

Measuring Changes in Media Bias using Natural Language Processing AI

Christopher Thomas
Edmunds

<17024395>

UXCFXK-30-3

Abstract

The goal of this project is to evaluate the viability, and practicality, of measuring changes in bias towards key figures in media, by tracking changes in how positive or negative a news media company is about particular key figures, using machine learning and natural language processing techniques. This includes people, organisations, or social groups.

Additionally, this project would like to evaluate if this can be achieved in a live application, where general users can access and view the data in a user-friendly fashion.

Acknowledgements

I would like to thank my supervisor Dr Shelan Jeawak, for providing invaluable insights in the conception of this project, my family, particularly Robert and Susan, for their endless support throughout my studies, Annie for being there for me through every difficult day, and Matthew and Bradley for patiently sitting through countless hours of incoherent rambling.

Table of Contents

Abstract.....	2
Acknowledgements.....	3
Table of Contents.....	4
Table of Figures.....	7
Table of Tables.....	8
Chapter 1: Project Background	9
1.1 Introduction	10
1.1.1 Motivation.....	10
1.1.2 Concept outline.....	10
1.1.3 Objectives.....	11
1.2 Literature Review	11
1.2.1 Media Bias.....	11
1.2.2 Attempts to Appraise Bias	12
1.2.3 About Machine Learning.....	14
Naïve Bayes Classifiers	14
Deep Convolutional Neural Network.....	14
1.2.4 Rule Based System	14
1.2.6 Understanding Data	15
Order of key figures:	15
Decision Tree for key figures:	15
1.3 Requirements.....	16
1.3.1 User and System Requirements.....	16
1.3.1.1 User Requirements	16
1.3.1.2 System Requirements	16
1.3.2 MoSCoW	17
1.3.2.1 Must Have	17
1.3.2.2 Should Have	18
1.3.2.3 Could Have	18
1.3.2.4 Won't Have	19
1.3.3 Elicitation and Analysis of Requirements.....	20
1.3.4 Project Timeline	21
.....	21
.....	21

1.3.5 Methodology.....	22
1.3.5.1 Agile	22
1.3.5.2 Waterfall	23
1.3.5.3 Conclusion.....	24
Chapter 2: Software Design	25
2.1 High Level Design	26
2.1.1 System Architecture.....	26
2.1.1.1 Choice of tools	26
2.1.1.2 User Interface	27
2.1.1.3 Database Design.....	29
2.1.1.4 Environment design	31
2.2 Low Level Design.....	33
2.2.1 State Machine Diagram.....	33
2.2.2 Use Case Diagram	35
2.2.3 Sequence Diagram	36
2.2.4 Choice of tools for Natural Language Processing	37
2.2.4.1 Sentiment Analysis.....	37
2.2.4.2 Named Entity Recognition	38
2.2.5 Why Chart.js.....	40
2.2.5.1 Why Flask	40
2.2.5.2 Why Beautiful Soup	40
2.2.6 Test Design	41
2.2.6.1 NLP Testing.....	41
.....	43
Chapter 3: Implementation	44
3.1 Understanding the BBC News Website.....	45
3.2 Web_Crawler	47
3.3 Post Processing of the Data	49
3.4 Measuring Sentiment.....	50
3.5 Choosing a Named Entity.....	51
3.6 Sentiment Database/SQLite Database.....	53
3.7 Flask Web Server.....	54
3.8 Base_HTML	56
3.9 Testing Results	57
NLP Testing.....	57
Chapter 4: Project Evaluation	58

4.1 Functional Limitations.....	59
4.2 Research.....	59
4.3 Design.....	60
4.4 Implementation	60
4.5 Time Management.....	60
4.6 Supervisor	61
4.7 Aims and objectives	61
Chapter 5: Further Work and Conclusions.....	62
5.1 Ideal Whitebox Designs.....	63
5.2 Graph Library	64
5.3 Web Crawling.....	64
5.4 Data Refactoring	64
5.5 Bespoke Model	65
5.6 Multiple Metrics.....	65
5.7 Fuzzy Wuzzy	65
5.8 Conclusion.....	66
Glossary.....	67
Table of Abbreviations	68
References / Bibliography.....	69
Bibliography	69
Appendix A: First Appendix.....	71

