose using a Gaussian in each object of the high-dimensional universe [18]. The density underneath the Gaussian defines the probability distribution over all the neighbour objects. Assigning the minimization of the difference between the distribution values in the high-dimensional and the low-dimensional spaces the goal of this technique.

Later, Laurens van der Maaten and Geoffrey Hinton described this technique from a physics perspective, imagining each object having springs connecting him and the remaining objects in the space. Each spring could result in an attractive or repulsive force on the object, depending on the distance between himself and the other connecting object in the high-dimensional space. The sum of all the forces places the objects previously in a multi-dimensional space into a low-dimensional and more straightforward to interpret space [19].

2.4.2 t-Distributed Stochastic Neighbor Embedding

Due to the difficulty in optimizing the cost function and, as Laurens van der Maaten and Geoffrey Hinton call, ”crowding problems” [19], they presented a new technique that aims to overtake these problems.

The authors implemented two improvements to the previous SNE. First, they used a symmetrized cost function introduced by Cook et al. [20] similar to the one used in SNE but with simpler gradients. Second, instead of using a Gaussian to estimate the similarity among the various objects in the low-dimensional space, a Student-t distribution is used [19].

This improvement to the already revolutionary SNE technique allows for better visualization of considerably large data while using less computational power.

CHAPTER 3

Design and Methodology

This chapter's main objective is to present and explain the steps taken throughout the work, from data gathering, through the model's development until the preparation of the results. In our work, several files were programmed using the programming language Python and some of these files can be found in the Appendices.

3.1 Data Collection

This work aims to understand the overtime geometry evolution of corpora about economy and finance from different periods; thus, the first step was to gather text data about the intended topic. In this case, the topic was economy and finance, focusing explicitly on the European Union data.

As said previously, news articles about finance and the economy are beneficial to projects like this as they can provide interesting information about people's perspectives about the state of the economy. Hence, the primarily focus was gathering the most considerable amount of news articles from the top economic and financial news agencies:

- From the Webhose web page [21], a dataset called "Financial news articles" was acquired containing several news articles from the 2015's period between July and October. This dataset contains news articles from several news agencies worldwide, each of them categorized with various labels as the date, news agency, country of interest.
- Although older, the "Reuters-21578, Distribution 1.0" dataset [22] was a vital addition to our database, considering that it brought us a large number of news articles from 1987. As the name suggests, this free dataset consists of a total of 10788 documents from Reuters financial newswire.
- The "Financial News Unprocessed" dataset [23] contains news articles from Reuters, and The Wall Street Journal collected in June 2017. It is available for public use.

Besides news articles, it was also essential to gather data from other sources to widen our dataset. Thus, reports were collected about the European economy from several EU organizations and the IMF:

- The Annual Reports from the European Central Bank were collected since 1991 as well as all the Economic Bulletins since 2015. The Annual Reports consist of reports on Eurosystem's monetary policies and descriptions about the European System of Central Banks' tasks and activities. Economic Bulletins contain information about the economic and monetary situation in which the Governing Council policies' decisions are based.

- With the intent of focusing on the EU's future and the Euro Zone, the European Commission publishes the Economic Forecasts. These contain economic outlooks for individual member states as well as for future ones. Economic Forecasts also include projections about some of the World's largest economies, as these economies can affect the EU and the Euro Zone performance.

- The Regional Economic Outlooks published by the IMF discuss economic developments and prospects for countries of specific regions, in this case, Europe. It also contains information about the outcomes created by the economic policies put in place and their effects on the region's economic performance.

Collected Data		
Name	Data Type	Period
European Union Annual Reports	Reports	Post-1991
European Union Economic Bulletins	Reports	Post-2015
European Union Economic Forecast	Reports	Post-2000
Financial News Unprocessed	News Articles	2017
IMF European Regional Economic Outlook	Reports	Post-2008
Reuters-21570	News Articles	Pre-2000
Webhose - Financial News Articles	News Articles	2015

Table 3.1: Description of the Collected Data per Type and Period.

The data collected was then divided into periods in order to study the evolution of the word geometry formed by the model. The periods chosen were intervals separated by significant events so it is possible to understand the effects of these events on the data. The events chosen were the turn of the millennium and the 2008's financial crisis as they were big turning events to the financial sector. Three TXT files were created, one from each period between the chosen events.

Due to the sheer amount of data collected to be used in the model implementation, it was not feasible to place it in the appendix. Hence, a permanent online project was created in GitHub to store the data, making it available to anyone who wants to use it [24].

3.2 Tools and Libraries

Oftentimes in programming projects, open-sourced toolkits and libraries are used to facilitate some processes, and this work is not an exception. Therefore, in this chapter, the tools and libraries are introduced and explained to facilitate their use throughout the program.

3.2.1 NLTK

The Natural Language Toolkit (NLTK) is an open-source program consisting of several modules that, along with a straightforward structure, aids in computational linguistics projects like this one [25]. It incorporates highly useful graphical tools to display data structures. It is also compatible with Python, a very intuitive and easy-to-write language.

3.2.2 GENSIM

Gensim is a Python library precious for data analysis because of its quick, memory-efficient, scalable algorithms for Singular Value Decomposition (SVD) and latent Dirichlet Allocation (LSA). It is ideal for operating with Natural Language Processing as it is a prominent package concerning processing extensive text data in word models [26].

3.2.3 Pickle

Pickle is a useful module that serializes and de-serializes Python objects by employing binary protocols. This module converts the Python objects and converts them into a character stream quickly and easily. The pickling and unpickling process occur without damaging the objects hierarchy, allowing reconstructing the object in another python script.

3.2.4 Multiprocessing

Sometimes, when programming with complex code, the machine's processing power is not enough or is not optimized, delaying the code's processing. Multiprocessing is a package that grants programmers more control over the processing power of a given machine. This module is remarkably useful in data processing and AI projects to enhance the device's performance to faster results.

3.2.5 OS

So as to interact with the device's operating system, the OS module presents several functions that grant the programmer easy access to information or to produce changes in the device.

3.2.6 RE

The RE module is a helpful tool in searching and manipulating text data. It uses matching operations to locate strings in the data and provides multiple functions that allow the simple manipulation of the located string.

3.2.7 Future

The __future__ module is an adaptability layer connecting Python 2 and Python 3. It enables the interpreter to compile some semantics available in futures versions easily in the current interpreter.

3.2.8 NumPy

NumPy, Numerical Python, is an open-source Python library used to do numerical computing with multi-dimensional arrays [27]. It is a useful tool to use when working on scientific computing since arrays are the conventional representation concerning numerical data.

3.2.9 Sklearn

Sklearn, SciKit-learn, is a machine learning library freely available on Python. It was created using other libraries like NumPy, pandas, and Matplotlib, providing several supervised and unsupervised learning algorithms.

3.2.10 Pandas

Pandas is a robust, quick, and accessible library to use that allows the programmer to store, interpret, and manipulate data. This tool, built upon the NumPy package, enables manipulating extensive tabular data and times series, useful to data science projects.

3.2.11 Matplotlib

In order to understand the behaviour of the data, it is useful to visualize it interactively. The Matplotlib library adds an extensive collection of functions to Python that allows easy and interactive visualizations of data, both static or animated.

3.2.12 Seaborn

Similarly to Matplotlib, Seaborn is a data visualization library for Python. It is built upon Matplotlib and produces engaging and instructive charts and graphs with a distinguished interface.

3.3 Corpus Cleaning

Before feeding the data into the model, it is necessary to clean it, i.e., it is necessary to remove all the unnecessary content from the corpus. The preprocessing of the corpus is essential in Natural Language Processing (NLP) in order to construct a functional model.

The preprocessing of the corpus consists of removing all non-alphabetic characters, such as numeric, punctuation and symbol characters, as well as removing stopwords. The removal of the non-alphabetic characters is straightforward to understand as they do not convey meaning to the corpus. However, stopwords are actual words, so their dismissal is due to their intricate meaning. Stopwords are the most common words in a natural language, and are words like ”and,” ”but,” ”a,” ”or,” and ”what”. Although necessary in a natural language, they add little significance to a corpus besides connecting the phrases.

An additional step while preprocessing the corpus is through its simplification, which can be achieved by a process called lemmatization. The lemmatization process consists of tagging words according to their morphology and then reducing the word to its simplest form. For example, the word ”cats”, a noun in plural, becomes ”cat”, ”best”, an adjective, becomes ”good” and ”was”, a verb in the past, becomes ”be”. Lemmatization is a powerful tool because considerably reduces the size of the corpus, consequently decreasing the processing time when training the model and improves the accuracy of the trained model.

For this purpose, a Python file was written in order to preprocesses the raw corpus to be trained. In the remaining of this subsection, the code will be explained step by step. The code can be found in full on Appendix A.

Firstly it is necessary to import all the essential libraries to the preprocessing of the raw corpus. These libraries are NLTK and Pickle, as well as some specific functions.

