# 

**Text Classification/Analysis Using Neural Networks and Word2Vec: How To Use The Technique**

|  |  |
| --- | --- |
| **Author:** | John Rey Juele |
| **Student ID:** | 15167798 |
| **Supervisor:** | Farshad Toosi |

**Final Year Project Report**

B.Sc. Computer Systems

Department of Computer Science and Information Systems

Academic Year – 2018/2019

Table of Contents

[0](#_Toc5194671)

[Project Summary 2](#_Toc5194672)

[Chapter 1: Introduction 3](#_Toc5194673)

[1.1. Context/Motivation 3](#_Toc5194674)

[1.2. Objectives 4](#_Toc5194675)

[1.3. Technologies 5](#_Toc5194676)

[Chapter 2: Research 7](#_Toc5194677)

[2.1. Artificial Neural Networks 7](#_Toc5194678)

[2.1.1. The Inspiration 7](#_Toc5194679)

[2.1.2. The Application 8](#_Toc5194680)

[2.2. Word2Vec 12](#_Toc5194681)

[2.3. Doc2Vec 15](#_Toc5194682)

[2.4. Application of Research/Plan of Approach 15](#_Toc5194683)

[Chapter 3: Implementation 16](#_Toc5194684)

[3.1. Dataset Acquirement 16](#_Toc5194685)

[3.2. Text Pre-processing 17](#_Toc5194686)

[3.3. Doc2Vec & Word2Vec Implementation 19](#_Toc5194687)

[3.4. Software Architecture & Design Patterns 19](#_Toc5194688)

[3.4.1. MVC (Model View Controller Architectural Pattern) 19](#_Toc5194689)

[3.4.2. Strategy 19](#_Toc5194690)

[3.4.3. Factory Method 19](#_Toc5194691)

[3.4.4. Command 19](#_Toc5194692)

[3.4.5. Observer 19](#_Toc5194693)

[Chapter 4: Testing & Discoveries 20](#_Toc5194694)

[Chapter 5: Conclusion & Analysis 20](#_Toc5194695)

[References 20](#_Toc5194696)

[Image Sources 21](#_Toc5194697)

[Dataset sources 22](#_Toc5194698)

# Project Summary

For my final year project, I am developing an application that will take a piece of text or a set of texts as input and will then label them with a topic or subject that best describes them. This project makes use of concepts and ideas that are brought out in the fields of machine learning and natural language processing and the objective is to use these to create software that fulfils a need in real life.

During the lifetime of this project I am looking into developing an application that incorporate software models capable of performing classification tasks, of which I am paying close attention to neural networks and the theory and the operations that make them work. I am also including “Word2Vec” (Mikolov, et al., 2013), a relatively recent technique devised by a team of researchers at Google. This technique converts words into numerical representations which can in turn help the computer process and understand language while at the same time keeping the semantics of the words intact. Word2Vec can also be extended to cover not just words but entire documents with the help of “Doc2Vec” (Mikolov & Le, 2014).

As the machine learning algorithms that I am using utilises a supervised style of learning, it is required to obtain a good set of training data (Brownlee, 2016). This can range from news articles, academic papers to non-formal texts sources such as Twitter and other social medias. There is no limit to the sources I can pull from as long as they are appropriately labelled due to the fact that the performance of the classification task greatly depends on the quality of the training data. The pre-processing of the text is also required in order to help the machine efficiently handle the text input we feed into it. This may include stemming the words, removing unnecessary punctuations and other pre-processing techniques on text data.

One of the goals of this FYP is to produce working software and therefore I will also be considering the characteristics and best practices to developing good software. I am approaching this project similar to how a business would approach such a development problem by keeping in mind the functional and non-functional requirements, devising a plan of action and to incorporate design patterns so that the application can remain in line with the “SOLID” design principles to software development.

# Chapter 1: Introduction

## Context/Motivation

As we are living in a digital age, information has become widely distributed and therefore easily accessible. Even though we have not completely moved on from physical books and other paper-back publications, more emphasis has been placed on making information available electronically. 80% of the information available to us, both in physical and electronic form, is in text format (Korde and Mahender, 2012) .

Helping computers understand and process human language has been an active area of research in computer science especially in the field of natural language processing (NLP). One of the subareas of this field is in the classification of text. Computers are being used as tools to automate work especially in analysing and processing text with a view of classifying them. With this in mind, text classification is a contributory building block in many NLP applications. Following on from this, the technological environment today has encouraged researchers to place more attention in incorporating and implementing mechanisms of text classification in everyday, practical tasks. We can see this in action with the prevalence of machine learning today regarding web searching, sentiment analysis and data filtering (Aggarwal and Zhai, 2012), as well as in digital assistants such as Apple’s Siri and Amazon’s Alexa.

It has always been an interest of mine to dabble in the field of machine learning and to gain an understanding on how the resources of computers can be manipulated to mimic and simulate human-like intelligence and learning. Communication and language is an integral part of being human as well as of society as a whole. I am curious to discover the nuts and bolts that operate underneath the machine learning technology of today, especially on how they perform human-like tasks such as understanding text.