Table of Figures

1	Figure 1.2.1: Media Bias Diagram	12
2	Figure 1.3.4.1: Project Timeline	21
3	Figure 1.3.4.2: Project timeline and Gantt Chart	21
4	Figure 1.3.5.1: Agile Methodology Example Diagram.....	22
5	Figure 1.3.5.2: Waterfall Methodology Example Diagram.....	23
6	Figure 2.1.1.2.1: Chart Design	27
7	Figure: 2.1.1.2.2: MVP Whitebox Design	28
8	Figure 2.1.1.4: Environment Component Diagram: Fullscreen version on next page	31
9	Figure 2.2.1: State Machine Diagram: Fullscreen version on next page.....	33
10	Figure 2.2.2: User Case Diagram	35
11	Figure 2.2.3: Sequence Diagram	36
12	Figure 2.2.5: Usages of Flask in Major Software Companies	40
13	Figure 3.1.1: BBC News Header	45
14	Figure 3.1.2 BBC News Latest Update Section	45
15	Figure 3.1.3: Checking for New Articles	46
16	Figure 3.2.1: Multiple Update Backlogs	47
17	Figure 3.2.2: Single Updates Backlog	47
18	Figure 3.2.3: BBC HTML View	48
19	Figure 3.2.4: BBC Text Body Class	48
20	Figure 3.2.5: BS4 Body Search Code	48
21	Figure 3.2.6: Content Cleaning Code	48
22	Figure 3.3.1: Pre-Refactoring Article Data	49
23	Figure 3.3.2 Post-Refactoring Data	49
24	Figure 3.3.3: Post Weighting Data	50
25	Figure 3.3.4: Incorrectly formatted data	50
26	Figure 3.3.5: Mislabelled Data 1	52
27	Figure 3.3.6: Mislabelled Data 2	52
28	Figure 3.3.7: Fetched Article Data.....	52
29	Figure 3.6.1: Test Data	53
30	Figure 3.6.2: Real Data	53
31	Figure 3.7.1: Chart.JS Documentation	54
32	Figure 3.7.2: Early Graph Attempt	54
33	Figure 3.7.3: Graph with All Data.....	55
35	Figure 3.8.1: Bootstrap Navbar Wide	56
36	Figure 3.8.2: Bootstrap Nav Bar Reactive	56
37	Figure 4.1.1: Final Website	59
38	Figure 5.1.1: Ideal Whitebox Design	63
39	Figure 5.1.2: Ideal Whitebox Design Labelled.....	63
40	Figure 5.2.1: Two Key Figure Graph	64

Table of Tables

Table 1 Must Have Functional Requirement	17
Table 2 Must Have Non-functional Requirement	17
Table 3 Should Have Functional Requirement.....	18
Table 4 Should Have Non-Functional Requirement.....	18
Table 5 Could Have Functional Requirement	18
Table 6 Could Have Non-Functional Requirement	19
Table 7 Won't Have Functional Requirement.....	19
Table 8 Won't Have Non-Functional Requirements	19
Table 9: Database Example	30
Table 10: NLP Testing table.....	41
Table 11: Web Scraping Testing Table	41
Table 12: Core Process Testing Table.....	42
Table 13: Front End Testing Table.....	42
Table 14 NLP Testing Results	57
Table 15 Web Scraping Testing Results	57
Table 16 Core Process Testing Results.....	57
Table 17 Front End Testing Results	57

Main Body Word Count excluding Table of contents, Table of figures and Table of Tables:10,965

Chapter 1: Project Background

The establishment of the problem the project is intending to tackle, as well as the various requirements that need to be met, and methodologies utilised.

1.1 Introduction

With this project, we have sought to discover whether it is possible, and practical, to algorithmically measure bias, and changes in bias, within media using AI and Machine Learning based approaches. As well as attempt to realise an approach for displaying this data, in a non-technical user-friendly way.

