Automatic Stance Detection

DT228
BSc in Computer Science

Michael Lenghel

C16434974

Dr. John Gilligan

School of Computer Science

Technological University, Dublin

08/03/2020

Abstract

This project develops a system which can automatically detect the stance of media outlets in relation to a particular topic such as Brexit. It uses techniques from opinion mining and topic modelling in order to identify objectivity and bias. The stance, or slant commonly referred to, of a media outlet refers to a position that the media presents to the public in relation to specific topics.

As an example, depending on the varying political views that certain news-reporting organisations prescribe to, the topic of Brexit has been reported on extensively with mixed reviews. Through automatically determining which news companies have a positive, neutral or negative view towards certain topics, bias can be uncovered and news sources which do not provide an objective view will be easier to disregard as lacking credibility or biased.

The range of applications of such a system is also explored. Areas of particular interest include fake news detection, how sentiment in the media towards the market correlates to stock trends, interview analysis and ethnography.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Michael Lenghel

08/12/2019

Acknowledgements

I would like to thank and acknowledge the generous support of my thesis supervisor Dr. John Gilligan for his encouragement, advice and guidance provided throughout the development of this project and dissertation. I would also like to thank my family and friends for their continued support and feedback during the writing of this thesis and project development.

Table of Contents

Table of Figures

1. Introduction

Stance detection is the measurement of whether the author of a text is in favour of a topic or against it. Stance detection is viewed as a subtask of opinion mining that lies above sentiment analysis. The divergence between sentiment analysis and stance detection is that stance detection systems determine the favourability towards a given target or topic, even if it is not explicitly mentioned within the text and generally, sentiment analysis classifies text into positive or negative, irrespective of topic. An example of stance detection is the overall portrayal of a topic such as Brexit, whereas an example of sentiment analysis is classifying a text into positive or negative, regardless of the content that was discussed.

This introductory chapter will look at introducing the components which facilitate stance detection. These techniques include:

- Natural Language Processing Techniques
- LDA Topic Modelling
- Data Sourcing News Articles by News Organisation
- A Novel Approach to Labelling Topics Generated by the LDA Topic Model
- Stance Detection Techniques to link the sentences that discuss similar topics
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model
- Linking the words generated by the topic model to their original sentences
- Opinion Mining Techniques including Sentiment Analysis
- Displaying this Information Clearly to the User

This thesis plans to unite the above components in order to be able to determine how a newspaper company reflects a specific stance towards certain topics and finally the necessity of a stance detection algorithm to determine bias and inform the public which news companies are credible or untrustworthy.

The ultimate objective of this project will be the ability to determine bias or neutrality within media outlets and display this information to the user with the aid of clear visualisations.

1.1. Project Background

1.1.1. Automatic Stance Detection

Opinion mining and sentiment analysis, deals with the computational treatment of opinion, sentiment and subjectivity in text [1] and will be the driving force behind automatically deriving the stance of news articles and ultimately the opinion score within each sentence that relates to a respective topic.

Opinion mining is the science of using text analysis to understand the drivers behind public sentiment, whereas sentiment analysis examines how people perceive a topic in a more basic form such as positive or negative. Opinion mining goes a level deeper in order to deduce the driver behind the specific response of the individual. Opinion mining in essence, is applied through the use of

monitoring hundreds of thousands of sentences and capturing the general consensus of how certain topics are perceived through applying a multitude of techniques on sentences that discuss the same topic.

A basic example of opinion mining is finding all of the positive and negative reviews that discuss a product and discovering if there are more negative or positive reviews. A basic example of sentiment analysis is checking the specific sentence: "The Omega toaster is built from quality metals and is highly durable" and classifying the sentiment into a category such as positive or negative. In this case the sentiment would be classified as positive.

The stance detection proposed in this thesis will be capable of not only determining the traditional three states "negative", "neutral" and "positive", but will go a step further and represent five states including "Very positive" and "Very negative".

At a low level, techniques of sentiment analysis will be used in order to generate individual sentiment scores for each text that refers to a topic to create the more complex stance score. Since the conception of sentiment analysis, the area has undergone a large shift of alterations. The cornerstone of this area where there was a manifestation of research was in 2004 where there was an immense increase of subjective text on the web. Consequently, 99% of the papers on sentiment analysis have been published after 2004 [2]. Since then the area has evolved significantly, so have the terms dictating this area. Within the inception of the concept of stance analysis it was originally coined "Points of View Analysis" which then as of early 2020 has evolved to "Stance Detection" and "Slant Detection".

There have been multiple methods developed in order to further improve the accuracy of stance detection and improve the underlying sentiment detection models. These methods range from:

- Manually tagging words sentences with their intent as implemented with *Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis* [3]
- Rule based systems such as the implementation within *Lexicon-enhanced sentiment analysis framework using rule-based classification scheme* [4]
- Supervised and unsupervised machine learning models through neural networks and deep learning which has been implemented within the research of the paper titled *Twitter Sentiment Analysis with Deep Convolutional Neural Networks* [5] published in late 2015.

This area is still evolving with many problems still without flawless solutions such as interpreting sarcasm as well as understanding grammatically incorrect language. It also must be noted that while a text can address a large number of topics, stances may be separate based on different topics. As a result, a number of opinion mining techniques are required in order to extract the individual topics discussed and calculate the individual stances for each topic.

Latent Dirichlet allocation (LDA) topic modelling developed by Blei in 2003 [6] has grown to be known as one of the most used topic modelling algorithms. LDA is essentially a generative probabilistic approach to build topics from multiple documents using word frequency. Its use within natural language processing has increased significantly in the last number of years and it is also a major area of research within computer science.

LDA is predominantly used by organisations to analyse qualitative data such as interviews and surveys automatically by building topics in conjunction with thematic analysis. Using LDA to build

topics in various media platforms such as twitter and newspapers is a more recent trend and currently an immense point of discussion within computer science.

Topic models, which in this context provide the function of grouping sentences that discuss a particular topic, identify latent patterns of word occurrence using the distribution of words in a collection of documents [7]. Topic modelling will be responsible for the automatic creation of topics through correlating the phrases or words within a body of texts that relate to a specific set of topics using statistical word distribution. The LDA topic model will create unlabelled topics, which dictates that while the words link to a logical topic, the topic name itself is not labelled due to the complexity of accurately defining a category to represent a topic.

Both approaches described above will use machine learning in order to build models that can create topics (through aggregating the sentences that fall into a topic) and understand the individual sentiments of each sentence. An average is then accumulated for each sentence that is linked to its respective topic in order to discover the overall stance towards that topic for the respective media outlet.

Traditionally, the topics that are created by the LDA topic model are not labelled. Instead arbitrary numbers are used to represent the topic names, referred to as imaginary names, and the researcher interprets and labels the topic manually as discussed within *Reading Tea Leaves: How Humans Interpret Topic Models* [8]. The interpretation of topics is performed through reading the most used words within the topic. As an example, if the words are "Trump", "Campaign" and "election" it is likely that the topic in question refers to the US president.

One of the objectives that developed from the implementation of this project is the requirement of a novel approach for the automatic labelling of topics. Labelling imaginary topics generated by an LDA model is a complex problem that to this day still has no optimal solutions.

Many approaches that attempt to label LDA topic models have been implemented including *Understanding LDA in Source Code Analysis* [9] which offers the approach of using the most dominant topic words to label the topics, but this approach has been shown to not always clearly represent and define the topic by David M. Blei within the peer-reviewed article Probabilistic Topic Models. [10] Multiple supervised and unsupervised approaches have also been implemented such as *Automatic Labelling of Topic Models [11]* which have attempted to use external data sources to label topics based on probabilistic searches. These attempts returned more correlative results than simply using the most dominant topic names, but heavily rely on the topics being well defined and distinct. The approach used within this project looks to learn from the previous attempts and implement a more accurate method of labelling the topics through its novel labelling algorithm which will be a major component of this thesis. The literature review will analyse and discuss approaches in more detail in order to form a better understanding of how to implement a more accurate automatic topic labelling algorithm.

A specific example of the operation of automatic stance detection within this project is finding all of the sentences within articles that discussed Brexit and determining the sentiment score on each occurrence through the use of subjectivity and polarity scores. Subjectivity will be ranked with a score between "0" and "1" where "0" is very objective and "1" is very subjective and this will test how neutral news companies are when discussing various topics. Polarity (a score between negative one and positive) one will test whether the statements are positive or negative towards the topics in

question where "negative one" is a strongly negative sentiment and "positive one" is a strongly positive sentiment. Based on the objectivity and strength of the sentiment scores it becomes straightforward to dictate whether or not specific news companies are biased towards topics such as religion, Brexit, political leaders or any other topic that is discussed at length within articles.

1.1.2. Value of Stance Detection

According to the Joint National Readership Survey (JNRS) 2012/2013 almost three million people in Ireland read newspapers regularly [12]. Within *The vicissitudes of attitudes and similar representational constructs in twentieth century psychology [13]* McGuire hypothesised the effects mass media has on the general public. McGuire noted on the intended effects of the media which were:

- (A.) The effects advertising has on purchasing
- (B.) The effects political campaigns have on voting
- (C.) The effects of public service announcements on personal behaviour
- (D.) The effects of propaganda on ideology and finally
- (E) The effects of media ritual on social control.

A recent example that pertains to (B.) the effects political campaigns has on voting can be demonstrated within the amount of money that different political candidates spend on voting. According to a congressional hearing that took place in November of 2017, Facebook executive Colin Stretch revealed the amount of money spent by Donald Trump and Hillary Clinton in the 2016 elections to be eighty-one million. [14] On the other hand, it is estimated that Bernie Sanders had spent less than five million on his ad campaign and it is widely believed that this was one of the key reasons Bernie Sanders lost the nomination in 2016.

In the book *Media Effects and Society*, the author Elizabeth Perse makes the case that in 1992, 206 billion was spent on advertising within the media and that it logically makes sense that if such a large portion of finance is dedicated to media outlets, then it logically has a quantifiable effect on the public. [15] In 2019, 563 billion dollars was spent on advertisements in just the United States of America alone. [16]

Based on the evidence provided, the topics represented by the media can undoubtedly influence and skew public opinion. The manner in which various issues such as mental health, political movements and technologies are presented is crucial in understanding the position that a news company is attempting to uphold in regard to that topic and this can become an issue where a newspaper company is biased towards certain topics.

A need for automatic media stance and bias detection is also evident through the large amount of research work residing within this area. Multiple dissertations, research papers and conferences have been dedicated to unlocking more powerful methods to process media information and extract more meaningful data.

The key components within this thesis that separates it from other projects is the methods employed to create the stance detection algorithm through the combination of both a sentiment

analysis model and LDA topic model, as well as a novel labelling of topics through the mapping of imaginary topics to real topics. Many sub-optimal solutions exist to the aforementioned components. An issue that relates to the labelling of topics is that it can be almost impossible to guarantee a generic topic name that fits what the sentence is describing if it is not well defined and discussed or too generic. Many attempts at guided machine learning models, word vectorisation and scraping different sources all attempted to find solutions to this problem but still have many problems.

1.2. Project Description

Automatic Stance Detection is a project that will investigate how popular news-reporting organisations in Ireland and abroad portray information to the public. *The paper Joint Sentiment / Topic Model for Sentiment Analysis* [17] defines stance detection as the act of detecting sentiments and topics simultaneously.

The main focus of this project is to measure media stance within various news reporting outlets to uncover bias towards different topics. This objective introduces new requirements in order to facilitate its successful execution such as the introduction of an accurate LDA topic model, an accurate sentiment analysis model, the extraction of news article through API's or web scraping where it is not available and ultimately a novel approach in order to label and define topics. This dissertation will also discuss the relevant applications of such a system. These applications include fake news detection, media stance to stock trends correlations and ethnography.

This project will source several thousand news articles from various news companies and attempt to identify the stance presented by the news companies towards a specific set of topics. Targeting which sentences discuss what topics will be performed using LDA topic modelling and then at a low level, sentiment analysis will be used to score each sentence into its positive, neutral or negative counter parts.

Stance detection is used in this project to portray the views held by the media to specific topics. Within this project, opinion mining techniques such as aspects of sentiment analysis are the key elements which aid in understanding the objectivity and sentiment within each sentence of the news, while LDA topic modelling is responsible for mapping the sentences to their appropriate topic. Sentiment analysis works by first applying natural language techniques in order to clean sentences of information that are irrelevant for analysis. A host of techniques such as creating bigrams, lemmatizing words, removing stop words that do not carry any weight when it comes to measuring sentiment such as "to" and "or" and many more. The quality of data cleaning greatly effects the accuracy of the sentiment model results and the more thoroughly techniques are applied, the greater the accuracy when calculating the stance detection scores for thousands of sentences.

The data cleaning performed to be able to determine the sentiment of a single sentence is as follows.

- Tokenization where each text is broken up into individual words.
- Stop word filtering where all stop words which are essentially words that carry no sentiment such as "to", "is" and "I" are removed from the text.
- Useless information such as emails and numerical values such as dates are removed.
- Stemming and lemmatization is performed where suffixes and prefixes are removed to transform the word into its base state, for example the word "going" is transformed to the word "go".
- A classification algorithm which determines the class a sentence belongs to, either positive, neutral or negative.
- Appling the sentiment class to the proposed sentence.

Figure 1 Stage of Developing Sentiment Analysis Algorithm [18]

Sentiment analysis identifies subjectivity and polarity from a text where a subjectivity score is how subjective a sentence is and the polarity is whether or not a sentence was positive, negative or neutral. There are many different approaches to a sentiment analysis implementation. These approaches include hand-crated rule-based systems as well as more advanced automatic approaches that use machine learning techniques. Note that opinion mining covers more categories then sentiment analysis and that more techniques will be employed in order to successfully implement the automatic stance detection system, particularly topic modelling which is the main source of complexity within this project.

Within machine learning sentiment analysis is modelled as a classification problem, where a classifier is input text and returns a category such as "positive", "negative", or "neutral". More sophisticated sentiment analysis classifiers return a score between two integers which allow further granularity in order to understand the differences between "very positive or very negative" and "slightly positive or slightly negative".

The models learn to associate a particular input with a particular sentiment score based on the sample data that is provided for training the model. The text input is converted to a feature vector that the model can interpret and the larger the data set, the more accurate the model can become. Naïve Bayes or neural networks are example of classification algorithms that can be used to understand the text.
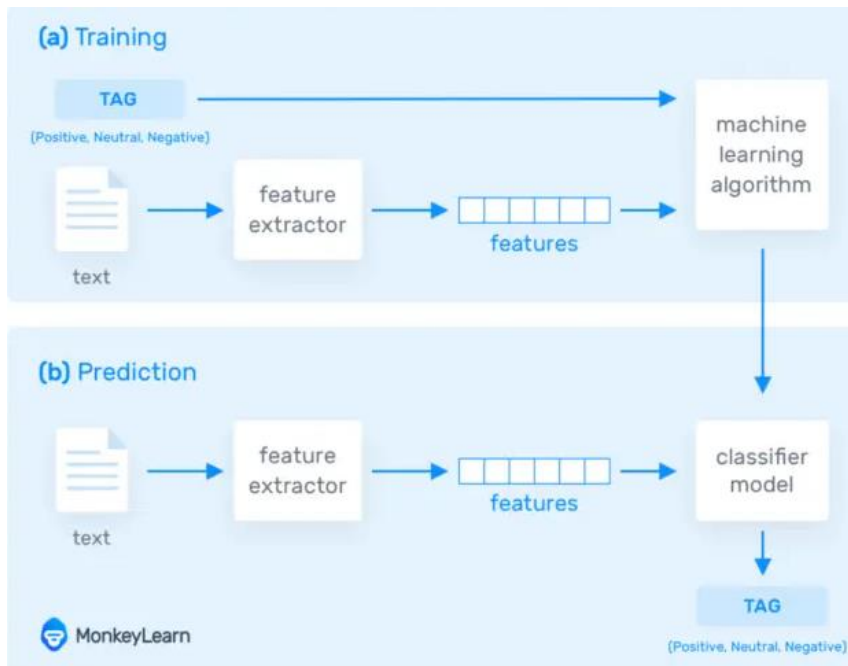
Figure 2 Machine learning classifier model [19]

Grouping sentences to certain topics is handled by LDA topic modelling which is a growing area within natural language processing and machine learning. Topic modelling can be used to discover the abstract topics that occur in a collection or group of documents through examining the themes in a large collection of documents. A more formal definition of topics is that a topic is a multinomial distribution over a fixed vocabulary [20].

I will be using topic modelling to build topics within newspapers articles. The topic modelling algorithm of choice that I will be using is LDA which is a probabilistic method. LDA maps all of the words within documents to topics in order to match them to imaginary topics. A more formal definition of LDA is "LDA is a generative statistical model that allows certain sets of observations to be explained by unobserved groups that may explain why some parts of the data are similar." [21].

The true power of LDA is that it can build a model that correlates certain words with a topic which can save a great deal of time as ordinary thematic analysis would require analysing texts by hand and manually mapping certain words which link to a topic. A basic example is that if in an interview a key topic that a company is interested in is streaming services, then the words correlated with streaming services could be "HBO, Netflix and Amazon prime" and all other streaming services deemed important. This process would then need to be applied to every topic which can be cumbersome where the topic count can be within the hundreds.

The approach that I am taking is to build a reference model to multiple topics such as religion, sports and politics, be able to expose when a news source is referencing a particular topic and then calculate their stance on this topic through a stance score where they discussed a particular topic across multiple papers. Afterwards, different newspapers such as "The Independent" and "The Daily Mail" can be compared on how they discuss this topic to determine if one holds a stronger stance on certain issues, favours certain political parties over others or even upholds a bias, be it negative or positive. Naturally this stance will be also be calculated over an aggregation of multiple media

16

outlets and therefore will be able to pinpoint how the media as a whole within the selected news companies perceive certain topics.

An example of a recent research project done within the scope of LDA is the journal publication [22] "Data Analysis and Visualization of Newspaper Articles on Thirdhand Smoke: A Topic Modelling Approach". Within this research a topic model is built on whenever third hand smoke is mentioned within the Chinese Media. This model was then used to understand the role the media plays in communicating this health concern.

The below diagram shows how LDA topic modelling generates topics. Firstly, it chooses a distribution of the topics which is displayed through the histogram on the right and then for each word chooses a topic assignment (which is represented as the coloured coins) and then selects the words corresponding to each topic.
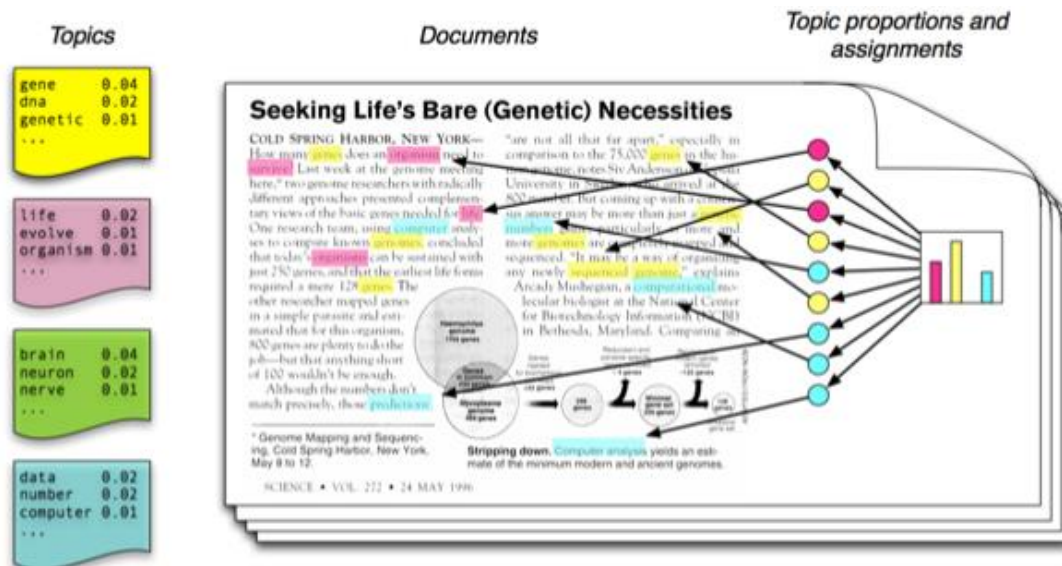
Figure 3 LDA Topic Modelling Through Mapping Words to Topics [6]

1.3. Project Aims and Objectives

The main objective of this dissertation will lead to the creation of an automatic stance detection system that can measure the stance or favourability of news outlets in regards to a specific set of topics. This system will implement a novel approach to label topics and provide clear graphs to display this information.

What makes the automatic stance detection in this project have value is that it was developed in order to provide new insights into how media organisations represent their views towards specific targets or topics and whether or not they are biased when discussing a specific set of topics. Performing this form of analysis manually is extremely time consuming and the ability to systematically determine stance within several thousand news articles according to topic could take weeks for a single news organisation, whereas it could take an hour for this system.

The breakdown of the components that facilitate automatic stance detection are as follows:

- Data Sourcing Several Thousand News Articles Per News Outlet
- Data Cleaning using Natural Language Processing Techniques
- Designing the LDA Topic Model
- A Novel Approach to Labelling Topics
- Stance Detection Techniques to Retrieve the Original Sentences from the Words Generated within the Topic Model
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model
- Finding and Implementing an Existing Sentiment Model
- Improving both Models
- Linking Components to Create the Stance Detection System
- Creating a Suitable Display to Graph the Data

The sourcing of data will be the first step in training the models. Initially, to prototype the LDA and sentiment models, a data source such as "20 News Groups" will be used. Once a working model has been developed, the next iteration will look to build up a large corpus of several thousand articles per news outlet either through external API's provided by news outlets or through the manual scraping of news articles through REST services.

The next phase requires the cleaning of data for both the LDA model and the sentiment Analysis model. Multiple techniques are used within natural language processing in order to remove irrelevant information that cannot be analysed. An example of some of the techniques include:

- Tokenization
- Stop word filtering
- Removal of emails, punctuation and any undescriptive characters
- The creation of bigrams, trigrams and four grams to connect words that are commonly used together and ensure context can still be understood in the case of topic models.

The next step is to build a suitable LDA model that can select themes from the articles and link the words of those themes to those models. There are a wide range of methods to implement this feature and many different libraries that can both optimise this model as well as improve the quality of models created. The LDA model that is built requires guided topics in order to be able to choose the topics that are created. This will need to be performed in order to allow different news outlets to be compared across the same topics.

While all of the data cleaning for the sentiment model will be performed and fine-tuned within this project, the classification of the sentiment model will be built using external libraries. Tests have shown that the classification model for sentiment analysis is difficult to fine tune accurately due to the large amount of edge cases that cause many misinterpretations such as sarcasm, grammar and spelling mistakes. In order to increase the accuracy of the LDA topic model and mapping of the imaginary to real name topics, less focus will be placed on the classification modelling algorithm for sentiment analysis and a well-rounded, accurate sentiment analysis model will be used.

Following the creation of an LDA model, the words that are linked to a topic map to an arbitrary number and the topic itself is said to be "unlabelled". This means that while the words link to a real-

world topic, the model does not label the name of the topic as a number of underlying issues relate to categorizing a specific topic to the top weighted words. An example is labelling the words "women, pregnancy, abortion". It is clear that the topic falls into either abortion or pregnancy but it is difficult to identify exactly which it is without looking at the context of the sentences. The intent will be to find a novel approach to the labelling of this topic.

Another component that is required is linking the specific words from the LDA model back to their original sentences. This step is critical as opinion mining techniques such as sentiment analysis will need to be performed on the sentences in order to obtain more accurate scores.

A method of visualization for the LDA topics will be another key objective in order to be able to accurately determine how accurately words map to their respective topics, as well as how strongly linked certain topics are to each other. A future feature that will be added if time is available is the introduction of mapping sentiment correlations to stock trends. This area is a major area of growth and could provide some key insights. For the following reason, a time series based graphing system will need to be implemented in order to allow for these new types of graphs within the future.

The stance or stance analysis score will need to be mapped to the created topic inside the LDA model which may prove to be a challenge as a great deal of processing separates the two methods and some ingenuity to map them together may be required.

Another key objective will be to evaluate the quality of the topics created through the LDA model, as well as how accurate the stance analysis scores were for each topic by comparing their accuracy in understandings topics to the accuracy of random participates. These participates will be required to fill out a questionnaire where they try to map the given words to a topic that the LDA algorithm created and see if they arrived at the same conclusion as the model.

Finally, the last objective is to further increase the accuracy of the models through better data cleaning, reducing the number of edge cases where the algorithm loses accuracy, increase the data size so that the model can be trained more accurately, creating the optimal number of topics and many more techniques that are commonly used in natural language processing and machine learning in order to further improve the accuracy of both the LDA topic model and sentiment analysis scores.

1.4. Project Scope

This system will not be able to make direct assertions about news-reporting organisations. The implementation will instead create scores that provide a probabilistic approach with some degree of error to dictate how positively or negatively the media and specific news outlets depict certain topics and it is then up to the user of the system to make a more profound judgement on the implication of the results. A negative score does not necessarily imply that the media is against this topic as it may indicate that they talked about it negatively in the past and that they do not hold the same position.

There are no plans to create this system on a remote server that will be hosted within a web application. Everything within this system will be ran locally on a machine that sets up the

environment using the documentation that is provided. The results will be displayed through a Grafana interface in order to be understood more clearly. The key results are the objectivity of a news outlet in relation to various topics, as well as their polarity.

The implementation of the system will be targeted towards the English language. The system will not look towards extensibility for multiple languages.

1.5. Thesis Roadmap

This section will provide a summary of each of the chapters covered within this report.

2 Literature Review

This chapter explores the background research related to LDA topic modelling, stance detection, opinion mining techniques such as sentiment analysis, the visualization of the models, sourcing real news articles separated by topic through web scrapers or API's. This chapter will also discuss the array of other relevant research that has been performed within this area of natural language processing and machine learning.

3 Experimental Design

The design chapter delves into and breaks down the different software methodologies that were considered and the thought process that was used to arrive to a conclusion. A view of the system architecture, use-cases and sequence operation will then be presented. A planned approach will also be discussed and analysed prior to the development stage.

4 Experimental Development

The development chapter describes the entire development process of the system using the technical architecture that was described within the design section. The challenges and issues encountered will also be broken down within this chapter providing both the solutions and trade-offs.

5 Testing and Evaluation

This chapter will describe how the system was tested and evaluated. The key components that will be tested will be the accuracy of the LDA model to create topics, as well as the accuracy of the opinion mining techniques. Both components will undergo multiple forms of testing and evaluation that will compromise both computer-only generated scores and how well the machine learning can resemble a real individual's ability to manually understand topics and stance. The evaluation of the individual components of the stance detection system will reflect on the accuracy of the system.

6 Conclusions and Future Work

The final chapter will reflect on the entirety of the project and will discuss the conclusions drawn and future work planned for the project. An overview of time management will also be provided through the use of a Kanban cumulative workflow diagram and GANTT charts.

2. Literature Review

2.1. Introduction

As outlined within the first chapter, the aim of this project is the implementation of an automatic stance detection system and the introduction of a novel topic labelling algorithm. The intent within the literature review is to explore systems that implement automatic stance detection and examine the individual components of opinion mining in more detail.

The key areas of research that are related to automatic stance detection will be presented. This includes the exploration of modern implementations and limitations of stance detection, LDA topic modelling and the complementary techniques and components that will be used to create and evolve the models such as the natural language processing techniques that will be employed, and the external methods to improve accuracy. Data sourcing and the current limitations of labelling topics will also be explored.

The literature review will also examine areas that are closely related to stance and bias detection such as fake news detection. Stance detection has many applications that are extremely significant and applicable within the modern world of business and research sciences. Applications range from bias and fake news detection to monitoring the effects that media sentiment has on consumer confidence within the market and consequently the stock market.

Accurate stance detection systems are currently in demand in a large number of corporations and it would be remiss to not closely examine the related applications in order to form a better understanding of the evolution and application of this project. The Venn diagram below show cases the correlation between stance detection, bias detection and fake news detection. Bias detection sets a baseline to determine how positive or negative the opinions are to determine if a text is biased. Fake news detection is a more complex field that determines if news is fake based on correlating current news sentiment within media.
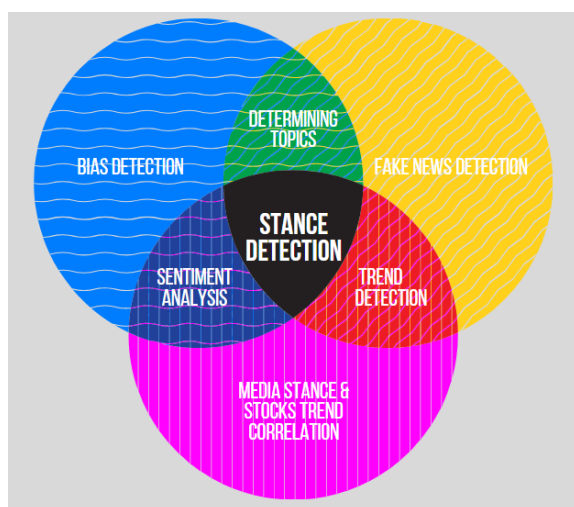
Figure 4 Venn Diagram of the Fields Related to Stance Detection

2.2. Manual Stance Detection

To reiterate, stance detection is a subtask of opinion mining which will use aspects of sentiment analysis and topic modelling in order to understand the position of media in regards to multiple topics. The key difference between sentiment and stance analysis is that stance analysis systems have the goal of determining the author's favourability towards a target or topic whereas sentiment analysis simply classifies a sentence as positive, neutral or negative.

The manual analysis of stance within a body of text is a cumbersome and tedious process that is difficult to apply at scale. Manual stance detection relies on iterating through qualitative data with the aim of finding the position of the author to the topic. The area of research with a suite of techniques to analyse this type of information is known as qualitative research which is defined as empirical research where the data is not in the form of numbers. [23]

A great deal of research has been performed in order to facilitate a better understanding of how to extrapolate meaning from texts. Different techniques are outlined below and discussed in order to determine the best method for manually identifying stance within newspaper articles.

A great deal of research has been performed within the area of qualitative research in order to facilitate a better understanding of how to standardise the extrapolation of meaning from texts.

The diagram below illustrates the different types of qualitative content analysis methods and the complementary algorithms that are directly linked. Summative uses keywords, conventional uses categories developed during analysis and directed is developed from categories created before the analysis. [24]

Coding Approach	Study Begins With	Derivation of Codes	Algorithms
Summative	Keywords	Keywords identified before and during analysis	Unsupervised and semi-supervised algorithms: **NMF, NTF LDA** and traditional clustering algorithms.
Conventional (Inductive)	Observation	Categories developed during analysis	
Directed (Deductive)	Theory	Categories derived from pre-existing theory prior to analysis	Supervised classification algorithms: Support Vector Machines

(Hsieh and Shannon, 2006)

Concentrate on Summative and Conventional (Inductive)

Figure 5 Types of Qualitative Content Analysis [24]

The method of qualitative data that seemed to apply the most to the area of manually creating topics through analysing word frequency patterns was thematic analysis which is used frequently within qualitative research within areas such as phenomenology and ethnography. Thematic analysis is a method of content analysis which contrasts with other qualitative analytic approaches such as

discourse analysis and grounded theory which are generally considered to be methodologies more so then methods. Thematic analysis emphasizes identifying, analysing and interpreting patterns of meaning (or "themes") within qualitative data. [25]

The basic key summary of the thematic method is as follows. Thematic analysis is implemented through the process of creating codes or words which will match to a theme. The codes are then later organised into themes. [28] An illustrative example is where the theme is "Phone Brands" and the codes are Nokia, Samsung and iPhone.

Thematic analysis provides the ideal manual representation of creating the LDA topic model. The disadvantages of thematic analysis are evident, when compared to the automatic approach discussed within this dissertation.

The machine learning model is capable of automating the creation of these codes and themes and then identifying them within thousands of articles in less than half an hour, while an analyst could spend hours iterating over just a handful of articles in order to assign codes to themes, and then find them all individually.
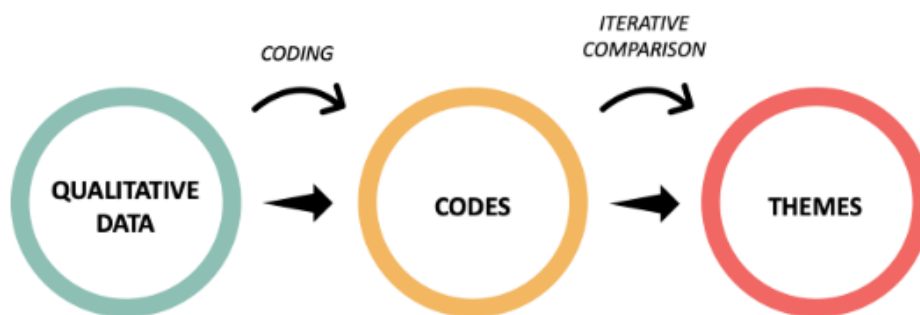
Figure 6 Thematic Analysis Process [29]

Once all of the occurrences of the codes and themes are found, the next phase of manual stance detection is opinion mining. Performing opinion mining manually is less intensive then thematic analysis as there is less setup, but it is still impractical at scale. Calculating the stance from tens of thousands of sentences is trivial in terms of time for a machine learning model, but for an individual, can take weeks of tedious work.

LDA topic modelling is essentially an altered automatic thematic analysis which can dynamically create these themes and match them to the words based on the patterns within word frequency. By being able to automatically create these topics the quality of the topics can be assured as a machine can consider many more articles than an individual could have read. The topics are also created based on frequency and as a result a machine algorithm is more likely to create the most relevant topics.

2.3. Alternative Existing Solutions to Your Problem

The two main components of automatic stance detection are opinion mining techniques including sentiment analysis and particularly topic modelling which are both currently a growing area of interest within computer science and there are a large range of projects within both fields. Multiple projects have been built around the area of detecting media bias, detecting points of view, fake news detection, stance detection and topic authenticity. The purpose of this section is to explore and discuss the various research projects related to these topics that share similarity to the solution presented within this dissertation and draw conclusions around their application, significance and learning points.

2.3.1. What is Automatic Stance Detection?

Automatic Stance Detection is an approach which determines the stance of the author using opinion mining and topic modelling in order to understand the position of media in regards to various topics. The automatic generation of topics is controlled by the topic modelling algorithm of choice which dynamically creates topics and binds the words most likely to be associated with that topic to their respective topics. At a low level, the sentiment analysis is performed through the application of a sentiment classifier on each occurrence of each topic through their respective sentences. A stance score is calculated through the average and whether this score is positive or negative dictates the stance of the news company towards those topics.