## Objectives

The main goal of this project is to develop an application that can read the text provided to it and to label each one as appropriate topic or subject. This goal can be broken down even further to three major milestone or sub-goals.

1. **A training and test dataset must be founded and readily available.** I used a supervised style of learning for my classification model. Therefore it was vital that these dataset be correctly labelled and was suitable for the task at hand. Upon obtaining the dataset, they should also be pre-processed. This includes but is not limited to stemming the words in the document, removing unnecessary punctuation and “stop words”.
2. **Develop the program using the SOLID design principles.** For the implementation of the program I intend to follow a 3-tiered closed architecture as well as including the MVC (Model View Controller) architectural pattern. It is also a goal of mine to add in necessary design patterns with a view of accommodating quality attributes such as flexibility and extensibility. The program will of course include a graphical user interface that will display the labelling. I plan to design the GUI as user friendly as possible by making it easy to use, clear and organised.
3. **Test and evaluate the end product**. Upon finishing the implementation, I will create different test cases for the outputs of the program. I also aim at evaluating the classification model that I will be incorporating into the program and to determine the accuracy of said model. Using the data from the tests I will be trying to tweak the different parameters of the program with a goal of increasing and improving the accuracy of the classification model.

This FYP introduces new concepts and ideas that I have not encountered before nor do I have an strong understanding of. Therefore I have personal goals that I set for myself. Python is a programming language that I have no knowledge of prior to this FYP. I hope to achieve a strong level of competency of the language through the project. Machine learning is an area of computer science that I find interesting, yet I have very little knowledge of. My means of this project, I intend to increase my current knowledge base in this area as I believe it would prove practical in the real world.

## Technologies

The below are the main technologies that I used for the FYP.

**Python**

Python was the programming language that I predominantly used to implement the application. The main reason why I decided to use python is because I did not use python previously and the FYP is an opportunity to learn and utilise the language in a meaningful way. Another reason why I chose python is the many libraries available as well as the documentation and tutorials that are available online.

**PyCharm**

PyCharm is the IDE that I used for this project. It is one of the IDE’s provided by Jet Brains who also have an IDE available for the Java programming language called “IntelliJ”. I am very familiar with IntelliJ and the features and settings in IntelliJ are also available in PyCharm and it is because of this that encouraged me to use it.

**Python Libraries**

* **Gensim**

Gensim was used to help create my Word2Vec and Doc2Vec models. According to the Gensim website its purpose is to “realize unsupervised semantic modelling from plain text”. The library provides the functionality in creating the models, training the models, finding similar words, obtaining document vectors as well as some useful utility functions such as text cleaning.

* **NumPy**

NumPy is the library that I used to perform the mathematical operations within my project. The library also makes available a more efficient implementation of arrays and matrices which I found to be useful in my research and in my own implementation of a simple neural network.

* **Scikit-Learn**

Scikit-learn is a library that provides already made implementations of many machine learning algorithms. This library was particularly helpful in the creation of my classification models as well as the evaluation of such models.

* **Beautiful Soup**

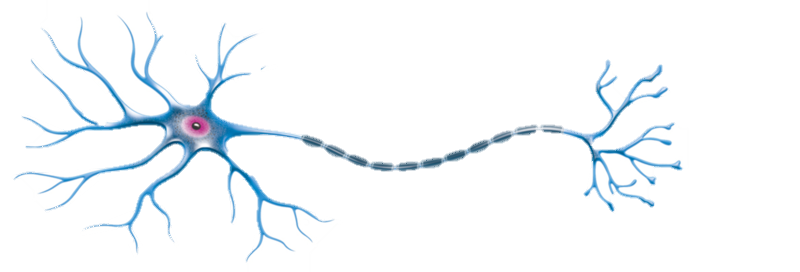
Beautiful Soup was the library that I used to parse through HTML files and to extract data from them. I used this library alongside python’s in built **urllib** package to automatically obtain information from websites and to create my dataset.

# Chapter 2: Research

## 2.1. Artificial Neural Networks

An artificial neural network (ANN) is an “information processing paradigm” that takes inspiration from the workings and the operation of the biological brain (Stergiou and Siganos, n.d.). ANN’s are not necessarily an algorithm; that is it follows a specific and concrete set of steps to solve a particular problem. They can be seen more as a paradigm or a framework as to how to structure and conceptualise the program.

### 2.1.1. The Inspiration

Artificial neural networks are inspired by our brains which are comprised by a large network of neurons. A neuron is just a single nerve cell that transmit electrical signals from one end to the other.

Axon

Cell Body

Nucleus

Dendrites

Terminals

Fig. 1. A Neuron

In the case of the diagram above, the electrical signal would enter the neuron through the dendrites which will then be processed inside the nucleus. It is only upon reaching a certain threshold that an output signal would be produced which is then sent through the axon and exists through the synapses within the neuron’s terminals (Rashid, 2016, pp. 41-43).