1.1.1 Motivation

Initially it is vital to establish that this is not a criticism of the existence of bias in media, all media has a degree of bias, and in the eyes of the authors of this project this is not a failure or a problem that needs to be solved.

The problem that this project is designed to solve, is the failure of certain actors within news media to acknowledge that they have a bias, inform consumers of their bias, or in some cases to wilfully deny their bias. As a lack of objectivity is not immediately visible from the perspective of the reader this carries certain risks.

Consequentially, uninformed media consumers can be swayed by news organisations with a particular agenda. This phenomenon has been widely cited as the cause of numerous problems in individuals within global society. From adopting bigoted and dangerous viewpoints, a cultivated ignorance of certain problems in society, and even radicalisation.

However, tracking bias is a particularly difficult goal, as often the definitions and perceptions of what qualifies as a biased piece of media vary from individual to individual.

Consequentially, we have decided that instead of decreeing a measure of bias in media, as other similar projects have done, it would be more valuable to provide a tool to enable a user to make that judgement for themselves from an informed standpoint

1.1.2 Concept outline

The tool from the user's standpoint will be a web page, wherein they can enter the name of an individual, company, organisation or political party. And then view a line graph of every article they are mentioned in, by date, from a specific outlet, with an associated value for each point in time ranging from 100% positive to 100% negative.

In the background there will be a program pulling articles from news websites and processing them using NLP methodologies to appraise how positive or negative they are about each individual who is named within the articles.

This should provide a helpful insight into not only how positive or negative news outlets have been when discussing key figures, but also how those perceptions may have changed over time.

1.1.3 Objectives

The intent of this project is that a user could, using this tool, discern whether their media of choice has consistently been overly negative or positive towards an individual or group. If so, they may wish to find insights from media that is more impartial, or at least biased in a differing direction for comparison.

Alternatively, it may allow a user who has noticed a change in the output of the media they consume, to check to see if the media they consume has changed its biases recently. A problem that occurs when news groups are influenced by governments, or financially invested in certain topics.

We will seek to measure this by recording the generalised sentiment towards individuals and organisations in all articles posted by various news media organisations, and chart them in a user friendly, insightful manner.

Consequently the overarching objectives can be summarised as such

- To access the body of text of all articles posted by news media organisations
- To appraise the various sentiments attached to the various key figures within those articles
- To store this information in a database
- To display the information in an easy-to-understand manner, ideally a line chart, on request from a user utilising a web page.

1.2 Literature Review

This section details an introduction to, and discussion of, the existing literature and environment which relates to the project that we are seeking to develop.

1.2.1 Media Bias

Within the current media climate, there exists certain incentives for allowing the bias of a news media organisation to become more extreme. For example, there is a documented relationship between biased media relating to certain groups, intended to incite animosity. And its ability to generate greater user engagement, and thus higher advertising revenue. (Rathje, (2021)). Since, as this paper proposes, there is a particular tendency towards social group animosity, it is reasonable to expect that any attempt to measure media bias would need to cover not just individuals but social groups as well.

Additionally, media often experiences a degree of state influence that can affect the type of media that is released. (Christian Davenport, (2010)) as this paper suggests, awareness of bias within media does have a mitigating effect on its effectiveness, suggesting bias analysis tools do have value.

Media bias developed under these circumstances, can often have far reaching effects within a society, such as altering elections, (Kaplan, (2007)) and inciting discrimination and violence within communities (Durrheim, (2005)). The severity of the consequences of many of these outcomes can be quite severe

These self-evidently, are issues within society worth tackling, but bias is an inherently difficult metric to measure. Perceptions of bias in media are extremely dependent on an individual's own biases,

meaning that objective appraisal is hard to achieve. (D'Alessio D.D, 2003) this suggests that any conventional measuring schema is likely to be unable to cope. For example, a methodology like that utilised by the BBFC for rating films, wherein a series of focus groups and test viewers provides an opinion of the rating they think is appropriate, based on a variety of criteria, is likely to be unable to cope with the polarising nature of media bias. As (Kaplan, (2007)) suggest, the more polarised an individual's political bias, the more biased their appraisal of political bias in all forms of media is, less forgiving of bias across the aisle and more forgiving of bias they agree with.