Both sentiment analysis and topic modelling require the use of machine learning in order to automatically process the data at a large scale. Sentiment analysis is integral in order to automatically understand the intent behind sentences, but grouping what the sentences are discussing to their correct category is the job of the topic model. The topic modelling algorithm of choice that is utilized within this thesis will be Latent Dirichlet allocation topic modelling which is a popular approach within machine learning to identify topics within documents. A popular definition of LDA is as follows, "Each document can be described by a distribution of topics and each topic can be described by a distribution of words" [30].

Stance detection techniques to map the words within the topic model to its original sentences and correctly apply the sentiment classifier on the occurrence of those sentences is also a critical job that introduces new challenges and complexities around stance detection.

As a basic example, if a media company receives a polarity score of "0.9" towards a topic such as a political party, this means that it is very likely that the news company desires to represent this political party in a positive light and biased towards this party. On the other hand, if a score of "0.1" is generated this means that the news company is biased and against this party. The scores are calculated through a number of different means. The sentiment analysis model is applied on each sentence and all sentences that are grouped under discussing this political party are averaged against each other in order to find how this party is generally presented within various media outlets.

A neutral base line will also be developed. Naturally, a score of 0.6 does not necessarily imply that a news company is biased as there are many factors within language that can slightly skew the score

that is output. This base line has the potential to extend further and appropriate research will need to be dedicated in understanding this base line.

2.3.2. Automatic Stance Detection within News Media

Within this section projects that use or implement components of automatic stance detection will be discussed and evaluated in order to form a better understanding of the capabilities of modern stance detection systems. Projects that will be explored include bias and fake news detection systems that implement forms of automatic stance detection. Components that will be evaluated include techniques of opinion mining such as sentiment analysis and LDA topic modelling.

A conference paper with the title "Large-Scale Sentiment Analysis for News and Blogs" [31] is a project that pointed out some techniques within the area of analysing media. This project was developed in early 2007 where techniques used to analyse texts were primarily sentiment analysis rather than stance detection and opinion mining. While accepting the differences with this work, it is still a useful source of information since it focuses primarily on the techniques used to form a scoring system to measure media outlook and a basic calculation of subjectivity.

Figure 7 Sentiment Analysis Graph Hops [31]

An interesting conclusion drawn from the above conference paper is how the subjectivity scores were calculated. The amount of subjectivity associated with each entity or topic is calculated through reading all of the news texts over a period of time and counting the sentiment score in each occurrence in order to get the average subjectivity for that topic. This subjectivity can be evaluated through the following calculation and acts as the base line for calculating the stance for each topic mentioned. This form of calculation is more basic then modern stance detection systems which use more fine-tuned stance detection techniques to measure the subjectivity score and calculate exactly what sentences constitute a topic, but this calculation acts as a good reference point for some of the overall goals of this project.

$$world\_subjectivity = \frac{total\_sentiment\_references}{total\_references}$$

Figure 8 Calculating Subjectivity Score for Topics [31]

Sentiment analysis has been used in many different research projects to achieve a wide range of different goals. In an article "Capturing Favourability Using Natural Language Processing" [32] published in 2003, studies were performed in order to understand how automatic sentiment analysis using natural language processing compares to that of a human's ability to understand the underlying sentiment. Humans were asked to manually review different products brands as either favourable or unfavourable and the system used reviews from the products website. While no affirmative conclusion was confirmed, the ratio displayed that on average their sentiment analysis detection model was correct only 75% of the time.

	polarity	brand A	brand B	bran d C	brand D
Human	favor.	437	169	80	39
	unfav.	70	65	51	41
System	favor.	52	22	9	3
	unfav.	4	5	2	1

Figure 9 Sentiment Analysis Positive and negative Distribution [32]

However, since 2003 sentiment analysis has undergone a large number of shifts and ultimately, improvements. Not only are models significantly more accurate, but the range of results from a sentence expand past binary (positive, negative) results. Using word vectors and multiple new optimisations made within the field, sentiment analysis algorithms have been found to be able to distinguish between very positive, slightly positive, neutral, slightly negative and very negative and the best sentiment analysis models (such as Google Clouds Natural Language API) has an excellent accuracy of 92.1%. [33] This is not to say that sentiment analysis as of 2020 is without faults. Sarcasm, word ambiguity and even multipolarity are areas that to this day sentiment analysis has not found solution that fully solve these problems. If a sentence has multiple polarities, for example "The colours in my laptop are good, but the audio is very weak", the sentence will return a negative or neutral result, but the model will not generate two scores for the separate topics, leaving more granular sentiment analysis solutions still to be desired.

In the article Comparing and Combining Sentiment Analysis Methods written in 2013, the differences in sentiment analysis accuracy across different libraries were investigated. The table in figure 34 shows the differences across the different libraries.
The wide range of different results indicates the large amount of different approaches that are available to the implementation of this system. Based on the systems requirements, libraries will need to be investigated to ensure that a high level of accuracy is achieved in order to obtain the most optimal and accurate results.

Metric	PANAS-t	Emoticons	SASA	Sentic-Net	Senti-WordNet	Happiness Index	Senti-Strength	LIWC
Recall	0.614	0.856	0.648	0.562	0.601	0.571	0.767	0.153
Precision	0.741	0.867	0.667	0.934	0.786	0.945	0.780	0.846
Accuracy	0.677	0.817	0.649	0.590	0.643	0.639	0.815	0.675
F-measure	0.632	0.846	0.627	0.658	0.646	0.665	0.765	0.689

Figure 10 Differences in Sentiment Library Accuracy [34]

"Computer-Assisted Content Analysis: Topic Models for Exploring Multiple Subjective Interpretations" (Chuang J, Wilkerson J, Weiss R, Tingley D, Stewart B, Roberts M, et al.) [35] is a paper written and published by the Princeton university. This paper discusses computer aided topic modelling from the perspective of allowing experts to interact with the creation of topic models in order to increase accuracy, reliability and adaptation of automatic model creation.

The paper discusses the limitations of topic modelling such as how multiple runs of a topic model can result to having multiple different solutions due to the underlying optimization [9]. Another point of conflict of topic modelling is that the output of models must frequently be manually updated which can introduce significant changes in the output produced as well as it has the possibility to replace the initial coding instructions that have been set up by the expert. Experts have rejected up to two thirds of the machine-generated topics [36]. Linking back to the first chapter, the limitations of topic modelling must be addressed. Particularly, the need for manual intervention in order to generate topic names. This problem seems to still be in effect with ideal solutions still not completely available.

Solutions are then presented within the paper that are focused on increase the usability of topic modelling in order to maximise its advantages. The first solution is to create high quality iterative models through the exploration of model space suited for their specific type of analysis. Unless this specific model can be manually constructed, manual coding will prevail in terms of accuracy, although it will still lag behind in being able to analyse large amounts of texts as content analysis is notoriously time and labour intensive.

The article also hypotheses that the visualisation of the topic model will reduce selection bias by the individual that writes the coding scheme. Generally, when picking codes in thematic analysis bias can occur when picking certain coding scheme and through the use of developing automatically generated coding schemes a large exposure to more coding schemes will increase the reliability of the coding implemented by the analyser.

Another recent article published at the start of 2019, which was briefly mentioned within the introduced, presents an implementation of LDA topic modelling that attempts to retrieve any discussion of thirdhand smoke within the Chinese Media [22]. The purpose was to discover how often the Chinese Media discusses this issue, how well it is discussed and how this has changed over time. A total of 2000 articles were recovered through the Wiser and Factiva databases.

An LDA topic modelling approach was implemented in order to discover the main keywords and topics discussed within thirdhand smoke articles. The percentage distribution for topics discussed are shown in the graph that has been added below (Figure 8) and the words that link directly to the topic have been added in the second graph (Figure 9).

The topics generated through the LDA model were performed on articles that only discussed third hand smoke and through using the word frequency distribution the LDA model successfully managed to create the words associated with its imaginary topics. Imaginary topics are created by the LDA model, but the LDA model does not understand the generic topic that the words link with. This dissertation did not focus on the automatic labelling of imaginary topics to their real-world counter parts and they were manually selected by hand. As discussed within the introduction, this is an objective that this dissertation will attempt to find a novel solution that will solve this underlying problem.

Findings within this article were able to determine that from 2013 to 2017 the number of articles written about thirdhand smoke has decreased from 78 a year to 41 a year within Chinese as opposed to the US where the number of articles increased from 52 to 105 respectively [22].
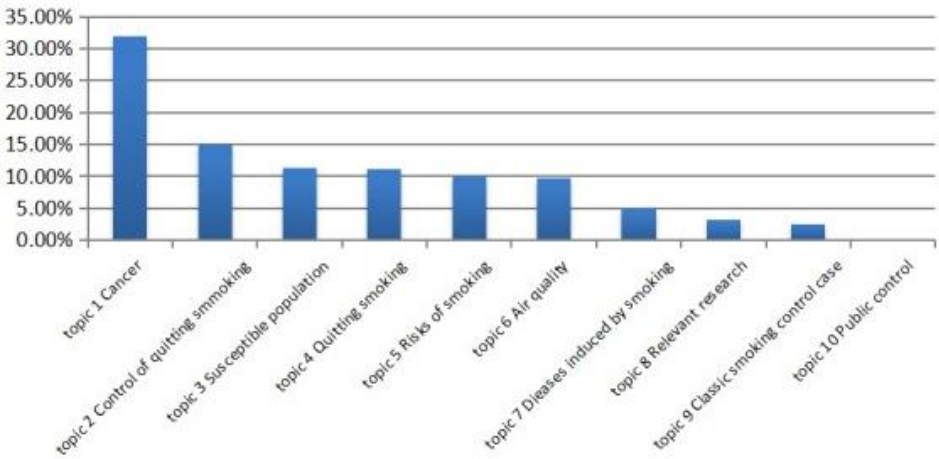
Figure 11 Thirdhand Smoking News Topic Model [22]

The figure below showcases the words that constitute the various topics generated by the LDA topic model.

Topic classification and keywords.

Classification group and topic name	Key words
Related diseases	
Topic 1: cancer	Lung cancer, cancer, tumor, treatment, patient, risk factor, pollution, professor, Chi prevention of air pollution
Topic 5: risks of smoking	Smoker, movement, body, nicotine, content, quitting smoking, experts, symptom re
Topic 7: diseases induced by smoking	Asthma, citizen, hospital, doctor, patient, treatment, smoker, time, long-term, breatl
Topic 3: susceptible population	Children, research, food, contact, cause, influence, environment, increase, body, cle smoker, female
Topic 4: quitting smoking	Quit smoking, smoker, smoke, hospital, drug, breath, doctor, work, smoker, content
Topic 8: relevant research	Introduction, reveal, cigarette, children, smoke, officer, smoker, increase, factor, pl:
Air and PM$_{2.5}$	
Topic 6: air quality	PM$_{2.5}$, indoor, concentration, severe, microgram, air, pollution, smog, influence, k:
Control and restrictions	
Topic 9: classic smoking control case	Shenzhen, tobacco control, activity, citizen, place, rule, investigation, smoker, worl indoor, public place, increase
Topic 10: public control	Public place, quit smoking, ban tobacco, rule, place, professor children, indoor, Chi worker, reveal
Topic 2: control of quitting smoking	Quit smoking, ban tobacco, third-hand smoke, public place, place, rule, ban, indoor Beijing, relevant, smoker, outdoor

Figure 12 Topic to key word link [22]

2.3.3. Bias Detection

Media bias detection in news articles is one of the most prominent areas of growth within stance detection and ties in directly with this project. This area comprises of determining if certain news outlets display bias towards certain topics. Bias analysis is generally performed through the use of content analysis. Marta Recasens, an expert in natural language processing, developed a method to detect bias using a dictionary of words that is also used by Wikipedia editors to ensure articles conform to Neutral Point of View (NPOV) rules [37].

One such journal article named "Automated identification of media bias in news articles: an interdisciplinary literature review" [38] examines bias detection through comparing by what means various news outlets report events differently. No physical implementation of such a system is developed within this article, but an in-depth explanation of how automating such an approach could work is provided. This article also delves into the advantages of automating such an approach in order to escape the time-consuming process of manually linking events to a baseline.

This article explains that it first derives a set baseline through articles which are generally perceived to be the most objective, such as police reports [39]. Events that are reported by multiple media outlets are then specifically chosen to be analysed. Two studies discovered that the volume of participants and the event type such as protests against legislation had a high impact on the number of news coverage by different media organisations [40].

To automate the above approach a sequence of steps was required. Firstly, relevant articles on events must be recovered, the articles must then be linked to the baseline data such as police reports and then statistics must be computed on the linked data.

The diagram below demonstrates the events that were examined based on different publishers across countries. The event in this scenario was where a publishing company in a country mentioned one of the four countries below (USA, Russian, Great Britain and the United Arab Emirates and Ukraine).

Figure 13 Publisher Bias Detection [13]

Another research projected developed within the Samsung R&D Institute focused on the implementation of a system that acts as a bias awareness news recommendation system. This system was built on the premise of scraping multiple news articles on a variety of topics from various news sources and then performing clustering on similar topics in order to calculate a bias score for each topic. The user can then generate a bias score for the article they are currently reading, as well as articles that are similar out of the previously web scraped articles.

The system architecture provided by the bias detection system is presented below. Newly created news articles are firstly scraped across multiple publishers, this is represented by the "crawler" and the content extraction is performed in order to obtain the articles and metadata such as which news company reported on the article. Topic modelling is then performed to construct the topics that were discussed followed by topic indexing which is effectively mapping the real topics to the imaginary topics created. Bias computation is then performed through a REST API. Firstly, the article is extracted based on topic, data cleaning and tokenization is then performed, passed to the REST API which returns an estimated bias score. The scores are than averaged across topics and sent to a database. Queries are then performed on this database each time a client accesses an article where a similar process occurs for that single article where it is compared to the other articles that discussed the same topic.
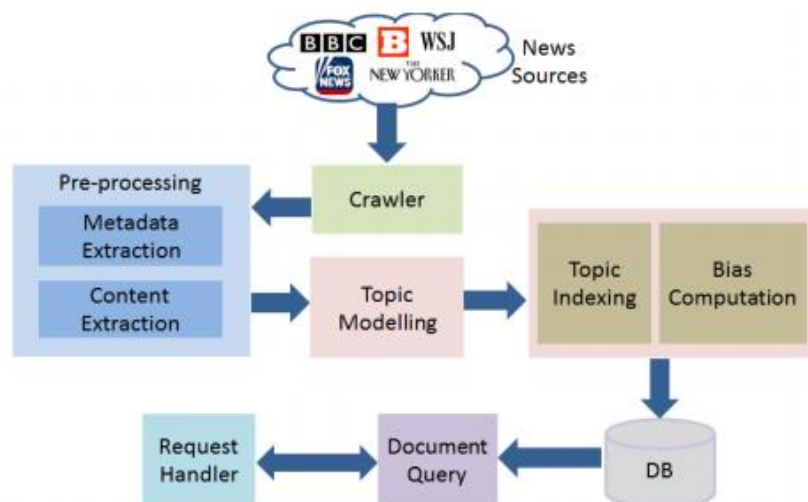
Fig. 1. Block diagram of the system for indexing news articles from different sources, along with their bias scores
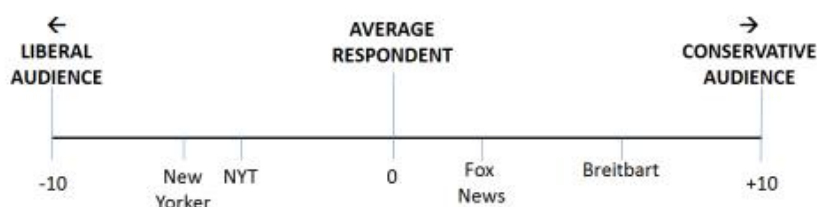
Figure 14 System Architecture for Bias Detection [41]

"Is the News Deceptive? Fake News Detection Using Topic Authenticity" [42] is a paper that proposes an approach in order to detect fake news in online social media using a machine learning classifier. The machine learning classifier detects certain key pieces of information within the account in order to check if the account posts fake news.

The approach firstly checks if the profile and background image or description indicate that the news company supports one side on a controversial issue as many fake news sites have this characteristic. If this is found to be true the news company is automatically assigned as fake news. The next step checks if the profile description indicates it is a news feed. If the profile is a verified account it is legitimate, but if it is not tweets should seem to be objective and have retweets cited from reliable accounts. There is some discrepancy as to whether or not the account can then be labelled valid or not as the news account could be very small and missing retweets from verified accounts as few people are subscribed and regularly read the accounts posts.

On the one hand the approach applied to tweets within this project would not directly be applicable to the system described in this dissertation as it attempts to use specific information from twitter such as retweets as well as profile images which do not exist for popular newspaper companies. On the other hand, the machine learning and classification model has multiple points of interest that are still similar to newspaper stance detection. The project proposes a novel approach in finding similarities between fake news accounts such as average post length, average retweets and linking predications between max and total friends of accounts. The application within stance detection could be to build a similar model in order to find similarities with newspaper companies that are biased. These similarities could also be factors such as common headline terms, average article length and average number of topics discussed. This approach has a large amount of potential as no approach uses this metadata in order to build bias predication and while it may be inaccurate at times it has the potential to introduce novelty to the approaches that have already been developed.
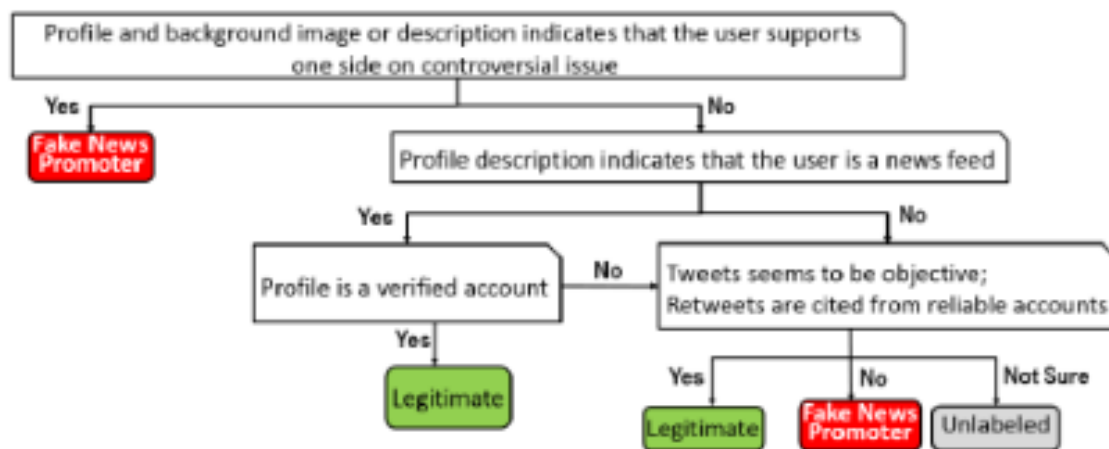
Figure 15 Manual Labelling [42]

2.4. Technologies you've researched

As per the discussion within the introduction, the key objectives are to implement topic modelling, data cleaning and a sentiment analysis that can be performed at large scale among thousands of articles in order to determine the stance or slant of news media in relation various topics. Consequently, requirements such as natural language processing libraries for data cleaning, lexical databases for understanding how closely words are linked and API's for connecting to real Irish news source databases were investigated within this section.

The technologies researched consist of natural language processing libraries, existing lexical databases which apply components of text analysis, qualitative research approaches and API's that can scrape newspaper articles.

2.4.1. Natural Language Processing Libraries

Python is at the forefront of natural language processing (NLP) and there is a plethora of different NLP libraries that are integrated with python. Different libraries have different advantages such as performance, accuracy and a more extensive feature lists, while others have specific trade-offs. The scope of this section is to discuss the most viable natural language processing libraries and single out the libraries that are most suited to this system.

The first natural language processing library that was uncovered within the research phase was NLTK or the natural language toolkit which was developed in python. This library provides a large suite of libraries for symbolic and statistical natural language processing within the English language. Advantages of this library include consistent use of data structures, extensibility to other third-party NLP languages, modularity between interactions of the components within the system and a through documentation. [43]

"SpaCy" is one of the fastest NLP languages that integrates the C programming language for some of the more process heavy tasks. A paper written in 2018 investigating "SpaCy" performance made the follow claim. "We employ the POS Tagger of SpaCy, in preference to the CMU TweeboParser, due to the heavy processing time of the latter. The TweeboParser was 1000 times slower as opposed to SpaCy" [44]. Spacy also provides more advanced features than libraries such as "TextBlob" such as entity linking and working with vectors. A disadvantage on the other hand is that NLTK has been shown to have better sentence tokenization as discussed in the same study.

"TextBlob" is an NLP library that extends the NLTK library and provides additional functionality such as noun extraction, word tagging and text classification. "TextBlob" is generally the library of choice for beginners within the area of NLP as NLTK itself can become very tedious for completing even simple tasks, whereas "TextBlob" can achieve even more complex functionality through a simple interface, the draw back being that TextBlob is slower than NLP languages such as SpaCy and less advanced in the feature suite that it provides. [45]

Scikit-learn is another NLP library that performs a wide array of both natural language processing and machine learning. This library focuses more than any of the others on machine learning having

natural integration with classification, regression, clustering and model selection. The advantages of this library are the wide array of various applications to machine learning.
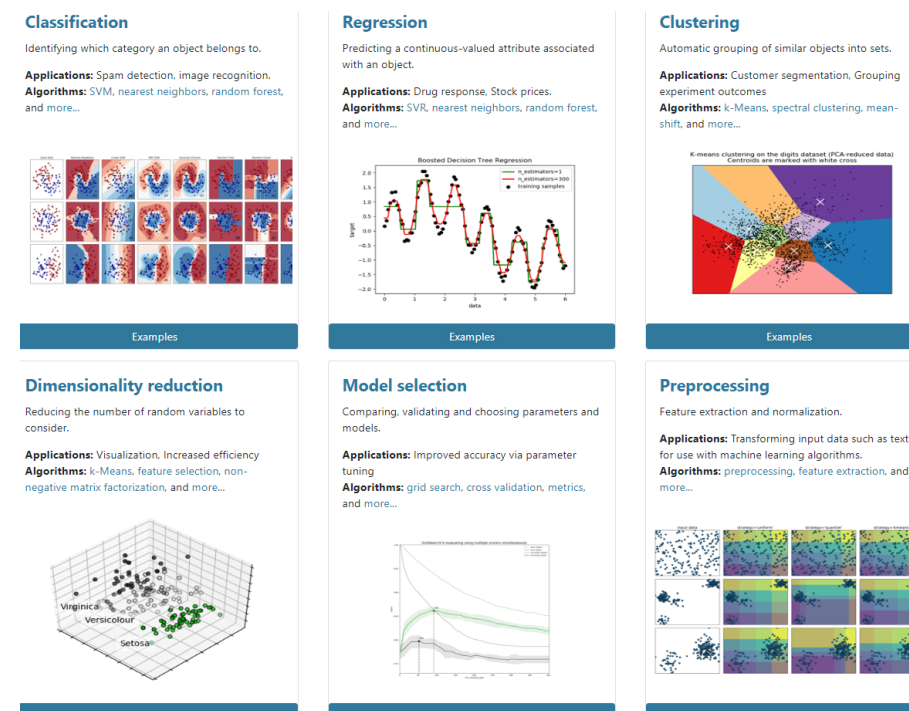
Figure 16 Sci-learn Machine Learning Applications [46]

Gensim is an open-source library for unsupervised topic modelling and natural language processing implemented in python. As seen in the previous section, a great deal of research projects uses genism in order to establish various models within natural language processing. Genism also has integration in order to assist in the extraction of semantic topics from documents in an efficient process. Gensim can work both with topic modelling and sentiment analysis with it's well optimized Word2Vec and Doc2Vec support.
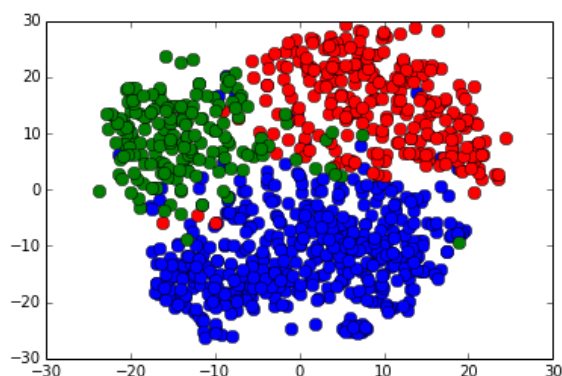
Figure 17 Gensim Classifier with food words (blue), sports words (red) and weather words (green) [47]

2.4.2. Natural Language Tools

Wordnet is a large English lexical database containing nouns, verbs, adjectives and adverbs which are grouped into sets of cognitive synonyms (synsets), each expressing a specific concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. [48] A lexical database such as wordnet is required in NLP as data cleaning processes such as lemmatization require breaking down each word into its normal form, removing prefixes and suffixes. This is a complex task that requires the assistance of such a lexical database.

WORDSEER is another online resource with an exposed API that performs textual analytics and visualisation to make text navigable and accessible. It splits phrases, nouns, years and even specific metadata such as the names of presidents from within texts.
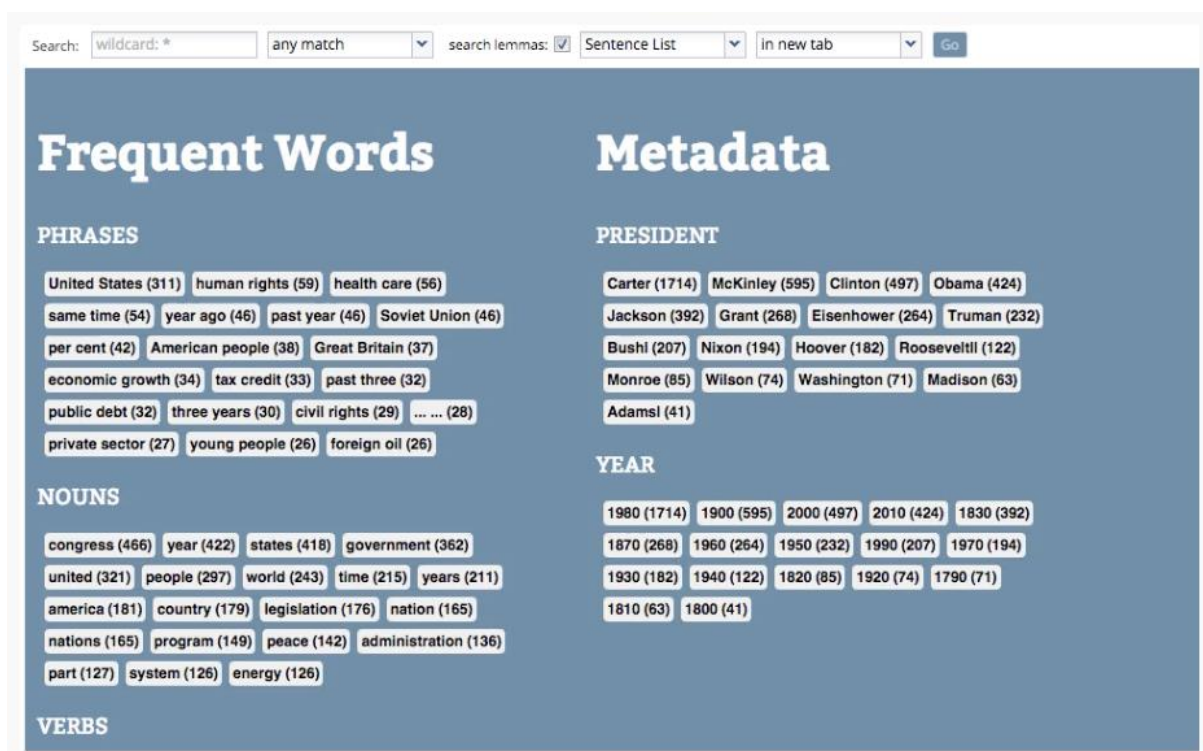
Figure 18 WORDSEER Visualisation of Text

2.4.3. API's

Initially, for the purposes of data training the prototype the "20 News Group Dataset" "BBC dataset" were considered. The "20 News Group Dataset" was chosen as a result of it having a simpler structure allowing it to be more easily parsed. After the basic model is created the sourcing of real news data will be the next step in the process.

For acquiring real life datasets several API's responsible for retrieving newspaper articles were investigated. The most powerful was "newsapi.org" that links directly to Irelands live top and breaking news headlines and articles. The data recovered is in json but an issue that was discovered

with this API is that there is a limit of 500 requests per day and only articles up to one month old can be retrieved for the free version. This may not be a crucial issue as within one month of making 500 daily requests, over fifteen-thousand news articles from Irish papers can be retrieved. [49] After further investigation it was discovered that unless this API's was payed for in a subscription, it would only return the first 180 characters of a newspaper making it unusable for the purpose of this project.

Another news API is the "MyAllies Breaking News" API that provides the access to real time news from across the glove. This API is completely free to use on RapidAPI and overcomes the previous limitation, but it does not have access to articles older than a single day.

A possible solution to overcome the limitation of a maximum of 500 requests a day is to implement a web scraper that can that can retrieve the textual information from articles daily. This will also provide the flexibility to include more information that is not provided within API's.

2.5. Other Research you've done

2.5.1. *Research of Bias in Media*

Research as to the exact steps of how bias occurs within media is a quintessential step in understanding how to build bias detection for news outlets. This research aids in mapping the stance detection score to each topic as it allows the creation of an estimation of how much bias naturally occurs within the media.

The diagram below shows the process of how bias is formed within the media. It firstly breaks down the factors that influence the medias perception. The political and ideological view of the company naturally influence how various topics are represented and this consequently will affect and skew the representation of political parties and movements through bias. However, factors such as advertisements will skew stance greatly [13]. Different articles can quickly switch between persuasive and informative quickly, where persuasive should generate outliers that have a very high stance score and informative could have an overly prevalent neutral score. The type of language used is highly relevant within this thesis where news outlets are normally highly informative, this may mean that more objective results will be obtained. Ideally within future iterations magazines and sources of text that are more persuasive will be analysed to better understand these issues.

Owners will affect in how articles report on events as the news company might not report negatively on an event involving one of their partners, advertisers or sponsors. Another large factor that influences media representation is target audience [13]. Target audience is a major factor as generally papers attempt to cater to the views and beliefs of the group that reads the paper in order to continue their readership of the paper and as a result this is another factor that effects media bias.

As can be seen in the centre of the diagram there are a lot of natural factors that further effect bias without considering opinionated bias. There can be discrepancies between different news companies and the gathering of data. Different news companies may be misinformed or miss key

facts that will affect the writing style and generate more negative stance scores than other news companies, even if they share the same perspective. Writing style can be more negative or positive for certain journalists as writing styles different. Local colloquialism would also make the sentiment analysis incapable of classifying the expression and generating a score as it may not have classified such phrases.

Finally, editing is a major contributor as it can both reduce and increase bias. Some papers may specifically remove certain words that are overly positive or negative with a more informative style of writing was required. Another factor within editing that effects bias detection is where miss spelled words are missed. Sentiment analysis cannot work on words that are miss spelt and as a result no sentiment score can be generated, and bias cannot be caught or dismissed in those cases.
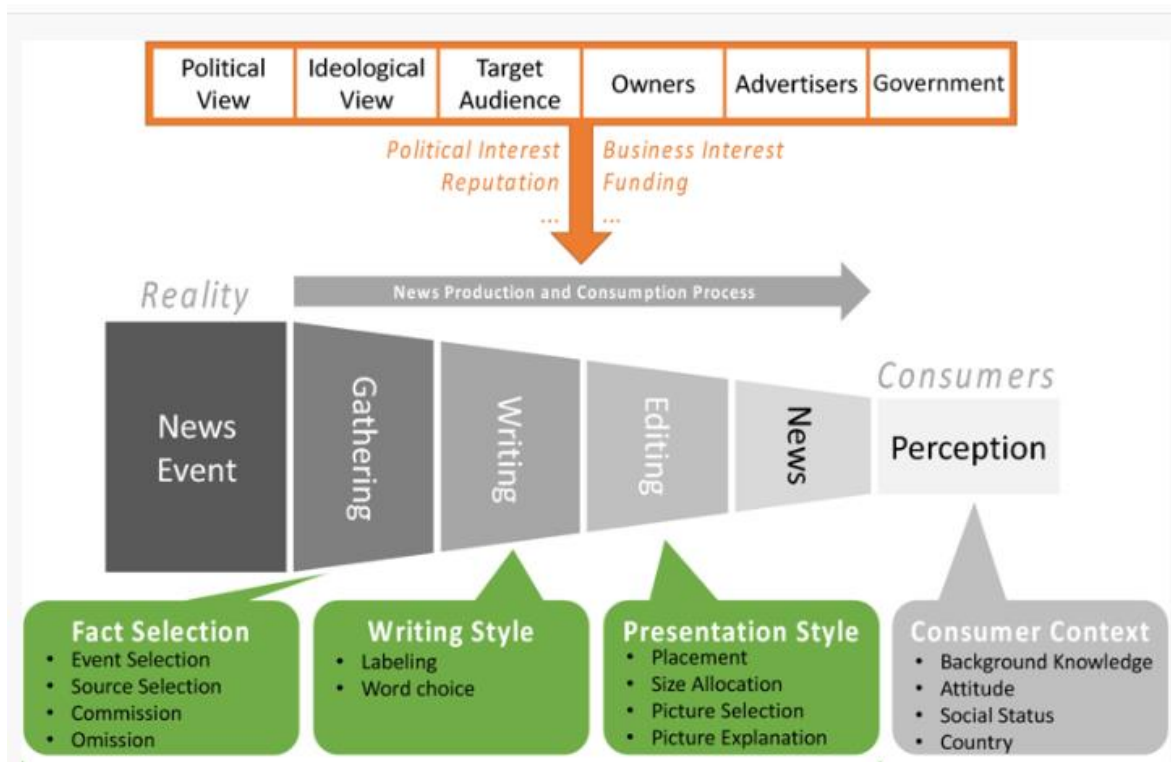
Figure 19 Bias within Media [13]

2.5.2. *Mapping of Imaginary Topics to Real World Topics*

An issue that has been an obstacle for LDA topic modelling is its inability to accurately label the topic name of the created topic. As discussed previously, multiple papers and projects highlight the difficulty associated with labelling topics. The paper *Computer-Assisted Content Analysis: Topic Models for Exploring Multiple Subjective Interpretations* [35] discusses how the majority of topic labelling algorithms are not reliable for analysis. Other papers that were previously discussed also highlight the difficulty of topic labelling such as *Understanding LDA in Source Code Analysis* [9], Probabilistic Topic Models [10] and *Automatic Labelling of Topic Models [11] showcase the difficulty of finding an accurate method of labelling topics.*

37

The LDA topic model by default labels topics through the use of an arbitrary number which are referred to as an imaginary topic. Conventionally, analysts label the topics from the context of the most weighted words. As an example, the words and phrases "Withdrawal from EU", "UK Referendum", "Vote" and "No Deal" clearly refer to Brexit. This becomes a bigger issue when not all of the words clearly relate to the same topic. A more complex topic would be labelling the words "Police", "Charge", "Drug". A labelling algorithm could look to label the topic in question as the topic "Police" or "Drug" or even "Law". This can be a task that is difficult even for humans without more context to the top weighted words.