Then the lemmatization function was defined as ”lemmatizer”, using the NLTK function ”WordNetLemmatizer”.

```python
lemmatizer = WordNetLemmatizer()
```

To aid in the cleaning process, word vectors were created. These vectors are, ”ord”, a vector containing special words that firstly escaped through the cleaning process but are necessary to the corpus, ”countries”, a vector comprised of country names, and ”stopW”, a vector containing the English language stopwords.

```python
ord_ = ['Europe', 'European', 'Union', 'Brexit', 'Parliament',
'Commission', 'Investment', 'Single', 'Market', 'Gulf', 'War',
 'ECB', 'Asia', 'OECD', 'UAE', 'NATO']

# Country List
countries = []
with open('Lists/Countries_upper.txt', 'r') as fp:
    for line in fp:
        countries.append(line.replace('\n', ''))

# StopWords List
stopW = []
with open('Lists/stopW.txt', 'r') as fp:
    for line in fp:
        stopW.append(line.replace('\n', ''))
```

So that it is possible to lemmatize the corpus, the tagging of the words is fundamental. Thus, two functions where define to execute this function, the function "tagging" and "nltk_tag_to_wordnet_tag". The function "tagging" receives a string containing a phase, converts the string into a vector including the words on the phrase with the function "word_tokenize", using the "nltk.pos_tag" function tags every word in the vector and ultimately returns the tagged vector. The purpose of function "nltk_tag_to_wordnet_tag" is to convert the NLTK tags into WordNet tags, as the "WordNetLemmatizer" uses WordNet tags instead of NLTK's.

```
# Tagging function for name removal
def tagging(phrase):
    phrase = word_tokenize(phrase)
    phrase = nltk.pos_tag(phrase)
    return phrase

# function to convert nltk tag to wordnet tag
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

Subsequently, the function that lemmatizes the sentences is defined with the name "lemmatize_sentence". This function receives a string input containing a sentence that is tokenized and tagged with NLTK tags using the "tagging" function. Creates a variable "wordnet_tagged" which contains all the words on the sentence and the respective WordNet tag, and an empty variable "lemmatized_sentence" that will receive, as the name suggests, the sentences lemmatized. Then, inside a for loop running on the "wordnet_tagged" variable, the tagged words are simplified and appended on the empty variable "lemmatized_sentence", and the ones that do not have tags are solely appended as they are. This function returns the input sentence lemmatized.

```
def lemmatize_sentence(sentence):
    nltk_tagged = tagging(sentence)
    wordnet_tagged = map(lambda x: (x[0],
                nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatized_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatized_sentence.append(word.lower())
        else:
            lemmatized_sentence.append(
                (lemmatizer.lemmatize(word, tag)).lower())
    return " ".join(lemmatized_sentence)
```

The corpus must be imported into the program before anything else. Thus, the file containing the corpus is imported into the "text" variable. Which is then tokenized by sentences using the NLTK function "sent_tokenize".

```
# Open Corpus
file = open('File.txt', encoding='latin-1')
text = file.read()
file.close()

# Tokenize
sentences = sent_tokenize(text)
text_ = [], trash = []
```

The cleaning process occurs inside a for loop, running through the vector containing the tokenized sentences. Firstly, the hyphens on the sentence are replaced by a space in order to prevent words like multi-dimensional to become attached when the non-alphabetic characters are removed. The sentence is tokenized into a vector that then passes through three steps. The first removes all the non-alphabetic characters, the second removes the stopwords and the third removes words smaller than two characters. A for loop is then introduced to remove all the words that are not able to be decoded with the "latin-1" Unicode. Using the "tagging" function and another for loop, the proper nouns, NNP, are removed from the text, with the objective of reducing the corpus size. Finally, the sentences are lemmatized using the "lemmatized_sentence" function and appended into a variable containing the clean corpus.

```
for sentence in sentences:
    sentence = sentence.replace('-', ' ')

    words = word_tokenize(sentence)

    # Removes all the Non-Alphabetic characters
    words = [word for word in words if word.isalpha()]

    # Remove StopWords
    words = [w for w in words if not w in stopW]

    # Remoce words smaller than 2 characters
    words = [w for w in words if len(w) >= 2]

    aux = [], aux_ = [], sent = []

    for word in words:
        if len(word) > 0:
            try:
                word.encode('latin-1')
                aux.append(word)
            except:
                trash.append(word)

    sent = ' '.join(aux)
    tags = tagging(sent)
    del aux, sent

    phrase = []

    for tag in tags:
        if tag[1] == 'NNP' and tag[0] not in countries
                           and tag[0] not in ord_:
            trash.append(tag[0])

        else:
            phrase.append(tag[0])

    phrase = ' '.join(phrase)


    # Simplify the text
    phrase = lemmatize_sentence(phrase)

    text_.append(phrase)
    text_.append('.')
```

In order to use the variables with the clean text and the removed words later, these variables are saved in TXT files.

3.4 Model Implementation

After cleaning the corpus, the Word2Vec Model has to be generated and trained. Thus, a Python file was built in which the corpus is prepared to be fed into the model, and the model is trained and saved. In this subsection, the code found in Appendix B is explained.

The first lines of the code contain the imported functions and libraries needed for the program's operation, followed by the import of the cleaned corpus into a vector. This vector is then tokenized using the "sent_tokenize" function that transforms each sentence in the corpus into a token.

```python
# Open Book
file = open('CLEAN_TEXT.txt', encoding='utf-8', errors='ignore')
book = file.read()
file.close()
print("Book loaded!")

raw_sentences = sent_tokenize(book)
```

Before tokenize the corpus by words, it is necessary to remove the full stop at the end of each sentence, consequently, the "word_tokenizer" function is defined.

```python
def word_tokenizer(raw):
    clean = re.sub("[^a-zA-Z]"," ", raw)
    words = word_tokenize(clean)
    return words
```

The variable "sentences" is created and filled with the tokenized corpus in a for loop.

```python
sentences = []
for raw_sentence in raw_sentences:
    if len(raw_sentence) > 0:
        sentences.append(word_tokenizer(raw_sentence))
```

The vector "sentences" has, in each element, the tokenized sentence and is now ready to be feed into the model. The size of the corpus is an important variable on the accuracy of the model. Thus, the length of the "sentences" vector is determined and printed.

```python
# Tokens Counter
token_count = sum([len(sentence) for sentence in sentences])
print("This corpus contains {} tokens.".format(token_count))
```

The extent of the files compiled for this work is presented in Table 3.2.

The model is defined as well as some of its parameters. In the "sg" parameter, the Skip-gram model was chosen instead of the CBOW because although CBOW is faster and provides good results for frequent words, Skip-gram represents well, not only frequent words but also rare ones. The "workers" parameter sets the number of worker threads, so, with the help of the "multiprocessig.cpu_counter" function, the workers were maximized to the number of cores of the machine in use. The

Corpus Size	
Corpus Period	Token Number
Pre-2000	1.124.496
2000 until 2008	780.732
Post-2008	10.542.623

Table 3.2: Number of tokens in each Corpus.

parameter "size" defines the size of the word vectors. The common values vary between 100 and 300, and due to the size of the corpus, the size was set as 300. [13] With a corpus size like the one used on this work, it's not practical the usage of words with very small frequencies. Thus, it was decided to disregard all the words that occurred less than eight times throughout the corpus by setting the "min_count" at eight. The "window" variable marks the range of surrounding context words, i.e., indicates the range in which the words before and after a given word are included in its context. This variable was set at eight because large windows usually achieve further information concerning the realm of the respective word. [28] The final variable to be established was "sample". This variable defines the threshold to which words with high-frequency of occurrence are down-sampled. The variable was set at 1e-4, a value within the range of (0, 1e-5) recommended by the package creators.

```
# Train The Model
Model = w2v.Word2Vec(
    sg = 1, #Skip-Gram
    workers = multiprocessing.cpu_count(),
    size = 300,
    min_count = 8,
    window = 8,
    sample = 1e-4
)
```

With the model parameters defined, a vocabulary is created with all the words in the corpus.

```
Model.build_vocab(sentences)
```

Finally, the model is trained and saved in a folder named "Trained". As a result of the data size and depending on the machine in use, the training process might take a few minutes.

```
Model.train(sentences, total_examples=Model.corpus_count,
                       epochs=Model.iter)

# Save The Model
if not os.path.exists("Trained"):
    os.makedirs("Trained")
```

```python
Model.save(os.path.join("Trained", "Model.w2v"))
```

3.5 Dimension Reduction

After training, the model produces a high-dimensional space geometry containing all the words in the vocabulary. This space has three hundred dimensions, established earlier while defining the model variable ”size”. In order to visualise the objects efficiently in the space to process the results, the high-dimensional space suffered a dimension reductions operation. Hence, a Python file was created to progress the dimension reduction along with its description is present within this subsection. The code can be found in Appendix C.

On the first lines, the necessary libraries, like Numpy, Word2Vec and TSNE, were imported. The trained model is imported and its vocabulary is then placed into a variable called ”vocab_len”.

```python
# Open the trained model
model = w2v.Word2Vec.load("Model.w2v")
print("Model loaded")

# Vocabulary length
vocab_len = len(model.wv.vocab)
print("Vocabulary length is ", vocab_len)
```

To aid the dimension reduction process, the vocabulary must be put inside a matrix, so a matrix is defined with the dimensions necessary to place all words and their respective tokens. Then, inside a for loop, the matrix is filled with the vocabulary.