While there have been a few attempts at measuring this data using techniques like this, they tend to be noticeably unobjective. The most prominent example of this being the “All Sides Media Bias Chart” which is based exclusively on comparative questionnaires, where a left and right leaning individual attempt to appraise the same online content. (All Sides Media Bias, 2020) While we feel this attempt is conceptually solid, its subjectivity produces overly broad results *See Figure 1.2.1* suggesting that a more objective methodology would be a better approach.

1 Figure 1.2.1: Media Bias Diagram

There have been multiple attempts at developing a schema or system capable of generating an effective measure of bias in media, but the most promising of these have been machine learning and AI models. They have consistently shown the best accuracy in objectively appraising bias in media, but as of yet there appears to be no agreed upon best method, with a wide variety of potential techniques being tried and tested.

1.2.2 Attempts to Appraise Bias

The first methodology we want to examine is an AI developed by “The Bipartisan Press”, an American media bias advocacy group. (Wang, 2019) Their methodology focusses on a bespoke machine learning model trained using a dataset of prelabelled data comprised of articles with associated biases. Their model was designed around BERT and LSTM

This methodology is interesting, but dependent on quality of the dataset used to train their machine learning model, in its development they developed two datasets and utilised two external ones while attempting to land on an optimal solution. Each solution was flawed in distinct ways, largely due to either biasing inherent to the subjectivity of the dataset, or high levels of noise. (Wang, 2019) Given this vulnerability to noise and subjectivity when utilising datasets that directly measure “bias” as a concept, we would likely lean away from using bias as a specific metric, instead leaning towards more objective metrics which are indicators of bias.

The other established methodologies we were able to discover do generally follow TPB's baseline technique, however they differ in the machine learning (or non-machine learning) models they use as well as the metric they are trying to measure to base their more subjective judgement of bias off.

In "Uncovering gender bias in newspaper coverage of Irish politicians using machine learning" the methodology utilised was to measure the frequency of appearances of Irish MPs in Irish news media, and to utilise various methodologies from Naïve Bayes Classifier machine learning models to Rule-based models (Leavy, 2019), to attempt to intuit the gender of the MP in question. With a consistent accuracy rate the model was able to note a distinct difference in the frequency at which Irish media would discuss female MPs, when compared to their male colleagues, suggesting a measurable bias regarding female MP's.

We would particularly like to focus on the metrics detailed in (F Hamborg, 2019). This Literature review details a wide variety of additional metrics used for attempting to measure bias including

- Event Selection: The types of events reported on
- Source Selection: Typical biases of the sources cited
- Commission and omission: Decision on omission or inclusion of supporting or disputing facts
- Word Choice: Choice of terms that indicate a bias vs alternative term
- Story Placement: The placement of stories to attract more or less attention
- Size allocation: The size of stories to attract more or less attention
- Picture Selection: The choice of negative or positive images
- Spin: The overall metric generated from some or all of the previous metrics.

All of these provide interesting choice of metric, and we believe all of them are valuable. Such a comprehensive list is reassuring however, as it implies that our approach is relatively novel.

Late in the research the author was doing for this project, the following article was found. (Tung, 2021) This was particularly interesting as it revealed a very similar approach to the problem, the use of named entity recognition and sentiment analysis in tandem to appraise bias. However, the main distinction being the source of the article data and the method of displaying the data.

While the article has merit, there are some aspects of the methodology that the author of this article has implemented, that we believe less practical than the methodology that we have employed. The first of these being the assignment of specific named entities to "left" or "right" categories. While this is useful for analysing bias towards very specific topics, it is poor for analysing bias for anything outside of the predefined groups. Our approach instead leans towards a flexible general-purpose tool.

Additionally, the articles reliance on a third-party live service "Diffbot Knowledge Graph" (Tung, 2021) to provide and display data was impractical as this is a limited paid for third party service. Our approach is focussed on developing a web scraper and website which will give us much more flexibility.

1.2.3 About Machine Learning

Within the context of this project, it is important to briefly summarise the design of certain machine learning models. Specifically, we would like to explore Naïve Bayes Classifiers, and deep convolutional neural network as these were the two models eventually utilised in the final product.