Automatic topic labelling is a key component of this dissertation as the objective is to have a stand-alone system that can infer the stance of any media outlet towards a set of various topics. Implementing automatic labelling of topics will allow this project to work at scale across a large set of media outlets without the need of an analyst or an interpreter to interact with the system. Without topic labelling a great deal of overhead would be added when analysing hundreds or thousands of news outlets as for each outlet up to twenty topics would need to be manually interpreted.

As discussed within the introduction, topic labelling still has no optimal solution that can automatically label topics with a high degree of accuracy. This dissertation will look to implement a novel topic labelling algorithm with a high degree of accuracy and this will be one of the most complex components within this project.

Labelling topics is a complex issue for a number of different reasons. Machine learning models that attempt to label topics generated by the LDA model results in a high degree of error when interpreting topics. This is because the LDA topic model retrieves the most used words that are associated with a topic, but not all words specifically relate to the topic in question. Defining a generic category that can span a select number of the most weighted words has also proved to a big challenge for modern machine learning classifiers. Many dissertations, books and articles have spent many years in an attempt to find an accurate machine learning model that can label topics but to this date no full proof method exists.

The publication *Automatic labelling of topic models* introduced a novel approach to label imaginary topics using Wikipedia articles. The authors theorised that if the top-10 topic terms within a topic were queried using Wikipedia articles, the titles of the Wikipedia articles could be used to identify the most generic topics. [50] This approach relies on the large volume of Wikipedia articles available on the internet, as well as the topic itself being well defined and relatively generic. The results found that it was uniformly better then unsupervised baselines for four different datasets, but the study overly relied on very basic topics such as "Books", "News" and "Blogs". These topics are not reflective of this project as topics will be less generic such as presidential candidates where the top ten terms may include the candidates' names but they will also often include extremely generic words such as "election, campaign and vote" which make it difficult to identify that the specific topic is a presential candidate, but more than likely make a generic assumption and simply return a generic topic such as "presidential candidate" or "election".

The article *Automatic Labelling of Topic Models Learned from Twitter by Summarisation* proposes a multi-document summarization task to characterise documents relevant to a topic. While previous work has shown that labelling a topic with the top most used words is not always representative of a

topic [51], this approach relies on term relevance relating to a topic using summarisation algorithms which intendent of an external source, provided a higher performance than the top-n terms base line created by unsupervised learning techniques [52].

The diagram below outlines the various different approaches tested within this research project. Research found that the frequency-based summarization techniques *(Sum Basic)* out performed graph *(Text Rank)* and relevance-based *(Maximal Marginal Relevance)* summarization techniques for generating topic labels [52].

	War	DisAc
GS	protest brief polic afghanistan attack world leader bomb obama pakistan	mine zealand rescu miner coal fire blast kill man disast
TT	polic offic milit recent mosqu	mine coal pike river zealand
SB	terror war polic arrest offic	mine coal explos river pike
TFIDF	polic war arrest offic terror	mine coal pike safeti zealand
MMR	recent milit arrest attack target	trap zealand coal mine explos
TR	war world peac terror hope	mine zealand plan fire fda
	Edu	**LawCri**
GS	school protest student fee choic motherlod tuition teacher anger polic	man charg murder arrest polic brief woman attack inquiri found
TT	student univers protest occupi plan	man law child deal jail
SB	student univers school protest educ	man arrest law kill judg
TFIDF	student univers protest plan colleg	man arrest law judg kill
MMR	nation colleg protest student occupi	found kid wife student jail
TR	student tuition fee group hit	man law child deal jail

Figure 20 Summarization Topic Labelling [52]

Many other approaches to topic labelling have been implemented, each with their own trade-offs such as using word vectors and letter trigram vectors. [53] This approach provides a framework for labelling topics through mapping candidate labels and topics to vectors in order to find which labels semantically relate to a topic. The candidate labels are found by calculating the similarity between various topics and candidate label vectors. This solution is used to help analysts determine topics more efficiently but it was found that it was found that analysts still need to read over the created topics to remove errors. Ideally this project can eliminate the aid of a reader interpreting topics.

Another approach is using neural embeddings to label a topic with a succinct phrase that summarises a theme or idea [54]. This solution was found to be highly accurate when monitoring simple well-defined topics such as blogs, books and news. No testing was performed on more difficult topics such as particular books, political leaders or movements and this brings into question how accurate this solution would be for a more complex topic list.

Both vector and letter trigrams as well as the summarization of themes using neural embeddings seem to still not accurately labels topics with sufficient accuracy to be reliable in most scenarios. For

this reason, the limitations and conclusions will be evaluated when constructing the approach to topic labelling in order to make the topic labels as accurate and dependable as possible.

Approaches that are considered after evaluating the literature review include finding the hypernym of the top weighted words within a topic distribution and labelling the topic with the top weight hypernym words, using a semi-guided machine learning technique to label the model with an external data source or a mixture of topic labelling algorithms that were reviewed to see if accuracy is improved.

2.6. Existing Final Year Projects

Several final year projects from previous years were looked at in the research phase of the project. The elimination process for choosing final year projects to investigate was finding similar themes such as natural language processing, sentiment analysis and machine learning.

Sentiment and Mood Interpreter for Logging Emotions

This final year project primarily focused on sentiment analysis for positive and negative moods. The sentiment analysis was performed on facial recognition (which was found to be 79% accurate) and diary entries (which was found to be 40% accurate). A large portion of the project also focused on the design, layout and usability of the application itself.

The application was designed using feature driven development with agile methodologies. Similarly, for the topic modelling within this dissertation, an iterative approach will also be essential as it will demonstrate a steady growth in the accuracy of the topics built. The feature driven approach will not work as well for this final year project as it will essentially focus on a few features and the complexity will follow from how they link together. For example, linking sentiment and topic analysis to determine sentiment towards certain topics.

Testing was implemented using percentage scores for facial / emotion detection and sentiment analysis. An iterative approach was then followed for monitoring how the percentage score changed as the analysis became more accurate. Edge cases that reduced the accuracy of the results were also flagged and different scores were built around them. An example of such an edge case is where the facial recognition is used in a dark room and in this scenario the accuracy of the facial recognition would have its own independent score rating.

NLPurchase – eCommerce Chatbot Final Year Project Report

This final year project makes use of natural language processing in order to make a chatbot that can successfully communicate with a customer on an eCommerence website. Natural language techniques such as lemmatization and removing stop words are used in order to break down the contents of the users input and allow a more standard, readable approach. In this newspaper topic and sentiment analysis thesis these data cleaning techniques are crucial, particularly for the topic modelling where thousands of newspaper article will each individually be scanned into memory and then split into a very large list of words where further analysis algorithms will be completed to extract meaningful topics and sentiment scores.

The design of the final year project followed an incremental cycle in order to allow the project to adapt to change, as well as provide continuous prototypes with simple complexity that gradually evolved over time. This design allows the project to be split into multiple stages, avoiding architectural risks very early in development as no crucial decisions are made very early in the analysis of the project.

Testing was performed continuously though user integration. A mixture of informal and formal testing was used. The informal testing was where multiple users between the ages of eighteen and twenty-four used the chatbot and multiple stages of its development and cycle and were proposed to fill in their criticisms, issues and proposed solutions when using the chat bot. The users would also give overall usability scores. The formal testing was implemented through a survey which contained questions based on the chatbot design guidelines. These guidelines compromised general usability and feature functionality.

2.7. Conclusions

With the knowledge and conclusions drawn from the literature review, the designing and development of the system can begin. The crucial learning from this research provided a stronger understanding of the current technologies and applications within this area as well as their advantages and disadvantages. Furthermore, a better grasp of what can currently be achieved with stance detection has also been ascertained.

This chapter has also further established the requirements necessary that will be discussed within the following chapter. Natural language Processing techniques will need to be designed in order to introduce an efficient and powerful data cleaning algorithm and a suitable LDA model will need to designed in order to produce accurate topics from the distribution of words.

The large scope of projects presented within this field also demonstrates how this field is currently an area of relevance. Multiple dissertations, papers and conferences over the last couple of years have been dedicated to further exploring this topic and unlocking more powerful tools and applications of this technology.

3. Experimental Design

3.1 Introduction

The aims and objectives that were discussed previously will be the core focus within the design section. The design will look to implement an automatic stance detection system that can measure the favourability of news media outlets in regards to various topics and accurately label topics and clearly display this information to users. The design will also allow for the introduction of features such as correlating stock trends to media sentiment. As indicated in previous chapters, the key components of automatic stance detection that will be designed comprised of the following.

- Data Sourcing Several Thousand News Articles Per News Outlet
- Data Cleaning using Natural Language Processing Techniques
- Designing the LDA Topic Model
- A Novel Approach to Labelling Topics
- Stance Detection Techniques to link the sentences that discuss similar topics
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model
- Finding and Implementing an Existing Sentiment Model
- Improving both Models
- Linking Components to Create the Stance Detection System
- Creating a Suitable Display to Graph the Data

The research that was discussed and evaluated within the literature review will feed into the design as it provides a better understanding of the limitations of automatic stance detection as well as the benefits and draw backs of various different approaches. As an example, a much better understanding of the limitations of labelling topics has been ascertained from the section *2.5.2 Mapping of Imaginary Topics to Real World Topics.* The section examines and evaluates the current solutions to labelling topics and identifies their short comings which will be considered within the designing of this system. The various approaches to stance detection examined in the literature review will also be considered, particularly examining the results and evaluations to optimise the accuracy of the models

The first section will look at the software methodologies employed in the creation of this project and then an overview of use cases will be created. The following section will analyse the system architecture from the perspective of the front-end, middle-tier and back-end.

Prototyping will be a key technique within the life cycle of the design and the development process. As the development continues, new iterations of the prototype will introduce changes to the design in order to adapt to the changing requirements as better approaches are discovered within the development phase. Linking directly to the incremental development methodology, prototyping is a technique which breaks a project into smaller portions with the scope of enabling simpler requirements. The strength of such a process is exemplified by Jason and Smith [55] where they comment on the users inability to identify their own requirements and as a result the need to exhibit the requirements of an experimental system such as a prototype in order to be able to more accurately estimate the requirements and the feasibility of the system.

3.2. Software Methodology

Multiple software methodologies were considered before arriving to a conclusion. Some of the methodologies considered includes the spiral and waterfall model, feature-driven development, extreme programming, and Kanban. Methodologies such as the waterfall method were instantly eliminated due to their well-known limitation of being unable to adapt to change in the requirements. As a result, the methodology of choice for this thesis was strictly required to be agile in nature and only the agile mythologies were seriously considered and examined.

The methodologies discussed below comprise of the spiral model, the feature-driven development model, extreme programming, Kanban and the CRISP-DM model.

Spiral model
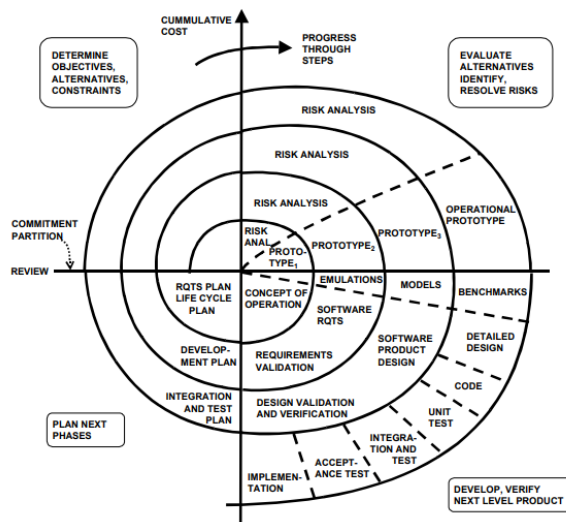
Figure 21 Spiral Model [56]

The spiral model is an agile methodology that is provides a several useful advantages that fit this project. It has one of the best risk analysis models as it is performed iteratively on each iteration. It also provides a realistic implementation as the project moves through loops in a spiral development process to be completed. As discussed in the literature review within the section 2.4.1. "Qualitative Analysis Research", the continuous iteration of codes and themes is extremely beneficial in order to better understand the data as many mistakes are initially made when understanding the data and it is only after a stronger familiarity with the data is made, that the codes and themes can be reliable, making the spiral model particularly useful in qualitative analysis research.

The disadvantages on the other hand bring in a costly model where a great deal of time is spent on each stage of the design and iteration process. The risk analysis also requires a highly specific expertise in order to design and deeply analyse each iteration. The spiral model is also particularly fine-tuned for large projects that span multiple teams and since this project will be independently completed it may not be suitable [57].
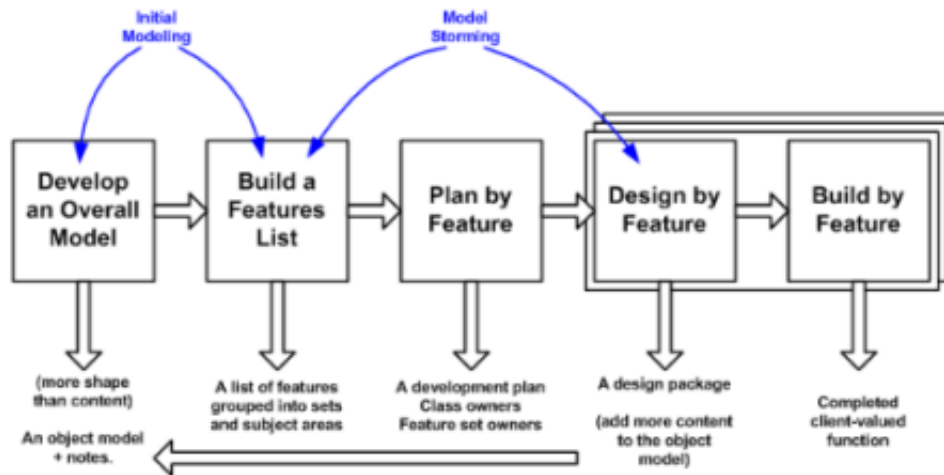
Figure 22 Feature-driven development model [58]

Feature-driven development supports industry recognized best practices, breaking down large projects into smaller features and a simple five step process than does not require a specific expertise. Disadvantages include an iterative process that is not as well defined as other agile methods and generally this methodology is not suited to a single software developer, but multiple teams working in parallel.

A key aspect of feature-driven development is its emphasis on communication between teams and collaboration between users. This thesis is not an application and will not go deeply within the area of human computer interaction. As a direct result a lot of the design principles behind feature-driven development stray away from this projects core focus and would not be suitable for this thesis.

Extreme Programming

Figure 23 Extreme Programming Model [59]

Extreme programming is an agile methodology that follows specific programming guidelines such as test-driven-development, continuous integration and prototyping. This methodology focuses far more on writing code than in very short iteration cycles where short tests are continuously integrated within the development process and while the design is left more lacking than the previous methodologies the level of testing done acts as its own documentation and analysis. Disadvantages include a requirement for skilled programmers that can conceptualize a large portion of the design and work independently or in pairs. Another disadvantage is the reduced level of design may lead to a larger risk within the project's development for edge cases.

Kanban

Kanban is an agile methodology that splits tasks into achievable blocks. It encourages continuous integration where all the work and progress are reviewed daily, providing powerful goal orientation and progress reports. These reports provide meaningful data that allows the developer to stay on track with continuous planning and analysis of progress,
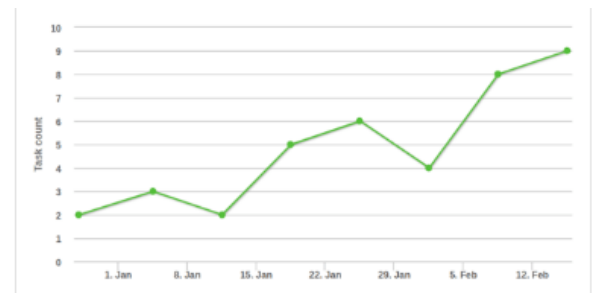
Figure 24 Kanban Progress Chart
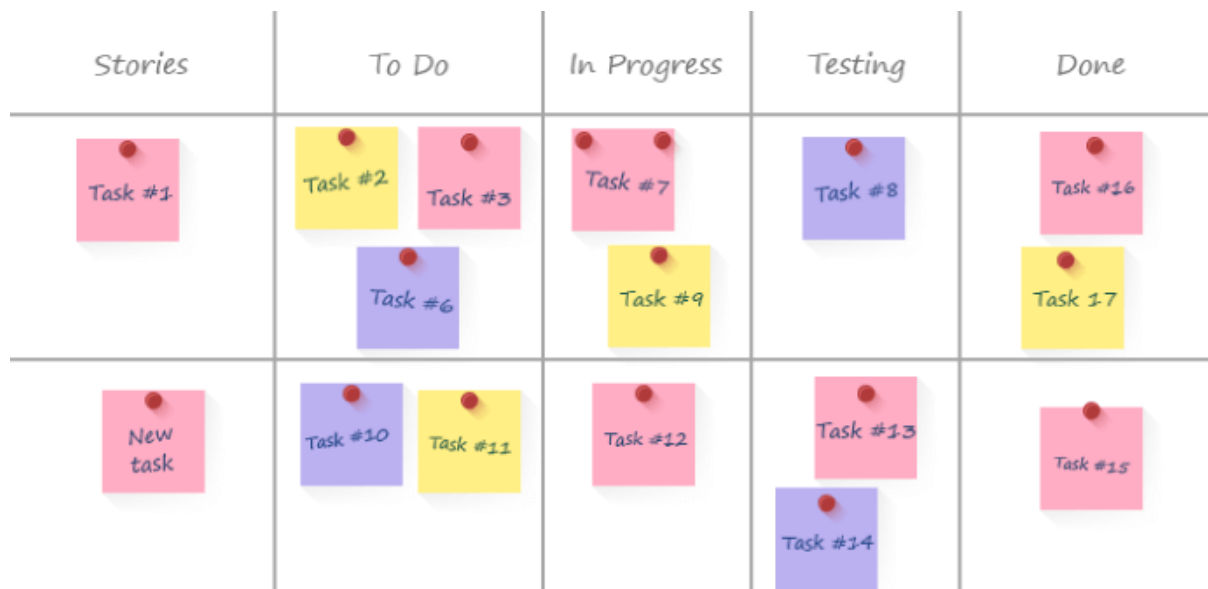
Figure 25 Kanban Example [60]

The Cross Industry Standard Process for Data-Mining is a model that is commonly used to solve machine learning problems [61]. This is a fine-tuned model that focuses on understanding data, pre-processing data and finally training and testing the model.

This methodology fits this project very well since the majority of the complexity within this thesis resides around the iterations of building the LDA topic model in each phase. This will compromise around understanding the requirements and cleaning the data and then training and testing the specified model.
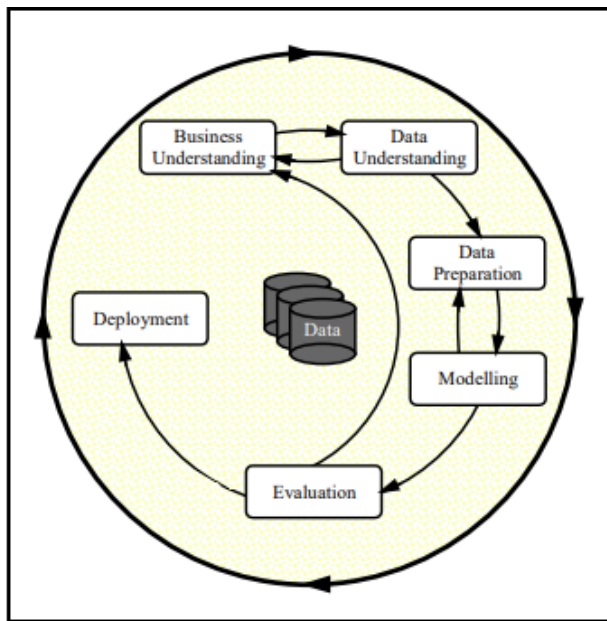
Figure 26 CRISP-DM [61]

The below figure shows the Kanban board that was used to schedule and plan work within the creation of this project. Work will be monitored through Pomodoro's done daily and the number of columns that are moved to the done column. A progress report will be provided at end of the project to show how tasks were scheduled over the last twenty weeks of the project.
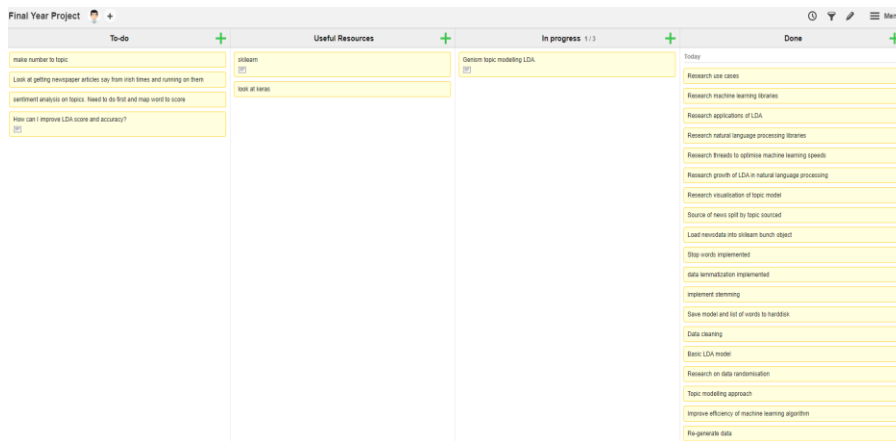
Figure 27 Kanban Board

3.3. Overview of Automatic Stance Detection System

Aspects from the Kanban, simplified spiral model and CRISP-DM methodology will be introduced within the development of this project. Each iteration will introduce a specific set of the key elements from each design methodology. The spiral model will be responsible for extracting the risk analysis and engineering testing methods. Kanban will be responsible for goal orientating each iteration, timeboxing using the Pomodoro technique as well as evaluating the success of each iteration continuously and finally CRISP-DM will provide standard machine learning specific approaches to mining, understanding, preparing modelling and evaluating the data.

Both design and code will be delivered in stages to allow an adaptation to change in each iteration. The following features will need to be completed and then iterated over to be made more accurate and provide more meaningful data.

1. Sourcing of training data to test the models.
2. Sourcing of real news articles with various topics from multiple media outlets through an API or web scraper.
3. A data cleaning algorithm that will implement natural language processing techniques to prepare the data for the sentiment and LDA topic model.
4. An LDA model that can accurately generate topics.
5. Methods to test the accuracy, coherence and perplexity of the LDA topic model through stages.
6. A mapping of the created imaginary topics to its respective real topics. (As discussed in the literature review, this will be a complex area in itself that is difficult to accurately implement with current technology).
7. Multiple methods to visualise the topics created by the LDA algorithm.
8. A sentiment analysis model.
9. A mapping of specific sentences to each topic.
10. Aggregating a specific slant or stance score for each topic.
11. A database to store the stance scores for media outlets.
12. A display for the various stance scores for each topic per represented media outlet.

The architecture does not rely on any client-server model. The complexity is sourced within the area of building topics from real newspapers and being able to map the topics to a stance score that reflects that newspaper companies' views towards these topics. Initially this system was designed to have no database due to CSVs providing the ideal method for modelling and graphing data, but due to the decision to use Grafana or wave front for displaying the results, a database was integrated in order to feed this information through Grafana's API natively. This in turn orchestrates a non-traditional approach for the interaction of the front, middle and back-end of the system. This is as a result of the logic layer only interacting with the database in order to upload results, rather than consume information, and the the front-end only consuming the data from the database in order to display the results without having any interaction with the middle-tier.

The diagram below shows the architectures interaction between the three layers. The logic layer is responsible for the core complexity of this project including the creation of a web scraper, a data cleaning algorithm, the creation of the required models and the processing of the created information. The logic layer in this instance is not at the middle of the architecture as it has no direct interaction with the front-end. The logic layer will calculate the required stance scores and send all of the relevant information to the created database. The information will then be sent through streaming from the database directly to the front-end as a data source.

The backend will be responsible for storing the results that are output by the logic layer. It will comprise of a single MySQL table that will be associated with one table that stores attributes such as news media company name, average and mode stance scores and the date this score was retrieved, allowing for the possibility of graphing changes of sentiment over time within future iterations.

The front-end will be fed the data from the back-end natively through streaming. The advantage of streaming the information in real time is that the sentiment overtime can be plotted as more recent articles are sourced from the various media outlets.
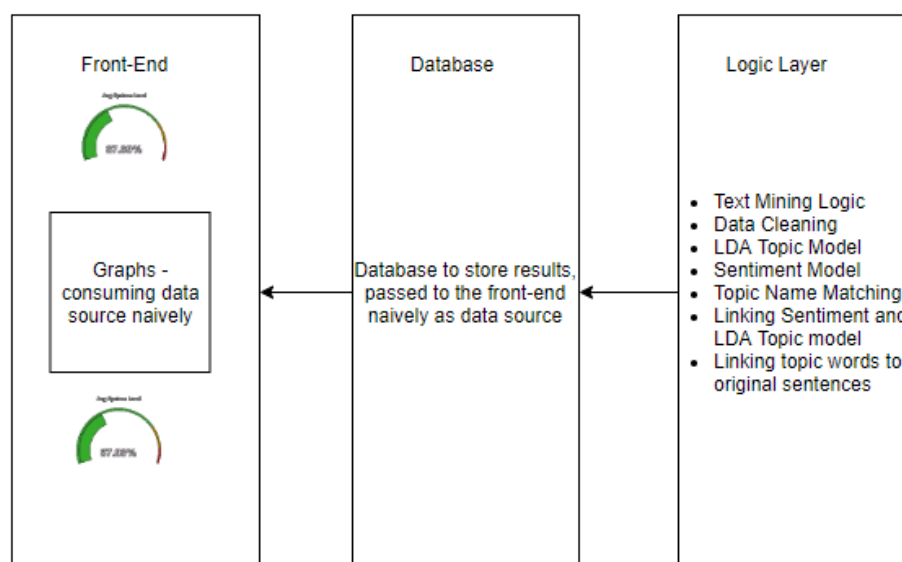
Figure 28 System Design

The diagram below shows a high-level sequence of events that occur within the system when generating the stance score of topics for each newspaper company. The processing is performed in stages where first both the training and real newspaper articles are sourced. The training data is then cleaned and an LDA and sentiment model is built using this data. The models will then be used on the real newspaper articles in order to split the topics again and build a stance score around each topic.
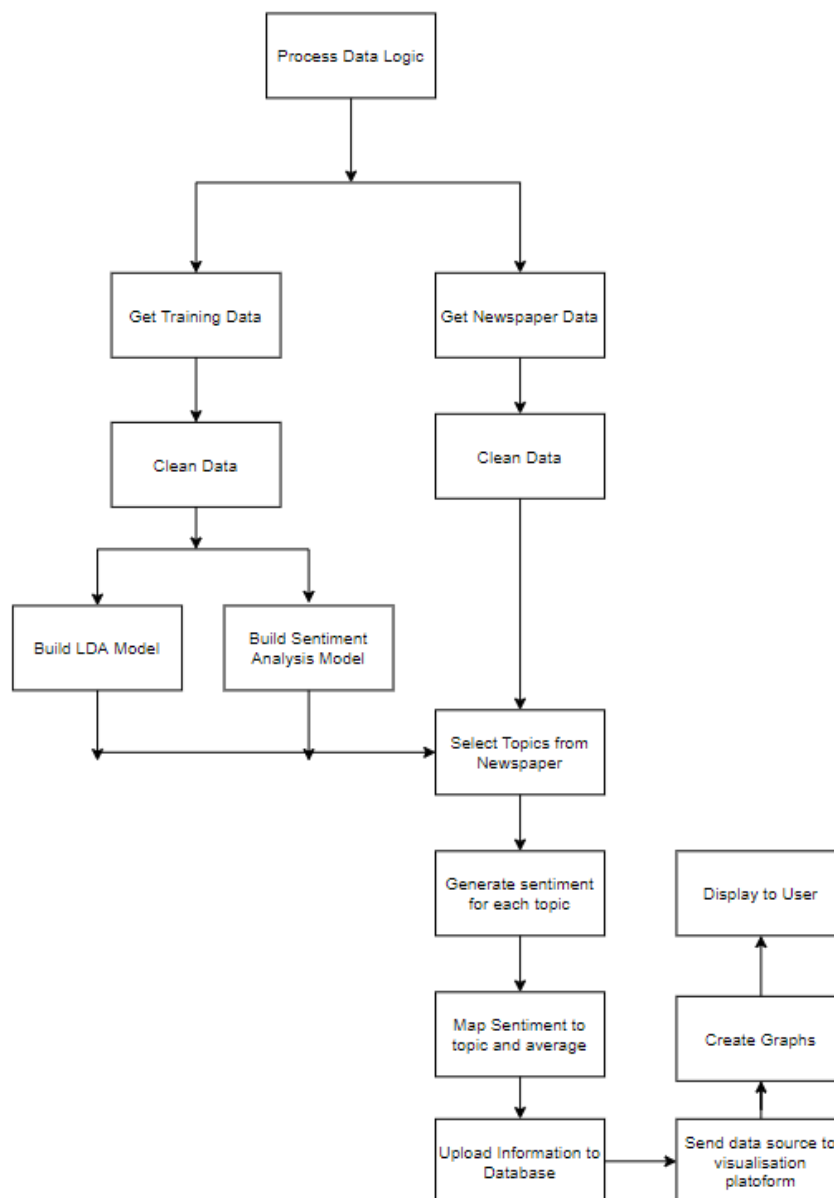
Figure 29 System Sequence of Operation Diagram

For the purposes of version control GitHub will be used. The iterative approach will be tightly linked to Kanban in order to keep track of progress through Pomodoro's, the tasks that need to be completed, the tasks that are completed and useful resources. In the final chapter the throughput chart will be included that will show case how time was spent when developing the project. Daily Pomodoro's will be incorporated within the development of this thesis.

3.4. Design of User-Display

The front-end layer will display two types of graphs. The first type of graph will describe the objectivity and stance of news outlets towards all of the listed topics. The second type of graph will outline exactly what words constitute a topic.

A textual representation of data will be provided, but the main display will be presented using a multi-platform open source analytics and interactive visualisation software. Multiple third-party libraries will also assist in visualising the created topics within graphs in order to provide more granular information for what constitutes a topic and to provide further insight into the quality of the created topics. The distance between topics and which topics are the most dominant and representative will also be visualized.

A dashboard will be created inside of a visualisation software such as Grafana or Wavefront in order to display most the stance graphs. The stance and objectivity scores will be created separately for each news company and the mode, average and median will be calculated as well as attempts to increase the amount of information available in displaying the data.
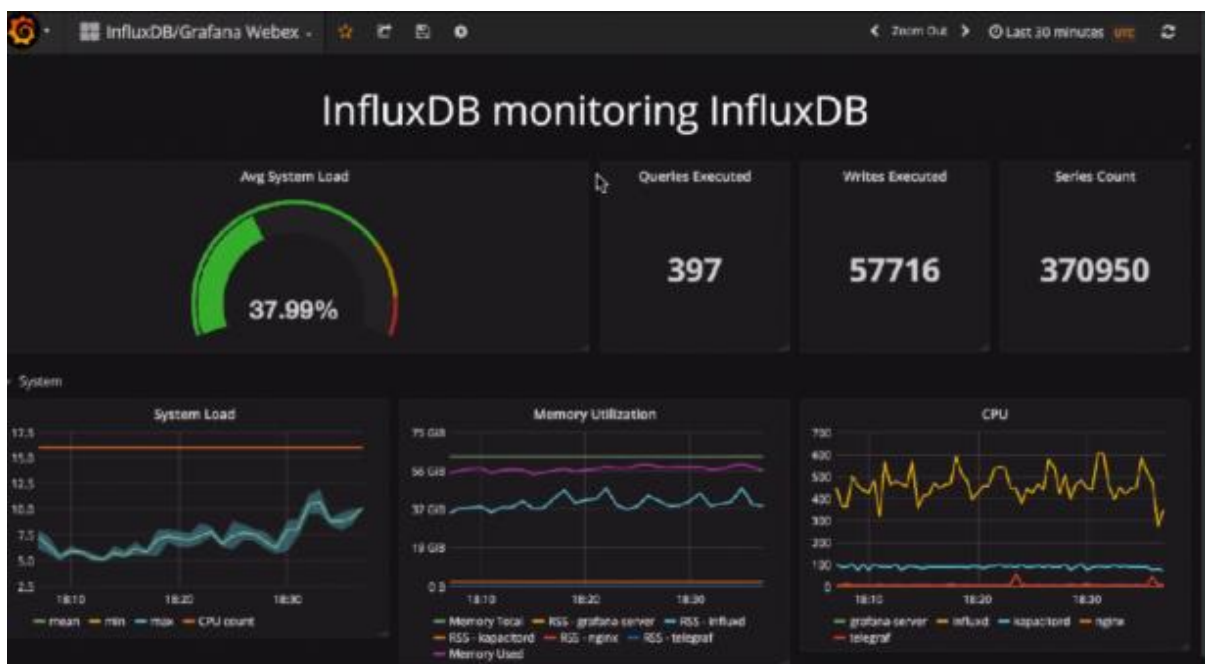
Figure 30 Example Grafana Dashboard [62]

There will be graphs that will be used to dictate the quality of the created topics, rather than the stance of media outlets and these are essential in order to form a better understanding of how the stance analysis was performed on these topics, as well as the significance of the results. Below I will discuss some of the different graphs that are planned to be implemented within the development phase.