### 2.1.2. The Application

We can model an artificial neuron like so:

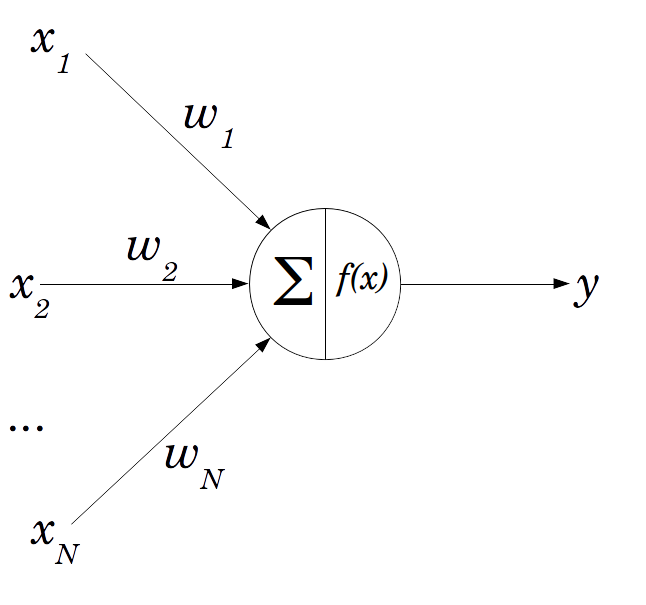


Fig. 2: A Perceptron

The official name of an artificial neuron is a “perceptron” which can act as a single processing unit of an ANN (Rosenblatt, 1958). The main components of the perceptron are:

* **The weighted inputs**

There can be N inputs to this neuron which are also weighted. The weights on each connection signifies the strength of that connection and therefore influences the processing and ultimately the output of that perceptron. These weights are assigned random values initially. They can be seen as the dendrites from the biological neuron.

* **The neuron**

Within the neuron itself is where the processing takes place. Two main processes take place. The “∑” or sigma signifies the summation of the inputs in the form:

As stated earlier under the biological brain, a certain threshold has to be met before the output is fired out of the neuron. This is modelled using an activation function: where we pass the sum into that function. There are many types of activation functions such as the step function and the sigmoid/logistic functions (Rashid, 2016, pp. 44-45). The result of that function will be that neuron’s output or in other words its *activation value.*

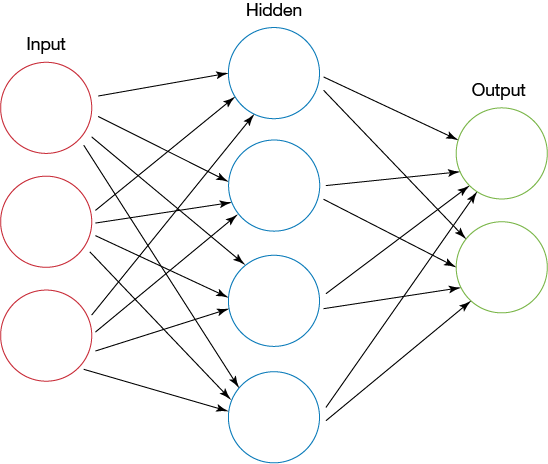
Just like the brain is a network of neurons, we can model a network of perceptrons like so:

Fig. 3: A feed forward neural network

Each circle is a neuron/perceptron where the output of that neuron will act as the input of the next neuron. I should also note that connections between neurons are weighted (as weights are not included on the above diagram). In a typical neural network architecture, there are three layers: the input, hidden and output layers. The number of nodes in the output and input layers are dependent on the problem domain while the number of nodes in the hidden layer is arbitrary.

In order to produce a guess out of the neural network, it must perform a feed forward mechanism through the network. Once executed, the activation value of each node in each layer is calculated until processing reaches the output nodes. The activation value of the nodes of the output layer is the neural network’s guess. I should also point out that there is a more efficient way of performing the feed forward algorithm. Everything on the neural network can be represented as matrices (Nielsen, 2018).