Naïve Bayes Classifiers

The Naïve Bayes Classifier Model we utilised is functionally a formula with weighted values that are altered depending on the accuracy of the model when tested.

It is based around the idea that to identify the probability chance that the object fits into a category, in this case a specific sentiment, you can test to see how many particular features of that object are present. The probability that a feature is attached to a category can then be altered through successive trainings.

This lack of interconnectivity between the features is what defines it as a “Naïve” classifier, although non naïve applications of the bayes algorithm do exist, this is beyond our scope. (Rish, 2001)

Deep Convolutional Neural Network

A DCNN is ultimately what we utilised for the Named Entity Recognition aspect of our project.

DCNN are particularly ideal for NER applications, as their assessments are contextualised by the data surrounding them, which is defined as the “convolutional feature”. In a sentence all of the words within the sentence will affect one another’s assessment. This is useful for extracting particularly pertinent details from a collection of data points.

Similar data points are then collated together to reduce the overall volume of the data inputs that need to be processed by the system. This is described as the pooling layer.

Finally, the data is passed to the neural network aspect of the DCNN. Here a collection of layers of nodes, also called neurons, are each effected by the data introduced from the input layers. The outputs of the data travelling through the various layers will ultimately reach an output layer that will provide your end result. Training this model is done by testing the validity of the output, selecting for the models that produce optimal results, then reproducing that model and randomly modifying a small part of the improved model (S. Albawi, 2017)

1.2.4 Rule Based System

For our sentiment analysis we ultimately were forced to settle on a rule-based AI system as opposed to a machine learning algorithm such as Naïve Bayes, or a Deep Learning methodology like the DNCNN due to speed constraints. *See 3.4 measuring sentiment* Below is a brief overview of how this methodology functions

The Textblob library we utilised makes external use of the pattern Library, this follows a simple rule-based system where adjectives are given sentiment values, and their frequency within a text is used to derive an overarching sentiment value. External data can be introduced to further train this model, but ultimately it is deterministic. There is a strong library of existing evidence on the practicality and validity of this method (L. I. Tan, 2015) but it is unfortunate that we could not utilise a more dynamic approach for this aspect.

1.2.6 Understanding Data

While developing the named entity recognition, it became apparent that a sentence could contain multiple named entities. It took a surprising amount of research to devise a solid methodology for choosing which one to utilise.

For devising this technique, we needed to research a field of study titled “Input Processing Theory” (VanPatten, 1993)

Order of key figures:

With VanPatten's book (VanPatten, 2004) he proposes the “First Noun Principle” which can be abbreviated as “learners tend to process the first noun or pronoun they encounter in a sentence as the subject or agent.”

From this we can discern that the subject of each of the sentences we process is likely to be the first named key figure, so we should develop our project to preference the first proper noun.

Decision Tree for key figures:

In addition, we needed a way to discern a decision tree for our eventually accepted four categories of named entity. After some deliberation we settled on the following technique

Our named entity recognition records the following four entity types, “Organisation”, “Geopolitical Entity”, “Person”, “Nationalities or religious or political groups”

These categories can be broken down into (New, 2004):

Singular Subject Nouns: An individual

An individual will almost always be the subject of a sentence. Therefore, we can place “Person” at the top of our hierarchy.

Plural Subject Nouns: A group

This leaves our three Plural Nouns. Ultimately, we chose to simply sort their hierarchy by scale, as typically a large scale is used more for context than specific comment (Mohamed, 2011). Therefore, our final hierarchy is

Person > Organisation > NRPG > GPE

Finally, we chose to base the decision tree as a secondary modifier, hence our final algorithm will always opt for the key figure who is the furthest down the hierarchy of entity types but will select from that lowest level whichever key figure appears first in the sentence.