Displaying the word frequency in each topic may also be useful to determine the dominance of each topic as discussed within the literature review. The choice of graph could be a histogram as it is one of the most informative and clear graphs in attempting to determine the total word frequency within a topic.
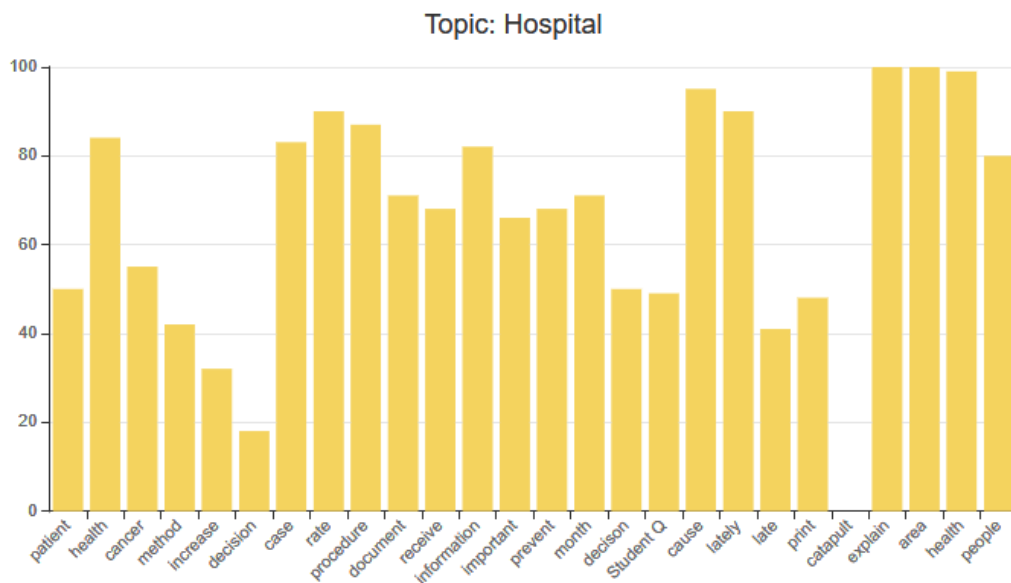
Figure 31 Word Frequency Topic Distribution Histogram

Below is a hand drawn design for how topics will be displayed in the LDA model. On the left a graph shows all the topics with numbers designating a topic. The larger a bubble the more commonly this topic is mentioned within the media. On the right is the display for a topic (9) which links to hospital as can be seen through the correlation of words such as patient, cancer, information etc.

Initially the LDA display does not process and correlate the imagery topic (which is the number) to the real topic as further processing and implementation is required to link it to the real topic. The distance between topics also represents how far away each topic is from each other. This is calculated using the similar words that are shared between topics. For example, if two topics were football and basketball, then they would be tightly linked as multiple words would be shared between them such as "lose, win, team".
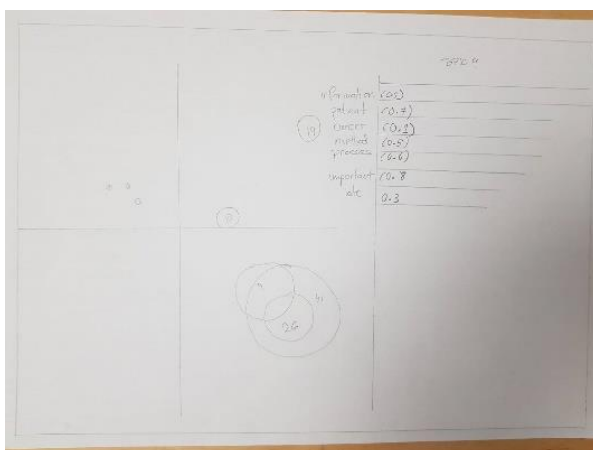
Figure 32 Hand drawn Topic Display

3.5. Design of Functionality

3.5.0 Components of Functionality

The middle-tier is the logic layer that drives the system and implements its core capabilities and it is this layer that the complexity from this system arises. The middle tier will implement:

- Web Scraping Articles from various Media Outlets
- Data Cleaning using Natural Language Processing Techniques
- A Latent Dirichlet allocation topic model
- Sentiment Analysis
- A Novel Approach to Labelling Topics
- Mapping the LDA topic Model to the Sentiment Model
- Calculating the Representative Stance Scores.

The use case diagram below shows the different pieces of functionality available to the user. The first piece of functionality allows the user to view the overall stance towards a topic created by the LDA model which provides more in-depth information such as the words used. The second option allows the user to see all the topics and stance scores generated with a clear interface such as Grafana. Finally, the last option shows all the topics without stance. This option also provides more information on the topics themselves such as what words make up that topic.
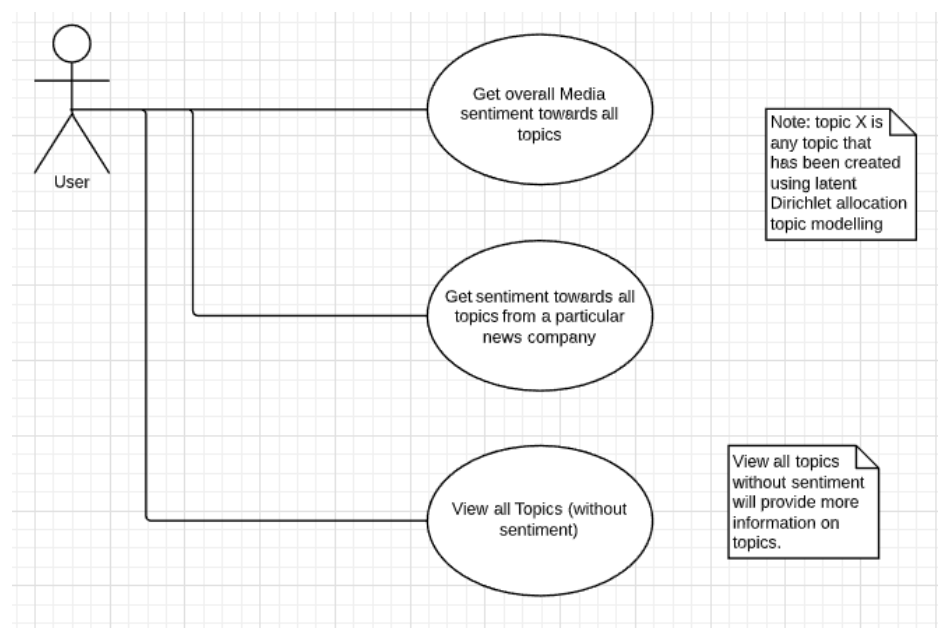
Figure 33 User Use Case Diagram for Viewing Topics and stance Scores for Media Outlets

3.5.1 Web Scraper

A suitable method will be required to scrape newspaper articles from API's and where API's are not available through an inbuilt web scraper that is targeted at specific media outlets. Due to the large amount of data required it is likely a thread capable web scrapper will need to be implemented in order to decrease the time it takes to capture large amounts of data, especially in order to build the LDA topic model.

The below design shows cases how URL's will be scraped for each news company that will be monitored. After running multiple tests, a thousand articles seem to be a sufficient number of articles to examine per topic.

URL Web Scraper Execution
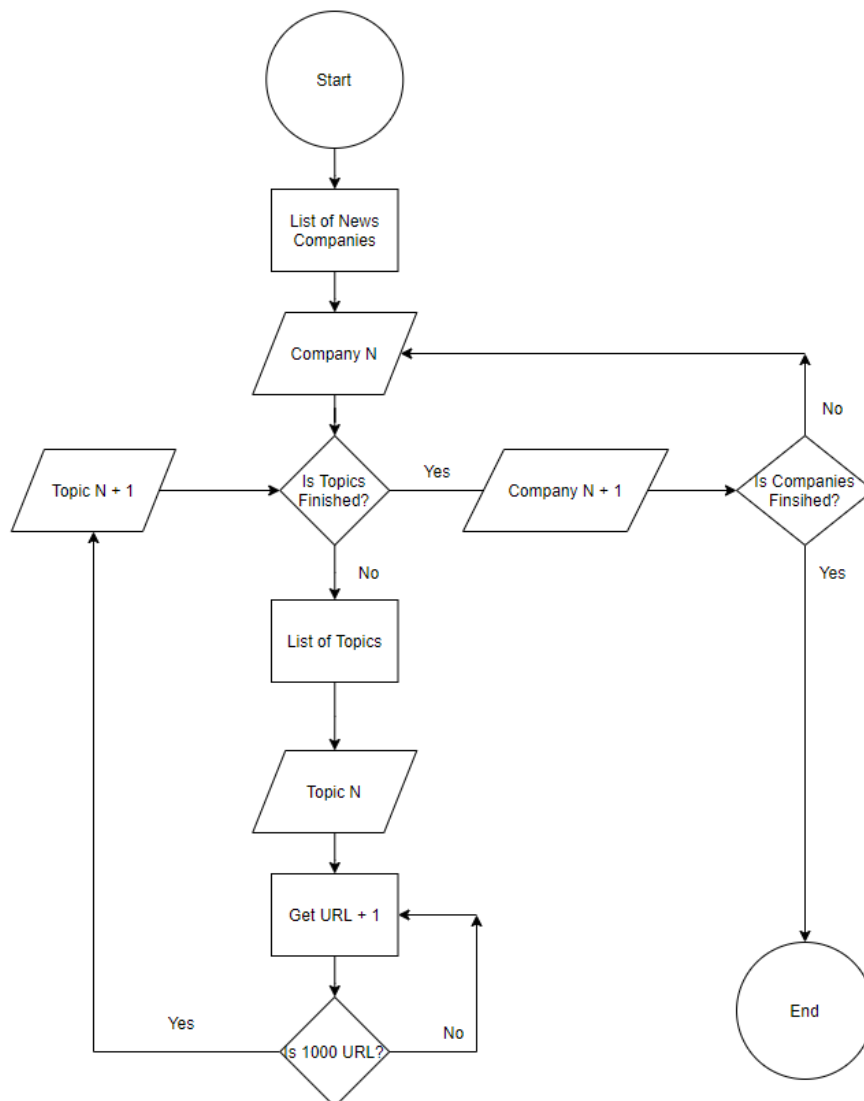
Figure 34 Web Scraper Architecture for URLs

The below diagram shows the path of execution for the article web scraper. This web scraper uses the URL's from the previous diagram and scrapes each article per news company.

Article Web Scraper Execution
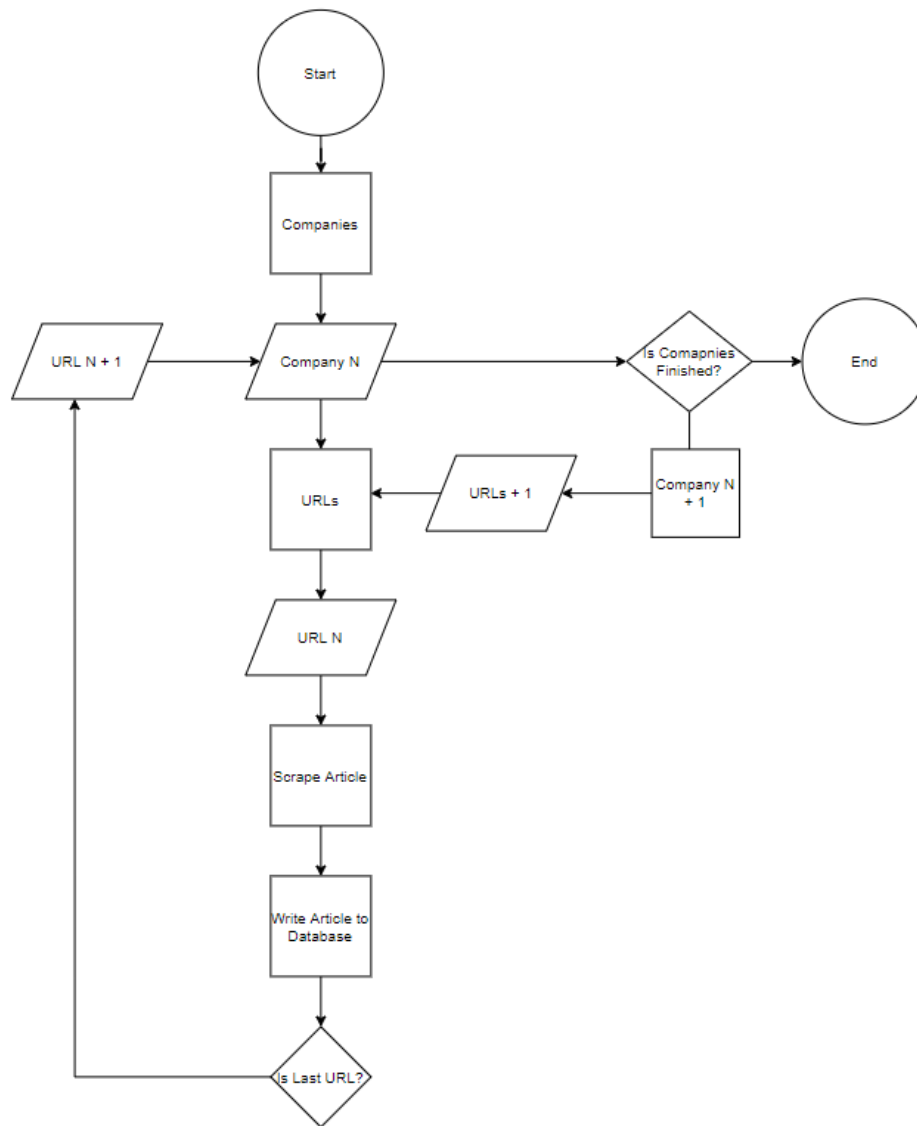
Figure 35 Web Scraper Architecture for Articles

3.5.2 Data Cleaning and building the LDA Topic Model

The technical architecture for data cleaning below shows the data processing required in order to apply the LDA model as well as the sentiment analysis model. This step is pivotal in natural language processing as the data needs to be cleaned in order to efficiently and accurately create the models. Within natural language processing the more work that is performed in the data cleaning layer, the more accurate the models will be and developing a more accurate and efficient data cleaning algorithm will be a future concern in order to increase the accuracy of the model.
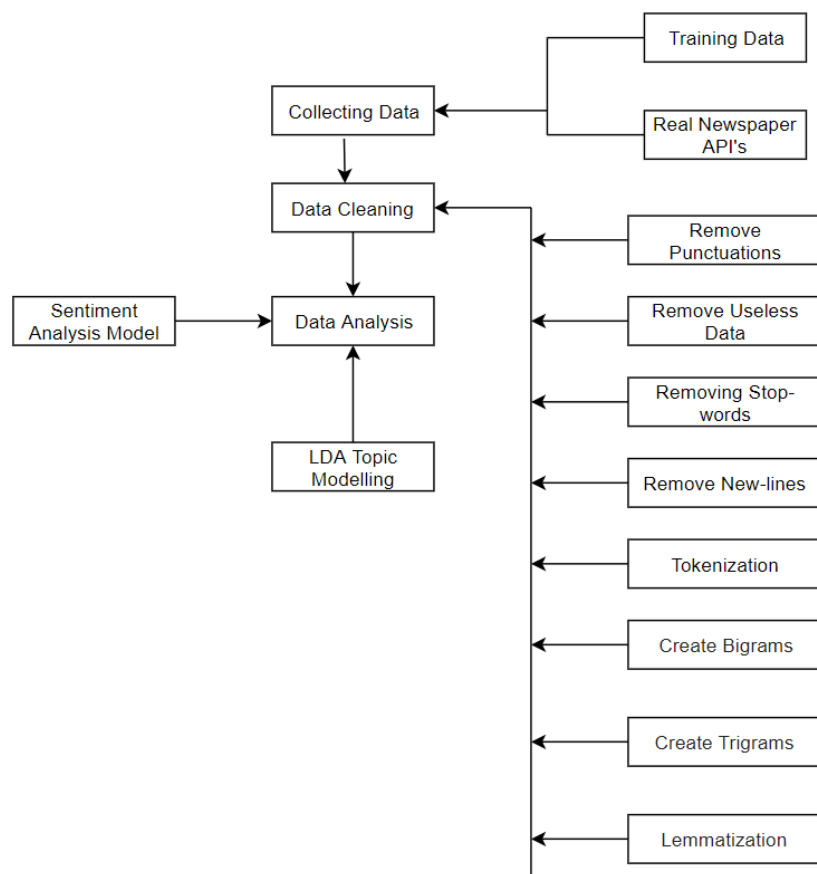
Figure 36 Process for Data Cleaning for Sentiment Analysis and LDA Topic Modelling

The implementation will require multiple natural language processing techniques as can be seen in the diagram above. Some of the techniques are straight forward such as removing punctuations and can be implemented with simple regex commands and other techniques require additional assistance from third party libraries. As an example, lemmatization will require removing all suffixes and prefixes from every word and transforming them into their neutral form. This is a complex area in natural language processing and contains a large amount of edge cases and complexity which are cumbersome to solve manually. A more feasible solution is to investigate and find an optimal library that balances speed and efficiency in order to perform the lemmatization on every word within the newspaper articles.

Building the LDA topic model will require the creation of a word dictionary that maps each word to an ID and corpus, which maps each ID to the frequency of each word. Multiple machine learning

libraries were investigated within the literature review and tested within the design phase. These libraries include Kera's, Mallet topic wrapper, TextBlob and more. From testing it was found that Gensim with Mallet's wrapper were very optimised when it came to training and creating the models. The number of topics generated were also experimented with and within the development section where the data sourcing, cleaning and model creation is optimal, the optimal number of topics that correlates with the highest accuracy were also graphed.

The below graph was generated within the implementation phase. The coherence score reflects the quality and accuracy of the topics. It seems that the optimal number of topics to generate when creating the topic model is around twenty. Earlier and after twenty topics there is a significant difference in the accuracy of the created topic model.
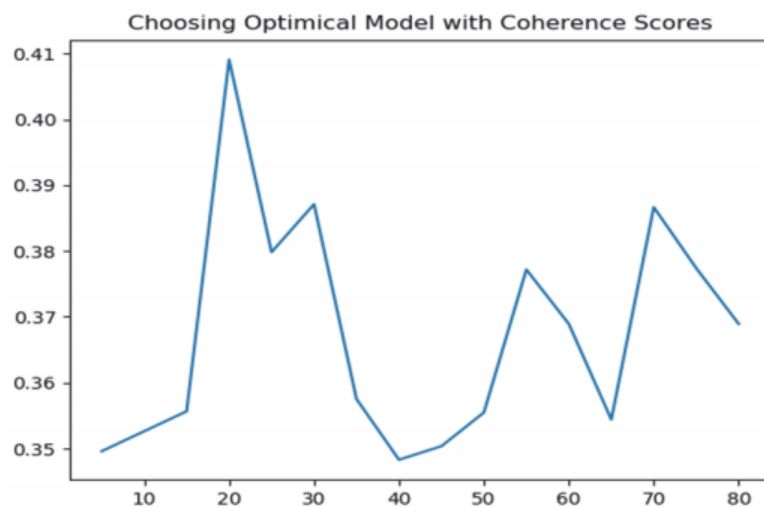
Figure 37 Coherence Score to Topic Number Relation

3.5.3 Opinion Mining Techniques

The opinion mining techniques such as the sentiment analysis model will need to classify each sentence and generate an average score for each topic known as the stance score. Mapping the sentiment score to each topic will then need to be implemented. This will be difficult as the models will be created separately and some method to link them will need to be created. The sentiment model is highly dependent on the data cleaning performed and this will be a key in increasing the accuracy of the sentiment model. More issues were later uncovered within the development phase that required the application of more opinion mining techniques such as the retrieval of the original sentences from the topic model.

3.6 Topic Labelling

A problem that emerged within a later iteration of a prototype was the requirement for mapping the imaginary topics to their real-life counterparts. The objective to implement an automatic topic labelling algorithm developed from the requirements of delivering a stand-alone system that did not need manual intervention.

Topic labelling is complex as understanding the specific words that relate to a topic and correlating them to a topic that encompasses some of them is not a trivial task. As discussed within the literature review, multiple research papers have been dedicated to improving topic labelling algorithms and to this day, modern implementations still lack a high degree of accuracy.

Topic labelling solutions have ranged from machine learning, to manually linking API's and web scraping correlative matches of word frequency to online encyclopaedias such a Wikipedia [63]. More primitive solutions have involved displaying the highest word count. Initially within earlier prototypes, topics were labelled through displaying the highest weighted words within each document but as discussed in the paper *Probabilistic topic models. Communications of the ACM* [6] this is not a sufficient and accurate method of labelling topics.

In initial prototypes to label the topics the hypernym of the top weighted words was used to label the topics. The hypernym is a word with a broad meaning constituting a category. API's such as wordnet were integrated in order to build semantic relationship diagrams between the top weighted words and determine a topic title. This solution was found to inaccurate as the top weighted words could have very different meanings. As an example, the top weighted words for Brexit were "Brexit" and "Deal". Through calculating the hypernym of these words no clear category will be formed.

Another solution that was implemented within the development phase was the creation of a probabilistic distance algorithm between the topic words and the topic names. This solution takes advantage of the fact that the articles that were sourced had specific topics that they discussed.

The algorithm is as follows.

- Each word within every article was mapped within a dictionary to its represented topic. As an example, the sentence "A no-deal Brexit was the potential withdrawal of the UK" would be tagged to the topic Brexit within a dictionary. When the topic is then created by the LDA topic model, all of the words that constitute the topic would be linked to their original sentences.
- Bigrams, trigrams and four-grams were also mapped within a dictionary after some processing to the respective articles.
- After the LDA model created the imaginary topics, the most weighted words that constituted each topic calculated their distance from the set number of topics. This was performed through calculating the number of words that linked to a respective topic, and since these words were the most common words within each topic, this should result in a high degree of accuracy.
- A problem that then occurred within the development phase was the mismatch of topics for topics that were less dominant and this resulted in duplicate labelling's. An example is where business and stocks share a lot of words and since business may more broadly be

discussed it could have been labelled over stocks. To solve this issue a "topic fight" algorithm was implemented within the development phase where both topics perform "probabilistic fights" for the right to the topic label based on word dominance. The loser must search for their new topic after the topic fight and stocks can now receive its correct label. This dominance fight will occur on each duplicate labelling.

The diagram below shows how words were mapped to their respective topics, original sentences and some extra information such as author and date to allow for the integration of future improvements to this project such as calculating how stance changes over time for a news outlet or how stance differs for different authors.
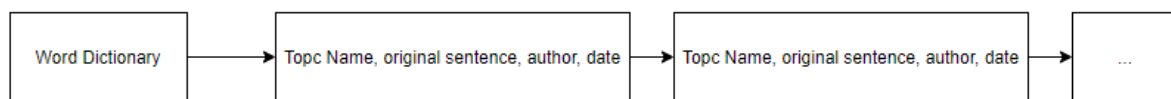
Figure 38 Word to Topic Dictionary

3.7. Design of Backend

The backend is only necessary to store the stance scores for news companies, so that it can be sent through Grafana's native API as a data source. Grafana is normally used for representing times series visualizations and this information is normally sent using time series data sources such as Prometheus and influx DB. MySQL is an alternative data source that can avoid storing the timestamps of when the information was received and furthermore represent the data without the dimension of time.

MySQL also has the alternative capability of allowing data sources to include time stamps. This means that in future iterations in a system that scrapes sentiment data regularly at fixed intervals, Grafana can be used to track sentiment trends within media over time. This functionality is particularly useful for fake news detection, and correlating sentiment trends to the stock market.

The scheme has no logical relationships and while NoSQL would be ideal due to the performance increase, Grafana does not support NoSQL at this time. The database will be a simple table that stores the attributes for individual news companies such as stance scores, average objectivity and

3.8. Technical Considerations

The data created by the middle-tier will be streamed to the front-end as a data source through a MySQL instance. The main display within the front-end will represent the stance the media outlets present towards the various topics. The second display will showcase what constitutes a topic, by outlining the words, as well as how many times each word occurred within a topic.

There is no direct user interaction between the system and the user. The graphs display results from the scraped articles and the models that were trained with the data.

3.9. Conclusions

Through the analysis of multiple agile methodologies, a mixed approach seems to be the most suitable for this thesis. The methodologies that will be employed include Kanban for task scheduling and progress management, a more basic spiral model for risk analysis and CRISP-DM as it follows the exact approach this system will be taking for text mining, creating models and finally evaluating the data.

Kanban and the spiral model will allow for early prototype development, the ability to adapt to change and monitor the completion of each task as well as setting and monitoring goals as they are completed with multiple progress reports generated by the Kanban tool to monitor progress. CRISP-DM will focus on the design of each step with how data is processed and prepared.

The design of different visualisation methods was discussed in order to determine the best methods for visualising the results as well as creating visualisations to improve understanding the accuracy of the created models. The functionality was designed in order to best suit the requirements and create an accurate representation of the stance of news outlets and their objectivity in regards to the predefined topics. The key learning points from within the literature review were considered when designing both the front-end and middle-end of the application in order to reduce possible risks within development.

Based on the key themes that were discussed within this chapter, the following chapter will delve into the development process and many of the issues covered in this chapter will be revisited. The development chapter will discuss the implementations of the design and any challenges or changes that developed as a result of implementation limitations.

4. Experimental Development

4.1. Introduction

This chapter will begin the outline and implementation of the automatic stance detection system that was discussed and planned within the previous chapters. This chapter will also discuss the key challenges faced within the development process due to unforeseen problems and their resolutions that stray from the original design.

As evaluated previously within the introduction and design chapter, the key objectives of the implementation are:

- Data Sourcing Several Thousand News Articles Per News Outlet
- Data Cleaning using Natural Language Processing Techniques
- Designing the LDA Topic Model
- A Novel Approach to Labelling Topics (A Major Area of Complexity Within this Project)
- Stance Detection Techniques to link the sentences that discuss similar topics
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model
- Finding and Implementing an Existing Sentiment Model
- Improving both Models
- Linking Components to Create the Stance Detection System
- Creating a Suitable Display to Graph the Data

As already discussed in the detail, the development of this project will introduce elements from the simplified spiral model, Kanban and the CRISP-DM methodologies in order to adopt the best features that fit this project from each model. The development will be carried out in multiple iterations with the use of prototypes for each new feature that will at first implement the basic functionality with limited accuracy and improve on each feature over new iterations. Kanban will be used for managing goals and the allocation of time on each feature and the specific stages employed for the implementation of text mining and the creation of models will prescribe to the CRISP-DM methodology.

4.2. Technical Architecture Summary

The projected was implemented in the python language, version 3.6 and Java libraries were integrated to make further improvements to the LDA topic model using Java 8. Multiple libraries were used in order to assist in natural language processing, creating the LDA model and visualising the topic model such as SpaCy, NLTK, Gensim and pyLDAvis respectively. The LDA model's accuracy score was computed using both a coherence and perplexity scores.

As mentioned within the design chapter, the architecture for the system follows a non-traditional approach in that the front-end will consume a data source such as the database natively and the logic layer (or middle-tier by convention) will be responsible for the creation of this data. The data is

uploaded to the database and has no direct interaction with the front-end. The diagram showing the relationship between the tiers can be seen here.
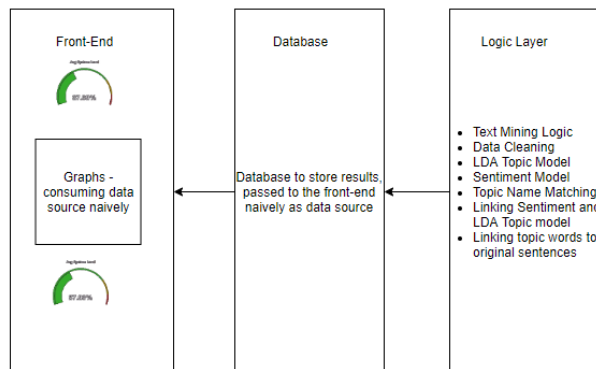
Figure 39 System Design

Due to the model architecture outlined above, the logic layer will be described first, followed by the front-end and finally the back-end. The reasoning for this ordering is that the functionality needs to be first described in order to understand the results that are presented.

The functionality comprises of most of the complexity within the system and will discuss the individual components that were discussed within the design section as new challenges that were not expected.

The design section will present the graphs and results that were created within the development of this project. This will comprise of a Grafana dashboard that will showcase the various stances and objectivity scores held by the news media outlets towards various topics. There will be another graph that will granularly represent what words constitute a topic, as well as the dominance of each word listed from highest to lowest.

4.3. Functionality

The GitHub repository that contains the source code, models, html visualisation file and corpus can be found at the following link:
https://github.com/MichaelLenghel/Automatic_Stance_Detection_In_Media

The functionality section is split into the following subsections.

1. Data Sourcing
2. Data Cleaning and Building the LDA topic model
3. Mapping Imaginary Topics to Real World Topics
4. Mapping Topic Words to Original Sentences
5. Drawing more Information from Results *and Improving Graph Representation*
6. Improving the Quality of Topics
7. Overview of Key Code

4.3.1 Data Sourcing

As discussed within the design section the initial scope of sourcing news articles was to use existing API's such as the global newsapi found at https://newsapi.org/ or the news outlets own public API's. Both of these methods were not suitable for text mining for a number of reasons. Firstly, while the global news API is capable of searching news worldwide, unless the premium version is payed for it will only return the first 128 characters of any article, making it unsuitable for this project due to its high costs. The second approach of using local news API's was also not possible for all news companies as very few news companies provide an open API that can be used to query their news articles. After further analysis, a more generic approach that could be used ubiquitously for all news organisations was required to be engineered.

The solution that was later decided on was creating a web scraper that would use REST standards. The web was designed to scrape URLs that relate to a specific topic and fall between a certain date to provide the relevant articles' URLs. Once the URLs were scraped, the web scraper would then loop through each link for each news company, parse the articles for the title, content, author and date it was written and save them to an existing database. The architecture diagrams for both pieces of functionality can be found below.

Once the web scraper was created a new issue was encountered. The web scraper required hours in order to scrape the required number of articles for a news company topic and this was not permissible as it would take many days of scraping to download all of the required articles for just one news company. The solution introduced was to make the application multithreaded and the introduction of threads significantly reduced the speed it took to copy the articles on to the local machine.

Initially when threads were introduced, this created a new problem as there were hundreds of threads all attempting to access a news website at the same second causing short IP blocks that would only last a couple of seconds. To rectify this problem less threads were used and this stopped the news site from blocking the local IP. The diagrams below are from the design section and reiterate how the web scraper functions.

The below diagrams show the sequence of events that occur within the web scrapers. The two web scrapers were split into two separate forms of execution to reduce coupling. The first web scraper retrieves multiple links from a news source, separated by topic. The topic is searched manually using REST standards and the links are parsed individually. The second web scraper uses the scraped links from the first web scraper and parses for key information about the article.

The diagram below shows how URL's are scraped. Once the link to the news source main page is known, REST standards are used in order to iterate through the links within a page when searching for a specific topic. Each link is parsed from the web page and stored in a corpus that will be later used by another web scraper to store the full articles within a web page.

Link Web Scraper Architecture
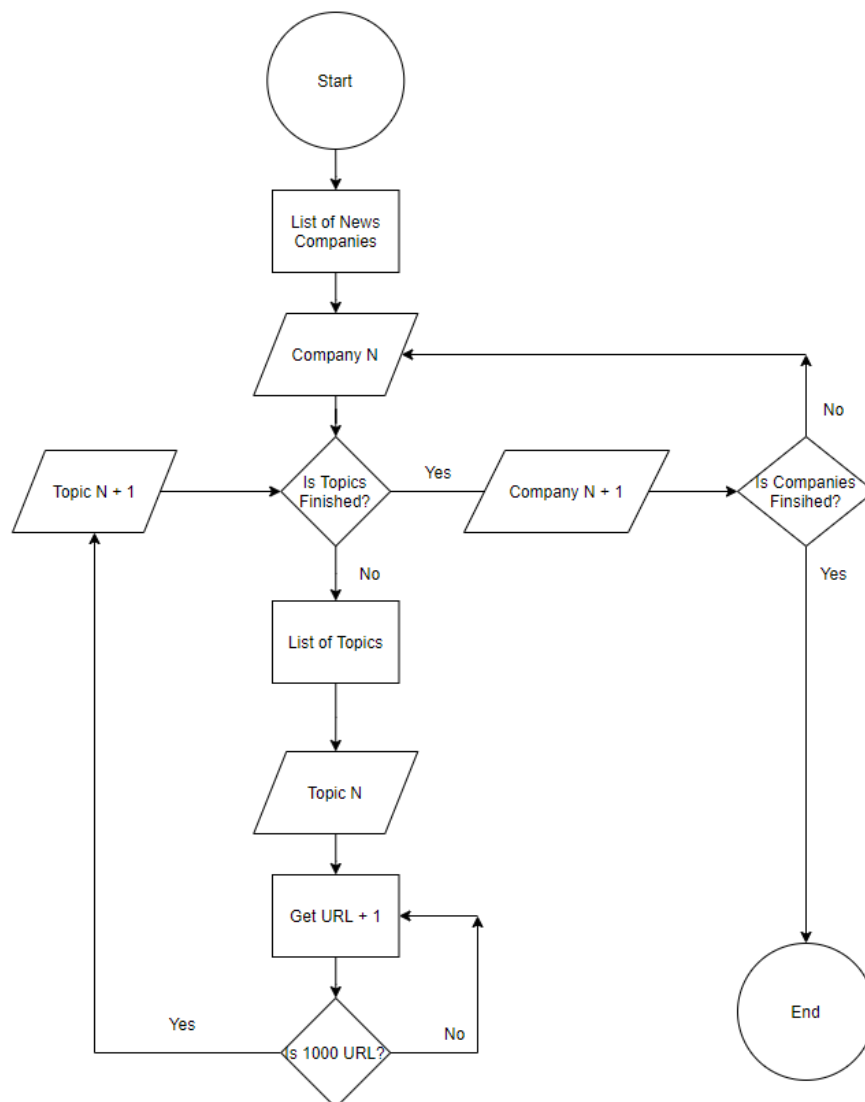
Figure 40 Web Scraper Architecture for URLs

The diagram below shows how articles are scraped from news sources. Each link that was recovered from the previous web scraper is looked up and the page is parsed to gather the following pieces of information:

- The Title of the Article
- The Full Text from the Article
- The Date
- The Author
- Topic Name

Article Web Scraper Architecture
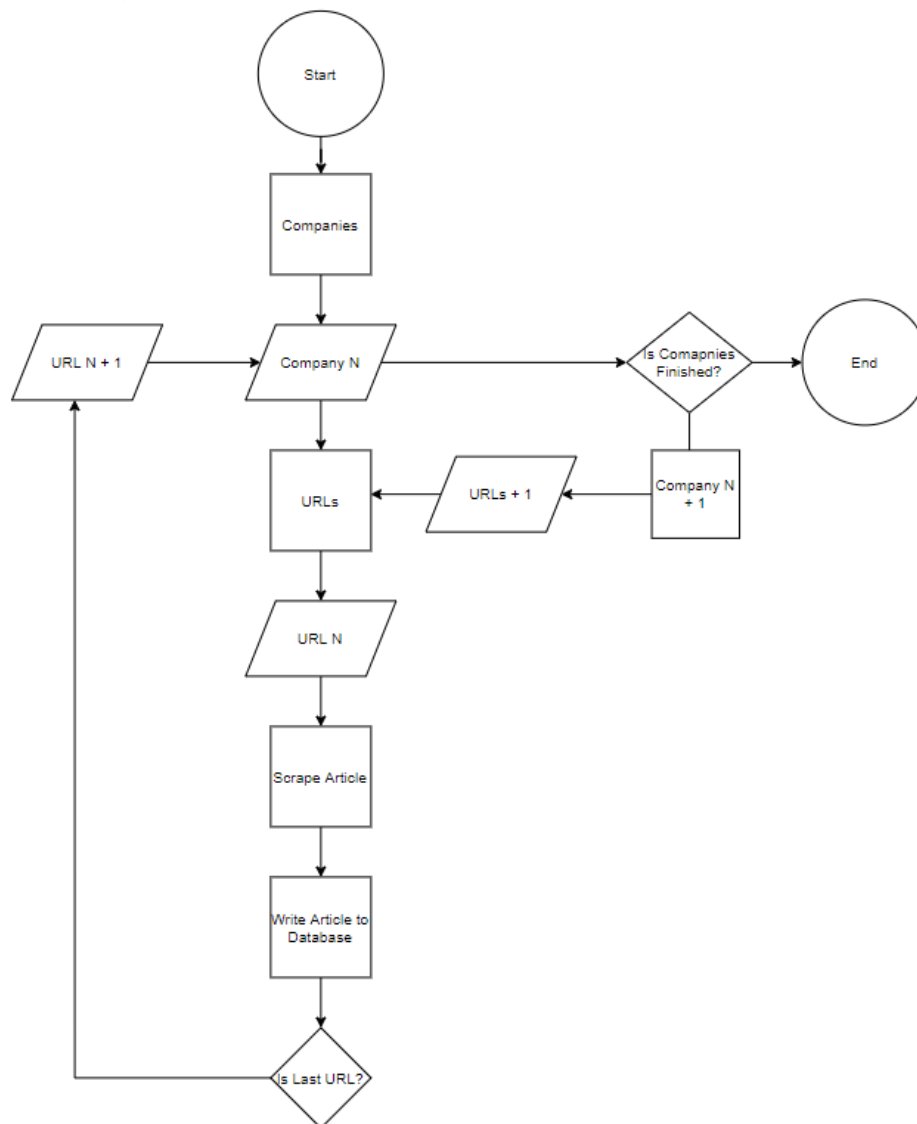
Figure 41 Web Scraper Architecture for Articles

4.3.2 Data Cleaning and Building the LDA Topic Model

The creation of the LDA model depends heavily on process of data cleaning. As discussed within the design chapter, this step is critical, as the higher the quality of data cleaning the more accurate the LDA will become.