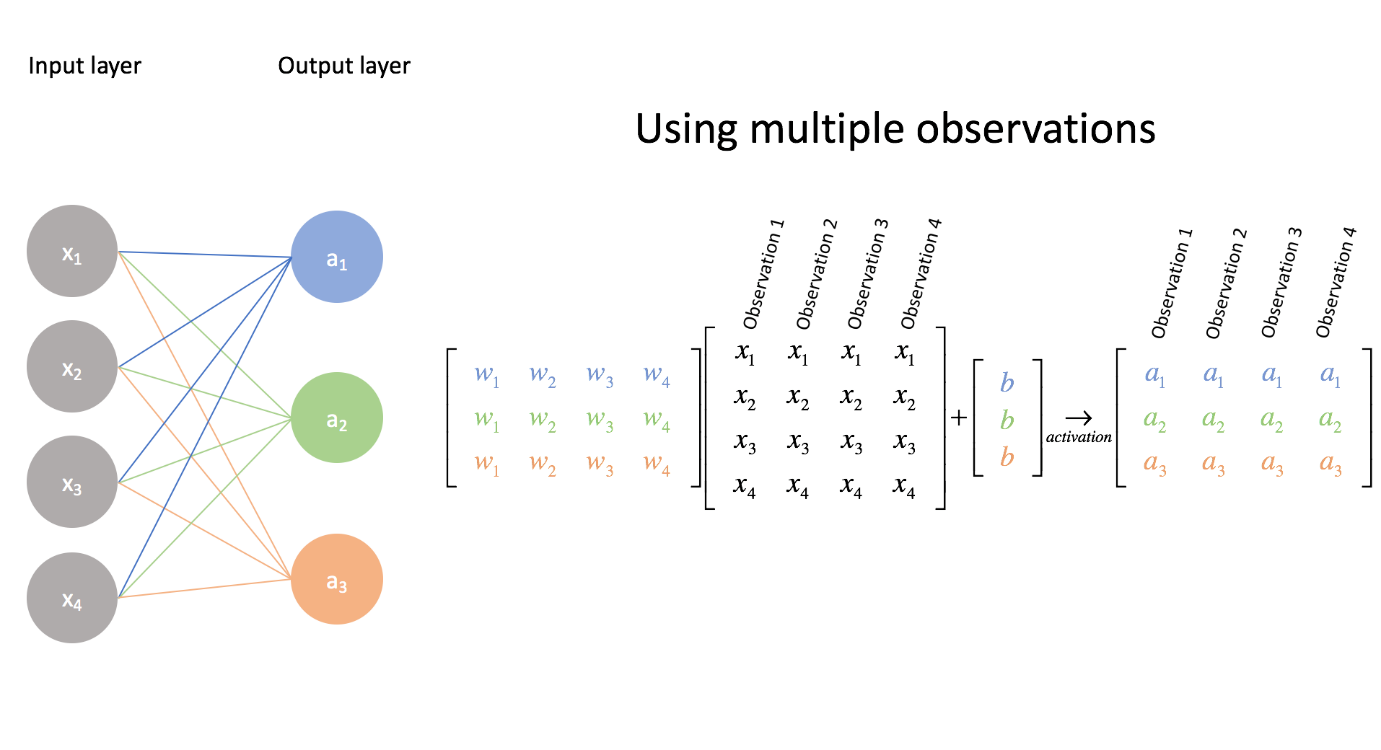


Fig. 4: Matrix representation of the NN

In order to find the activation values of the nodes in the next layer, a dot product operation can be done between one matrix with another. With the above diagram, we perform the operation between the matrix representing the weights between the input and hidden layer and the matrix representing the input layer in order to produce the activation values in the hidden layer. The diagram above has not included the activation function however. So upon obtaining each values from performing the multiplication, we apply the activation function on each of the values in the result in order to find activation value of the nodes. We can do the same operation in order to find the output layer values. Through linear algebra operations we can simplify the feedforward algorithm especially programmatically.

However running the feed forward on a neural network for the very first time may not operate and produce a result that is expected. That is due to the fact that it is untrained. The neural network must learn from already existing data in order to help it produce a more accurate guess. As I have pointed earlier, the weights between each layer greatly influence the activation values of the next neuron. We can also see them as the decision makers of the network. As the weights are initialised with random values, they will likely produce inaccurate activation values in the next node. So from that idea, when training the neural network, what really is happening in the background is that the weights are being adjusted towards a more accurate value.

A concept found in supervised learning is the idea of errors. This is very similar to how humans learn from making mistakes. We compare our work with the actual answer and to use the mistakes we have learnt to adjust our current knowledge. So in the case of neural networks, we make it produce a guess for a particular set of inputs and to compare its guess with the actual answer that is correlated to that set of inputs. Let’s put this mathematically:

Upon discovering the error of the output layer (as the error produced from the overall guess of the NN is the error of the output layer), we use those errors to traverse or propagate back towards the input layer of the NN while at the same time finding the errors of each node in each layer. This is where the mechanism of “backpropagation” comes in (Klein, 2011).