1.3 Requirements

1.3.1 User and System Requirements

1.3.1.1 User Requirements

The broad user requirements can be described as follows

- Must be able to enter a key figure
- Must be returned a graph of the key figures' sentiment in the news over time
- Must be easy to use

1.3.1.2 System Requirements

The broad system requirements can be described as follows

- Must be able to retrieve all new daily articles from a news website
- Must be able to break that data down into sentences
- Must be able to detect key figures in articles
- Must be able to detect sentiment attached to those key figures
- Must be able to save that data to a database in a usable fashion
- Must be able to average out and weight that data appropriately

1.3.2 MoSCoW

Reference Key:

KFNN: Key Figure Neural Network

SENN: Sentiment Analysis Neural Network

DB: Database

WC: Web Crawler

DWS: Dedicated Web Server

1.3.2.1 Must Have

Table 1 Must Have Functional Requirement

Reference	Functional Requirement
KFNN-01	The ability to discern a key figure in a sentence with at least 70% accuracy
SENN-01	The ability to discern the sentiment of a sentence with at least 70% accuracy
SENN-02	Generate a net sentiment score for a key figure in an article
DB-01	Store the key figure, sentiment score, article reference, article source, and date of publish in a table
WC-01	Must be able to web crawl newest articles from at least one news source, daily
WC-02	Must be able to process articles into a format readable by the neural networks
DWS-01	Must be able to take input from user of key figure, and display a relevant graph of sentiment over time

Table 2 Must Have Non-functional Requirement

Reference	Non-Functional Requirement
DWS-03	Must support colour blindness palettes
DB-03	Must have sufficient security against SQL injection (since database queries are being used)
WC-03	Must be able to handle incorrect, flawed or missing data or webpages
WC-04	Must be able to access, download and process all data on each supported source, within a 24-hour time period
DB-02	Must not take up an unreasonable volume of hard disk space
DWS-02	Must be sufficiently useable for an individual from a non-technical background

1.3.2.2 Should Have

Table 3 Should Have Functional Requirement

Reference	Functional Requirement
WC-05	The ability to source articles from multiple news sources simultaneously
DWS-04	The ability to display multiple key figures on the same graph
DWS-05	The ability to display multiple sources on the same graph
WC-06	The ability to instruct the program to search a specific term on a news site to gain a more comprehensive wider timeframe of how that key figure is perceived, beyond what has existed since the program began to run
DB-04	Periodic sorting of table (to speed up lookup times for key figures)
DWS-21	The ability for the user to input a timeframe and only retrieve data from within that timeframe

Table 4 Should Have Non-Functional Requirement

Reference	Non-Functional Requirement
DWS-06	Statement of clarity on the imperfect accuracy of neural networks, establishing this is for detecting trends as opposed to extremely accurate data
DWS-07	Should be extremely easy to use for anyone from any degree of technical skill
DWS-08	Should be accessible on a wide range of internet enabled devices
DWS-09	Audio/Text to Speech/ Differently sighted accessibility specific options
DWS-22	Should be reactive and resize for multiple screen sizes

1.3.2.3 Could Have

Table 5 Could Have Functional Requirement

Reference	Functional Requirement
DWS-10	Double checking articles for edits, treating edited article as new article as opposed to over-righting existing article (media has historically been accused to editing out problematic statements after the damage has been done)
SENN-03	Method of improving Neural Network model (similar method to captcha? Asking users to rate the sentiment of a sentence, the key figure of a sentence, and allowing the user to enter if what they enter agrees with the consensus, thus validating a piece of training data that can then be used, as well as combating security breach attempts)
DWS-17	Ability to use multiple types of graphs, as opposed to just line graphs against time
DWS-18	The ability to export the graph as an image file

DWS-19	The ability to generate a .txt file of all data appearing in a graph
--------	--

Table 6 Could Have Non-Functional Requirement

Reference	Non-Functional Requirement
DWS-20	General aesthetic improvements, nice intro logo/animation, quality CSS

1.3.2.4 Won't Have

Table 7 Won't Have Functional Requirement

Reference	Functional Requirement
WC-07	Support for non-web-based news media (newspaper, tv, etc.) could make use of voice and text recognition neural networks to pre-process data
DB-05	Full comprehensive database going back as far as online articles exist (no way to view all articles, only searchable terms)
DWS-11	Embedding into other websites
DWS-12	Offline Support (an app that appraises the biases of all the text you read could be insightful)

Table 8 Won't Have Non-Functional Requirements

Reference	Non-Functional Requirement
DWS-13	Database sharding to support large tables
DB-06	Oversight to ensure general accuracy (and improve accuracy) of neural networks
DWS-14	Dedicated web server
DWS-15	DDOS/Etc advanced web protection
DWS-16	Any form of server redundancy

1.3.3 Elicitation and Analysis of Requirements

For the development of this project, it is vital to establish who the key stake holders are.