```python
# Define Matix
word_vectors_matrix = np.ndarray(shape=(vocab_len, 300),
                                    dtype='float64')
word_list = []
i = 0

# Fill the Matix
for word in model.wv.vocab:
    word_vectors_matrix[i] = model[word]
    word_list.append(word)
    i += 1
    if i == vocab_len:
        break
```

In order to convert the high-dimensional space in only two dimensions, the TSNE function is defined followed by the creation of the matrix holding the transformed two-dimensional space. Ultimately, both the matrices are stored for later use.

```
# Compress the word vectors into 2D space
tsne = TSNE(n_components = 2, random_state = 0, metric="cosine")
word_vectors_matrix_2d = tsne.fit_transform(word_vectors_matrix)

# Save Matrix
np.save("Mtx_name", word_vectors_matrix)
np.save("Mtx_2d_name", word_vectors_matrix_2d)
```

3.6 Word Space Plot

With the two-dimensional matrix defined by the code in Appendix C, is now possible to plot the words in a graph and to interact with the word space. In order to plot the words, a Python file was programmed and it is displayed in Appendix D.

The code starts with the import of various functions necessary to run it, like Numpy, Pandas, Word2Vec, Mathplotlib and Seaborn. The saved model and matrices containing the model vocabulary are also imported.

```
# Open the trained model
model = w2v.Word2Vec.load("Model.w2v")
print("Model loaded")

# Vocabulary length
vocab_len = len(model.wv.vocab)
print("Vocabulary length is ", vocab_len)

# Open the trained model matrix
word_vectors_matrix_2d = np.load("Mtx_name.npy")

# Open the multi-dimensional matrix
word_vectors_matrix = np.load("Mtx_2d_name.npy")
```

Then, a for loop fills the ”word_list” vector with all the words on the vocabulary.

```
word_list = []
i = 0
for word in model.wv.vocab:
    word_list.append(word)
    i += 1
    if i == vocab_len:
        break
```

With the help of the Pandas library, a data frame is created to build a table with three columns, one for the words and two for the x and y coordinates.

```python
# Word points DataFrame
points = pd.DataFrame([
    (word, coords[0], coords[1])
    for word, coords in [
        (word, word_vectors_matrix_2d[word_list.index(word)])
        for word in word_list
    ]
], columns=["Word", "x", "y"])
```

To create a plot environment, the context is set at "poster" with the "sns.set_context" function, and a figure is created with the "plt.subplots" function. Then, the words are plotted with a red marker and the axes defined. Ultimately, the word tags are plotted with the help of a for loop.

```python
sns.set_context("poster")
fig, ax = plt.subplots()

# Plot the word points
ax.plot(points.x, points.y, 'ro', markersize=15)

# Defining Axes
offset = 1.0
ax.set_xlim(min(points.x) - offset, max(points.x) + offset)
ax.set_ylim(min(points.y) - offset, max(points.y) + offset)

# Plot the point tags
k = 0
for i, j in zip(points.x, points.y):
    corr = -0.05  # correction for annotation in marker
    ax.annotate(points.Word.values[k], xy=(i + corr, j + corr))
    k += 1