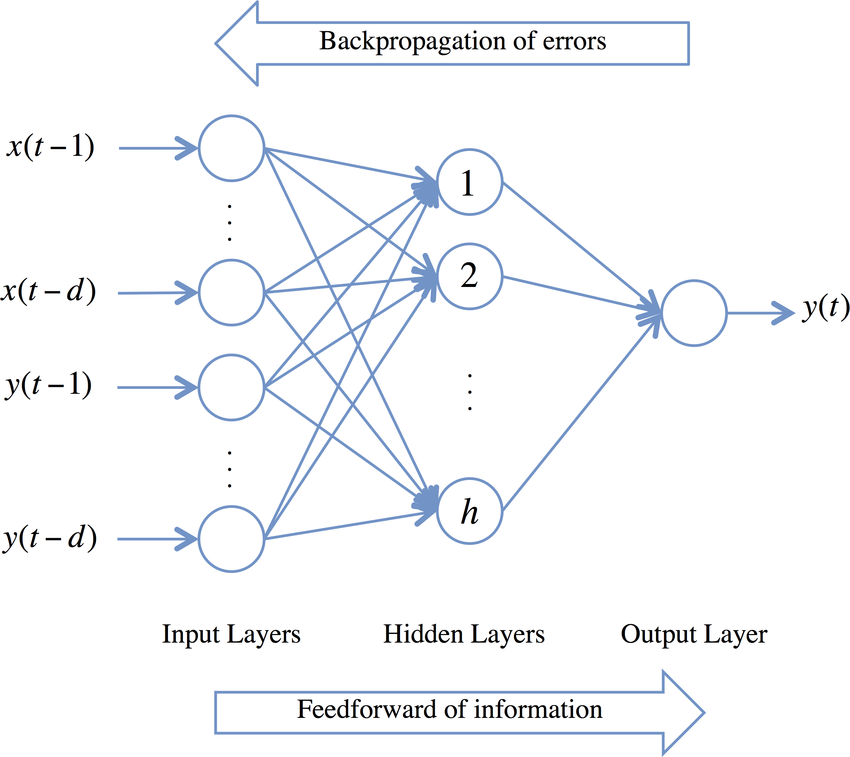


Fig. 5: Feed Forward & Backpropagation

Once we have obtained all the errors of each node in the neural network it will help us to figure out how much we should adjust each of the weights in the networks. The goal of the learning process is the minimize the errors and get them as close to zero as possible. This brings up a new idea of “gradient descent” (Rashid, 2016, pp. 83-93). From this discussion we can conclude that the effectiveness of a neural network is not determined fully by the efficiency of the implementation but by the quality of the training data. As this is a supervised model, the guesses produced by neural networks depend fully on the data it learnt from.

Gaining an understanding on the functionality of a neural network is very important for this project as it is the foundation of many of the core features of the project application. We can see one in use in the next section with regards to Word2Vec.

## 2.2. Word2Vec

Word2Vec is a mechanism that converts words into “word embeddings” or numerical representations of words (TensorFlow, 2018). There are many different ways of creating word embeddings such as TF-IDF vectorisation, count vectors and co-occurrence matrices. However, what is great about Word2Vec is that it maintains the semantical relationships that exist between words. We can illustrate this by performing matrix calculations between word vectors. For example, the word vector produced by:

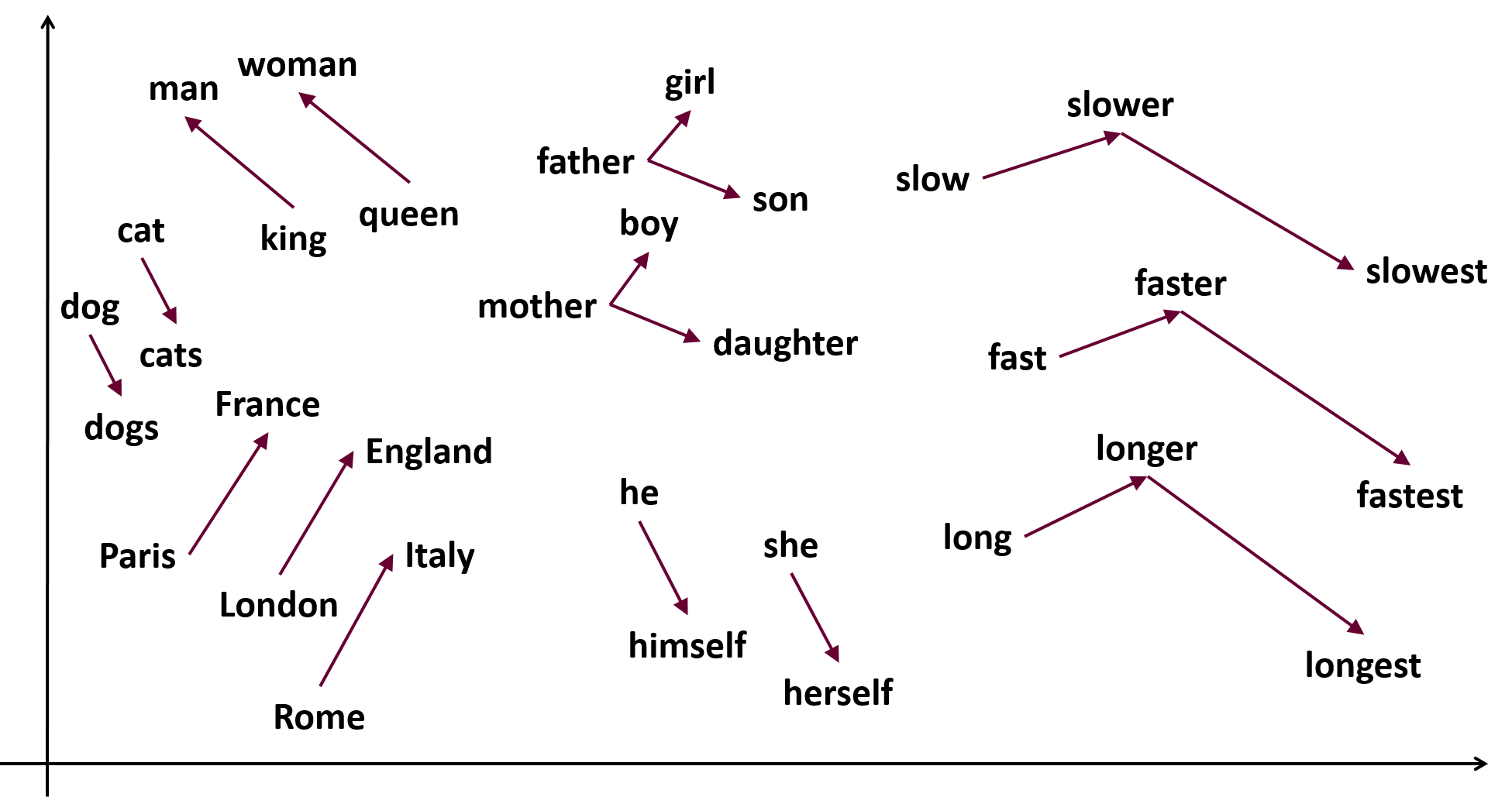
is a vector that is very similar to the word vector of “Queen” (Mikolov, et al., 2013).