Within the implementation, the articles go through a large number of different processes in order to remove unanalysable data. Regex is use to remove new lines, emails, quotes, headers and any other non-meaningful data such as stop words. Bigrams are then created which combine any words that are commonly used together as well as trigrams and four-grams using modelling. Lemmatization is then performed which removes tenses from words in order to have them in their most neutral form.

Data Cleaning and Building LDA Model Process
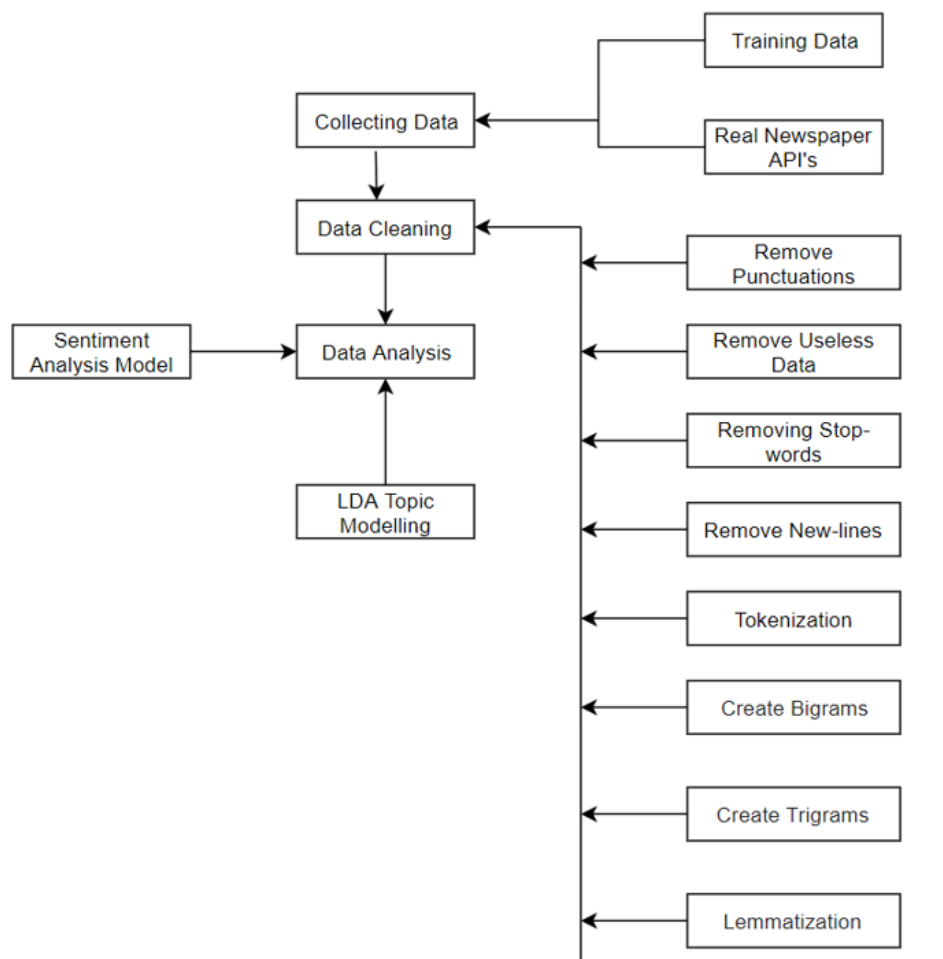
Figure 42 Text Mining and Building LDA Model

Once the data is cleaned, the word corpus and dictionaries are created. The word corpus contains the IDs of words and their frequency and the word dictionary maps IDs to the words. An LDA model

is then created with fifty words per topic and a total of N topics in the current instance of the project. The LDA mallet wrapper is introduced to improve accuracy which provides considerable advantages to only using genism's wrapper and these advantages are further discussed in the evaluation phase.

The number of topics greatly affects the accuracy of the LDA model and this is a topic discussed at length within the area. The quality of data cleaning as well as data size also have spin off effects on the number of topics created. After manual testing and computing the coherence scores based off of multiple topics, it was found that the ideal number of topics was nineteen, providing the highest coherence score from the tested ranges of five to forty.
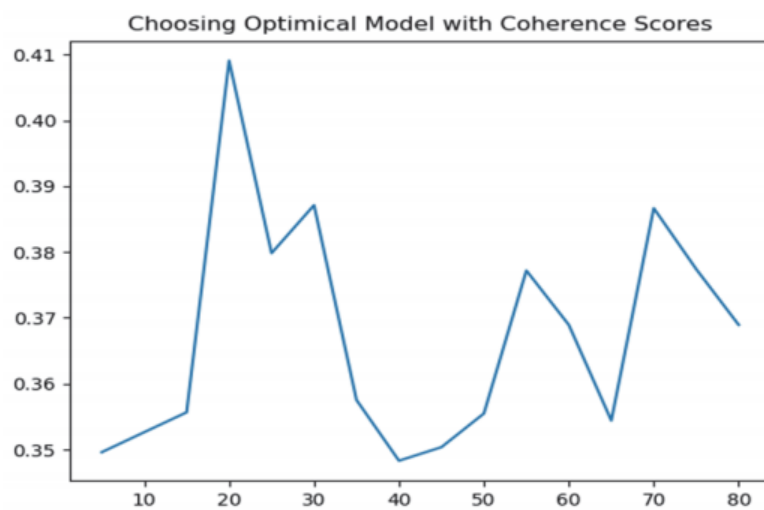
Figure 43 Choosing Optimal Number of Topics [64]

4.3.3 *Mapping Imaginary Topics to Real World Topics*

As discussed heavily within the background research and design section, the mapping of imaginary topics to real world topic names is an extremely difficult area with no complete solutions at the time of writing this thesis. The final solution that was implemented within this thesis is a novel one that attempts to map the distance from the most heavily weighted words within an imaginary topic to their most likely topic using a probabilistic method.

Initially multiple approaches were taken to attempt to solve the issue of labelling topics. Within initial prototypes calculating the hypernym of the top weighted words was used as a method to label the topics. The hypernym is a word with a broad meaning constituting a category. An example of a hypernym is "pigeon, crow, eagle and seagull" which are hypernyms of the word seagull. API's such as wordnet were integrated in order to build semantic relationship diagrams between the top weighted words and determine a topic title. API's such as wordnet were integrated to create these hypernyms but it was found to be inaccurate as the top weighted words within a topic may not have direct association with each other. As an example, the top weighted words for Trump were "Trump" and "Impeachment". Through calculating the hypernym of these words no clear category will be formed.

Another attempt implemented searching Wikipedia with the top weighted terms and using the results to dictate the more general meaning of the topics. This attempt while accurate in some instances and an improvement to the previous approach of calculating hypernyms also fell short and returned inaccurate results. In the previous example of Trump, this method would have worked, but in the example of basketball where the top words are "win", "lose", "team" this automatic real name topic discovery system would be faulty. This approach required using the Wikipedia API and requesting pages that discussed these top weighted words.

The novel approach implemented within this project required taking advantage of the known set of topics that were discussed within the articles. This was possible as the articles were scraped all linked to specific topics such as Trump or Brexit.

A multi-level word dictionary uses each word within the articles as a new key. Each word within the dictionary points to an array that contains the topic name where the article was found when scraping for the word, the original sentence, the URL the word is from, the name of the author that wrote the word and the date the article containing the word was published. The design of the dictionary can be seen below.
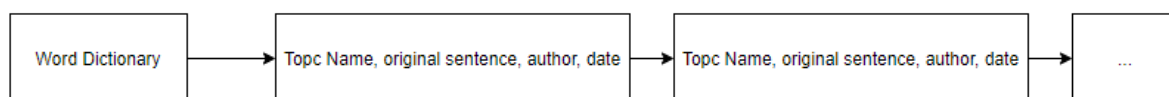
Figure 44 Topic Word Dictionary Mapping

Once this word dictionary is generated, the most weighted words from each imaginary topic are checked against this dictionary and where the topic name appears the most within this check is the real name of the topic until the topic dominance fight begins.

The dominance "fight" (Implemented inside the method "topic_fight" method) is used to compare two topics which are competing for a topic name that share similar attributes. As an example, the imaginary topic that is actually China may receive the real topic name business due to the similar words associated with the topic of business (Tariff, global, economy, cost), but when the topic that represents business is assigned a name, the business topic is already being used by China. This method to fight topics makes both topics compete to see which topic has more similar words and have the right to the topic name of business, where the loser is re-evaluated again and can find its correct topic name now that business is excluded from the topic's options, allowing China to find its correct label.

This method requires knowing what topics are being searched for beforehand, but provides a very accurate mapping of imaginary topics to real name topics. The only issue encountered is that very generic topics cannot be mapped to a real name counterpart and this is a problem that where it is difficult to find a perfect solution as there is not one answer that can fit every generic topic. As an example, even a person may struggle to label the topic of the words "birth, woman, health, choice" which can refer to the topic of pregnancy, child birth or even abortion but it can be very difficult to automatically and assertively map a specific topic without prior context. The meaning of a topic like this can also have different meanings to

different people as complete objectivity is not easy to maintain. For these words the LDA topic model mapped abortion as the topic name and this was indicative of the scraped articles.

4.3.4 Mapping Topic Words to Original Sentences

Mapping of the words from the topic model to their original sentences is crucial as the sentiment analysis model must be applied on the full sentences rather the single words it returns. The problem is that the LDA topic model splits each word individually from its sentences in order for the model to calculate and map to which topic the words are generally referring towards. For the purposes of mapping of words to their original sentences, the probabilistic method discussed above is used due to its high accuracy.

Once the topic names are calculated, the word corpus can be looped through again and now only the sentences of words that map to a particular topic name are returned. This is very efficient since every single sentence will naturally belong to this topic and sentiment analysis can be ran in isolation for each topic, regardless of the order of sentences.
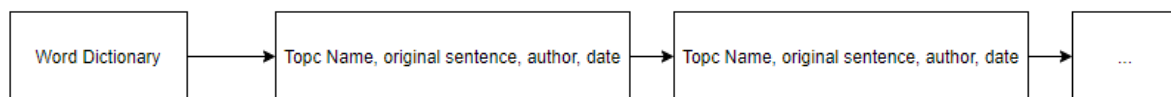
Figure 45 Topic Word Dictionary Mapping

Since the number and type of topics is guided, the possibility is available to map each word to its corresponding sentence and topic name inside a multi-level dictionary and then calculate the sentiments in each sentence of a word that coincides with a topic, dictated by the topic labelling algorithm.

When it came to retrieving the sentences for each word the topic labelling algorithm mapped the topic name to the imaginary topic using a similar distance algorithm to the mapping of topic labels. Each topic word retrieved every sentence that was tagged with that particular topic. The advantage of this approach is that the accuracy of word to sentence mapping to topic is 100% assuming the topic was labelled correctly.

The disadvantage of this method is a high space complexity where a large number of sentences are stored for each word that occurred in thousands of articles. Where space was an issue, an optimisation is to make each sentence immutable after creation to have only one occurrence of each unique sentence and map words to the same unique sentences. This optimisation is similar to how strings are immutable in python and any new declaration of the same string points to the same object within memory unless detached manually. This optimisation makes the space complexity S^T in the worst case where S is the number of unique sentences and T is the number of topics that separate each sentence.

Figure 46 Basic Word to Sentence Mapping

4.3.5 Drawing more Information from the Results and Improving Graph Representation

After analysing the topic stance results for the news organisations, many stances for both the Daily Mail and the Independent managed to remain close to neutral without showing a strong stance with the exception of a few topics such as drugs and Brexit.

Graphs of the mode were introduced as the average can infer the general representation of a topic, but due to outliers it may skew the data inconsistently. The mode on the other hand will display how the topic was most consistently represented, regardless of outliers.

The addition of the mode showed that most sentences were generally neutral, but after further examination many sentences were far from objective with very positive and very negative statements read in the output. This seemed to be as a result of the many sentences being neutral such as "The Bishop left the building" which would return a sentiment score of zero and skew the results accuracy due to the large amount of bland sentiments, but a few select statements would encompass the real sentiment such as "The Bishop showed a tremendous display of courage". In order to improve the results in this instance, results that returned zero were eliminated, to be able to receive a more accurate representation of the results.

4.3.6 Improving the Quality of Topics

Various challenges were encountered within the development process that were initially not planned for within the design phase. Initially the model computed a very low coherence and perplexity score meaning that the quality of topics was very low and a number of different actions to improve these scores were taken. Firstly, a more powerful, exhaustive data cleaning algorithm was introduced to remove irrelevant data. Data cleaning methods introduced include, creating bigrams which combine two words that are commonly used together into one word and removing all special symbols, emails using powerful regex expressions. The data size for building the model was also increased to over seven thousand in order to further increase the accuracy of the model. Modifying the number of topics created (discussed in more detail within the evaluation chapter) and adding a new LDA mallet wrapper were also employed to further increase accuracy.

Another issue that was encountered was the time it takes to create a model. For a relatively small data set of 7000 articles the LDA model, as well as data cleaning took over 20 minutes for completion. In order to increase the efficiency different libraries were looked into that were more

optimised for natural language processing. The library that was previously discussed within the background research phase was SpaCy which has a large portion of its functionality implemented in the C programming language. The library that was used prior was exclusively Scikit-learn which had a slower implementation of the data cleaning algorithms since it is completely implemented within the python language. Certain regex expressions were also further simplified and less exhaustive approaches to remove all useless characters were created.

4.3.7 Overview of Key Code

The overview of the code has been added to the **appendix section** due to the large amount of code involved and discussed. Within this section a quick overview of the code involved will be presented and if more detail is required it can be found within **Appendix A**. Note that only the main files are discussed in order to maintain clarity and helper files which are responsible for re-formatting articles and output are not noted below.

The main code files discussed are:

1. Scrape_articles.py
2. Generate_lda.py
3. Visualize_lda.py

The file "scrape_articles.py" is responsible for scraping URL's and articles from any specified company once the REST standards for that website are known. In order to save time and scrape articles much faster, the application is multithreaded and sets up a daemon system that splits the job of scraping articles into separate tasks.

The "generate_lda.py" file is responsible for data cleaning, creating the word corpus and dictionary and creating the base LDA model. It is also responsible for creating a separate word corpus which maps every word to its original respective sentence, author, date it was published, and URL for traceability.

The "visualise_lda.py" file is responsible for the visualising the created topics, testing the quality of the topics created using perplexity and coherence scores, improving the LDA model using a new wrapper, mapping the LDA topic model to the sentiment model, as well as implementing a novel algorithm to map the imaginary topic names to their real counterparts.

4.4. User-Display

The user-display below is broken into two different subsections. The first subsection describes and showcases the stance of media towards various topics. The second subsection is for providing more granular information and displaying what constitutes a topic such as the words that make up a topic (with weights) and how dominant certain topics are in regards to each other.

4.4.1 Media Outlets' Stance Representation

The representation of media outlets' stance is implemented through Grafana, a multi-platform open source analytics and interactive visualization software. The two graphs that were used to display this information include bar gauges and horizontal bar graphs.

Bar gauges were used to represent the average objectivity and mode stance scores due to the relative representation of different stance. In this instance, the gauge starts from the lowest stance score and rotates towards the highest. This provides a clear understanding of the differences between the which topics are represented the most positive and the most negative within a media organisation.

Both the bar chart and bar gauge stance scores have been truncated to start from zero in order to provide a clearer view of the differences in how stance are represented. The only topics that encountered a negative stance score were housing by the independent and drugs by the daily mail and drugs were significant (-4.7) which made the graphs less clear since it relatively measured the differences.

The first two figures represent the differences in objectivity per topic represented relatively to each other for the Independent and the daily mail. As can be seen when comparing the two figures, the Independent on average seems to be more objective then the daily mail in its use of language. Using this algorithm, the least objective topic for the Independent is the topic of "business" with an objective score of 0.22, whereas the least objective topic for the daily mail is the topic "online" with a score of 0.47.

The scores are calculated using the top ten weighted words within a created topic. As an example, the words that represent Trump are words such as "Trump, Campaign, Election, Candidate, Impeachment and so on". These results do not completely map the objectivity of the news media to Trump as some of the words analysed are more general as is the limitation of LDA topic modelling, but when compared to different news outlets do provide some level of insight.

Figure 47 The Independent Average Objectivity
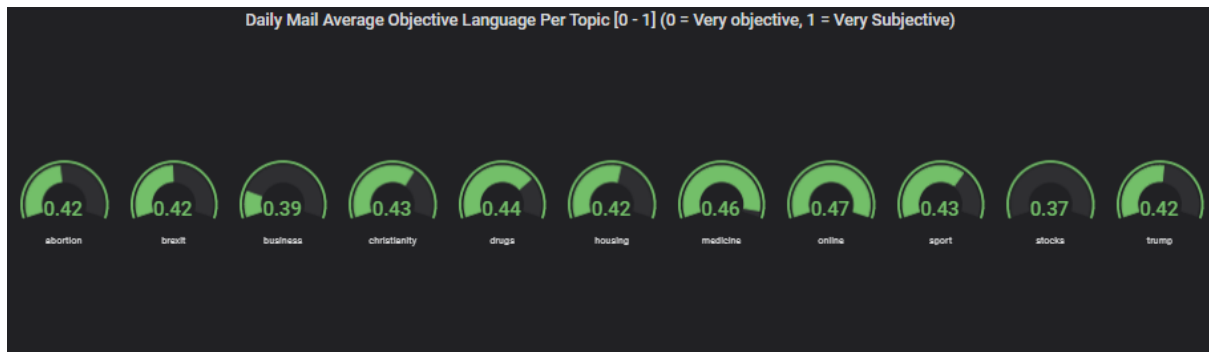
Figure 48 Daily Mail Average Objectivity

Horizontal bar graphs were used for displaying the average stance as similarly to the bar gauge graphs it also provides clear, relative understanding of how different topics were perceived by media outlets as well as clearly defining the relative differences in topic stance.

As can be seen in the bar chart below a more negative view was taken by the Independent towards Brexit than the daily mail and this lines up with the news organisations views.

For the topic of drugs, the more positive trend for drugs in the independent could be as a result of the Independent using significantly more objective language when describing drugs then the Daily Mail which can be seen clearly as the daily mail used an average objectivity of 0.44 when discussing drugs whereas the independent had an average objectivity score of 0.028 as can be seen in the diagrams below.
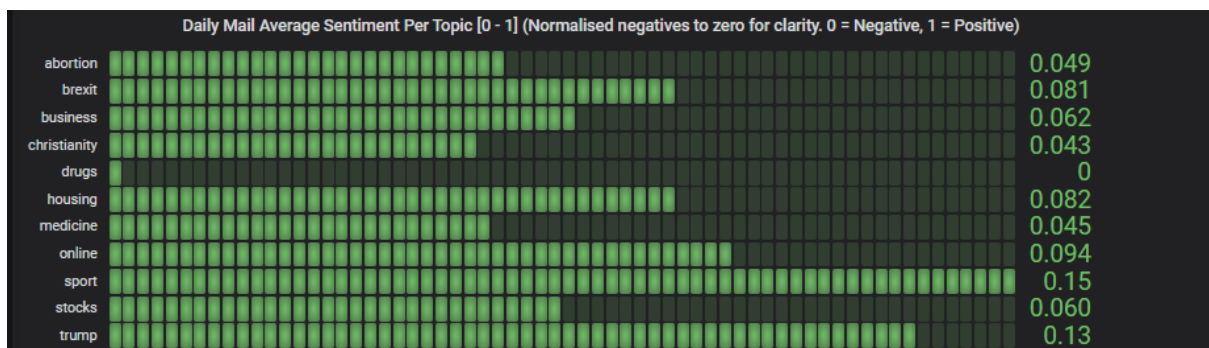
Figure 49 Daily Mail Average Sentiment
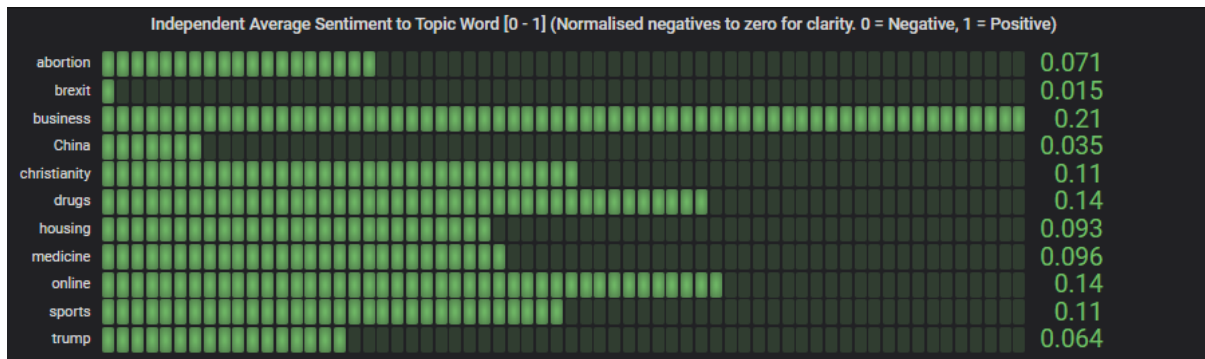
Figure 50 The Independent Average Sentiment

Bar gauges display the mode as the mode seems to step in ".1" increments and the bar gauges seemed to be clearer than the bar graph for representing this information.

The modified mode stance in this instance represents the stance that was most often displayed for each topic. The average can infer the general representation of a topic, but due to outliers it may skew the data inconsistently. The mode on the other hand will display how the topic was most consistently represented, regardless of outliers.

The results for the mode seem to be much higher than the results for stance sentiments above, and this is because the mode initially was either zero or close to zero for almost all of the topics, proving that both papers generally attempted to be more objective, but in order to better understand with the papers did use non-objective language and what their views were when not objective, sentiment results that returned zero were omitted and this is the data that follows. This data shows much more positive results for various topics due to the average zero results not skewing the data lower.
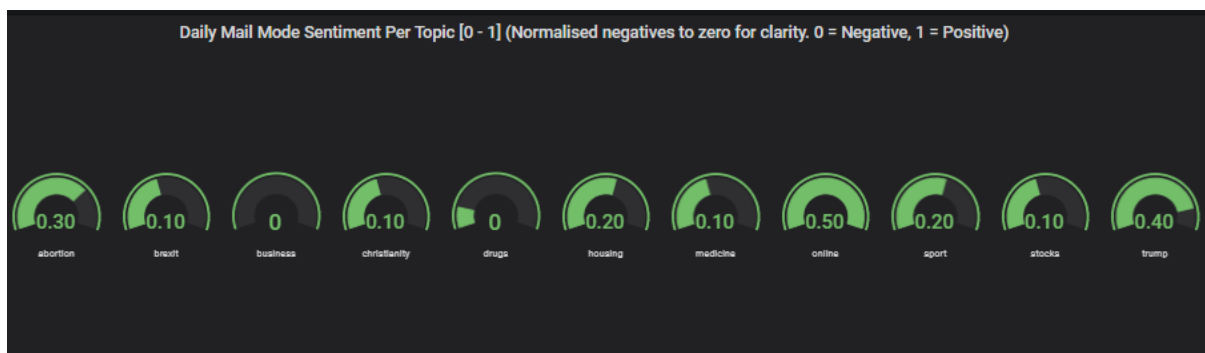
Figure 51 Daily Mail Modified Mode Sentiment
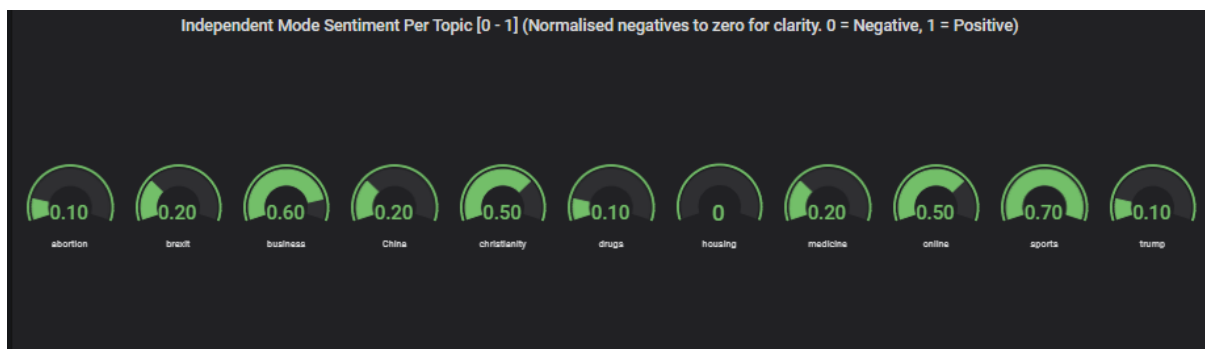
Figure 52 The Independent Modified Mode Sentiment

4.4.2 Representation of Imaginary Topics

This section discusses the layout for the display of the created imaginary topics, labelled as numbers. The display was created with the assistance of the pyLDAvis.gensim library which was specifically created in order to provide access to powerful methods for visualising topic models.

In the topic graph below, each number within the graph represents an imaginary topic with a list of words that relate to that topic. As an example, the topic which has been clicked is marked in red (42) and all of the words that are associated with this topic can be seen on the right of the graph. Words such as information, patient, cancer, cause and case instantly can be related to a hospital or health. Therefore, the number 42 represents this real-life topic of health or hospital and the algorithm which maps the imaginary number to the real-life topic is then ran to convert it using a probabilistic method.

The distance between topics represents how closely linked the different topics are to each other. As an example, the diagram below which represents the topic 26 is completely covered within topic 41s circle. This represents that there is a strong link between the two topics and when examining the contents of the two topics they share many words, however the words that are used are generic in 41 (would, go, make, think) and in 26 the words that are used are presumably from political articles where this more common direct speech from anecdotes is present (people, group, believe, speak, die) and many generic words are often tightly coupled. The interface shown below matches the hand drawn graph which was the proposed method for displaying topics in the design chapter.
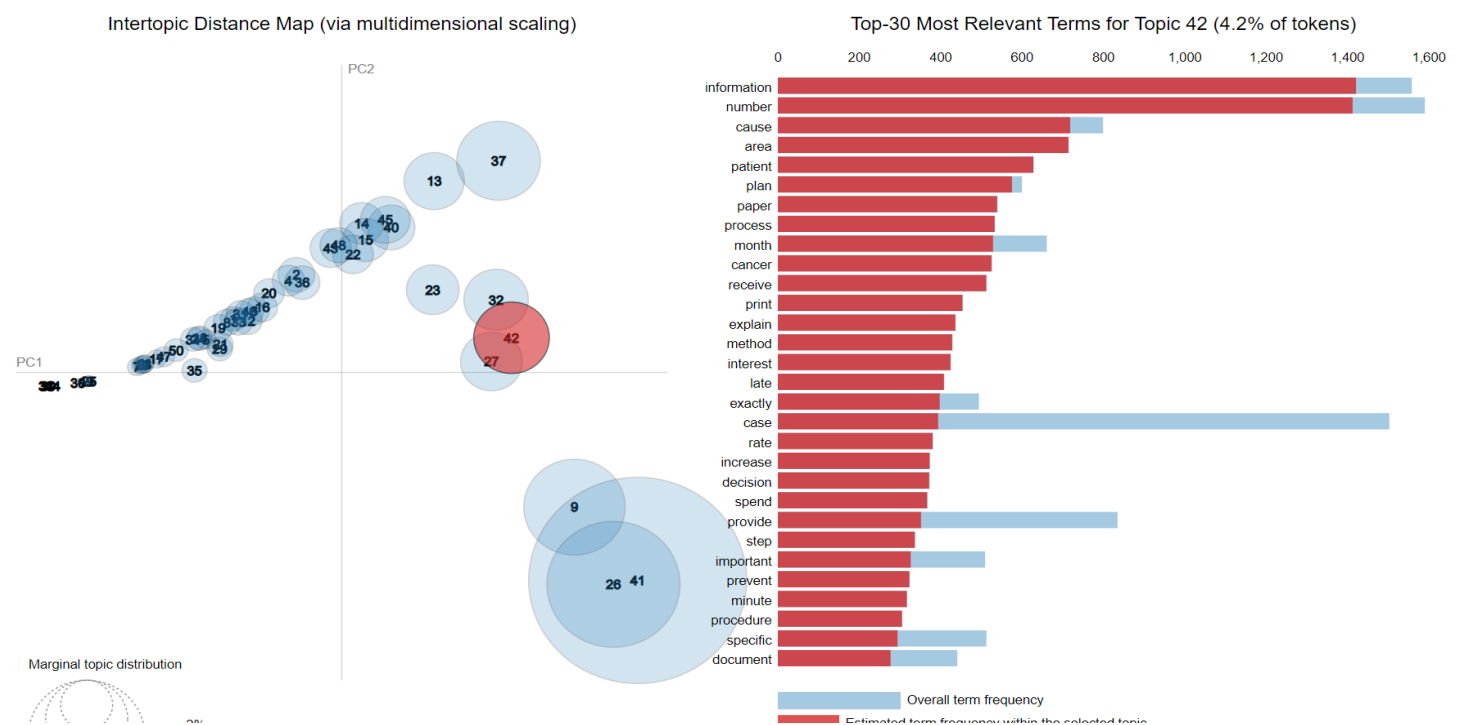
Figure 53 Topic Modelling Visualisation for Topic 42 (Hospitals)

The word count for each topic is also present within the graph, outlining how extensively different words are used within the various topics.
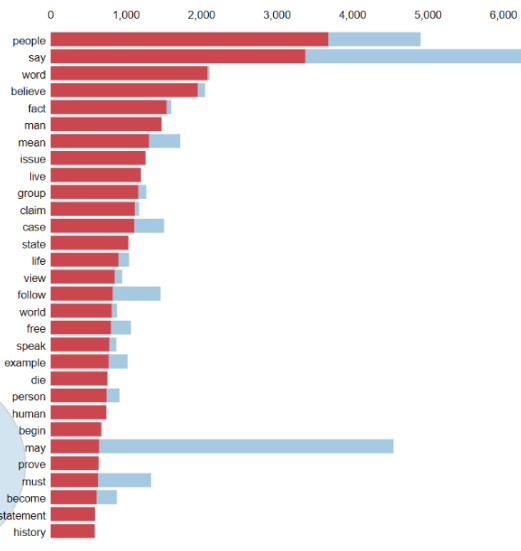
Figure 54 Topic Modelling Visualisation for Topic 26 (Politics)
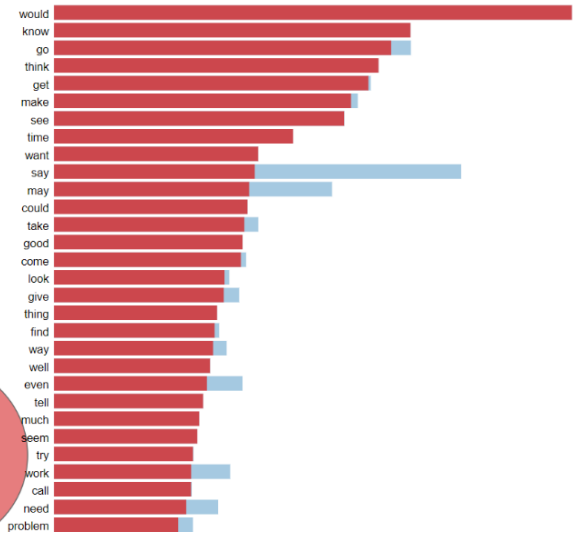
Figure 55 Topic Modelling Visualisation for Topic 41 (Generic Words)

4.4.3 Stance Detection Example

The below figure shows the results achieved through the execution of the automatic stance detection system. The meaning behind the columns is as follows:

- The first column shows the imaginary topic label which are arbitrary numbers from zero to eleven.
- The second column was created by the novel topic labelling algorithm using probabilistic distance and topic dominance to the advantage of the system.
- The third column describes the stance associated with topic word itself.
- The fourth and fifth columns are the key columns which describe the average and mode stance associated with this topic.

```
Index: 0 Topic: sport Topic Word Average Stance: 0.278941714227455685 Average Stance: 0.14917241379310349 Mode Stance: [0.2] Median Stance: 0.010714285714285718 Topic Word Averag
Index: 1 Topic: drugs Topic Word Average Stance: -0.4272729872729874 Average Stance: -0.026031148299241753 Mode Stance: [-0.05] Median Stance: -0.03333333333333 Topic Word Ave
Index: 2 Topic: abortion Topic Word Average Stance: 0.11306231016594753 Average Stance: 0.1020759791180417 Mode Stance: [0.2857142857142857] Median Stance: 0.125 Topic Word Avera
Index: 3 Topic: stocks Topic Word Average Stance: 0.08872581122581125 Average Stance: 0.06025514752195067 Mode Stance: [0.1] Median Stance: 0.0666666666666667 Topic Word Average
Index: 4 Topic: business Topic Word Average Stance: 0.08938366212562604 Average Stance: 0.0622062932993655556 Mode Stance: [-0.2] Median Stance: 0.05952380952380953 Topic Word Ave
Index: 5 Topic: trump Topic Word Average Stance: 0.14952749383643216 Average Stance: 0.12961878248545927 Mode Stance: [0.4] Median Stance: 0.13636363636363635 Topic Word Average
Index: 6 Topic: online Topic Word Average Stance: 0.05035723197706868 Average Stance: 0.0942592781551347 Mode Stance: [0.5] Median Stance: 0.1 Topic Word Average Subjectivity: 0.
Index: 7 Topic: medicine Topic Word Average Stance: 0.06625663875046132 Average Stance: 0.0453151616167556 Mode Stance: [0.1] Median Stance: 0.0750000000000001 Topic Word Averag
Index: 8 Topic: housing Topic Word Average Stance: 0.05556248669363423 Average Stance: 0.0815051732756525 Mode Stance: [0.25] Median Stance: 0.08 Topic Word Average Subjectivity
Index: 9 Topic: christianity Topic Word Average Stance: 0.05952978930870088 Average Stance: 0.04319353877136356 Mode Stance: [0.1] Median Stance: 0.06818181818181818 Topic Word A
Index: 10 Topic: brexit Topic Word Average Stance: 0.05440161988549084 Average Stance: 0.08124640294672168 Mode Stance: [0.13636363636363635] Median Stance: 0.0833333333333333 T
```

The below figure shows the top weighted terms that relate to each labelled topic above.

```
0.4 Words: [['team', 'play', 'point', 'game', 'player', 'season', 'start', 'lose', 'make', 'sport', 'lead', 'time', 'score', 'open', 'win']]
an Subjectivity: 0.4166666666666667 Words: [['police', 'charge', 'drug', 'year', 'woman', 'man', 'officer', 'find', 'claim', 'victim', 'sentence', 'case', 'court', 'arrest', 'kill']]
an Subjectivity: 0.4 Words: [['abortion', 'state', 'law', 'woman', 'file', 'rule', 'court', 'decision', 'issue', 'federal', 'group', 'legal', 'public', 'ban', 'bill']]
666667] Median Subjectivity: 0.33333333333333337 Words: [['year', 'market', 'fall', 'rise', 'economy', 'low', 'report', 'impact', 'global', 'share', 'growth', 'price', 'expect', 'month', 'high']]
an Subjectivity: 0.375 Words: [['company', 'business', 'year', 'pay', 'sell', 'buy', 'cost', 'property', 'money', 'home', 'product', 'customer', 'firm', 'online', 'financial']]
bjectivity: 0.4125 Words: [['campaign', 'trump', 'candidate', 'vote', 'election', 'voter', 'democratic', 'support', 'state', 'president', 'debate', 'political', 'primary']]
454545454545453 Words: [['comment', 'statement', 'post', 'people', 'claim', 'make', 'social_media', 'video', 'online', 'family', 'continue', 'share', 'add', 'include', 'give']]
Subjectivity: 0.43 Words: [['patient', 'drug', 'study', 'doctor', 'treatment', 'health', 'woman', 'find', 'medicine', 'medical', 'year', 'cancer', 'risk', 'people', 'research']]
0.41111111111111115 Words: [['government', 'party', 'leader', 'deal', 'year', 'prime_minister', 'leave', 'number', 'talk', 'rule', 'make', 'election', 'trade']]
edian Subjectivity: 0.4083333333333334 Words: [['church', 'year', 'people', 'fire', 'faith', 'world', 'work', 'christianity', 'religious', 'cathedral']]
ity: [0.5] Median Subjectivity: 0.4 Words: [['government', 'year', 'prime_minister', 'leave', 'make', 'deal', 'trade', 'number', 'role']]
```

4.5. Backend

The backend is only included as a means to send the results of the news outlets stance to the front-end as a native data source to Grafana and as a result is not an area of complexity within this project. Below, the table created encapsulates the information for a news company in the table named 'topic'.'sentiments'. The backend is using a simple MySQL server with select permissions enabled for a created Grafana user that will access this data.

```sql
CREATE TABLE `topic`.`sentiments`(
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `company` varchar(45) DEFAULT NULL,
  `topic` varchar(45) DEFAULT NULL,
  `year` year(4) DEFAULT NULL,
  `avg_topic_word_sentiment` float(5,4) DEFAULT NULL,
  `avg_sentiment` float(5,4) DEFAULT NULL,
  `avg_objectivity` float(5,4) DEFAULT NULL,
  `mode` float(2,1) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1;
```

Figure 56 MySQL Sentiment Table

5. Testing and Evaluation

5.1. Introduction

This chapter will discuss the methods for testing and evaluating the automatic stance detection system that was discussed in the previous chapters. The objectives were to create a system that can accurately determine media objectivity and bias. This consisted of the relevant components to ensure this task could be performed, primarily through the creation of an accurate LDA topic and sentiment model and the introduction of a novel technique to label topics. The evaluation will be performed through examining the accuracy of these components through a number of different techniques.