plt.show()
```

This code produces a graph similar to the one shown in Figure 3.1, where the units of the axes are intrinsic to the word universe and have no relations to the units of the universe we live in. Based on the concept of distributed representation [29], each word is represented by more than one axis. In this case, each word is represented in relation to all the other words, creating a space based on total interconnectivity among words. This process was a result of the TSNE technique.

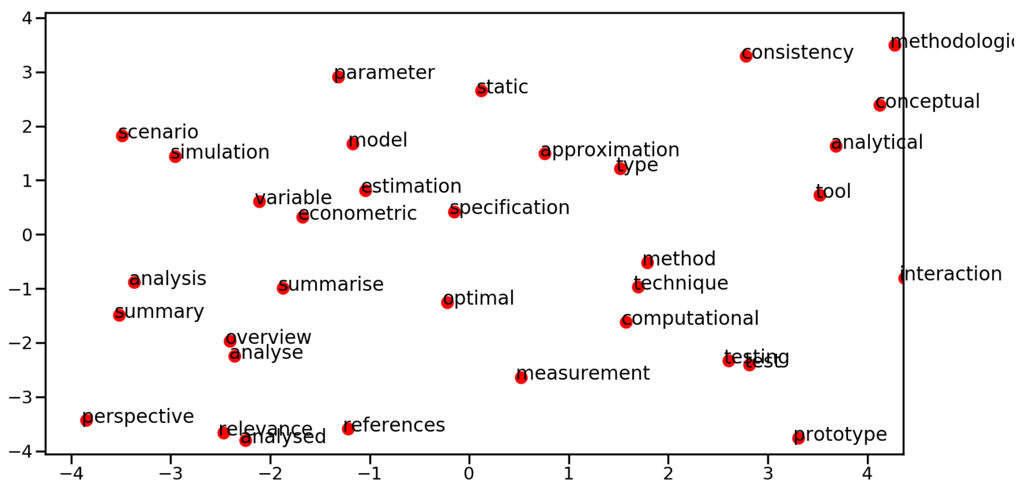


Figure 3.1: Zoomed section of the word space created with the Post-2008 corpus.

The word spaces created using the models and their two-dimensional representation are discussed in chapter four.

CHAPTER 4

Results and Discussion

In this chapter, as the objective of constructing a geometric space based on natural language text was achieved, the results gathered throughout this master thesis are here exhibited and debated. The assembled space offers us the possibility to interact, measure and observe its behaviour through different periods.

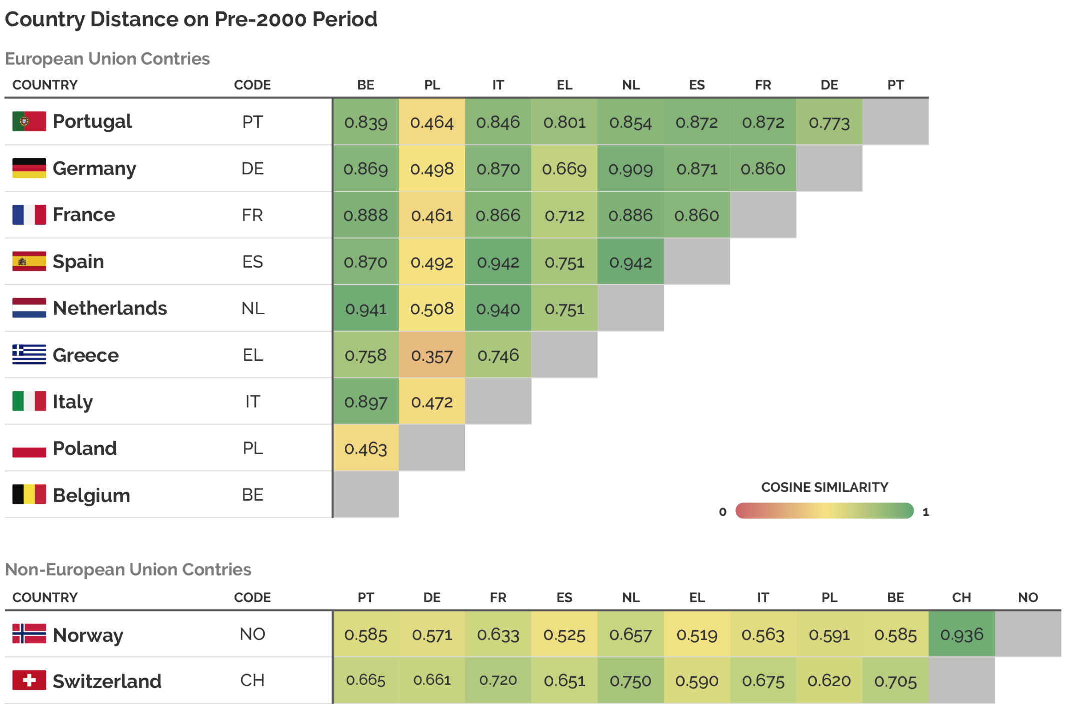
The chapter is divided into two sections. The first section presents the data collected from the high-dimensional space created with the model described earlier. The second section displays the plots generated with the dimensionally reduced word space.

4.1 Word *Distance*

From our multi-dimensional word space, it is possible to measure the words' closeness through the cosine similarity. The cosine similarity can be obtained using the function "similarity" from the Gensim library. This function receives two strings that must be words present in the corpus vocabulary and returns the cosine similarity between them.

Since our generated word space places together words that have a similar meaning or belong within the same topic according to peoples perspective emerged the idea of measure the *distance* between the European countries according to that perspective. This measurement was carried out, comparing the cosine similarity between the European country names in the corpus during the three different periods of gathered data.

The similarity results are displayed on the three different periods of collected data. The Figures 4.1, 4.2 and 4.3 represent respectively, the period before the year 2000, the period between 2000 and 2008, and the period after 2008. The tables within the figures show a colour gradient to help interpret the similarities between words. This colour gradient progresses from red, meaning zero similarity between words and cosine similarity = 0, till green, meaning the cosine similarity = 1 and the words are entirely similar, i.e., it is the same word.



Country Distance on Pre-2000 Period

European Union Contries

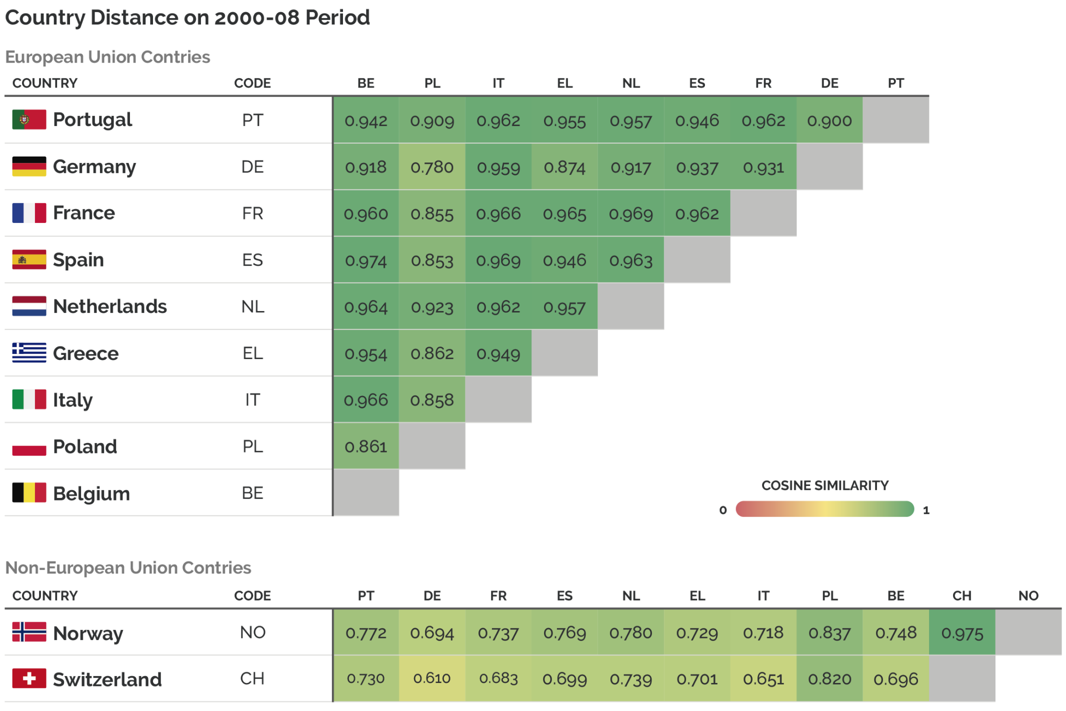
COUNTRY	CODE	BE	PL	IT	EL	NL	ES	FR	DE	PT
Portugal	PT	0.839	0.464	0.846	0.801	0.854	0.872	0.872	0.773	
Germany	DE	0.869	0.498	0.870	0.669	0.909	0.871	0.860		
France	FR	0.888	0.461	0.866	0.712	0.886	0.860			
Spain	ES	0.870	0.492	0.942	0.751	0.942				
Netherlands	NL	0.941	0.508	0.940	0.751					
Greece	EL	0.758	0.357	0.746						
Italy	IT	0.897	0.472							
Poland	PL	0.463								
Belgium	BE									

COSINE SIMILARITY 0 1

Non-European Union Contries

COUNTRY	CODE	PT	DE	FR	ES	NL	EL	IT	PL	BE	CH	NO
Norway	NO	0.585	0.571	0.633	0.525	0.657	0.519	0.563	0.591	0.585	0.936	
Switzerland	CH	0.665	0.661	0.720	0.651	0.750	0.590	0.675	0.620	0.705		

Figure 4.1: Country *distance* measured by cosine similarity on the Pre-2000 Period.



Country Distance on 2000-08 Period

European Union Contries

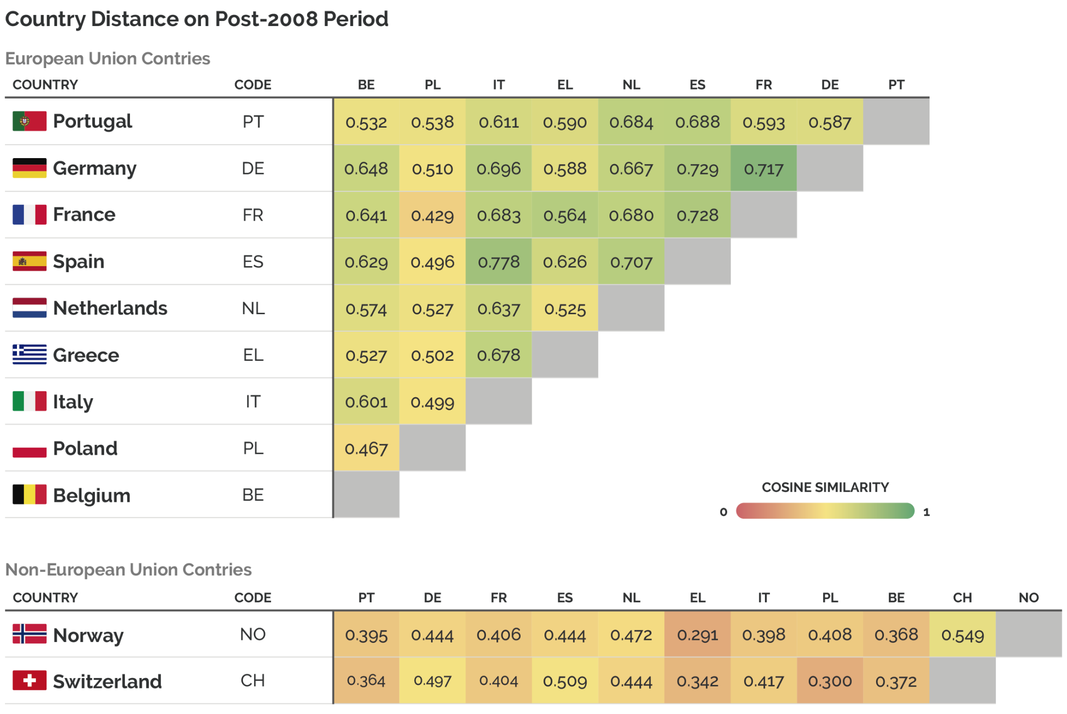
COUNTRY	CODE	BE	PL	IT	EL	NL	ES	FR	DE	PT
Portugal	PT	0.942	0.909	0.962	0.955	0.957	0.946	0.962	0.900	
Germany	DE	0.918	0.780	0.959	0.874	0.917	0.937	0.931		
France	FR	0.960	0.855	0.966	0.965	0.969	0.962			
Spain	ES	0.974	0.853	0.969	0.946	0.963				
Netherlands	NL	0.964	0.923	0.962	0.957					
Greece	EL	0.954	0.862	0.949						
Italy	IT	0.966	0.858							
Poland	PL	0.861								
Belgium	BE									

COSINE SIMILARITY 0 1

Non-European Union Contries

COUNTRY	CODE	PT	DE	FR	ES	NL	EL	IT	PL	BE	CH	NO
Norway	NO	0.772	0.694	0.737	0.769	0.780	0.729	0.718	0.837	0.748	0.975	
Switzerland	CH	0.730	0.610	0.683	0.699	0.739	0.701	0.651	0.820	0.696		

Figure 4.2: Country *distance* measured by cosine similarity on the 2000-2008 Period.



Country Distance on Post-2008 Period

European Union Contries

COUNTRY	CODE	BE	PL	IT	EL	NL	ES	FR	DE	PT
Portugal	PT	0.532	0.538	0.611	0.590	0.684	0.688	0.593	0.587	
Germany	DE	0.648	0.510	0.696	0.588	0.667	0.729	0.717		
France	FR	0.641	0.429	0.683	0.564	0.680	0.728			
Spain	ES	0.629	0.496	0.778	0.626	0.707				
Netherlands	NL	0.574	0.527	0.637	0.525					
Greece	EL	0.527	0.502	0.678						
Italy	IT	0.601	0.499							
Poland	PL	0.467								
Belgium	BE									

COSINE SIMILARITY 0 – 1

Non-European Union Contries

COUNTRY	CODE	PT	DE	FR	ES	NL	EL	IT	PL	BE	CH	NO
Norway	NO	0.395	0.444	0.406	0.444	0.472	0.291	0.398	0.408	0.368	0.549	
Switzerland	CH	0.364	0.497	0.404	0.509	0.444	0.342	0.417	0.300	0.372		

Figure 4.3: Country *distance* measured by cosine similarity on the Post-2008 Period.