Initially, it would appear that virtually all demographics, regardless of socio-economic strata have some degree of concern as to media bias. However, from an objective standpoint It would appear that news media organisations and their employees may indeed have a degree of vested interest in supporting and maintaining media bias, as opposed to opposing it (Kaplan, (2007))

Therefore, it is reasonable to assume that the individuals most likely to benefit from this tool, are media consumers. Particularly media consumers who may not be readily able to detect the biases that they are falling victim to (D'Alessio D.D, 2003). Additionally political activists and groups who are struggling to combat misinformation (Durrheim, (2005)) may find that the introduction of a tool for greater objectivity supports their goals. Consequentially we would label the key stake holders as.

Stake holder 1: General News Media Consumers

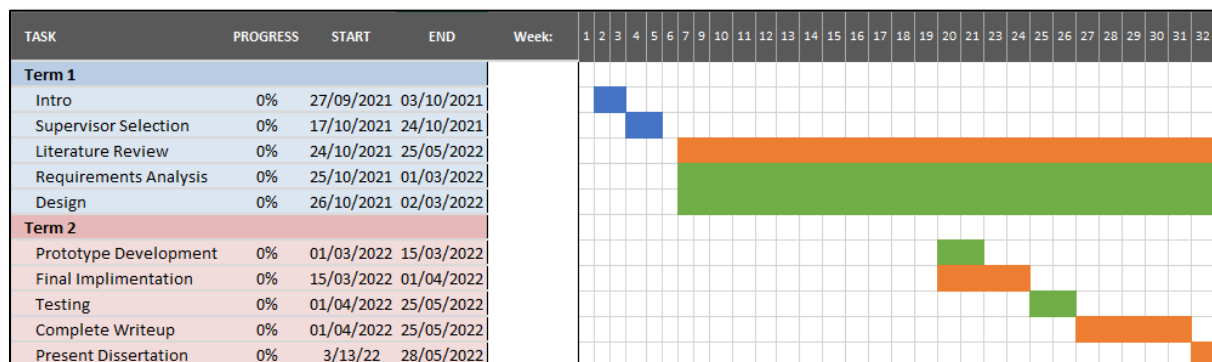
Stake Holder 2: Political Activists

1.3.4 Project Timeline

A timeline and associated Gantt chart were produced to help plan out the development of this project, they show the intended goals and progress points for the production of this project. See *Figure 1.3.4.1: Project Timeline* and *Figure 1.3.4.2: Project Timeline and Gantt Chart*

TASK	PROGRESS	START	END
Term 1			
Intro	0%	27/09/2021	03/10/2021
Supervisor Selection	0%	17/10/2021	24/10/2021
Literature Review	0%	24/10/2021	25/05/2022
Requirements Analysis	0%	25/10/2021	01/03/2022
Design	0%	26/10/2021	02/03/2022
Term 2			
Prototype Development	0%	01/03/2022	15/03/2022
Final Implimentation	0%	15/03/2022	01/04/2022
Testing	0%	01/04/2022	25/05/2022
Complete Writeup	0%	01/04/2022	25/05/2022
Present Dissertation	0%	3/13/22	28/05/2022