Fig. 6: Vector relationships between words

Word2Vec makes use of shallow neural networks (NN with only one hidden layer) that are trained to do a specific purpose. The two main purposes are described by the two architectures of Word2Vec: **Continuous Bag of Words** and **The Skip-gram model**.

**The Skip-gram model**

The skip-gram model is a neural network that is trained to predict the neighbouring word (context words) given a specific word in the text corpus.

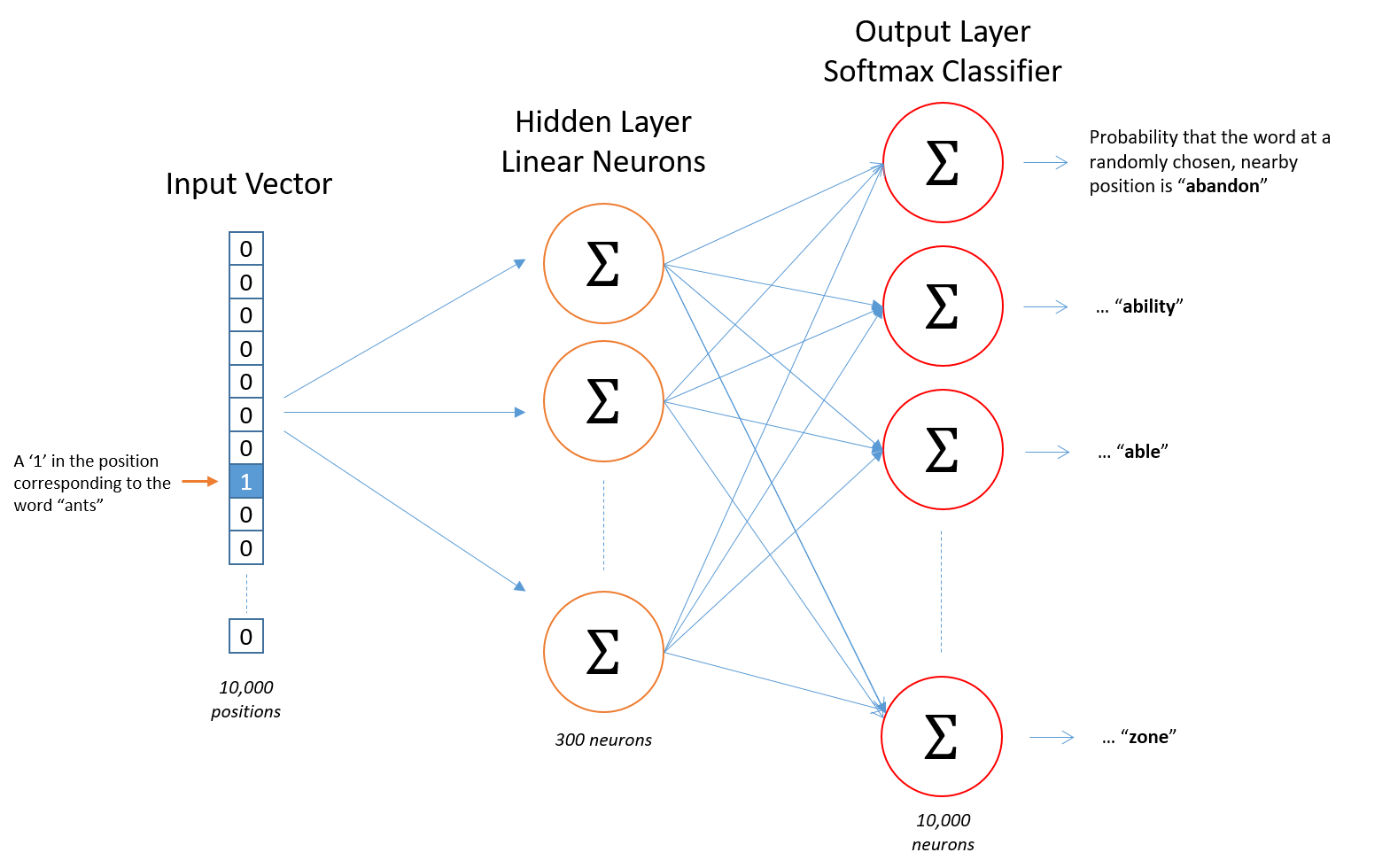


Fig. 7: The skip-gram architecture

**Continuous Bag of Words (CBOW)**

Continuous Bag of Words is a neural network that is trained to guess a particular word given the word(s) that surrounds that word.