Although with caution, we are able to hypothesise the interpretation of the data portrayed in the previous figures. Focusing on the idea that all the collected data derives from financial news and economic reports, the word space construction is based on the assumption of the people who wrote the data. This implies that, ideally, our word space portrays a word representation based on the assumptions about the economy from the people of a specific period. Thus, it is theoretically possible to evaluate the differences in the cosine similarity between the words and understand the evolutions in people's perspective throughout time.

Looking at the figures previously shown, it is helpful to have in mind the events that took place at the European stage in the last three decades.

Starting with Figure 4.1, with data from the last decade of the twentieth century, at first glance, it is possible to observe that all the countries belonging to the European Union at this time are fairly close to one another, including the newly joined members states of Portugal and Spain. Poland, as a Non-European Union member at this time, is placed further away from the European Union members states. Norway and Switzerland, while close, keep a *distance* from the other countries.

In Figure 4.2, with data from the first eight years of the new millennium, we can witness that, in general, all countries came closer to each other as the thought of a unified Europe was a shared perspective among the financial sector. With the introduction of the Euro, countries like Portugal, Spain and Greece became also closer to their European neighbours. In 2004, Poland joined the Union and can now be seen closer to every country. The Non-European Union countries, although still at a considerable *distance*, are also closer than before to the Union members and even closer to each other.

In Figure 4.3, with post-2008 data, we can see a severe divergence between all the countries in the aftermath of the 2008 financial crisis. Even though the bigger economies like Germany, France, Italy and Spain maintain some closeness, the *distances* between the countries are much greater than any period during the last three decades. The most significant differences can be seen among the Non-European Union members, as the *distances* between them and the Union members increased immensely, alongside the *distance* between them.

Without jumping to conclusions, as these are preliminary results of an ongoing work that needs further research, we can just hypothesize regarding this behaviour.

As closer we get to 2008, the closer the countries are plotted, as the European project grew and consolidated its presence in the financial sector. However, after the devastating 2008 crisis, the perception was that the European Union was not so consolidated as people thought. The *distance* between the countries increased as the relationship between them soured and internal conflicts arise as a consequence of the crisis. People's perspective post-crisis was in general an estrangement among the European Union member states and a weaker union than before 2008, and the three figures seem to resemble this divergence.

4.2 Space Visualisation

An interesting way of interpreting the result data is through the plotting of the two-dimensional reduced word space using the process explained on subsections 3.5 and 3.6.

Due to the limited available memory of the machine used in this work, it was impossible to create a matrix and to reduce the high-dimensional space with all the words in the vocabulary. Hence, the vocabulary was reduced to two thousand words. This helped not only to work with the available memory but also with the visualisation of the word space once plotted.

This section is broken into three subsections in order to present and discuss the different periods in which the results are divided.

4.2.1 Pre-2000 Word Space

In Figure 4.4, we can see the representation of the word space created by the 2000 corpus, containing data from the period before the year 2000. The image above shows the plotting of the words, derived from the word space, in which the words are represented as red dots. The picture below shows the plotting of the words and their respective tags.

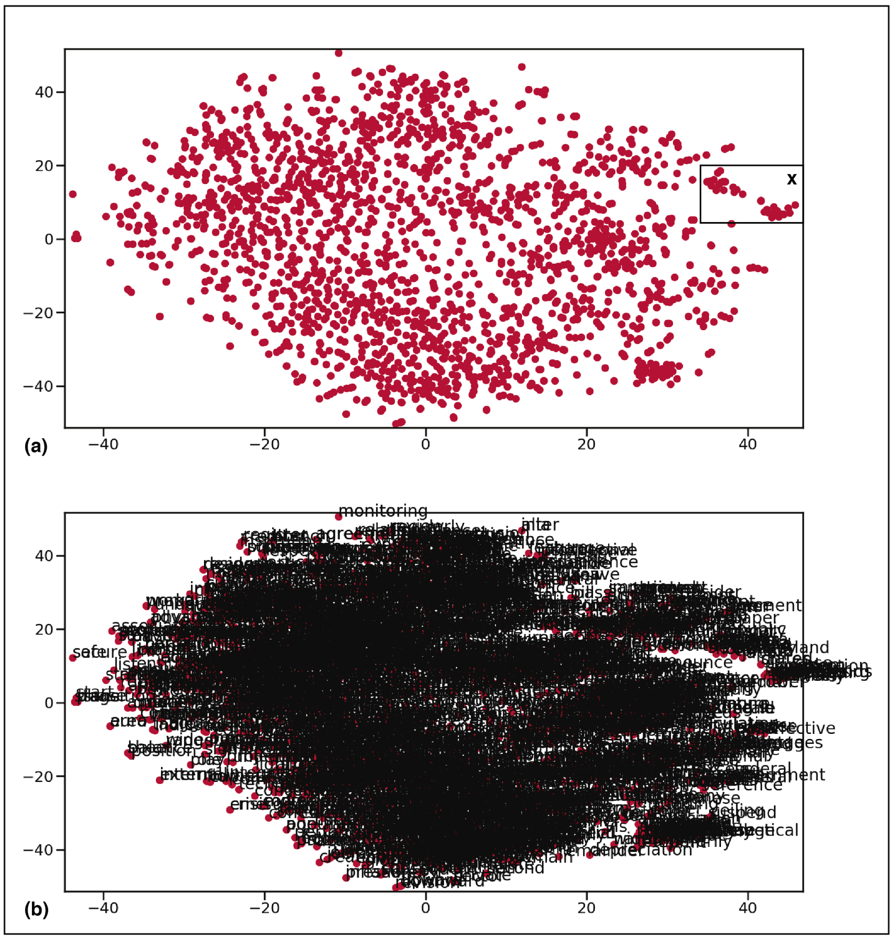


Figure 4.4: Representation of 2000 words from the word space created with the Pre-2000 corpus. (a) Representation of word without tags and Area X containing the countries names. (b) Representation of word with tags.

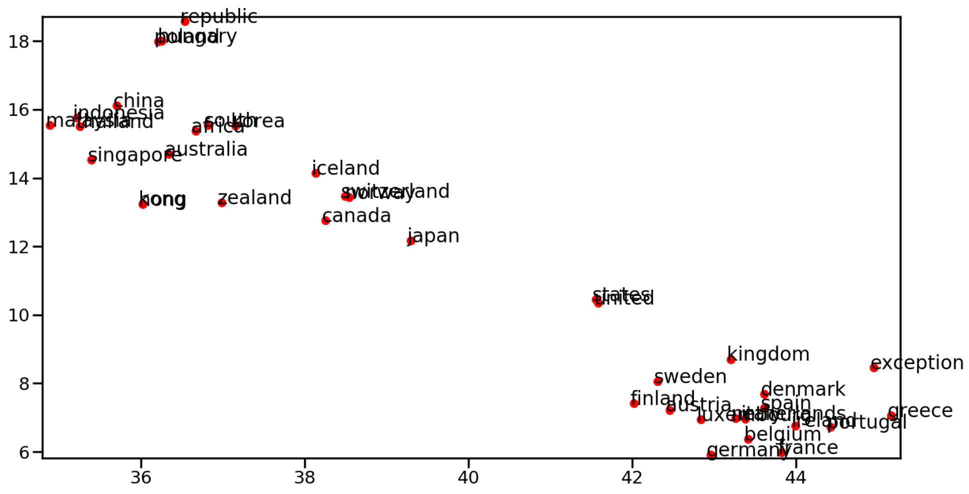


Figure 4.5: Area X from Figure 4.4: Zoomed section of the word space, created with the Pre-2000 corpus, containing the countries' names.

Although difficult to locate, with patience we are able to discover some interesting regions on the plot. One of this zones is located on the top left corner of Figure 4.4, and is displayed on Figure 4.5. This section contains the countries names arranged according to their similarity. We can observe two more densely packed regions in the countries area. One, shown in Figure 4.6, contains the countries belonging to the European Union and the other contains the remaining countries further apart from the Europe region.

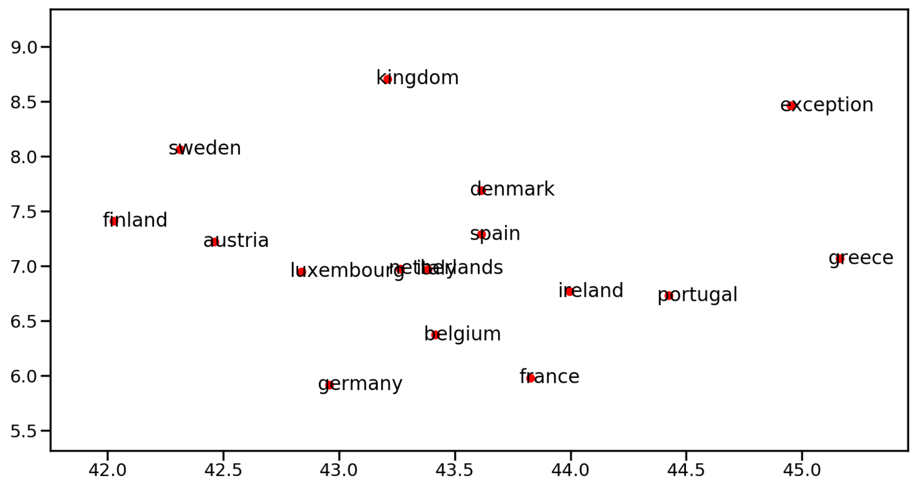


Figure 4.6: Zoomed section of Figure 4.5 containing Europe countries.