The purpose of testing is to ensure that the functionality of code works as expected. Testing will be performed through automated tests to ensure that the functionality and logic of the code works as expected and as new code is introduced to catch edge cases early and prevent them.

The evaluation of the automatic stance detection system will be performed through evaluating its specific components. The LDA and sentiment model will be evaluated through a mix of computer-generated scores as well as testing their ability to perform their tasks to that of random participants.

What makes this evaluation unique is the introduction of the random participants in order to evaluate the accuracy of the models. Comparing the results of the models to that of human provides a higher degree of accuracy then only using computer generated scores.

5.2. Stance Detection System Components and Performance

In summary, a stance detection system that can determine stance within media outlets to determine bias has been implemented. This has been achieved through the successful creation of the following components that constitute the stance detection system.

- A web scraper that can source articles from multiple news outlets.
- A data cleaning algorithm to pre-process data before the creation of models.
- An LDA topic model that can dynamically create topics through linking words that refer to particular topics together.
- A novel approach to automatically labelling topics.
- Stance Detection Techniques to link the sentences that discuss similar topics.
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model.
- Opinion mining techniques that implement aspects of sentiment analysis.
- Creating a Suitable Display to Graph the Data

System performance has been shown to be relatively fast when creating the models, taking less then half an hour. The web scraper when ran on a desktop with an i5 6500k processor was found to take up to a full day to scrape the required number of articles for a news company. This limitation is due to concerns to not overwhelm the web server for the media outlets rather than the process being

slow as the web scraper has been multi-threaded but only two threads out of the intended 256 threads were used as media outlets websites are not capable of providing hundreds of requests for articles in a second to a single IP.

5.3. System Testing
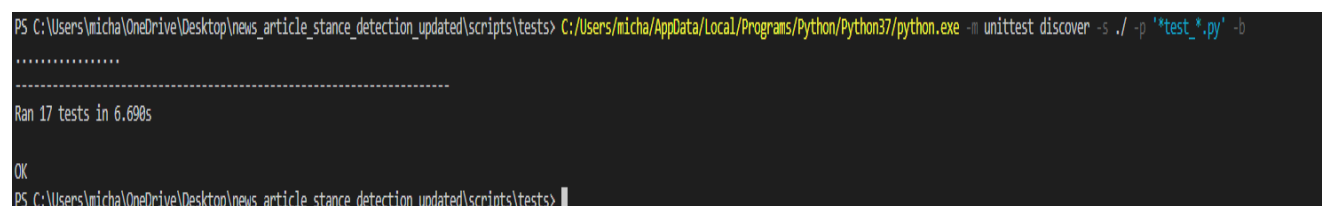
The project was tested iteratively throughout the development process. Each feature that was introduced was first prototyped to quickly introduce the new feature at a basic level and it was then continuously improved through each iteration. The advantage of using prototypes was that it reduced the time it took to implement all of the required pieces of functionality as it allowed for an adaption to change as the development process progressed. Prototyping allows the simulation of important aspects, unveiling the previously not-fully appreciated design issues [71].

User feedback was integrated through random participant surveys. The survey questions were created by the model's predications and the user's ability will be compared to that of the model's ability. The random participants surveys feed into the evaluation of the project which will be discussed in more detail within section 5.3

5.3.1 Unit Tests

Unit-tests were performed in order to ensure that individual units of the software such as methods are well tested and account for edge cases. Unit testing is the first level of software integration and ensures that as new code is introduced, each prior unit does not break in terms of functionality. Unit tests also act as documentation that explains what the code is performing at various stages.

Below the performance of the unit tests as well as a small subset of the unit tests can be seen in the following figures.

```
PS C:\Users\micha\OneDrive\Desktop\news_article_stance_detection_updated\scripts\tests> C:/Users/micha/AppData/Local/Programs/Python/Python37/python.exe -m unittest discover -s ./ -p '*test_*.py' -b
.................
----------------------------------------------------------------------
Ran 17 tests in 6.690s

OK
PS C:\Users\micha\OneDrive\Desktop\news_article_stance_detection_updated\scripts\tests>
```

Figure 57 Unit Tests Execution

The below unit tests test the output of the sentiment retrieval method, the retrieval of the word corpus and the calculation of the median and mode. These tests ensure that output is as expected in different scenarios. In the second last test "test_calculate_sentiment_mode" an edge case was caught that was not previously known and that returned errors at what seemed like random intervals when the mode was calculated for sentiment. The error was that if multiple modes were

calculated, it returned the entire list rather than a single float value. This test caught the error instantly and shows the value of unit testing, as an appropriate fix was then applied.

```python
def test_getSentiment_polarity(self):
    # Test if sujective in range from - 1 to 1 where -1 is negative and 1 is positive

    # Negative test case
    self.assertLess(self.vl.getSentiment('This referendum is terrible')[0], 0)
    # Positive test case
    self.assertGreater(self.vl.getSentiment('This referendum is amazing')[0], 0.5)

def test_getSentiment_objectivity(self):
    # Test if objecvtive in range of 0 to 1 where 0 is objective and 1 is opinion

    # Negative test case
    self.assertEqual(self.vl.getSentiment('This referendum is terrible')[1], 1)
    # Positive test case
    self.assertLess(self.vl.getSentiment('This referendum exists')[1], 0.5)

def test_retrieve_word_corpus(self):
    # Ensure does not return an empty list
    self.assertNotEqual(self.vl.retrieve_word_corpus, [])

def test_calculate_sentiment_median(self):
    # Test polarity median
    self.assertEqual(self.vl.calculate_sentiment_median(self.median_pol, self.median_obj)[0], 3)

    # Test subjectivity median
    self.assertEqual(self.vl.calculate_sentiment_median(self.median_pol, self.median_obj)[1], 12)

def test_calculate_sentiment_mode(self):
    avg_pol = sum(self.median_pol) / len(self.median_pol)
    avg_obj = sum(self.median_obj) / len(self.median_obj)

    # Test polarity median
    self.assertEqual(self.vl.calculate_sentiment_mode(self.median_pol, self.median_obj, avg_pol, avg_obj)[0], 3)

    # Test subjectivity median
    self.assertEqual(self.vl.calculate_sentiment_mode(self.median_pol, self.median_obj, avg_pol, avg_obj)[1], 12)

def test_get_topic_sentences(self):
    # Check filters topic
    self.assertEqual(self.vl.get_topic_sentences('Brexit', ['Brexit:::Auther:::Date:::Brexit is something', 'China:::Auther:::Date:::China is something']), ['Brexit is something'])

if __name__ == '__main__':
    unittest.main()
```

Figure 58 Unit Tests Code

5.3.2 *Integration tests*

Integration testing is the next level of testing that checks multiple components within the software at the same time. While unit tests are useful to ensure different sections of code work well in isolation, integration takes this a step further and adds another level of integrity that ensures the software works correctly. This type of testing requires acting like a user of the application through actions such as:

- Calling a HTTP REST API
- Calling a Python API
- Calling a web service
- Running a command line

These integration tests can be run in the same way that unit tests are written, following an input, execution and assertion pattern. The significant differences is that the integration tests are checking multiple components at once such as a network socket or if an API is reachable.

were also performed where multiple units are combined in order to expose faults within the interaction of units. Big bang or top down testing were considered in order to combine all of the units and test them appropriately.

The integration tests below show that the news company websites are reachable, as well as ensure the correct version of python is installed, as well as java (which is required for creating the LDA model).

```python
class test_generate_lda_model(unittest.TestCase):
    def setUp(self):
        sys.path.append('..')
        import scrape_articles as sa
        self.articles = sa.WebScrapeArticles()

    # Check Independent is reachable
    def test_independent_connection(self):
        self.assertNotEqual(self.articles.makeConn("https://www.independent.ie/"), "Could not make connection 404")
        self.assertNotEqual(self.articles.makeConn("https://www.independent.ie/"), "")

    # Check Daily Mail is reachable
    def test_daily_mail_connection(self):
        self.assertNotEqual(self.articles.makeConn("https://www.dailymail.co.uk/home/index.html"), "Could not make connection 404")
        self.assertNotEqual(self.articles.makeConn("https://www.dailymail.co.uk/home/index.html"), "")

    def test_python_version(self):
        python_version_pattern = "3.7"
        self.assertRegex(sys.version, python_version_pattern)

    def test_java_installed(self):
        # Java message that will be expected (Checking for version prone to fault)
        java_version_stub = "Java(TM) SE Runtime Environment"
        self.assertRegex(subprocess.run(["java", "-version"], stdout=subprocess.PIPE).stdout.decode('utf-8'), java_version_stub)
```

Figure 59 Integration Testing Code

5.3.3 *System testing*

System testing was found to lack a strong application within this system. System testing is a form of black-box testing and requires testing requirements such as the environment setup, monitoring usability, security and documentation. The environment has already been tested with integration tests and the usability is only performed through display graphs. Security and documentation are also safely performed due to the lack of user interaction and the clear documentation provided.

5.3.4 *Acceptance Testing*

The last stage within testing is acceptance testing. This stage is also implemented within Grey-Box testing and ensures that the required functionality is fully implemented and working as expected. This form of testing is performed through such actions as beta testing a product by the end user. In this instance, the acceptance testing will be performed through the user surveys that are discussed within the next section which will analyse the accuracy of the LDA and sentiment model. As the system has no direct user interaction, the use of survey to determine the accuracy of data performs the same functionality as beta users testing the product, as the product in this instance is the data returned.
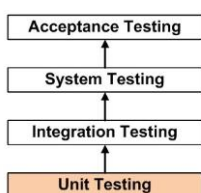
Figure 60 Stages of Integration Testing [26]

5.3.5 Grey-Box Testing

Grey-Box testing was integrated in order to ensure that both the internal logic and paths of execution, requirements and functionality are successfully tested. Grey-box testing combines both White-Box and Black-Box testing. It is a more advanced form of testing that aggregates both methods in order to provide a different angle of testing to ensure that different paths of execution work as expected (from White-Box testing) and find errors in performance, incorrect functions, initializations and more (from Black-Box testing). The aim of Grey-Box testing is to verify system implementations against its specification. [68]

White-Box testing was performed through the use of integration and unit tests. White-Box testing focuses on ensuring the various paths of execution within the code work as expected and the unit's tests and integration tests ensured that edge cases were caught early and that the system works. Where the integration tests checked that web servers and the correct version of libraries were installed, unit tests focused on the specific logic of the code.

Black-Box testing was performed through user evaluation where the internals were not examined and the fundamental outputs were compared to that of users. This form of testing ensures that the results are meaningful and accurate. The diagrams below show how Grey-Box testing works.
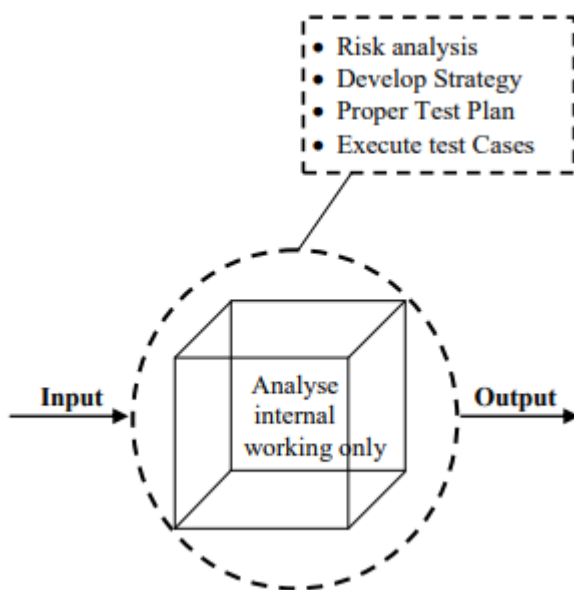
Figure 61 White-Box Testing [69]
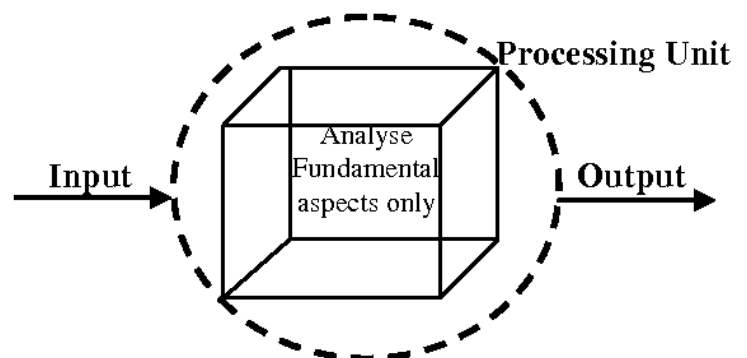
Figure 62 Black-Box Testing [69]
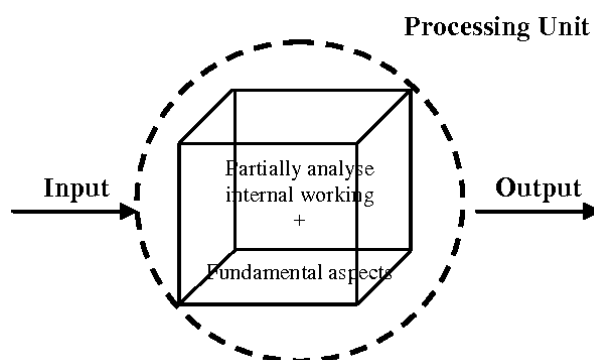
Figure 63 Grey-Box Testing [69]

81

As can be seen from the above diagrams Grey-Box testing encapsulated both the internal working analysis from White-Box analysis and functional aspects of Black-Box testing.

GitHub is used as version control to ensure that any changes that reduce accuracy can be easily rolled back. GitHub also acts as a failsafe in the scenario where sections of the project may be deleted.

The following section will look at the evaluation of the stance detection system. While testing was performed to ensure that the logic and individual pieces of code worked as expected, the evaluation section will measure the accuracy of the stance detection system results.

5.4.0 Automatic Stance Detection System Evaluation

The section will look at the evaluation of the Automatic Stance Detection System. The evaluation will be performed through evaluating individual components of the stance detection system and having these evaluations reflects on the accuracy of the overall stance detection system. The evaluation will primarily focus on determining the accuracy of the LDA topic model, the topic labelling algorithm and the accuracy opinion mining techniques such as sentiment analysis. This form of evaluation will be performed through the following three methods described below.

- Computer generated scores to test the individual components of the stance detection system and having these scores reflect on the accuracy of the system.
- Comparing the accuracy of models to twenty random participants.
- Testing the models against unseen data where the result is known.

The subsections within this section are as follows:

- 5.4.*1 – Present and Discuss Results Obtained*
- *5.4.2 – Evaluation of Models using Random Participant Surveys*
- 5.4.*3 – Quick Note on User Bias*
- 5.4.*4 - Discuss Computer-Generated Scores used to Evaluate LDA topic and Sentiment models as well as the topic labelling algorithm*
- *5.5 – Mapping Topic Words to Original Sentences*

5.4.1 Automatic Stance Detection - Results

5.4.1.1 Parameters for Experiment

As discussed within the development and literature review, understanding the results and creating a baseline for a negative, positive and neutral representation of a topic is a challenging area. The parameters for the results are as follows.

- The Daily Mail will be the news outlet examined.
- The topics that have been provided to the stance detection system are: Abortion, Brexit, Business, Christianity, Drugs, Housing, Medicine, Online, Sports, stocks and Trump.
- Results that were negative have been normalized to zero as drugs at (-0.477) was a strong outlier and this made the relative scale within the bar gauge harder to understand.
- The mode and average stance results are provided and discussed as to which are better.

5.4.1.2 Daily Mail Results

5.4.1.2.1 Daily Mail Mode Results (Bar Gauge)
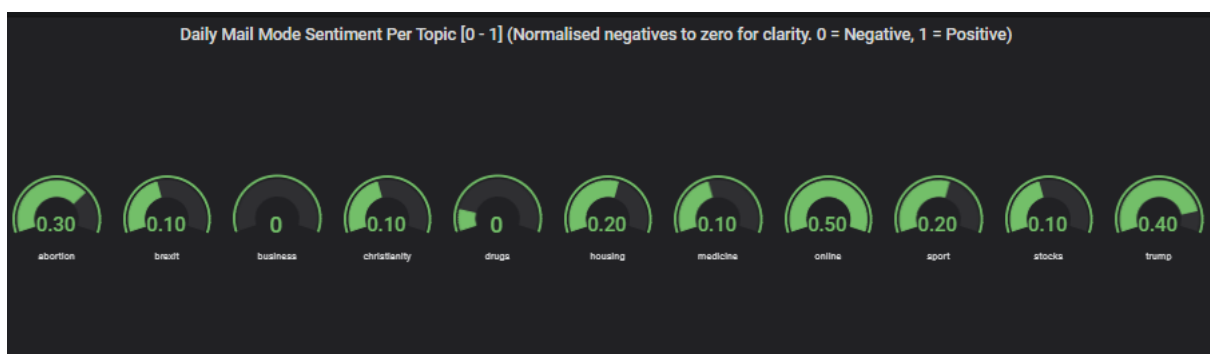
Figure 64 Daily Mail Mode Sentiment

5.4.1.2.2 Daily Mail Average Results (Horizontal Bar Graph)
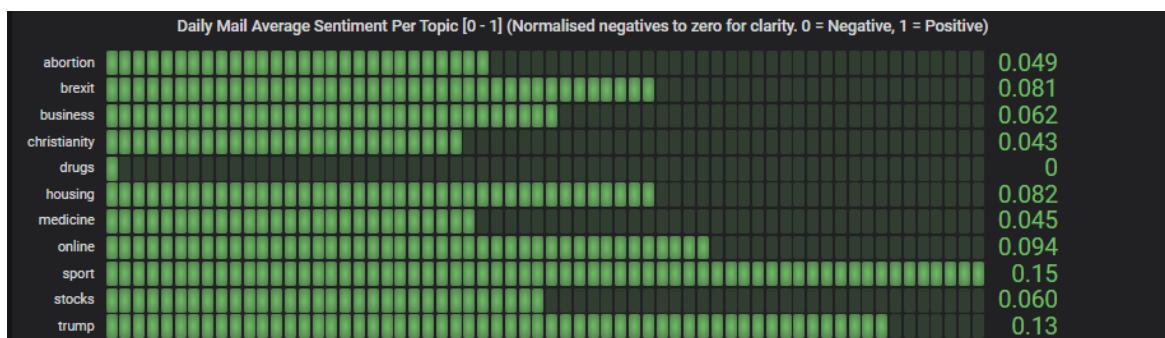
Figure 65 Daily Mail Average Sentiment
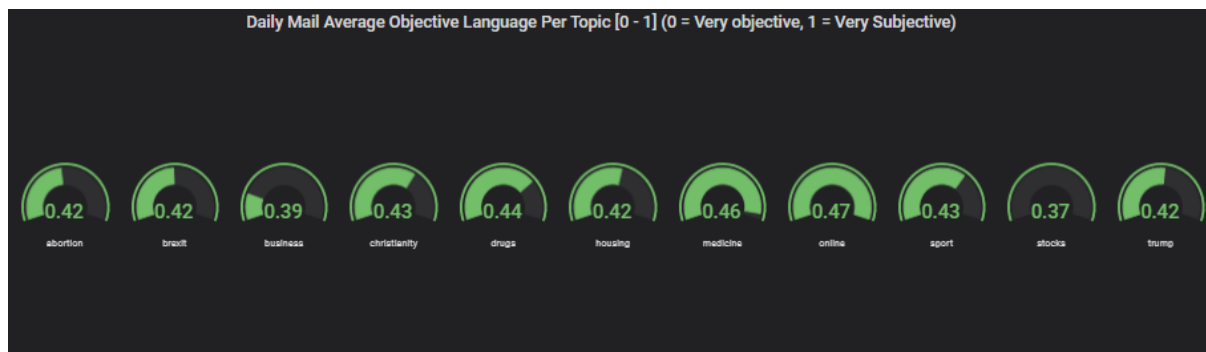
Figure 66 Daily Mail Average Objectivity

5.4.*1.3* Results Explained

As a sample example and summery of how the results were obtained, the topic of "drugs" will be discussed. Initially, hundreds of articles that discussed drugs as the central topic were sourced from the daily mail using a web scraper. Information extracted includes the authors name, the date the article was published as well as the title and contents of the article. A word corpus was then built which maps each word to its original sentence and topic. Data cleaning was then performed and natural language processing techniques were used in order to pre-process the data for the creation and running of the models. Bigrams, trigrams and four grams-all the way up to ten-grams (which represent phrases instead of words that refer to a topic) were also mapped to their topics and original sentences.

The LDA model then built the topic of "drugs" through linking all the words and phrases that mapped to this topic and taking into account which words occurred more frequently then others. These words were matched to an arbitrary number known as an imaginary topic such as "1". In this instance the most used words for the topic of "drugs" were "Police, Charge, Drug, Year, Officer, Arrest, Kill, Claim, Victim, Sentence, Case, Court, Cocaine, Young, Death, Illegal and problem".

A novel labelling algorithm was then applied to all topics. This labelling algorithm used a distance algorithm for each word within the topic of "drugs". This distance algorithm calculated the number of times each word was tagged with a topic, and which ever topic had more tags won and labelled the topic respectively. This solution encountered a new issue where some topics would have dominant word counts and duplicative label multiple topics. In this instance a "topic fight" algorithm would occur where any topic that was labelled twice would calculate its topic score against that of the other topic with the same label. The winner that had a higher topic count within the distance algorithm kept its topic name and the loser would search for a new topic and without the previous topic in the list it would find its correct topic. An example of this issue is the topic of "stocks" and

"business". "stocks" became labelled as "business" due to the word dominance, but the topic fight algorithm allowed them to find their correct topic".

The full sentences needed to then be retrieved in order to apply opinion mining techniques such as aspects of sentiment analysis. This was performed through a similar probabilistic distance algorithm which took the words mapped to their original sentence and topic, and now that the topic was known for each word, the original sentences were retrieved. The order that they were retrieved in did not matter and this allowed the original sentences to retrieved for each word for sentiment analysis accurately at 100%.

Once sentences were retrieved per each topic the stance was calculated through applying opinion mining techniques to measure the individual subjectivity (how positive or negative a sentence was) and objectivity scores.

The results were then output to a database and the database was found to Grafana, a visualisation platform as a data source. Finally, the graphs above were constructed and the results were displayed.

5.4.1.3.2 Reflecting Results Accurately

The average result for how a topic is most represented is not always a strong indication of how a topic is perceived when detecting the stance of news outlets. News organisations in particular use many arbitrary sentences that do not necessarily reflect sentiment, but are setups for the sentences that do reflect the sentiment. This was briefly mentioned within the literature review when researching how to measure media bias [13]. Informative news sources use many more objective sentences and this can greatly affect the subjectivity and polarity scores when looking at the average.

Examples could be "The Pope walked towards the stadium.", "The people followed" and the sentences that do reflect sentiment have less weight as the average brings down the overall result towards zero. Through attempting to sanitize the subjectivity scores that contained these empty sentiments and using the mode value it was found to have a better reflection of calculating sentiment for these news outlets. As can be seen when comparing the average and mode, results seem to be skewed towards 0 within the average due to the high number of sentences that are none expressive, whereas in the mode a different picture is painted. The topic of "Drugs" has a very negative stance score and this is expected, but now business also becomes a negative topic. The topics online and Trump are also seen to have a significantly higher scores due to the lack of non-explicatory sentences which dragged down the results. This would seem to suggest that the mode is a better method of calculating the stance then the average, at least within news outlets where the majority of language is objective. Objectivity can still be calculated from calculating the separate objectivity score.

Through analysing the descriptive words of the media, it was found that it is rare for a public news media to use very positive terms such as "amazing" or terrific" due to their informality in comparison with the large amount of negative words such as "destructive" or "outrageous". This seems to point to a points system that is skewed towards a more negative direction, and this should

85

possibly lower the expectation for positive language. This seems to be why higher results are fewer in number to lower results as can be seen within the figure "Daily Mail Mode Sentiment".

Using the system described above, it stands to reason that a possible approach is if the result is zero the sentiment is negative, any result that falls between 0.1 and 0.3 is neutral and anything above is positive.

5.4.2 Evaluation with Random Participants

5.4.2.1 LDA Topic Labelling Evaluation

A method of evaluation that applies to both the LDA topic model and aspects of sentiment model testing is to check the accuracy against a document that has manually computed either sentiment scores or topics. This requires new data for evaluation as training data that the model was created upon cannot be used. Through comparing the manually generated topics names against that of the user an accurate score can be generated around how closely the model resembles a human's capability of deciphering either sentiment or topic creation. The user's ability to guess the topic generated by the LDA model can also be used to evaluate the accuracy of the LDA topic labelling algorithm.

5.4.2.1.1 Parameters for the LDA Topic Survey

- The topics were chosen at random from the list of topics that were generated by the LDA topic model. These topics are 1. Abortion, 2. Brexit, 3. Medicine, 4. Trump, 5. Housing.
- The words provided for each topic are the highest weighted words within that given topic.
- Each user fills out the topic they believe comprises the given words.
- The premises of the analysis is to determine the accuracy of the LDA topic model and the LDA labelling against that of random participants.

5.4.2.1.2 Results of Survey

LDA Survey

LDA Topic Survey

The lists of words within this survey have automatically been generated. The purpose of these surveys is to check whether or not the words relate to a clear topic. In order to complete this form, please attempt to guess what topic the list of words are referring to. In the majority of cases the topic is the first word in the list, but in some cases the first word is not the topic.

For example from the list ["HBO", "Netflix" and "Amazon Prime"] The topic is most likely Streaming Services. In this case the topic not the first word.

An example where topic is not in the list is the following: ['trade', 'country', 'market', 'economy', 'economic', 'world', 'global', 'export', 'tariff'] . In this instance the topic is "Business" and all the words within the list are words used to describe this topic.

1. From the following list, what is the main topic: ['abortion', 'woman', 'law', 'life', 'issue', 'case', 'week', 'legislation', 'pregnancy', 'baby'] ?

2. From the following list, what is the main topic: ['deal', 'brexit', 'government', 'leave', 'vote', 'border', 'agreement', 'prime_minister', 'irish'] ?

3. From the following list, what is the main topic: ['medicine', 'drug', 'patient', 'doctor', 'health', 'treatment', 'product', 'medical', 'people', 'animal'] ?

4. From the following list, what is the main topic: ['trump', 'campaign', 'election', 'close', 'president', 'political', 'white_house', 'call', 'show'] ?

5. From the following list, what is the main topic: ['house', 'housing', 'people', 'home', 'property', 'build', 'government', 'cost', 'year', 'social'] ?

Figure 67 LDA Survey used to Evaluate Model

For testing the quality of the topics, twenty random participants were asked to fill out this survey and the results are as follows.

For the first topic list refereeing to abortion:

- Nine correctly guessed abortion
- Six Guessed Women's rights
- Three Pro-Life
- One guessed pregnancy
- One guessed life

For the second topic list refereeing to Brexit:

- Thirteen correctly guessed Brexit
- Two guessed Vote
- Two guessed Politics
- One guessed Boris Johnson
- One guessed EU
- One guessed Law

For the third topic list refereeing to Medicine:

- Fourteen correctly guessed medicine or hospital or health
- Three guessed drugs
- Two guessed the topic coronavirus
- One guessed Animal Testing

For the fourth topic list refereeing to Trump:

- Six correctly guessed Trump
- Six guessed election
- Four guessed Election
- Two guessed USA
- One guessed government
- One guessed political

For the fifth topic list refereeing to Housing:

- Nineteen correctly made a reference to the housing epidemic
- One guessed real-estate

From the above results, the LDA topic model was capable of guessing the correct topic for each sequence of words with 100% accuracy, whereas the random participants managed to guess 61% correctly with 39% out of 100 being something related to the current topic. This does not mean the current LDA model is perfect as it was found that the requirement for this LDA topic model and labelling algorithm required dominant topics in order to distribute the words correctly. But it seems that as long as the topics discussed are known in advance and the topics in question are not generic,

the LDA topic labelling algorithm has a very high accuracy in labelling topics. The LDA topic model itself is also very capable of finding the dominant words within non-generic topics.

A dominant topic is a topic where the words used to describe the topic are specific such as Brexit, or Trump and are not shared across multiple topics. The example previously used to discuss topics that where not dominant were "football" and "GAA" where the top words were "win", "lose" and "team" and classified the topics under a more generic topic name of "sports". Another example is the topic of "stocks" and "business" which often had too many similar terms, and as a result the topic of "stocks" always became labelled as "business", although as described earlier, the relabelling of stocks was rectified using a "topic fight" algorithm that forces the duplicate topics to fight for the topic through a probabilistic word distance fight. The loser should then be able to find their appropriate topic.

Where generic topics were removed and only twelve topics were left from the original nineteen, it was found that the LDA topic model managed to match all twelve topics to their most commonly used terms and the topic labelling algorithm was capable of labelling eleven out of twelve topics correctly, revealing an accuracy of 92% when it came to labelling non-generic topics.

5.4.2.2 Sentiment Evaluation

Note that the measuring of sentiment is not a major area of complexity within this dissertation and as a result the evaluation of sentiment has been simplified.

The below survey was provided to twenty users and the sentences were picked from the positive / negative sentences dataset at random. The survey found that 76 out of 100 questions were answered correctly. The positive results model show case an equal accuracy as the twenty random participants at 76.1% and the negative score accuracy is slightly lower at a total accuracy of 68.6%.

Choose the correct sentiment for each question

1. If you sometimes like to go to the movies to have fun , wasabi is a good place to start.

○ Positive

○ Negative

2. It's so juvenile , only programmers could possibly find it funny .

○ Positive

○ Negative

3. I approached the usher and said that if she had to sit through it again , she should ask for a raise .

○ Positive

○ Negative

4. It will delight newcomers to the story and those who know it from bygone days.

○ Positive

○ Negative

5. It uses an old-time formula , it's not terribly original and it's rather messy -- but you just have to love the big , dumb , happy movie.