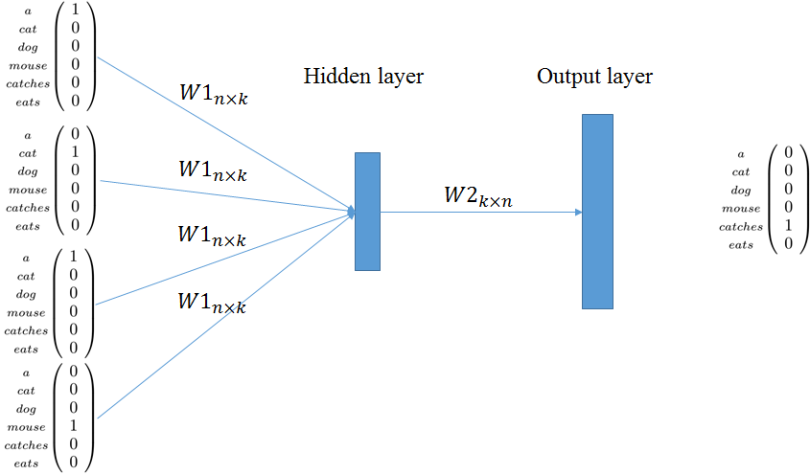


Fig. 8: The CBOW architecture

As these neural networks are being trained to fulfil their target functions, the weights connecting each layer in their networks are being adjusted towards a more accurate value. It is from these weights that we can derive the word vectors for the words in the text corpus (McCormick, 2016). As I have pointed earlier everything on the neural network can be represented as matrices.

Fig. 9: The weight matrix representing the weights between the input and hidden layer of a hypothetical Word2Vec model

We can also represent the input word to the NN like so

Fig. 10: A one-hot representation of the input word “Computer”

In order to find the word vector for the word “Computer” (As an example) we perform a simple dot product between the two matrices.

By doing such an operation, what we’re really doing is that we’re selecting the word vector from all the stored weights representing all the possible words in the corpus and this selected word is projected into the hidden layer. So from the above example the word embedding for the word computer is:

## 2.3. Doc2Vec

## 2.4. Application of Research/Plan of Approach

# Chapter 3: Implementation

## Dataset Acquirement

As stated earlier, the quality of the performance of the application depends heavily on the quality of the dataset I use to train it with. So the first step I took was to find a reliable source of text data that is appropriately and correctly labelled. The ideal dataset for me is when the text corpus:

* Is labelled
* Contains documents such that the context of that document accurately describes the label that is put onto it.
* Is to the point and exact. In other words, the text should revolve around its subject matter while simultaneously trying to avoid mixing other topics/subject into itself.
* Has little to no spelling mistakes or other grammatical errors.

I should point out however that documents may have correlations between each other with regards to subject matter or topic. Therefore avoiding the mixture of different topics in the same document cannot be 100% avoided. For example, a document that talks about “Google” can dabble between the topics of “Tech” and “Business”. I embraced this fact while searching for a dataset source. However complications arose when this is accepted too much and I will talk about this more in depth in the “Conclusions” section of this report.

I wanted to have a large supply of text data as a large corpus can lead to better output accuracy. My initial plan was to use academic papers from sources such as “Nature.com”, “IEEE”, “ACM” as the papers precisely stick to their subject matter, are labelled and contain very little to no grammatical errors as they are typically proof-read. Unfortunately, it was not at all simple to acquire documents from these types of sources. As I needed a large amount of documents, I was obliged to automate the acquirement of the dataset as downloading or acquiring the documents manually will be an impossible task to do. However, this approach cannot be done for sources I mentioned above as payment is required to get the documents. It is possible to obtain these documents for free as a student of an university but due to the fact that I needed to automate the process, it was not possible to login as a student by means of a python script.

I was provided with a dataset from a member of the department that contained labelled news articles from BBC (Greene & Cunningham, 2006). This was close to what I wanted but it only contained about 2,000 documents which is not enough for what I wanted. I stumbled upon another dataset on Kaggle (Misra, 2018). This dataset contains a JSON file which had about 200,000 lines of objects in this format:

These json objects represent articles from “Huffpost”. This json file appealed to me as it mentions the link to the main article as well as it being labelled. The dataset contained many topics but I only chose 6 of them: “BUSINESS”, “POLITICS”, “SCIENCE”, “SPORTS”, “TECH” and “TRAVEL”. I wrote a python script that parsed through each json object, obtained the link and its category (topic) and scraped the website for its text using the obtained link. My script then saved the text in a .txt file and placed it in the folder whose name is equivalent to the category of the current json object. In this manner, I did not have to manually copy paste the text from the news articles by hand. At the end of the script, I had 6 folders named after the topics that I chose containing about 23,000 .txt files. This then acted as my dataset for this project.