4.2.2 2000-08 Word Space

With the word space created over data from the first eight years of the twenty first century, the words were plotted of its vocabulary in Figure 4.7. This figure contains two graphs, one contains only red dots that represent words for the vocabulary, the other contains the tags alongside the red dots.

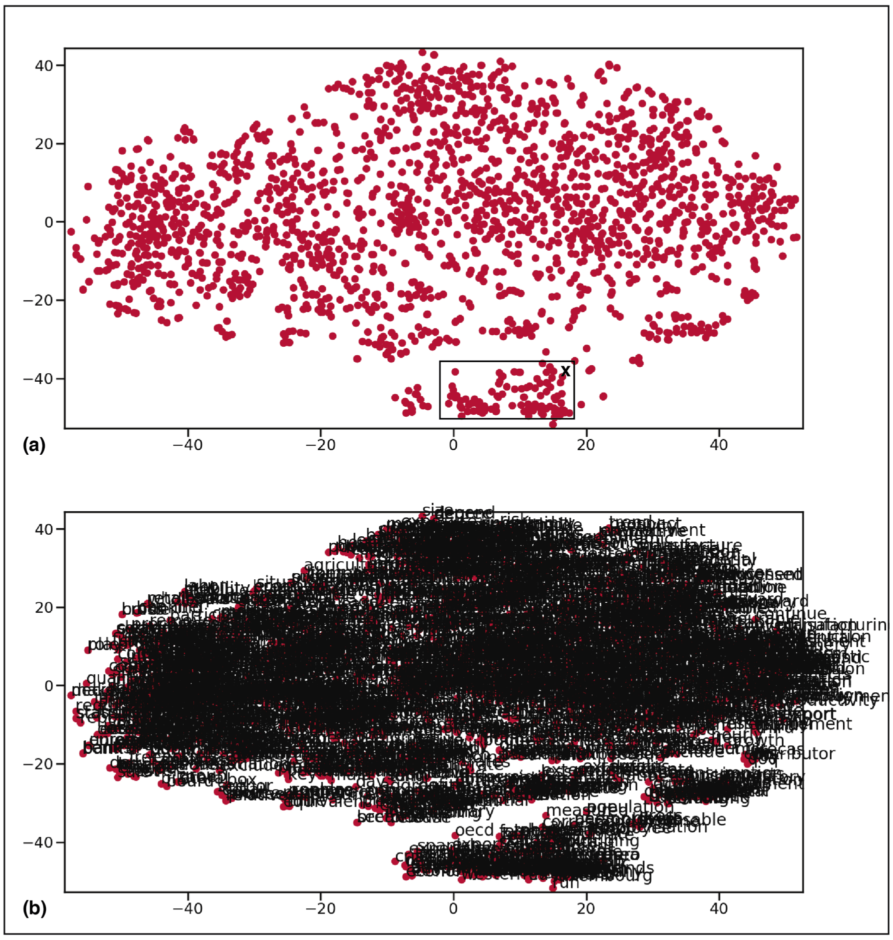


Figure 4.7: Representation of 2000 words from the word space created with the 2000-08 corpus. (a) Representation of word without tags and Area X containing the countries names. (b) Representation of word with tags.

Similarly to the plots of the pre-2000 word space, the names of the countries are all grouped in one region. In this case, the area is located on the bottom of the graph and it is outlined in Figure 4.8. This region can also be divided into European Union member states and Non-members. A zoomed section, containing the European Union member states, can be seen in Figure 4.9.

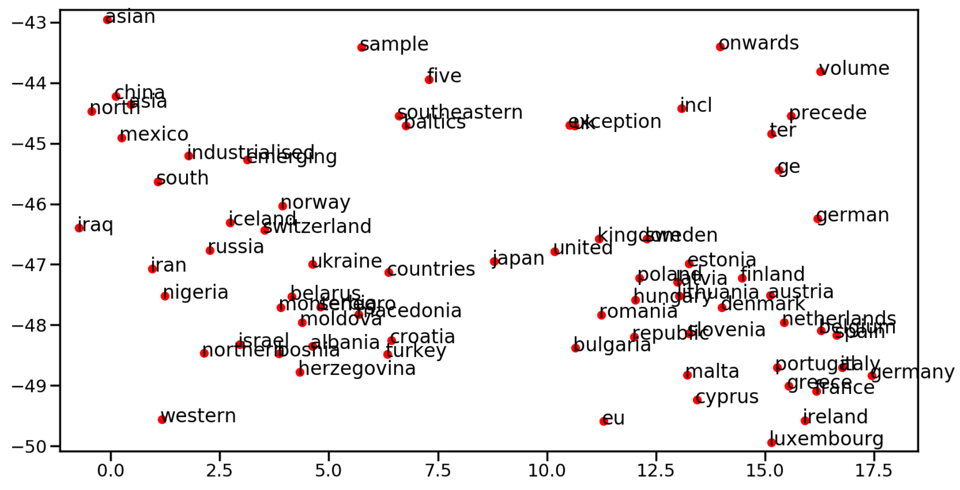


Figure 4.8: Area X from Figure 4.7: Zoomed section of the word space, created with the 2000-08 corpus, containing the countries' names.

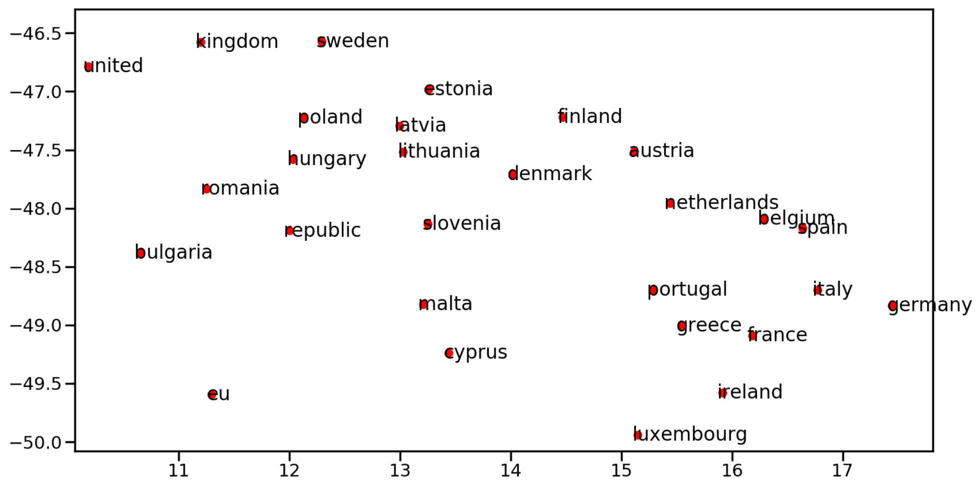


Figure 4.9: Zoomed section of Figure 4.8 containing Europe countries.

It is also interesting to note the smaller region on the center top area of Figure 4.8, depicted in Figure 4.10, containing the words ”uk” and ”exception”. The proximity between these two words might reveal an abiding account of exceptions in UK’s economic behaviour towards the remaining European countries. Portraying a long-lasting rivalry between the United Kingdom and the European Union that eventually lead to their parting in 2020.

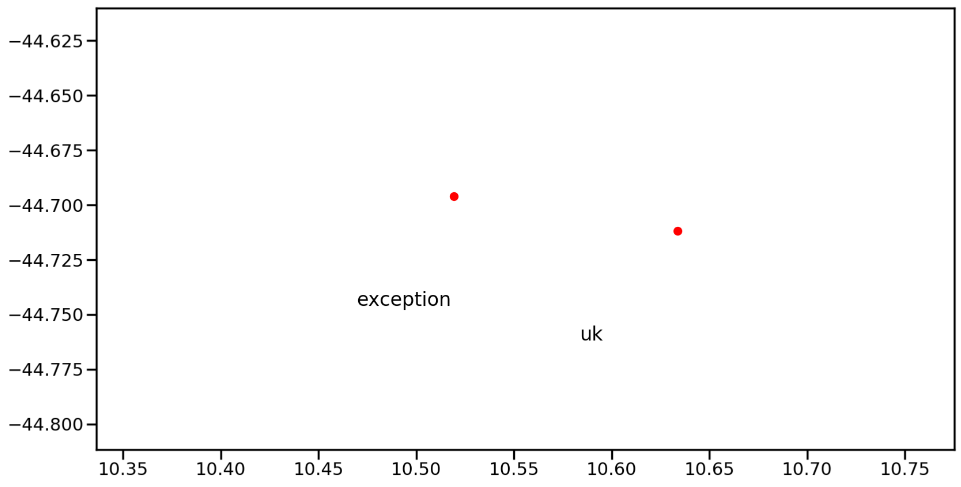


Figure 4.10: Zoomed section of Figure 4.8 containing the word ”exception” and ”uk”.

4.2.3 Post-2008 Word Space

Finally, with the word space created by the data from the aftermath of the financial crisis, the words we plotted from the vocabulary into the graph shown in Figure 4.11. Similarly to the Figures 4.4 and 4.7, Figure 4.11 contains two graphs with the representation of the words with and without the respective tags.

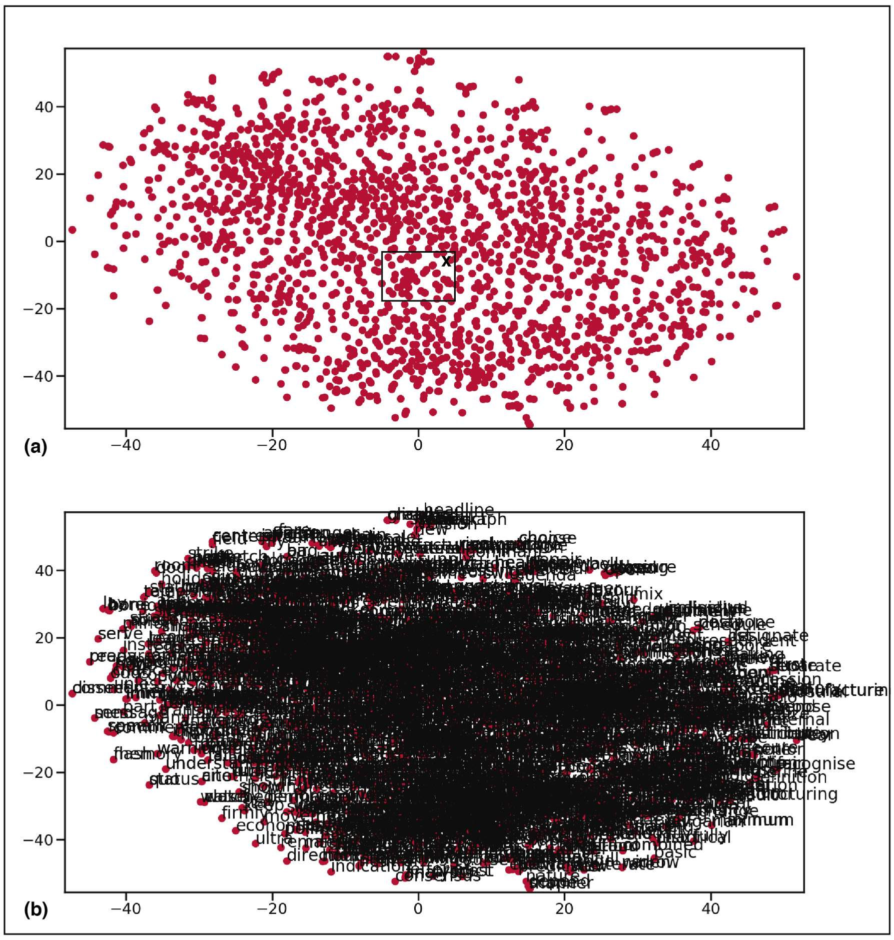


Figure 4.11: Representation of 2000 words from the word space created with the Post-2008 corpus. (a) Representation of word without tags and Area X containing the countries names. (b) Representation of word with tags.

However, this plot displays a different behaviour then those previously shown. In this period, the country names are much more sparsely distributed throughout the plot. As seen in the previous subsection, the contrast in the word space post-financial crisis shows a severe divergence in the representation of the name of the countries compared to preceding periods. The region containing the country names, located at the centre of the graph of Figure 4.11, is plotted in Figure 4.12.

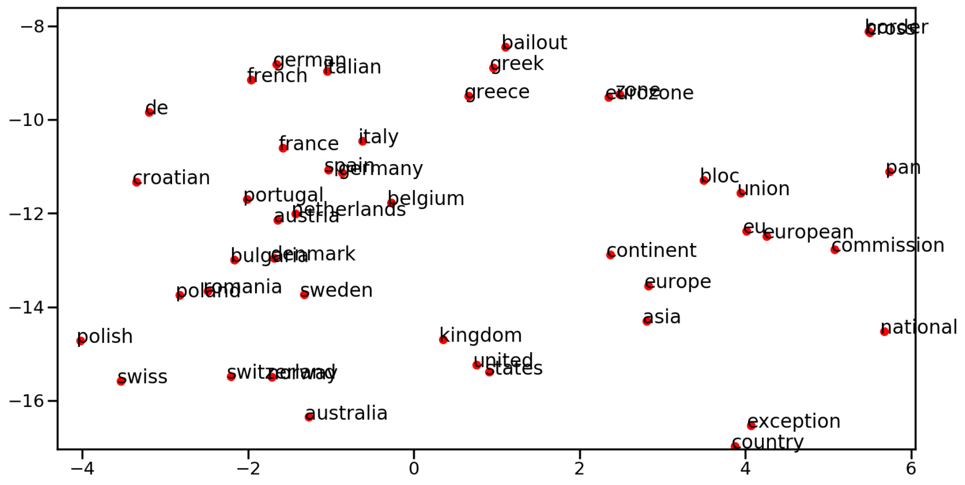


Figure 4.12: Area X from Figure 4.11: Zoomed section of the word space, created with the 2000-08 corpus, containing the countries' names.

The plots of the reduced word spaces shown in this subsection present a similar behaviour of that seen in the prior. In which the objects that represent the country's names came closer to each other until the year 2008, but after the financial crisis occurs a divergence between them.

CHAPTER 5

Final Remarks

This project was set out to build a geometric space based on the automatic interpretation of text about finance and economy so that the space could be used to understand the changes in people's perspective about this topic.

The first step taken was the gathering of the data to be interpreted. For this, financial news and economic reports were cleaned and preprocessed, as explained in chapter three. A Word2Vec model was constructed and trained with the gathered data, creating a multi-dimensional word space. To improve the interpretation of the high-dimensional space, a dimension reduction mechanism was applied to the word space, making it into a two-dimensional space.

The geometric space was successfully built, providing us with interesting results about people's perspective about the economy throughout the years. It is possible to gauge that, in both the measurements of the *distances* and the plot of the word spaces, there is a severe difference in people's perspective about the relation among European countries after the year 2008. During the end of the millennium until 2008, the measured *distances* between European countries seem to follow a pattern of approximation. However, suddenly after 2008, there is an abrupt rupture in the approximation behaviour followed by fast seclusion of every country. This behaviour of the word vectors is also observed in the plots of the reduced word spaces. The 2008 financial crisis might be to blame for this distancing between countries, but further research is necessary to link these behaviours to actual events.

All the materials and procedures are available for public use on `https://github.com/ms-bernardo/Geometries-Based-on-Automatic-Text-Interpretation`.