○ Positive

○ Negative

Figure 68 Sentiment User Survey

5.4.3 Effect of User Bias when understanding Topics and Sentiment

When conducting any type of research that is subjective, bias always play a role and has the possibility to impact results if not accounted for correctly. In this instance user bias may have skewed the results due to user bias. As an example, within the third topic of medicine, two individuals guessed coronavirus even though there was no mention of the illness, but due to the epidemic that is taking place the guess was altered. This inconsistency could have been avoided through conducting user surveys to first determine any bias that may be present and only after the screening allow certain participants to complete the survey if the chance for bias was deemed low.

5.4.4 Evaluating Computer Generated Scores

5.4.4.1 LDA Perplexity and Coherence Scores

As introduced previously, the intent within this evaluation is to have the LDA topic model and opinion mining techniques such as the sentiment model reflect on the overall accuracy of the system.

A method of evaluating the LDA and sentiment topic models is to use probabilistic algorithms that are often used to define the accuracy of machine learning models. For LDA two such scores are Perplexity and Coherence scores. Perplexity score is a statistical measure of how well the model predicts a topics relation to the words through comparing word distribution and topic mixtures where the lower the score the better. The coherence score is used for assessing the quality of the learned topics through checking the number of words that appear in multiple topics where the higher the score, the higher the quality of the topic.

Through various improvements and modifications within the creation of the LDA model, the quality of topics was improved over time. Initially with the LDA prototype and a more basic data cleaning algorithm, the coherence score was 0.32 and perplexity was -23.56. After the data cleaning algorithm was revised and improved the coherence score rose to 0.41 and the perplexity rose to -22.84.

Once the training data was changed to real newspaper data, and the number of topics was adjusted to nineteen a higher accuracy was again reached as the coherence score rose to 0.45 and the perplexity lowered to -21.3. The LDA mallet wrapper was then used over the Gensim wrapper and this again improved the coherence score to 0.55 and lowered the perplexity to -15.67.

5.4.4.2 Sentiment accuracy - Testing Against Known Results

The library used to extract sentiment was TextBlob and not an area of major complexity within this project. The main factor which affected the sentiments accuracy within this context was the quality of the data cleaning algorithm which has been discussed in detail within chapter three and four. In order to assess the accuracy of the sentiment analysis performed, sentences were tested against a list of positive sentences found in the file "positive.txt" and a list of negative sentences in the list "negative.txt". This data set includes a lot of confusing and ambiguous sentences that are difficult even for a human to understand providing a good baseline to test against.

The results showed that the positive results were 76.1% accurate and the negative were 68.6% accurate. Each list included 5225 separate sentences and due to their high amount of difficulty the sentiment evaluation seems to be very capable to manage analysing the sentiment within newspaper articles, where more clear language is used.

5.5. Mapping Topic Words to Original Sentences by Topic

The first opinion mining technique that must be evaluated is how each word from the topic model is mapped to its original sentence. This is a crucial step as each sentence must be measured for stance to calculate the sentiment, rather than individual words. There are a multitude of various techniques that can be applied at this stage including rule-based systems and sentiment models.

An example of mapping a word to its original sentence is as follows. Before the LDA topic model creates the topics from the articles, each word within the article's points to its original sentence and the topic that it was discussing. The LDA topic model then creates the topics and each word from the topics then reconstructs its original sentiments in order to apply techniques of opinion mining on the extracted sentences.

The approach applied within this project took advantage of the guided nature of the set number of topics that were sourced. Each word within the topic model was tagged within a multi-level dictionary that mapped each word to every sentence it was used in as well as the name of the topic where the sentence was used.

When it came to retrieving the sentences for each word the topic labelling algorithm mapped the topic name to the imaginary topic using a similar distance algorithm. Each topic word then retrieved every sentence that was tagged with that particular topic. The advantage of this approach is that the accuracy of word to sentence mapping to topic is 100% assuming the topic was labelled correctly.

The disadvantage of this method is a high space complexity where a large number of sentences are stored for each word that occurred in thousands of articles. Where space was an issue, an optimisation is to make each sentence immutable after creation to have only one occurrence of each unique sentence and map words to the same unique sentences. This optimisation is similar to how strings are immutable in python and any new declaration of the same string points to the same object within memory unless detached manually. This optimisation makes the space complexity S^T in the worst case where S is the number of unique sentences and T is the number of topics that separate each sentence.

The diagram below shows the architecture for mapping each sentence to its original topic Name.
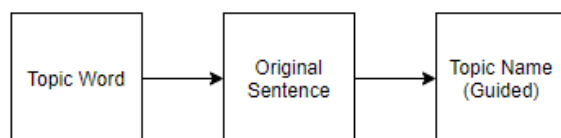
Figure 69 Mapping Words to Original Sentences by Topic

In summary, at the cost of a higher space complexity, an accuracy of 100% was achieved in mapping topic words to their original sentences. Knowing which words related to which sentences was not achieved, but this is arbitrary as long as all the words match the correct sentences based on topics.

5.6. Conclusions

This chapter examined both the methods of testing and evaluations of the presented system. The testing section discussed the testing methodologies that were integrated such as unit testing and Grey-Box testing. The evaluation section delved into how to critically ensure the LDA topic model's capability to produce topics and the sentiment model's ability to understand human sentiment through three alternative methods. The first evaluation is completed through comparing the model's capability to that of random individuals through random participant surveys. The second computes scores to critically determine that the algorithms model is accurate using probabilistic algorithms. The final method of evaluation uses new data that already has already had sentiment scores and topics generated by an individual and this data is compared to their results.

6. Summary, Conclusions and Future Work

6.1. Introduction

This chapter is an overview of the work discussed in this project, the conclusions and key findings that can be drawn from this project and finally the implications for future research. Within the summery of work completed a report on how time is spent on the project is provided through the use of two GANTT charts and a Kanban board that were used to organise work for this project. The workflow diagram is also provided to show how time was spent within each column.

As discussed in detail within the previous chapters, the current system has been shown to implement stance detection with a strong degree of accuracy. This has been performed through examining the components of stance detection individually and allowing this evaluation to reflect back up on the overall capability of the stance detection algorithm created.

6.2. Summary of Work Completed

Two GANTT charts have been added to the appendix section (Appendix B) which showcase the difference between the planned and implemented approach.

The challenges that were encountered and resolved in this project are as follows:

- Sourcing real news articles using API, and then upgraded to web scraper. [DONE]
- A method of parsing the json data provided by the web API's. [DONE]
- Data Cleaning using Natural Language Processing Techniques [DONE]
- Basic LDA topic model [DONE]
- A novel approach to labelling topics. [DONE]
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model [DONE]
- Stance detection techniques to unite the individual components into the main system [DONE]
- Basic Sentiment model (Implemented using third-party libraries) [DONE]
- Methods to increase the accuracy of the LDA model. [DONE]
- Methods to increase the accuracy of the sentiment model [DONE]
- An increased level of knowledge in order to implement Grey-Box testing. [DONE]
- A method of mapping each sentiment analysis result to where the topic is flagged and ultimately create the stance scores. [DONE]
- A user interface that can display topics as well as their sentiment score and link them to specific newspaper companies. [DONE]
- Adding random participants to the evaluation and testing phases. [DONE]
- Improving current work complete in iterations. [DONE]

Some of the challenges encountered are as follows:

- The researching of various LDA and sentiment model implementations to gather a better understanding for the trade-offs when implementing such models.
- Implementing novel automatic probabilistic methods in order to map word-frequency to topics.
- Acquiring an API key from a news source API source such as: https://newsapi.org/s/ireland-news-api and writing code to retrieve articles daily. (Limit of 500 a day using above source).
- JSON parser and regex expressions for finding interesting metadata.
- Using different LDA model libraries, increasing data size and distribution of topics, more fine-tuned data processing and more methods to specifically increase model accuracy.
- Research Grey-Boxing projects and Udemy courses on this topic.
- Research how sentiment analysis works and implement a suitable model.
- Both models will be computed separately and some pre-processing will be required in order to link them, possibly using the hash maps data structure.
- A user interface can be displayed with a suite of topic modelling libraries such as "matplotlib" and "pyLDAvis" exist for visualisation. The key interface will be accessed through command line which will create various html files for visualising different graphs and components. A textual result in a CSV will also be computed of all newspaper included and their bias score as well as the overall bias score by media.
- Random participants will be included in the development and evaluation process through requesting individuals to perform in a study where they will answer questions within surveys that will dictate how accurate the sentiment and topic models are in assessing the results.

6.2.1 Kanban Board - Final Year Project

The image below is the Kanban report structure that was used within the development of this project. Planning was performed through picking a number of tasks in advance for each week and dedicating a set amount of Pomodoro's daily. A pomodoro is a half an hour block time allocation of work, where the worker is not allowed to perform any other action other the task they are working on.
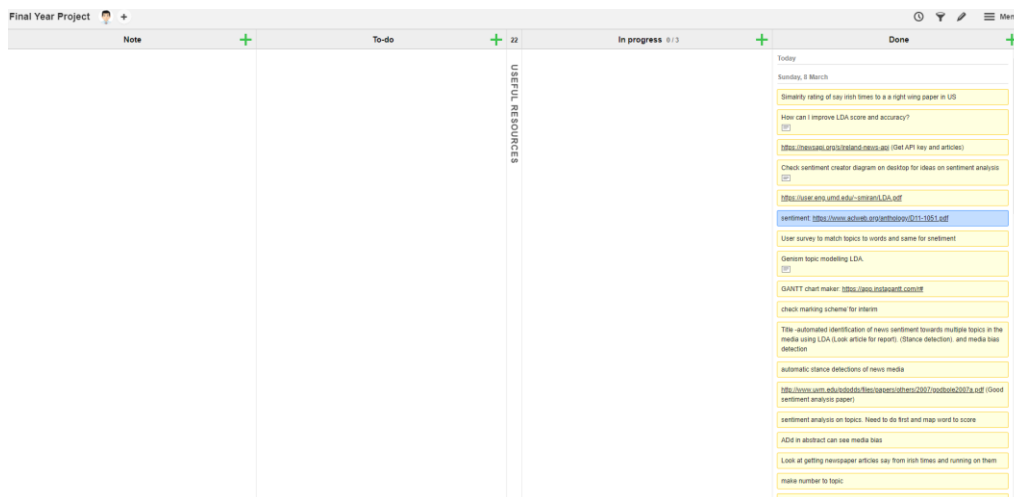
Figure 70 Dissertation Kanban board

6.2.2 Kanban Board – Cumulative Work Flow Diagram

The Kanban board was used to monitor progress and ensure that work was consistently performed. The below diagram shows how tasks were completed over the final twenty weeks of the project. The bottom color signifies tasks completed, and the reason that tasks are not equally distributed is that new tasks were regularly added, that were quickly completed and move the done column.
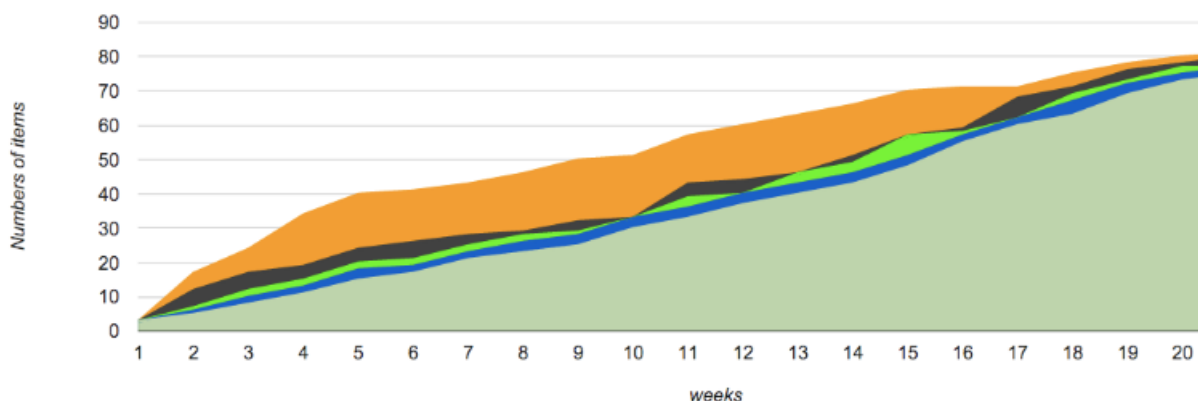
Figure 71 Cumulative Work Flow Diagram

6.3. Conclusions

The objective of this thesis was the successful creation of an automatic stance detection system that can measure the stance or favourability of news outlets in regards to a specific set of topics. This systems core functionality, not taking it account sourcing and displaying data, comprised of the following components:

- The LDA Topic Model
- A Novel Approach to Labelling Topics
- Stance Detection Techniques to link the sentences that discuss similar topics
- Stance Detection Techniques to map the Opining Mining Techniques to the Topic Model
- Opinion Mining Techniques such as Sentiment Model

The graph below represents the implementation and function of the automatic stance detection system. Results found that the system achieved its objective in measuring the objectivity and subjectivity of the stance targeted towards the created topics and ultimately quantify bias. To make the graphs easier to read, the negative results were moved to zero, where "drugs" has a negative score of -0.477 and the topic of "business" has a score of -0.2. The highly negative score for the topic of "drugs" seems to indicate that the stance detection system is capable of monitoring which phrases relate towards which topics and them calculate their stance. When looking at the sentences that discussed drugs words such as "kill, charge, police, victim, sentence" were used which reveals the origin highly negative score. It seems that news media rarely takes a strong negative view that is consistent within text and very few topics receive negative scores within the news organisations of *the daily mail* and *the independent*.

Figure 72 Daily Mail Mode Sentiment

To evaluate the strength of the LDA topic model and topic labeller a survey was created within *section 5.3.4* to test how well users could guess the names of topics, given the top ten weighted words from each topic created by the LDA topic model. Results found that from a sample of twenty participants, 61% guessed the topic correctly, with 39% guessing a topic that was closely related. The LDA topic labeller was also executed on the given topics and all of the topics were correctly labelled. When the set of topics was increased from five to fourteen, it was found that the LDA topic model

correctly identified the words that constituted all fourteen topics and the labeller labelled 11/14 topics, indicating an accuracy of 79%.

To evaluate the sentiment model, its accuracy was compared to that of random participants. Two lists of complex positive and complex negative sentences were used. The sentiment model was run against the complex sentences list and it was found that the positive sentences were 76.1% accurate and the negative sentences 68.6% accurate. In comparison, the random participants received an overall sentiment score of 76% accuracy using the provided sentences within *section 5.3.4.2.* Testing for user bias in judged sentiments was discussed and noted, but no definitive measures were performed in order to ensure that individual bias within the random participants was eliminated.

While the core value of this project is the ability to measure bias within media outlets, the existing automatic stance detection system can also be applied to other areas, particularly qualitative analysis. As an example, this system could be used to measure the stance towards various topics within structured interviews. Within corporations or hospitals through monitoring the speech of staff members, underlying issues could be uncovered where a strongly negative stance is uncovered. The LDA topic model would generate relative topics that are discussed in each use case. For nurses a lack of equipment could be indicated with a negative score towards the topic of equipment. For employees in a software company, a negative sentiment could be relayed towards the topic of "management". The application of the stance detection system to qualitative analysis seems to have a multitude of applications.

Multiple challenges were faced within the creation of this project. The limitation of this automatic stance detection system comprises of requiring "Dominant Topics". A dominant topic is a topic that has specific words that relate to it. An example of such a topic is the topic of "Trump". Words and phrases such as "Trump", "US president", "impeachment" and "campaign" all have strong correlations that describe this topic and are not commonly shared among many topics.

Examples of weak topics that share more generic words that comprise multiple topics are "Football" and "GAA" where words such as "win", "lose" and "team" are shared among both topics and as a result the two topics become classified under a more generic topic such as "sports". This pattern applied itself to other sports such as basketball and rugby.

The implications of this limitation alter the usability of this project. The requirement for dominant topics means that this stance detection system works best when only a few separate topics are discussed that do not heavily relate to each other. In the worst-case scenario this system would monitor a text that discusses six or seven topics equally and as a result the quality of the created topics would be lower.

If this system was applied within qualitative analysis research the implication of this limitation would be that it would provide less value at the start of the research phase as the text would be more general and discuss more topics, and more value as the research reaches a later stage, where the text is more specific.

The results presented above indicates that the stance detection system is capable of measuring bias within media outlets through the calculation of stance scores, given a clear topic.

6.4. Future Work

A major area of growth within the field of computer science and business is using sentiment analysis to correlate the effects that this has on stock trends. Many consumers rely on influential institutions and celebrities for information about the broad market and depending on the sentiment towards this market and more specifically certain stocks, they can have a strong effect on the growth and decrease of stocks as consumer confidence grows or decreases. The value of such a system could be extremely useful if correlation could accurately be plotted and used to predict the value of stocks within the future

Fake News Detection is a growing area that is strongly correlated with bias detection in news articles. Through identifying the stances that newspaper companies present on a given day, a probability score can be calculated to determine if another piece of external news is fake or not. This score will determine how different the bias scores are too news for that day, while also taking into account what topics are discussed. Many fake news outlets will centralise rumours around specific celebrities or events and if these new pieces of information are not discussed in reputable news companies it may be likely that it is in fact fake news. This can be iterated on as a product as a plugin could in real time analyse an article that is in the browser and compare the results to news similar news during that day.

An area with particular interest that this project brushes off of includes pattern matching correlative trends across strongly related topics. An example of such a correlative trend is where the stance for one topic such as climate change improves, stance towards the beef industry deteriorates. This project has the potential to expose such tightly linked correlations and assist in a better understanding of how public stance affects the real world. Mapping media stance onto google trends graphs is another interesting data point that could be captured. This could expose how interest in a topic correlates with its stance.

Ethnography is an account of social life and culture in a particular social system based on the actions of people in a particular culture observed. This thesis can be applied in the area as it a comparative scale across different subjective views and can be analysed objectively across different cultures. Examples could be the level of stance represented towards religions, government, social movements and any other topic that is discussed in detail. From interviews that can be transcript, to novels and the news media a wide range of sources can be used to obtain this data.

Applying stance analysis to psychoanalysis bring forth a number of ethical barriers that prevent the recording and record keeping of psychiatric sessions in order to maintain a strong level of privacy between doctor and patient. The application seems to specifically extend towards researching on a mass scale the differences in cultural stance identified by different groups and better understanding how these belief systems effect various techniques within their application.

Bibliography

1. Pang B, Lee L. Opinion Mining and Sentiment Analysis. INR. 2008 Jul 7;2(1–2):1–135.
2. Mäntylä MV, Graziotin D, Kuutila M. The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. Computer Science Review. 2018 Feb;27:16–32.
3. Wilson T, Wiebe J, Hoffmann P. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing [Internet]. Vancouver, British Columbia, Canada: Association for Computational Linguistics; 2005 [cited 2020 Mar 24]. p. 347–354. Available from: https://www.aclweb.org/anthology/H05-1044
4. Asghar MZ, Khan A, Ahmad S, Qasim M, Khan IA. Lexicon-enhanced sentiment analysis framework using rule-based classification scheme. PLoS One [Internet]. 2017 Feb 23 [cited 2020 Mar 24];12(2). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5322980/
5. Severyn A, Moschitti A. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval [Internet]. Santiago, Chile: Association for Computing Machinery; 2015 [cited 2020 Mar 24]. p. 959–962. (SIGIR '15). Available from: https://doi.org/10.1145/2766462.2767830
6. Blei D. Probabilistic topic models. Communications of the ACM [Internet]. 2019 [cited 4 December 2019];(55):77–84. Available from: https://dl.acm.org/citation.cfm?id=2133826
7. Jacobi C, Atteveldt W van, Welbers K. Quantitative analysis of large amounts of journalistic texts using topic modelling. Digital Journalism. 2016 Jan 2;4(1):89–106.
8. C hang, Jonathan, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David M. Blei. 2009. "Reading Tea Leaves: How Humans Interpret Topic Models." InAdvances in Neural Information Processing Systems, 288–296.
9. Binkley D, Heinz D, Lawrie D, Overfelt J. Understanding LDA in source code analysis. In: Proceedings of the 22nd International Conference on Program Comprehension [Internet]. Hyderabad, India: Association for Computing Machinery; 2014 [cited 2020 Mar 24]. p. 26–36. (ICPC 2014). Available from: https://doi.org/10.1145/2597008.2597150
10. Blei D. Probabilistic topic models. Communications of the ACM (55):83–84. Available from: https://dl.acm.org/citation.cfm?id=2133826
11. Lau JH, Grieser K, Newman D, Baldwin T. Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. Portland, Oregon: Association for Computational Linguistics; 2011. p. 1536–1545. (HLT '11).
12. 2012/2013 JNRS - Readership [Internet]. News Brands Ireland. 2013 [cited 2019 Dec 1]. Available from: https://newsbrandsireland.ie/jnrs-20122013/
13. Hamborg F, Donnay K, Gipp B. Automated identification of media bias in news articles: an interdisciplinary literature review. Int J Digit Libr. 2019 Dec 1;20(4):391–415.

14. Watch: Facebook, Twitter & Google to testify in Senate Russia hearings [Internet]. [cited 2020 Mar 25]. Available from: https://www.youtube.com/watch?v=mDfAFzh6doM&feature=youtu.be

15. M. Pearse E. MEDIA EFFECTS AND SOCIETY. 2nd ed. Mahwah, New Jersey: LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS; 2001.

16. Global advertising spending 2019. Statista. [cited 2020 Mar 25]. Available from: https://www.statista.com/statistics/236943/global-advertising-spending/

17. Lin C, He Y. Joint sentiment/topic model for sentiment analysis. In: Proceedings of the 18th ACM conference on Information and knowledge management. Hong Kong, China: Association for Computing Machinery; 2009 [cited 2020 Mar 24]. p. 375–384. (CIKM '09). Available from: https://doi.org/10.1145/1645953.1646003

18. T.P. A Sentiment Analysis Approach to Predicting Stock Returns [Internet]. Medium. 2018 [cited 2019 Dec 4]. Available from: https://medium.com/@tomyuz/a-sentiment-analysis-approach-to-predicting-stock-returns-d5ca8b75a42

19. Sentiment Analysis [Internet]. MonkeyLearn. 2018 [cited 2020 Mar 8]. Available from: https://monkeylearn.com/sentiment-analysis

20. Natural Language Processing for Beginners: Using TextBlob [Internet]. Analytics Vidhya. 2018 [cited 2019 Dec 5]. Available from: https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/

21. Chen J, Yamada Y, Ryoke M, Tang X. Knowledge and systems sciences. 1st ed. Tokyo: Springer Singapore; 2018.

22. Liu Q, Chen Q, Shen J, Wu H, Sun Y, Ming W-K. Data Analysis and Visualization of Newspaper Articles on Thirdhand Smoke: A Topic Modeling Approach. JMIR Med Inform [Internet]. 2019 Jan 29 [cited 2019 Dec 6];7(1). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371067/

23. Punch KF. Introduction to Social Research: Quantitative and Qualitative Approaches. SAGE; 2005. 342 p.4

24. aneeshabakharia. Algorithms for the thematic analysis of twitter datasets [Internet]. Education presented at; 06:19:41 UTC [cited 2019 Dec 5]. Available from: https://www.slideshare.net/aneeshabakharia/algorithms-for-the-thematic-analysis-of-twitter-datasets2

25. Antezana L, Lagos C, Cabalin C. La opacidad de la política en la prensa chilena: un análisis de suplementos semanales. Estudios sobre el Mensaje Periodístico. 2017 Nov 20;23(2):727–46.

26. Unit Testing [Internet]. Software Testing Fundamentals. 2011 [cited 2019 Dec 7]. Available from: http://softwaretestingfundamentals.com/unit-testing/

27. Liu Q, Chen Q, Shen J, Wu H, Sun Y, Ming W-K. Data Analysis and Visualization of Newspaper Articles on Thirdhand Smoke: A Topic Modeling Approach. JMIR Med Inform [Internet]. 2019 Jan 29 [cited 2019 Dec 1];7(1). Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6371067/

28. Mortensen D. How to Do a Thematic Analysis of User Interviews [Internet]. The Interaction Design Foundation. [cited 2019 Dec 5]. Available from: https://www.interaction-design.org/literature/article/how-to-do-a-thematic-analysis-of-user-interviews

29. Experience WL in R-BU. How to Analyze Qualitative Data from UX Research: Thematic Analysis [Internet]. Nielsen Norman Group. [cited 2019 Dec 5]. Available from: https://www.nngroup.com/articles/thematic-analysis/

30. Ganegedara T. Intuitive Guide to Latent Dirichlet allocation [Internet]. Medium. 2019 [cited 2019 Dec 8]. Available from: https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158

31. Godbole N, Srinivasaiah M, Skiena S. Large-Scale Sentiment Analysis for News and Blogs. In 2007.

32. Nasukawa T, Yi J. Capturing Favourability Using Natural Language Processing. 2003. p. 76

33. Kwiatkowski S. Machine Learning as a Service: Part 1 [Internet]. Medium. 2018 [cited 2020 Mar 9]. Available from: https://towardsdatascience.com/machine-learning-as-a-service-487e930265b2

34. Gonçalves P, Araújo M, Benevenuto F, Cha M. Comparing and combining sentiment analysis methods. Proceedings of the first ACM conference on Online social networks - COSN '13. 2013;.

35. Chuang J, Wilkerson J, Weiss R, Tingley D, Stewart B, Roberts M, et al. Computer-Assisted Content Analysis: Topic Models for Exploring Multiple Subjective Interpretations. 2014.

36. David Hall, Daniel Jurafsky, and Christopher D Manning. Studying the history of ideas using topic models. In EMNLP, pages 363–371, 2008.

37. Recasens M, Danescu-Niculescu-Mizil C, Jurafsky D. Linguistic Models for Analyzing and Detecting Biased Language. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) [Internet]. Sofia, Bulgaria: Association for Computational Linguistics; 2013 [cited 2019 Dec 6]. p. 1650–1659. Available from: https://www.aclweb.org/anthology/P13-1162

38. McGuire WJ. The vicissitudes of attitudes and similar representational constructs in twentieth century psychology. European Journal of Social Psychology. 1986;16(2):89–130.

39. Gruenewald J, Pizarro J, Chermak SM. Race, gender, and the newsworthiness of homicide incidents. Journal of Criminal Justice. 2009 May 1;37(3):262–72.

40. Oliver PE, Maney GM. Political Processes and Local Newspaper Coverage of Protest Events: From Selection Bias to Triadic Interactions. American Journal of Sociology. 2000 Sep 1;106(2):463–505.

41. Patankar A, Bose J, Khanna H. A Bias Aware News Recommendation System. In: 2019 IEEE 13th International Conference on Semantic Computing (ICSC) [Internet]. Newport Beach, CA, USA: IEEE; 2019 [cited 2019 Dec 7]. p. 232–8. Available from: https://arxiv.org/ftp/arxiv/papers/1803/1803.03428.pdf

42. Elyashar A, Bendahan J, Puzis R. Is the News Deceptive? Fake News Detection Using Topic Authenticity. In 2017.

43. Loper E, Bird S. NLTK: The Natural Language Toolkit. CoRR. 2002 Jul 7;cs.CL/0205028.

44. Dutt R, Hiware K, Ghosh A, Bhaskaran R. SAVITR: A System for Real-time Location Extraction from Microblogs during Emergencies. arXiv:180107757 [cs] [Internet]. 2018 Nov 19 [cited 2019 Dec 5]; Available from: http://arxiv.org/abs/1801.07757

45. Natural Language Processing for Beginners: Using TextBlob [Internet]. Analytics Vidhya. 2018 [cited 2019 Dec 5]. Available from: https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/

46. scikit-learn: machine learning in Python — scikit-learn 0.22 documentation [Internet]. [cited 2019 Dec 5]. Available from: https://scikit-learn.org/stable/

47. Labs DD. Modern Methods for Sentiment Analysis [Internet]. Medium. 2017 [cited 2019 Dec 5]. Available from: https://medium.com/district-data-labs/modern-methods-for-sentiment-analysis-694eaf725244

48. WordNet | A Lexical Database for English [Internet]. [cited 2019 Dec 5]. Available from: https://wordnet.princeton.edu/

49. Documentation [Internet]. News API. [cited 2019 Dec 5]. Available from: https://newsapi.org

50. Lau JH, Grieser K, Newman D, Baldwin T. Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. Portland, Oregon: Association for Computational Linguistics; 2011. p. 1536–1545. (HLT '11).

51. Mei Q, Ling X, Wondra M, Su H, Zhai C. Topic sentiment mixture: modeling facets and opinions in weblogs. In: Proceedings of the 16th international conference on World Wide Web - WWW '07 [Internet]. Banff, Alberta, Canada: ACM Press; 2007 [cited 2020 Mar 9]. p. 171. Available from: http://portal.acm.org/citation.cfm?doid=1242572.1242596

52. Cano Basave AE, He Y, Xu R. Automatic Labelling of Topic Models Learned from Twitter by Summarisation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) [Internet]. Baltimore, Maryland: Association for Computational Linguistics; 2014 [cited 2020 Mar 9]. p. 618–624. Available from: https://www.aclweb.org/anthology/P14-2101

53. Kou W, Li F, Baldwin T. Automatic Labelling of Topic Models Using Word Vectors and Letter Trigram Vectors. In: Zuccon G, Geva S, Joho H, Scholer F, Sun A, Zhang P, editors. Information Retrieval Technology. Cham: Springer International Publishing; 2015. p. 253–64. (Lecture Notes in Computer Science).

54. Bhatia S, Lau1 J, Baldwin2 T. Automatic Labelling of Topics with Neural Embeddings [Internet]. Arxiv.org. 2016 [cited 9 March 2020]. Available from: https://arxiv.org/pdf/1612.05340.pdf

55. Janson M, Smith L. Prototyping for Systems Development: A Critical Appraisal. MIS Quarterly. 1985;9(4):305.

56. admin. What is spiral model?7 Advantages and Disadvantages of model [Internet]. Am7s. 2019 [cited 2019 Dec 2]. Available from: https://www.am7s.com/spiral-model/

57. https://resources.sei.cmu.edu/asset_files/SpecialReport/2000_003_001_13655.pdf

58. Ambler S. Feature Driven Development (FDD) and Agile Modeling [Internet]. Agilemodeling.com. 2019 [cited 2 December 2019]. Available from: http://agilemodeling.com/essays/fdd.htm

59. Andrew Powell-Morse. Extreme Programming: What Is It And How Do You Use It? [Internet]. Airbrake Blog. 2017 [cited 2019 Dec 2]. Available from: https://airbrake.io/blog/sdlc/extreme-programming

60. What is a Kanban Board? Why and When to Use Kanban? [Internet]. Productivity Land. 2019 [cited 2019 Dec 5]. Available from: https://productivityland.com/what-is-kanban-board/.

61. Wirth R. CRISP-DM: Towards a standard process model for data mining. In: Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining. 2000. p. 29–39.

62. Visualize Time-Series Data with Open Source Grafana and InfluxDB [Internet]. The New Stack. 2017 [cited 2020 Mar 17]. Available from: https://thenewstack.io/visualize-time-series-data-open-source-grafana-influxdata/

63. Lau JH, Grieser K, Newman D, Baldwin T. Automatic labelling of topic models. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1. Portland, Oregon: Association for Computational Linguistics; 2011. p. 1536–1545. (HLT '11).

64. Wahyudin I, Tosida ET, Andria F. Data Science dan Big Data.

65. 1

66. Bhatia S, Lau1 J, Baldwin2 T. Automatic Labelling of Topics with Neural Embeddings [Internet]. Arxiv.org. 2016 [cited 9 March 2020]. Available from: https://arxiv.org/pdf/1612.05340.pdf

67. Augenstein I, Rocktaschel T, Vlachos A, Bontcheva K. Stance Detection with Bidirectional Conditional Encoding [Internet]. 2nd ed. London: arXiv; 2016 [cited 24 March 2020]. Available from: https://arxiv.org/pdf/1606.05464.pdf

68. De Nicola G, di Tommaso P, Rosaria E, Francesco F, Pietro M, Antonio O. A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems. In: Dal Cin M, Kaâniche M, Pataricza A, editors. Dependable Computing - EDCC 5. Berlin, Heidelberg: Springer; 2005. p. 305–17. (Lecture Notes in Computer Science).

69. De Nicola G, di Tommaso P, Rosaria E, Francesco F, Pietro M, Antonio O. A Grey-Box Approach to the Functional Testing of Complex Automatic Train Protection Systems. In: Dal Cin M, Kaâniche M, Pataricza A, editors. Dependable Computing - EDCC 5. Berlin, Heidelberg: Springer; 2005. p. 305–17. (Lecture Notes in Computer Science).

70. Khan ME, Khan F. A Comparative Study of White Box, Black Box and Grey Box Testing Techniques. In 2012.

71. Buchenau M, Suri JF. Experience prototyping. InProceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques 2000 Aug 1 (pp. 424-433). ACM.