## Text Pre-processing

Once the dataset is gathered, a mechanism was needed to be set in place in order to “clean” the text that was to be fed into the software system. There are multiple techniques that can be used to pre-process text and these include (Ganesan, 2019):

* Lowercasing
* Stemming
  + “Chopping off” the ending of words in the hopes of obtaining the root word. For example, “-ing”, and “-ed” would be chopped off from words such as “ending”, “ended” to get the word “end”.
* Lemmatization
  + Similar to stemming such that the goal is to get the root word. However where it differs is that it also encompasses the special cases. For example the word “better” would be reduced to the word “good”.
* Stop-word removal
  + Removal of words that do not carry any a lot of meaning. These words/letters include “a”, “I”, “is”, “are” etc.
* Normalization
  + Transforming text to a standard format. Different situations may call for different format standards.
* Noise removal
  + Removing characters or text that is unwanted. Example of this are HTML or XML tags, “hashtag” symbol in tweets etc.

The techniques above does not have be implemented together and the techniques that are used will depend on the situation you are in. In my case I am using the Gensim library that provides Word2Vec and Doc2Vec functionalities. The library also include text cleaning functions. I used the function called “simple\_preprocess” which converts all the words to lowercase, removes HTML tags, numbers and stop-words and all punctuations. Once all of that is done, the function tokenizes the text or in other words, the text becomes an array/list of words. For example the text: “I was too <a>little</a> to reach 50cm above me, oh well!” would converted to: ['was', 'too', 'little', 'to', 'reach', 'cm', 'above', 'me', 'oh', 'well']. These text cleaning techniques used by Gensim proved to be sufficient for my use in the project.

## Doc2Vec & Word2Vec Implementation

## Software Architecture & Design Patterns

### MVC (Model View Controller Architectural Pattern)

### Strategy

### Factory Method

### Command

### Observer

# Chapter 4: Testing & Discoveries

# Chapter 5: Conclusion & Analysis

# References

Aggarwal, C. C. & Zhai, C., 2012. A Survey of Text Classification Algorithms. *Mining Text Data,* January.pp. 163-222.

Ganesan, K., 2019. *How to Perform Text Preprocessing for Machine Learning & NLP? | Kavita Ganesan.* [Online]   
Available at: http://kavita-ganesan.com/text-preprocessing-tutorial/  
[Accessed 15 January 2019].

Greene, D. & Cunningham, P., 2006. *Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering,* Pittsburgh: ICML.

Klein, B., 2011. *Backpropagation in Neural Networks.* [Online]   
Available at: https://www.python-course.eu/neural\_networks\_backpropagation.php  
[Accessed 16 Oct 2018].

Korde, V. & Mahender, N., 2012. Text Classification and Classifiers: A Survey. *International Journal of Artificial Intelligence & Applications (IJAIA),* 3(2), p. 85.

McCormick, C., 2016. *Word2Vec Tutorial - The Skip-Gram Model.* [Online]   
Available at: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/  
[Accessed 20 October 2018].

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. *Efficient Estimation of Word Representations in Vector Space.* Arizona, Proceedings of Workshop at ICLR.

Mikolov, T. & Le, Q., 2014. *Distributed representations of sentences and documents.* Beijing, JMLR.org, pp. 1188-1196.

Misra, R., 2018. *News Category Dataset.* [Online]   
Available at: https://rishabhmisra.github.io/publications/

Nielsen, M., 2018. *Using neural nets to recognize handwritten digits.* [Online]   
Available at: http://neuralnetworksanddeeplearning.com/chap1.html  
[Accessed 15 Oct 2018].

Rashid, T., 2016. *Make Your Own Neural Network.* Germany: CreateSpace Independent Publishing Platform.

Rosenblatt, F., 1958. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review ,* 65(6), pp. 386-388.

Stergiou, C. & Siganos, D., n.d. *Neural Networks.* [Online]   
Available at: https://www.doc.ic.ac.uk/~nd/surprise\_96/journal/vol4/cs11/report.html#References  
[Accessed 24 December 2018].

TensorFlow, 2018. *Vector Representations of Words.* [Online]   
Available at: https://www.tensorflow.org/tutorials/representation/word2vec  
[Accessed 17 October 2018].

# 

# Image Sources

**UL LOGO:**

<http://www.mmrrc.ul.ie/dotnetnuke/portals/6/UL_Logo_1ver.png>

Fig. 1:

<https://pngimage.net/nerve-cell-png-5/>

Fig. 2:

<https://www.lucidarme.me/simplest-perceptron-update-rules-demonstration/>

Fig. 3:

<https://www.ibm.com/developerworks/library/cc-artificial-neural-networks-neuroph-machine-learning/index.html>

Fig. 4:

<https://www.jeremyjordan.me/intro-to-neural-networks/>

Fig. 5:

<https://www.researchgate.net/figure/Feedforward-Backpropagation-Neural-Network-architecture_fig3_303744090>

Fig. 6:

<https://www.samyzaf.com/ML/nlp/nlp.html>

Fig. 7:

<https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b>

Fig. 8:

<https://towardsml.com/2018/06/12/understanding-word-embeddings/>

# Dataset sources

D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML 2006.