Further improvements could be performed in order to enhance the results to create a more accurate and responsive word space. These include an increase in data size, specifically on the data before 2008; chose smaller periods of time so more changes in behaviour can be observed; more thorough cleaning and preprocessing of the data, so to reduce the vocabulary and increase the model training accuracy.

This work, although preliminary, could pave the way for new and interesting projects using automatic text interpretation to map the perspectives of people. These projects can further advance our knowledge concerning the financial sector, helping economists predict a crisis or enhance mechanisms searching for fake news, for a more democratic information industry.

In a physics panorama, there are several prospects following this work. With a constructed space, it is possible to, for example, look for symmetries, that according to Noether's Theorem could present us with a conservation law for expanding systems like our word space, or grant a new mechanism for physicists and economists to understand the economy's behaviour.

Bibliography

[1] "Ec-iils joint discussion paper series no. 1: Financial crises: A review of literature," *Economic Policy, CEPR*, Nov. 2011.

[2] M. Almunia, A. S. Bénétrix, B. Eichengreen, K. H. O'Rourke, and G. Rua, "From great depression to great credit crisis: Similarities, differences and lessons," *Economic Policy, CEPR*, 2009.

[3] F. Valencia and L. Laeven, "Systemic banking crises : A new database," *IMF WORKING PAPERS*, Sep. 2008.

[4] M. D. Bordo and O. Jeanne, "Boom-busts in asset prices, economic instability, and monetary policy," *NBER Working Paper Series W8966*, May 2002.

[5] M. Terrones and E. G. Mendoza, "An anatomy of credit booms: Evidence from micro and aggregate data," *IMF WORKING PAPERS*, Sep. 2008.

[6] C. M. Reinhart and K. S. Rogoff, *This Time Is Different: Eight Centuries of Financial Folly.* Princeton University Press, 2009.

[7] L. Einav and J. Levin, "The data revolution and economic analysis," *NBER Working Paper No. 19035*, May 2013.

[8] P. Bacchetta and E. van Wincoop, "The great recession: A self-fulfilling global panic," *American Economic Journal: Macroeconomics*, vol. 8, no. 4, pp. 177–98, Oct. 2016.

[9] F. de Saussure, *Course in General Linguistics.* London: Duckworth, [1916] 1983.

[10] B. B. Mandelbrot, "An informational theory of the statistical structure of languages," in *Communication theory: papers read at a Symposium on "Applications of Communication Theory"*, W. Jackson, Ed., 1953.

[11] H. A. Simon, "On a class of skew distribution functions," *Biometrika*, vol. 42, no. 3/4, pp. 425–440, 1955.

[12] J. P. da Cruz, "Emergent behavior in multiplicative critical processes and applications to economy," PhD thesis, Universidade de Lisboa (Portugal), 2014.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *International Conference on Learning Representations*, Jan. 2013.

[14] L. Saitta, A.Giordana, and A. Cornujols, *Phase Transitions in Machine Learning.* Cambridge University Press, 2011.

[15] K. V. Kalidindi, “Deconstructing word embeddings,” *CoRR*, vol. abs/1902.00551, 2019. [Online]. Available: `http://arxiv.org/abs/1902.00551`, (accessed: 08.11.2020).

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Conference and Workshop on Neural Information Processing Systems*, Oct. 2002.

[17] A. Mnih and G. E. Hinton, “A scalable hierarchical distributed language model,” *Neural Information Processing Systems*, 2008.

[18] G. E. Hinton and S. T. Roweis, “Stochastic neighbor embedding,” S. Becker, S. Thrun, and K. Obermayer, Eds., pp. 857–864, 2003.

[19] L. van der Maaten and G. E. Hinton, “Visualizing data using t-sne,” *Journal of Machine Learning Research*, Sep. 2008.

[20] J. Cook, I. Sutskever, A. Mnih, and G. Hinton, “Visualizing similarity data with a mixture of maps,” M. Meila and X. Shen, Eds., ser. Proceedings of Machine Learning Research, vol. 2, PMLR, 21–24 Mar 2007, pp. 67–74.

[21] Webhose, *Financial news articles*, Sep. 2019. [Online]. Available: `https://webhose.io/free-datasets/financial-news-articles/`.

[22] Reuters, *Reuters-21578, distribution 1.0.* [Online]. Available: `http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html`, (accessed: 16.11.2020).

[23] Finance-And-ML, *News article and full details dataset.* [Online]. Available: `https://github.com/Finance-And-ML/News-Article-And-Full-Details-Dataset`, (accessed: 16.11.2020).

[24] M. Bernardo, *Geometries based on automatic text interpretation.* [Online]. Available: `https://github.com/ms-bernardo/Geometries-Based-on-Automatic-Text-Interpretation`, (accessed: 16.11.2020).

[25] E. Loper and S. Bird, “Nltk: The natural language toolkit,” *Association for Computational Linguistics*, Jul. 2004.

[26] P. S. Radim Řehůřek, “Gensim – statistical semantics in python,” 2011. [Online]. Available: `http://www.euroscipy.org/conference/euroscipy2011`, (accessed: 25.01.2020).

[27] T. Oliphant, “Guide to numpy,” Jan. 2006.

[28] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Association for Computational Linguistics, Jun. 2014, pp. 302–308.

[29] D. E. Rumelhart and J. L. McClelland, Eds., *Parallel Distributed Processing (Volume I: Foundations).* MIT Press, 1986, pp. 77–109.

Appendices

APPENDIX A

Corpus Cleaning