Appendix

Appendix A

Screape_articles.py:

The multi-threaded web scraper provides the functionality to scrape URL's and links from news outlets using REST API's. All code follows a separation of concerns in order to ensure that it is requires no extra work to scrape new news outlets once the URL is provided and the REST standards upheld by the news company are known. This includes how many pages per result are provided and in what tags encompass the title and paragraphs of the article. Below only a few select methods are shown.

```python
def buildArticles(self, company_tag):

    for topic in self.refined_news_categories:
        if '.' in topic:
            topic = ' '.join(topic.split('.'))

        self.pull_articles(topic, company_tag)
        self.q = Queue()
        self.run = True
# Using threads pull articles
def pull_articles(self, topic, tag=None):
    NUM_THREADS = 2
    self.call_tag = tag

    # Get all links in an object
    if self.call_tag == self.independent_tag:
        links = self.getCorpusLinks(self.independent_tag, topic)
    elif self.call_tag == self.daily_mail_tag:
        links = self.getCorpusLinks(self.daily_mail_tag, topic)
    elif self.call_tag == self.irish_times_tag:
        links = self.getCorpusLinks(self.irish_times_tag, topic)
    elif self.call_tag == self.new_york_times_tag:
        links = self.getCorpusLinks(self.new_york_times_tag, topic)

    for _ in range(NUM_THREADS):
        # Create the new thread using threader method
        t = threading.Thread(target = self.pull_articles_threader)
        t.daemon = True
        # Start the thread
        t.start()

    for link in links:
        self.q.put(link)
```

```python
        self.q.join()
        self.run = False

    def getDailyMailArticle(self, urlTopic):
        url, topic = urlTopic.split('|')
        article = ''
        # Header holds the date of article, url and author
        header = ''

        try:
            response = requests.get(url) # , headers=ua)

            soup = BeautifulSoup(response.text, 'lxml')

            # Get the author
            author = soup.find('p',attrs={'class':'author-section byline-
plain'})

            # Get the date of the article
            date = soup.find('span',attrs='article-timestamp article-
timestamp-published')

            header = '{ ' + author.text + ' ' + date.text + ' ' + url + ' }'
+ '\n'

            # Get the paragraphs of the article
            articleDivs = soup.find('div', attrs={'itemprop':'articleBody'})
            paras = articleDivs.findAll('p', attrs={'class':'mol-para-with-
font'})

            for para in paras:
                article += ''.join(para.findAll(text=True)) + ' '

            if article == '':
                paras = articleDivs.findAll('p')
                for para in paras:
                    article += ''.join(para.findAll(text=True)) + ' '

            with self.lock:
                # Write articles to corpus
                self.write_to_file(header + article, self.daily_mail_tag, top
ic)

        except AttributeError:
            print('Failed to find what we looked for at link, ' + url)
            # When failed to read file, IP blocked,
            # send file to other dir and can read it later
            self.write_dead_file_to_dir(url, self.daily_mail_tag)
```

```
        print('Sleeping to stop IP block...')
        time.sleep(20)

    except UnicodeEncodeError:
        print('UniCodeEncodeError occured at link' + url)

    return url
```

The generate_lda code below is one of the LDA topic models that was implemented.

generate_lda.py:

```
def gen_bunch(news_path):
    # Cateogies in data set
    news_categories = ['alt.atheism', 'comp.graphics', 'comp.os.ms-
windows.misc'
              , 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware', 'comp.
windows.x'
              , 'misc.forsale', 'rec.autos', 'rec.motorcycles', 'rec.sport.
baseball'
              , 'rec.sport.hockey', 'sci.crypt', 'sci.electronics', 'sci.me
d'
              , 'sci.space', 'soc.religion.christian', 'talk.politics.guns'
              , 'talk.politics.mideast', 'talk.politics.misc', 'talk.religi
on.misc']

    # Setup path to test corpus
    NEWS_GROUPS_TEST_PATH = os.path.abspath(os.path.join(os.path.dirname(__fi
le__), news_path))

    # Print the path
    # print(NEWS_GROUPS_TEST_PATH)


    ##### Need to implement a method for including custom categories! ####

    # Load all test data.
```

```
    # news_test = load_files(NEWS_GROUPS_TEST_PATH, description='News Paper Te
st Topics from 20 news groups'
    #                                  , categories=news_categories, load_content=T
rue , shuffle=False, encoding='latin1'
    #                                  , decode_error='strict')


    # Shuffling the data in order to increase distribution of topics and not o
verly simplify NLP patterns
    # news_test.data is everything in one big string
    news_test = load_files(NEWS_GROUPS_TEST_PATH, description='News Paper Test
 Topics from 20 news groups'
                                    , categories=news_categories, load_content=Tru
e , shuffle=True, encoding='latin1'
                                    , decode_error='strict', random_state=30)

    # Note:
    # Shows the topic and document ID + the article.
    # print(news_test.filenames[0])
    # print(news_test.data[0])

    # Get all of the file names
    # for integer_category in news_test.target[:10]:
    #     print(news_test.target_names[integer_category])

    return news_test
```

As discussed within the design phase, a method for cleaning and processing the data from the news articles was then implemented through regex operations, natural language processing techniques such as creating bigrams, removing stop words and lemmatizing words.

```
def multiple_replacements(article):
    empty_str = ""

    # Replacing all dashes, equals, cursors
    replacements = {
        "-" : empty_str,
        "=": empty_str,
        "^": empty_str,
    }

    # Replace newlines
    article_list = re.sub('\s+', ' ', article)

    # Replace emails
    article_list = re.sub('\S*@\S*\s?', '', article_list)

    # Replace quotes
    article_list = re.sub("\'", "", article_list)
```

```python
    # Replace headers of data (author, date and url)
    article_list = re.sub(r'\{[^)]*\}', '', article_list)

    # Create a regular expression using replacements and join them togetehr.
    # re.compile creates the pattern object
    # re.escape avoids using special characters in regex
    reg = re.compile("(%s)" % "|".join(map(re.escape, replacements.keys())))

    # For each match, look-up corresponding value in dictionary
    return reg.sub(lambda value: replacements[value.string[value.start():value.end()]], article)


def split_to_word(articles):
    # Iterate over every article
    for article in articles:
        # Yield to not overload the memory with the big data set.
        # Deacc parameter removes all punctuations as well as spliting each word.
        yield(gensim.utils.simple_preprocess(str(article), deacc=True))

def create_bigrams(articles, bigram_model):
    return [bigram_model[article] for article in articles]

def remove_stopwords(articles):
    return [[w for w in simple_preprocess(str(article)) if w not in stop_words] for article in articles]

def lemmatize_words(bigram_model):
    # Only considers nouns, verbs, adjectives and adverbs
    return [[w for w in lemmatize(str(article))] for article in bigram_model]

# This method is about a minute faster for a data set of 7000 than the one above
def lemmatization(articles, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):

    articles_lem = []

    # Load the spacy lammatixation model for english
    spacy_lem = spacy.load('en_core_web_sm', disable=['parser', 'ner'])

    for article in articles:
        w = spacy_lem(" ".join(article))
        articles_lem.append([token.lemma_ for token in w if token.pos_ in allowed_postags])

    return articles_lem
```

Words are tokenized and a bigram model is created in order to understand which words are commonly used together and link them appropriately. This is implemented before the create bigrams method is called:

```
    # brigrams model
    # Need bigrams as it cuts word that go together down into one
    bigram = gensim.models.Phrases(article_word_list, min_count=8, threshold=
100)

    bigram_model = gensim.models.phrases.Phraser(bigram)
```

Lemmatization is performed on nouns, verbs, adjectives and adverbs as can be seen within the code below:

```
    # Lemmatize - By default only nouns, verbs, adjectives and adverbs
    # lemmatized_article = lemmatize_words(bigram_words)

    lemmatized_article = lemmatization(bigram_words, allowed_postags=['NOUN',
'VERB', 'ADJ',  'ADV'])
```

The corpus and word dictionary to map topics to an ID and ID's to their frequency is then created with the following code:

```
    # Create dictionary. This maps id to the word
    word_dict = corpora.Dictionary(test_data_cleaned)

    # Create corpus. This directly contains ids of the word and the frequency
.
    corpus = [word_dict.doc2bow(data) for data in test_data_cleaned]
```

The LDA model is then constructed using the corpus and word dictionary. Each document is iterated through ten times, the data is shuffled in chunks of 100 documents and fifty topics are created as follows:

```
def build_lda_model(articles, word_dict, corpus):
    # Build LDA model
    # Retry with random_state = 0
    lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                        id2word=word_dict,
                                        num_topics=50,
                                        random_state=100,
                                        update_every=1,
                                        chunksize=100,
                                        passes=10,
```

```
                                            alpha='auto',
                                            per_word_topics=True)
```

The model is then saved through:

```python
def save_model(lda_model):
    # Below doesn't work due to access denied issues, datapath is the alternat
ive

    # MODEL_PATH = "../models/"
    # model_file = os.path.abspath(os.path.join(os.path.dirname(__file__), MO
DEL_PATH))

    model_file = datapath("model")
    lda_model.save(model_file)
```

The method create_word_corpus maps each word to their respective sentences (including bigrams) is included in the generate.py files and is as follows below. For each company it iterates over each word within the articles and maps that word to its respective sentence, the topic name that it belongs to, the URL that word was pulled from (for traceability), the authors name and finally the date the article the word was published in was written.

A separate process was followed to ensure

```python
def create_word_corpus(articles, company_tag):
    word_corpus = {}
    unique_words = []
    print('STARTING TO CREATE WORD CORPUS for company ' + company_tag + "..."
)

    # Default of where articles start after headers
    sentence_start = 4

    if company_tag == "DAILY_MAIL":
        sentence_start = 2
    elif company_tag == "INDEPENDENT":
        sentence_start = 4
    else:
        print("Company type used not supported")
        return

    for article in articles.data:
        # Extract the parts of the article
        article_parts = article.split('\r\n')
```

```python
        topic_name = article_parts[0].strip()
        print(topic_name)

        # DAILY_MAIL articles are formatted differently
        if company_tag == "DAILY_MAIL":
            header_line = article_parts[1].strip()[1:]
            words = header_line.strip().split(" ")
            try:
                author_name = ' '.join(words[1:3])
            except IndexError:
                print('Failed to return a author on article ' + article )
            try:
                # Do regex for date since more consistent
                article_index = header_line.find('Published:')
                article_end = header_line.find('https:')
                article_end -= 3
                article_date = header_line[article_index:article_end]
            except IndexError:
                print('Failed to return a valid date on article ' + article )
        elif company_tag == "INDEPENDENT":
            try:
                author_name = article_parts[1].strip()[1:]
            except IndexError:
                print('Failed to return a author on article ' + article )

            try:
                article_date = article_parts[2].strip()
            except IndexError:
                print('Failed to return a valid date on article ' + article )
        else:
            print('Company ' + company_tag + " not supported...")
            return

        # print('Topic name = ' + topic_name)
        # print('Author name = ' + author_name)
        # print('Article date = ' + article_date)
        # print('Link retrieved article from = ' + article_link)

        # Loop over every paragraph in the article
        for part in article_parts[sentence_start:]:
            # Catches the full sentences (used for sentiment analysis)
            sentences = part.split('.')

            # Loop through each sentence in paragraph
            for sentence in sentences:
                words = sentence.split(' ')
                # Loop through each word in the sentence
                for word in words:
```

```python
                    # Ensure a word and not a number
                    if word.isalpha():
                        if word not in unique_words:
                            unique_words.append(word)
                            # New element so add it to the dictionary only aft
er instantiating
                            word_corpus[word] = []
                            word_corpus[word].append(topic_name + ':::' + auth
or_name + ':::' + article_date + ':::' + sentence)
                        else:
                            word_corpus[word].append(topic_name + ':::' + auth
or_name + ':::' + article_date + ':::' + sentence)

    # for word in word_corpus:
    #     print('WORD: ' + word + ' POINTS TO ', word_corpus[word])

    # Add the bigrams to the word corpus
    print('Starting to add bigrams to the word corpus...')
    bigram_word_corpus = retrieve_bigram_word_corpus(company_tag)
    topic_name_bigram = ''
    author_name_bigram = ''
    article_date_bigram = ''
    sentence_bigram = ''

    # Convert bigram_word_corpus to a list:
    for bigram_word in bigram_word_corpus:
        bigram_word_list = []
        topic1_name = topic2_name = author1_name = author2_name = article1_dat
e = article2_date = article1_sentence = article2_sentence = ''

        # Get words frm bigram
        print(bigram_word)


        if bigram_word.count('_') == 1:
            # Get bigrams
            word1, word2 = bigram_word.split('_')
        elif bigram_word.count('_') == 2:
            # Get trigrams, but only include first and second word for now
            word1, word2, _ = bigram_word.split('_')
        elif bigram_word.count('_') == 3:
            # Get quadrams
            word1, word2, _, _ = bigram_word.split('_')
            # Get pentams
        elif bigram_word.count('_') == 4:
            word1, word2, _, _, _ = bigram_word.split('_')
        else:
            print('Should not reach here!!!')
```

```python
            continue

        # print('word1 = ', word1)
        # print('word2 = ', word2)

        if word1 in word_corpus.keys():
            if word2 in  word_corpus.keys():
                # In future revisions don't include only last bigram in array,
 take maximum count
                for word1, word2 in zip(word_corpus[word1],  word_corpus[word2
]):
                    # Get word_parts for word1
                    word_parts = word1.split(':::')
                    # Get topic_name from bigram
                    topic1_name = word_parts[0]
                    # Get author_name for bigram
                    author1_name = word_parts[1]
                    # Get article_date for bigram
                    article1_date = word_parts[2]
                    # Get article sentence for bigram
                    article1_sentence = word_parts[3]

                    # Grab the parts of the word2
                    word_parts = word2.split(':::')
                    # Get topic_name from bigram
                    topic2_name = word_parts[0]
                    # Get author_name for bigram
                    author2_name = word_parts[1]
                    # Get article_date for bigram
                    article2_date = word_parts[2]
                    # Get article sentence for bigram
                    article2_sentence = word_parts[3]

                    # For now have both if and else have same result, but the
else will be more
                    # complicated in future revisions, taking account more dat
a
                    if topic1_name == topic2_name:
                        topic_name_bigram = topic1_name
                    else:
                        topic_name_bigram = topic1_name

                    if author1_name == author2_name:
                        author_name_bigram = author1_name
                    else:
                        author_name_bigram = author1_name
```

```python
                        if article1_date == article2_date:
                            article_date_bigram = article1_date
                        else:
                            article_date_bigram = article1_date

                        if article1_sentence == article2_sentence:
                            sentence_bigram = article1_sentence
                        else:
                            sentence_bigram = article1_sentence

                        # Add the bigram
                        if bigram_word not in bigram_word_list:
                            bigram_word_list.append(bigram_word)
                            word_corpus[bigram_word] = []
                        else:
                            word_corpus[bigram_word].append(topic_name_bigram + ':
::' + author_name_bigram + ':::' + article_date_bigram + ':::' + sentence_bigr
am)


    return word_corpus
```

Visualising_lda.py
The model is read from hard disk using:

```python
# Get the model from genism path
def retrieve_modal():
    # Path to model
    model_file = datapath("model")
    # Load
    lda = gensim.models.ldamodel.LdaModel.load(model_file)

    print('Finished retrieving model...')
    return lda

def retrieve_word_article_list():
    MODEL_PATH = "../newspaper_data/newspaper_list.txt"
    newspaper_list_file = os.path.abspath(os.path.join(os.path.dirname(__file
__), MODEL_PATH))

    MODEL_PATH_WRITE = "../newspaper_data/newspaper_list_writing.txt"
    newspaper_list_file_write = os.path.abspath(os.path.join(os.path.dirname(
__file__), MODEL_PATH_WRITE))

    newspaper_article_list = []
    newspaper_word_list = []
```

```python
    with open(newspaper_list_file, 'r') as filehandle:
        for line in filehandle:
            # String splitting removes first bracket and new line + closing br
acket
            current_line = line[1:-2]

            # Split each word into the list
            current_list = current_line.split(', ')

            for word in current_list:
                # Append the word and remove closing and opening quotation
                newspaper_word_list.append(word[1:-1])

            newspaper_article_list.append(newspaper_word_list)
            newspaper_word_list = []

    print('Finished retrieving article word list...')
    return newspaper_article_list
```

The coherence and perplexity scores are generated through:

```python
def compute_complexity(article_word_list, word_dict, corpus, lda):


    # Coherence Score: 0.4132613386862506
    # Coherence score is the probability that a word has occured in a certain
topic, so the quality of the topic matching.
    coherence_model_lda = CoherenceModel(model=lda, texts=article_word_list, d
ictionary=word_dict, coherence='c_v')
    coherence_lda = coherence_model_lda.get_coherence()
    print('\nCoherence Score:', coherence_lda)

    # Perplexity: -21.294153422496972
    # Calculate and return per-
word likelihood bound, using a chunk of documents as evaluation corpus.
    # Also output the calculated statistics, including the perplexity=2^(-
bound), to log at INFO level.
    print('Perplexity:', lda.log_perplexity(corpus))  # a measure of how good
the model is. lower the better.
```

Within the visualisation.py, an LDA wrapper was also implemented using the Java library that in effect improves upon the results of the previously built LDA model

```python
def build_mallet_lda_model(articles, word_dict, corpus):
    from gensim.models.wrappers import LdaMallet

    MALLET_PATH = '../mallet_2/mallet-2.0.8/bin/mallet.bat'
```

```
    mallet_file = os.path.abspath(os.path.join(os.path.dirname(__file__), MAL
LET_PATH))
    os.environ.update({'MALLET_HOME':mallet_file})

    lda_mallet_model = gensim.models.wrappers.LdaMallet(mallet_file, corpus=c
orpus, num_topics=19, id2word=word_dict)

    print('CREATED MALLET MODEL')
    return lda_mallet_model
```

The "correlate_top_words_sentiment" method is responsible for retrieving the top most weighted words for each topic and calculating a sentiment score for each word occurrence in the articles.

```
# Uses top 30 words per topic to calculate polarity and subjectivity of each t
opic
def correlate_top_words_sentiment(lda_mallet_model, word_corpus, company_tag,
fight=False):
    topic_sentiments = {}
    topic_word_dict = defaultdict(defaultTopicValue)
    topic_name_polarity = []
    topic_name_subjectivity = []
    word_list = []
    polarity = []
    subjectivity = []
    isSciObj = False
    used_topic_words = []
    counter = 1

    # Try for 10 num words as well as 30 and see which is more  accurate
    for index, topic in lda_mallet_model.show_topics(num_topics=19, formatted=
False, num_words=15): #, num_words= 15):
        imaginery_topic = topic

        # getRealTopic. Can remove code involing fightingTopic and will make i
t more accurate as only a few topics will come through
        # But this increases how many results come through. Delete any duplica
te topics that mismatch necessary using fightingTopics
        topic_name, winner, losing_topic = getRealTopic(imaginery_topic, word_
corpus, used_topic_words, topic_word_dict, fight)

        if fight:
            # The duplicate has been fixed by getRealTopic to another topic, r
eplace topic_name to losing_topic and perform operations for it
            if winner == "EXISTING":
                topic_name = losing_topic
            elif winner == "":
```

```python
                # If winner and losing_topic are empty, that means this is a new topic and dont need to replce variables
                    pass
            else:
                # The imaginery topic is a stronger topic than the existing one
                # Firstly move the existing topic which is weaker then current one to new topic and append a number as it may also be beaten by another topic
                topic_sentiments[losing_topic + "L" + str(counter)] = copy.deepcopy(topic_sentiments[topic_name])
                topic_word_dict[losing_topic +  "L" + str(counter)] =  copy.deepcopy(topic_word_dict[topic_name])
                counter = counter + 1

        used_topic_words.append(topic_name)
        print('Topic name is: ' + topic_name)

        for w in topic:
            if w[0] in word_corpus.keys():

                ### Algo to remove generic words goes here

                # Only get the w[0] of the topic that has topic_name in the sentence of that word
                sentences = get_topic_sentences(topic_name, word_corpus[w[0]])
                # sentences = get_topic_sentences2(topic_name, lda_mallet_model)
                # Add the word to used words (these are words we are checking sentiment for)
                word_list.append(w[0])

                sentences_cleaned_list = generate_lda_model.clean_data(sentences, company_tag, isSciObj)

                for sentence in sentences_cleaned_list:
                    sentence_str = ' '.join([str(word) for word in sentence])
                    pol, subjec = getSentiment(sentence_str)
                    # many zeros from falsely understood sentences
                    # print("Adding polarity: " + str(pol))
                    # print("Adding subjectivity: " + str(subjec))

                    # Do not include zeros as they skew data too close to zero
                    # and does not show the weight of bias when used.
                    # This allows to discern opinion more easily
                    if pol != 0:
                        polarity.append(pol)
                        if topic_name in sentence_str:
```

```python
                            topic_name_polarity.append(pol)
                    if  subjec != 0:
                            subjectivity.append(subjec)
                            if topic_name in sentence_str:
                                topic_name_subjectivity.append(subjec)
            else:
                continue

        # Calculate average, mode and median for polarity and sentiment
        if len(polarity) != 0 and len(subjectivity) != 0:
            avg_polarity = sum(polarity) / len(polarity)
            avg_subjectivity = sum(subjectivity) / len(subjectivity)
        if len(topic_name_polarity) != 0 and len(topic_name_subjectivity) != 0\
:
            avg_topic_name_polarity = sum(topic_name_polarity) / len(topic_nam\
e_polarity)
            avg_topic_name_subjectivity = sum(topic_name_subjectivity) / len(t\
opic_name_subjectivity)
        else:
            avg_topic_name_polarity = None
            avg_topic_name_subjectivity = None

        polarity_mode, subjectivity_mode = calculate_sentiment_mode(polarity,
subjectivity, avg_polarity, avg_subjectivity)
        polarity_median, subjectivity_median = calculate_sentiment_median(pola\
rity, subjectivity)
        # Add values to sentiment dictionary, as well as words
        topic_sentiments[topic_name + str(counter)] = [avg_topic_name_polarity\
, avg_polarity, polarity_mode, polarity_median, avg_topic_name_subjectivity, a\
vg_subjectivity, subjectivity_mode, subjectivity_median]
        topic_word_dict[topic_name + str(counter)].append(word_list[:])
        counter = counter + 1

        print('Added topic: ' + topic_name)
        print('For topic {}, probably: {} the average polarity is {} and the a\
verage subjectivity is {}'.format(index, topic_name, avg_polarity, avg_subject\
ivity))

        # Clean up
        word_list[:] = []
        polarity[:] = []
        subjectivity[:] = []
        topic_name_polarity[:] = []
        topic_name_subjectivity[:] = []


    return (topic_sentiments, topic_word_dict)
```

The "getRealTopic" method below uses the sentence word corpus created earlier in order to map the distance between each word used in each article to one of the created topics. This is a probabilistic method that heavily relies on clearly defined topics rather than generic topics that can easily be confused together. As an example, the topic of basketball and football is not dominant enough in that they share too many words (team, win, lose, competition) in this model and would get categorised under a topic of sports.

```python
# Will return the topic that most matches the given 30 words
def getRealTopic(imaginery_topic, word_corpus, used_topic_words, topic_word_dict, fight):
    real_topic = ''
    topic_list = []
    winner = ""
    losing_topic = ""

    for i , w in enumerate(imaginery_topic):
        # Do for only first 10 words, but try with 30 and check if accuracy is changing
        if i == 10:
            break
        print("Before adding real topic: ")
        # For w[0] get all sentences where the word is used (from a premade corpus dictionary that matches a word to each occurence and the topic name in the folder)
        if w[0] in word_corpus.keys():
            for word in word_corpus[w[0]]:
                # Grab the topic where the word is found
                word_parts = word.split(':::')
                topic_list.append(word_parts[0])
            real_topic = most_frequent_topic(topic_list)
            print("Real topic is: " + real_topic)

            if fight:
                # If this topic already, fight them to find stronger one
                if real_topic in used_topic_words:
                    winner, losing_topic = topic_fight(imaginery_topic, real_topic, topic_word_dict, word_corpus)
                else:
                    return real_topic, "", ""

        # If winner == "EXISTING" -> no need to replace topic_word_dict
        # If winner == "IMAGINERY" -> need to replace in topic_word_dict
        return real_topic, winner, losing_topic
```

The method topic_fight is used to compare two topics which are competing for a topic name and share similar attributes. As an example, the topic China before it is provided a name may receive the name business due to the similar words associated with the topic of business (Tariff, global, economy, cost), but when the topic that represents business is assigned a name, the business topic is already being used by China. This method makes both topics

compete to see which topic has more similar words and have the right to the topic name of business, where the loser is re-evaluated again.

```python
def most_frequent_topic(list):
    return max(set(list), key = list.count)


def topic_fight(imaginery_topic, topic, topic_word_dict, word_corpus):
    winner = ""
    losing_topic = ""
    imaginery_topic_list = []
    existing_topic_list = []

    # Get number of topic occurences of imaginery topic
    for i , w in enumerate(imaginery_topic):
        # Do for only first 10 words, but try with 30 and check if accuracy is
 changing
        if i == 10:
            break

        if w[0] in word_corpus.keys():
            for word in word_corpus[w[0]]:
                # Grab the topic where the word is found
                word_parts = word.split(':::')
                imaginery_topic_list.append(word_parts[0])
            imaginery_topic_count = imaginery_topic_list.count(topic)

    # Get number of topic occurences of existing topic
    for i , w in enumerate(topic_word_dict[topic]):
        # Do for only first 10 words, but try with 30 and check if accuracy is
 changing
        if i == 10:
            break

        if w[0] in word_corpus.keys():
            for word in word_corpus[w[0]]:
                # Grab the topic where the word is found
                word_parts = word.split(':::')
                existing_topic_list.append(word_parts[0])
            existing_topic_count = existing_topic_list.count(topic)

    if imaginery_topic_count > existing_topic_count:
        winner = "IMAGINERY"
        existing_topic_list.remove(topic)
        losing_topic = most_frequent_topic(existing_topic_list)
    else:
        winner = "EXISTING"
        imaginery_topic_list.remove(topic)
        losing_topic = most_frequent_topic(imaginery_topic_list)
```

```
    return winner, losing_topic
```

Format.py

To format the sentences the format.py file is used to write the real topics names and the ten most weighted words next to each other as follows:

```python
import os
import sys

def sort_topic_name(topic_name):
    return topic_name[1]

def format_sentiments(company_tag):
    SENTIMENTS_PATH = "../newspaper_data/sentiments/" + company_tag + '.txt'
    FULL_SENTIMENTS_PATH = os.path.abspath(os.path.join(os.path.dirname(__fil
e__), SENTIMENTS_PATH))

    topics = []

    with open(FULL_SENTIMENTS_PATH, 'r') as filehandle:
        for line in filehandle:
            line_parts = line.split(' ')
            topic_name = line_parts[3]
            topic_word_sent = line_parts[8][:6]
            avg_word_sentiment = line_parts[11][:6]
            avg_topic_objectivity = line_parts[25][:6]

            topic_info = ("Topic: " + topic_name + " Topic Word Sentiment: " +
 topic_word_sent + " Average Word Sentiment: " + avg_word_sentiment + " Averag
e Word Objectivity " + avg_topic_objectivity + "\n", topic_name)
            topics.append(topic_info)

    topics.sort(key = sort_topic_name)

    for topic in topics:
        print(topic[0])


def main():

    companies = ["INDEPENDENT", "DAILY_MAIL"]

    if len(sys.argv) < 2:
        print("Must pass a company as argument. Accepted companies: [ " + comp
anies[0] + ", " + companies[1] + " ]")
        return ''
    else:
```

```python
        company = sys.argv[1]

        if company in companies:
            format_sentiments(company)
        else:
            print("Must pass a company as argument. Accepted companies: [ " +
companies[0] + ", " + companies[1] + " ]")



if __name__ == '__main__':
    main()
```

test_visualise_lda.py

```python
import unittest
import sys

class test_visualise_lda(unittest.TestCase):
    def setUp(self):
        sys.path.append('..')
        import visualise_lda as vl
        self.vl = vl
        self.median_pol = [1, 3, 3, 4, 5]
        self.median_obj = [10, 11, 12, 12, 14]


    def test_getSentiment_polarity(self):
        # Test if sujective in range from - 1 to 1 where -
1 is negative and 1 is positive

        # Negative test case
        self.assertLess(self.vl.getSentiment('This referendum is terrible')[0
], 0)
        # Positive test case
        self.assertGreater(self.vl.getSentiment('This referendum is amazing')
[0], 0.5)

    def test_getSentiment_objectivity(self):
        # Test if objecvtive in range of 0 to 1 where 0 is objective and 1 is
opinion

        # Negative test case
        self.assertEqual(self.vl.getSentiment('This referendum is terrible')[
1], 1)
        # Positive test case
        self.assertLess(self.vl.getSentiment('This referendum exists')[1], 0.
5)

    def test_retrieve_word_corpus(self):
```

```python
        # Ensure does not return an empty list
        self.assertNotEqual(self.vl.retrieve_word_corpus, [])

    def test_calculate_sentiment_median(self):
        # Test polarity median
        self.assertEqual(self.vl.calculate_sentiment_median(self.median_pol,
self.median_obj)[0], 3)

        # Test subjectivity median
        self.assertEqual(self.vl.calculate_sentiment_median(self.median_pol,
self.median_obj)[1], 12)

    def test_calculate_sentiment_mode(self):
        avg_pol = sum(self.median_pol) / len(self.median_pol)
        avg_obj = sum(self.median_obj) / len(self.median_obj)

        # Test polarity median
        self.assertEqual(self.vl.calculate_sentiment_mode(self.median_pol, se
lf.median_obj, avg_pol, avg_obj)[0], 3)

        # Test subjectivity median
        self.assertEqual(self.vl.calculate_sentiment_mode(self.median_pol, se
lf.median_obj, avg_pol, avg_obj)[1], 12)

    def test_get_topic_sentences(self):
        # Check filters topic
        self.assertEqual(self.vl.get_topic_sentences('Brexit', ['Brexit:::Aut
her:::Date:::Brexit is something', 'China:::Auther:::Date:::China is something
']), ['Brexit is something'])


if __name__ == '__main__':
    unittest.main()
```

test_scrape_articles.py

```python
import unittest
import sys

class test_scrape_articles(unittest.TestCase):
    def setUp(self):
        sys.path.append('..')
        import scrape_articles as sa
        self.articles = sa.WebScrapeArticles()
        self.reg_pattern = "http"
        self.topic = 'brexit'
        self.page_number = 1
        self.url_independent = 'https://www.independent.ie/world-
news/coronavirus/met-eireann-predicts-sunny-dry-week-but-some-closures-
```

```python
announced-as-outdoor-revellers-ignore-social-distancing-advice-
39065233.html|Brexit'
        self.url_daily_mail = 'https://www.dailymail.co.uk/tvshowbiz/article-
8141933/Sneaky-Pete-star-Giovanni-Ribisi-covers-face-mask-gloves-stocks-
supplies.html?ns_mchannel=rss&ico=taboola_feed|Brexit'


    def test_getDailyMailArticleLinksNotEmpty(self):
        # Check returns an element
        self.assertNotEqual(self.articles.getDailyMailArticleLinks(self.topic
, self.page_number), [])


    def test_getDailyMailArticleLinksCorrectData(self):
        # Check includes correct syntax
        self.assertRegex("". join(self.articles.getDailyMailArticleLinks(self
.topic, self.page_number)), self.reg_pattern)


    def test_getIndependentArticleLinksNotEmpty(self):
        # Check returns an element
        self.assertNotEqual(self.articles.getIndependentArticleLinks(self.top
ic, self.page_number), [])


    def test_getIndependentArticleLinksCorrectData(self):
        # Check includes correct syntax
        self.assertRegex("". join(self.articles.getIndependentArticleLinks(se
lf.topic, self.page_number)), self.reg_pattern)


    def test_getIndependentArticleNotEmpty(self):
        # Check returns a non empty string
        self.assertNotEqual(self.articles.getIndependentArticle(self.url_inde
pendent, False), '')


    def test_getIndependentArticleCorrectData(self):
        # Check includes a sentence from the article
        sentence_pattern = "Spring has sprung, but this year, the hope and ent
husiasm that usually comes with it has been subdued by the coronvirus crisis a
nd the unknown prospect of what that might bring."
        self.assertRegex(self.articles.getIndependentArticle(self.url_indepen
dent, False), sentence_pattern)


    def test_getDailyMailArticleNotEmpty(self):
        # Check returns a non empty string
        self.assertNotEqual(self.articles.getDailyMailArticle(self.url_daily_
mail, False), '')


    def test_getDailyMailArticleCorrectData(self):
        # Check includes a sentence from the article
        sentence_pattern = "Giovanni Ribisi made his health his priority."
```

```
        self.assertRegex(self.articles.getDailyMailArticle(self.url_daily_mai
l, False), sentence_pattern)




if __name__ == '__main__':
    unittest.main()
```
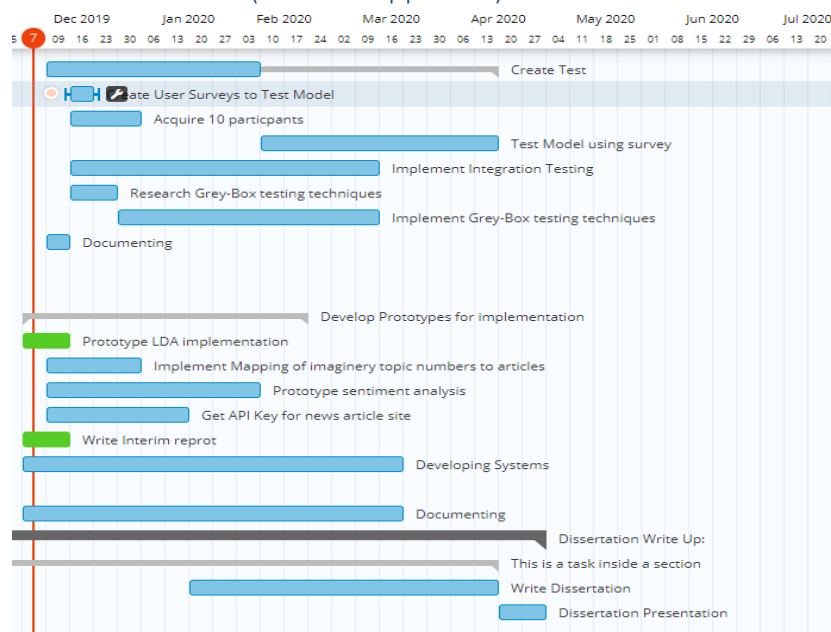
Appendix B

First GANTT Chart (Planned Approach)

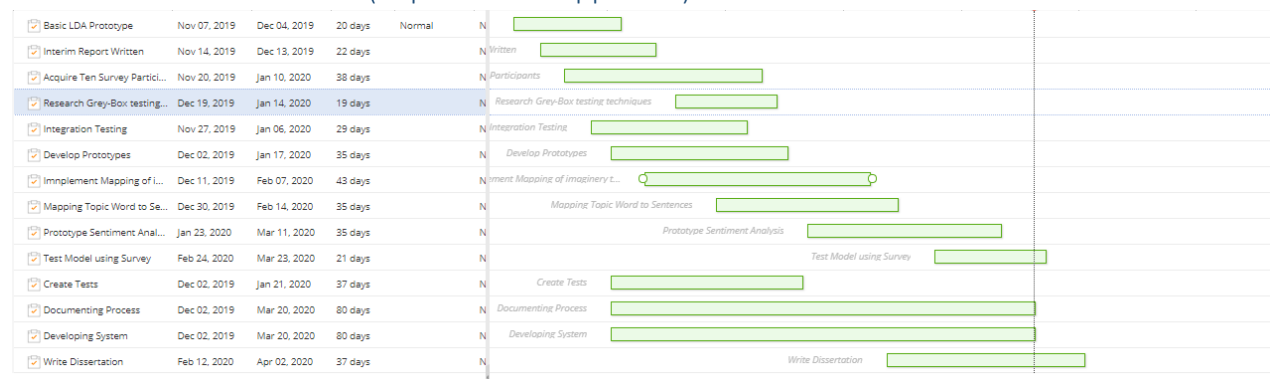*Figure 73 GANTT Chart*

6.2.2. Second GANTT Chart (Implemented Approach)

Figure 74 GANTT Chart 2