```
import nltk
import pickle

from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.tokenize import word_tokenize
from nltk import sent_tokenize

lemmatizer = WordNetLemmatizer()

ord_ = ['Europe', 'European', 'Union', 'Brexit', 'Parliament',
'Commission', 'Investment', 'Single', 'Market', 'Gulf', 'War',
 'ECB', 'Asia', 'OECD', 'UAE', 'NATO']

# Country List
countries = []
with open('Lists/Countries_upper.txt', 'r') as fp:
    for line in fp:
        countries.append(line.replace('\n', ''))

# StopWords List
stopW = []
with open('Lists/stopW.txt', 'r') as fp:
    for line in fp:
        stopW.append(line.replace('\n', ''))

# Tagging function for name removal
def tagging(phrase):
    phrase = word_tokenize(phrase)
    phrase = nltk.pos_tag(phrase)
    return phrase

```

```python
# function to convert nltk tag to wordnet tag
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

def lemmatize_sentence(sentence):
    nltk_tagged = tagging(sentence)
    wordnet_tagged = map(lambda x: (x[0],
                        nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatized_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            lemmatized_sentence.append(word.lower())
        else:
            lemmatized_sentence.append(
                (lemmatizer.lemmatize(word, tag)).lower())
    return " ".join(lemmatized_sentence)

# Open Corpus
file = open('File.txt', encoding='latin-1')
text = file.read()
file.close()

# Tokenize
sentences = sent_tokenize(text)
text_ = [], trash = []

for sentence in sentences:
    sentence = sentence.replace('-', ' ')

    words = word_tokenize(sentence)

    # Removes all the Non-Alphabetic characters
    words = [word for word in words if word.isalpha()]

    # Remove StopWords
    words = [w for w in words if not w in stopW]

    # Remoce words smaller than 2 characters
    words = [w for w in words if len(w) >= 2]

    aux = [], aux_ = [], sent = []

    for word in words:
        if len(word) > 0:
            try:
                word.encode('latin-1')
                aux.append(word)
            except:
                trash.append(word)

    sent = ' '.join(aux)
    tags = tagging(sent)
    del aux, sent

    phrase = []

    for tag in tags:
        if tag[1] == 'NNP' and tag[0] not in countries
                           and tag[0] not in ord_:
            trash.append(tag[0])

        else:
            phrase.append(tag[0])

    phrase = ' '.join(phrase)


    # Simplify the text
    phrase = lemmatize_sentence(phrase)

    text_.append(phrase)
    text_.append('.')

with open("CLEAN_TEXT.txt", "w", encoding='utf-8') as txt_file:
    for text in text_:
        txt_file.write("".join(text) + " ")

with open("REMOVED_NAMES.txt", "w", encoding='utf-8') as txt_file:
    for row in trash:
        txt_file.write("".join(row) + " ")
```

APPENDIX B

Model Training

```python
from __future__ import absolute_import, division, print_function
from nltk import sent_tokenize
from nltk.tokenize import word_tokenize

import multiprocessing
import os
import re
import gensim.models.word2vec as w2v

# Open Book
file = open('CLEAN_TEXT.txt', encoding='utf-8', errors='ignore')
book = file.read()
file.close()
print("Book loaded!")

raw_sentences = sent_tokenize(book)

def word_tokenizer(raw):
    clean = re.sub("[^a-zA-Z]"," ", raw)
    words = word_tokenize(clean)
    return words

sentences = []

for raw_sentence in raw_sentences:
    if len(raw_sentence) > 0:
        sentences.append(word_tokenizer(raw_sentence))

# Tokens Counter
token_count = sum([len(sentence) for sentence in sentences])
print("This corpus contains {} tokens.".format(token_count))

```

```python
# Train The Model
Model = w2v.Word2Vec(
    sg = 1, #Skip-Gram
    workers = multiprocessing.cpu_count(),
    size = 300,
    min_count = 8,
    window = 8,
    sample = 1e-4
)

Model.build_vocab(sentences)

Model.train(sentences, total_examples=Model.corpus_count,
                       epochs=Model.iter)

# Save The Model
if not os.path.exists("Trained"):
    os.makedirs("Trained")

Model.save(os.path.join("Trained", "Model.w2v"))
```

APPENDIX C

Dimension Reduction

```python
from __future__ import absolute_import, division, print_function
import numpy as np
import gensim.models.word2vec as w2v
from sklearn.manifold import TSNE

# Open the trained model
model = w2v.Word2Vec.load("Model.w2v")
print("Model loaded")

# Vocabulary length
vocab_len = len(model.wv.vocab)
print("Vocabulary length is ", vocab_len)

# Define Matix
word_vectors_matrix = np.ndarray(shape=(vocab_len, 300),
                                    dtype='float64')
word_list = []
i = 0

# Fill the Matix
for word in model.wv.vocab:
    word_vectors_matrix[i] = model[word]
    word_list.append(word)
    i += 1
    if i == vocab_len:
        break


# Compress the word vectors into 2D space
tsne = TSNE(n_components = 2, random_state = 0, metric="cosine")
word_vectors_matrix_2d = tsne.fit_transform(word_vectors_matrix)

```

```python
# Save Matrix
np.save("Mtx_name", word_vectors_matrix)
np.save("Mtx_2d_name", word_vectors_matrix_2d)
```

APPENDIX D

Word Space Plot

```
from __future__ import absolute_import, division, print_function
import numpy as np
import gensim.models.word2vec as w2v
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Open the trained model
model = w2v.Word2Vec.load("Model.w2v")
print("Model loaded")

# Vocabulary length
vocab_len = len(model.wv.vocab)
print("Vocabulary length is ", vocab_len)

# Open the trained model matrix
word_vectors_matrix_2d = np.load("Mtx_name.npy")

# Open the multi-dimensional matrix
word_vectors_matrix = np.load("Mtx_2d_name.npy")

word_list = []
i = 0
for word in model.wv.vocab:
    word_list.append(word)
    i += 1
    if i == vocab_len:
        break

# Word points DataFrame
points = pd.DataFrame([
    (word, coords[0], coords[1])
    for word, coords in [
        (word, word_vectors_matrix_2d[word_list.index(word)])
        for word in word_list
    ]
], columns=["Word", "x", "y"])

sns.set_context("poster")
fig, ax = plt.subplots()

# Plot the word points
ax.plot(points.x, points.y, 'ro', markersize=15)

# Defining Axes
offset = 1.0
ax.set_xlim(min(points.x) - offset, max(points.x) + offset)
ax.set_ylim(min(points.y) - offset, max(points.y) + offset)

# Plot the point tags
k = 0
for i, j in zip(points.x, points.y):
    corr = -0.05  # correction for annotation in marker
    ax.annotate(points.Word.values[k], xy=(i + corr, j + corr))
    k += 1

plt.show()
```