2 Figure 1.3.4.1: Project Timeline

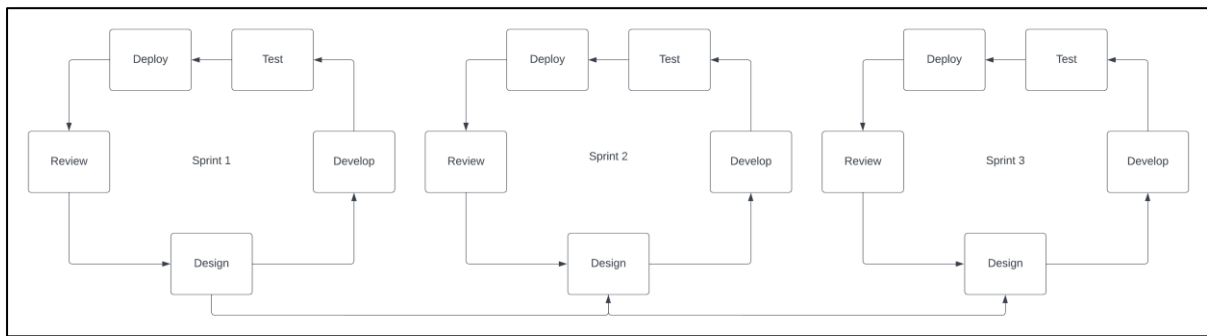


3 Figure 1.3.4.2: Project timeline and Gantt Chart

1.3.5 Methodology

This section will discuss the main software development methodologies utilised in the production of this project. The purpose of a software methodology is to guide the various stages of production that a project will go through in its conception. There are two primary methodologies that were considered.

1.3.5.1 Agile



4 Figure 1.3.5.1: Agile Methodology Example Diagram

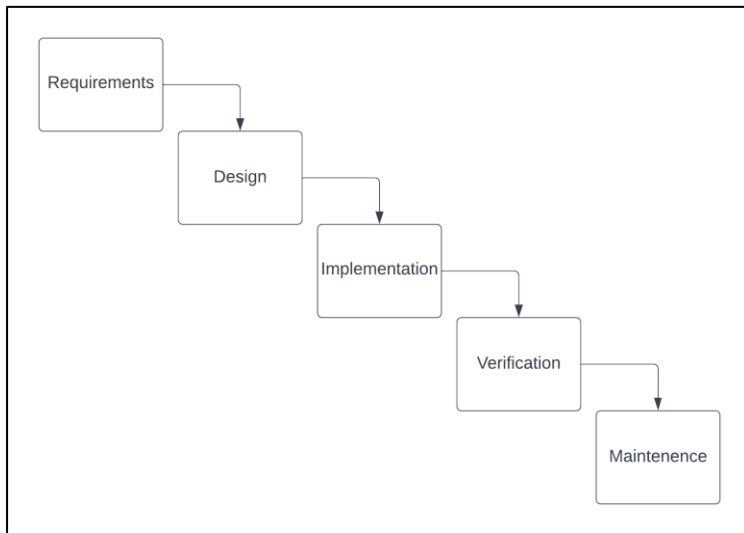
The Agile Methodology is based around developing the software in a series of repeating incremental steps. Where each loop constitutes a “sprint” to complete, test and push a series of predefined targets. At the end of each sprint uncompleted tasks are pushed to the next sprint and the project repeats itself. See Figure 1.3.5.1: Agile Methodology Example Diagram

Agile is extremely flexible compared to other methodologies, allowing you to defer work that is incomplete to another sprint, without halting progress totally. Additionally, it supports a “Dive straight in” approach where you can iteratively develop without significant preplanning, ideal for complex projects with large amounts of functionality. The iterative approach also has the advantage of supporting frequent reaffirmation of goals, and iterative testing approaches, preventing large end of project bug lists buried in complex code.

However, it does have certain limitations, mainly that it is very hard to predict the time it will take to produce the project, which can lead to a project that is overly ambitious for the time frame.

1.3.5.2 Waterfall

The Waterfall methodology is based around predesigning and establishing every aspect of your project sequentially, starting at labelling out the requirements, then designing every aspect of the code, then implementing all your designs, then verifying all your results, before ultimately maintaining your final code as long as the lifespan persists see *Figure 1.3.5.2: Waterfall Methodology Example Diagram*



5 *Figure 1.3.5.2: Waterfall Methodology Example Diagram*

The key advantages of this methodology are that it produces an extremely well-defined scope, making long term planning considerably more practical than in agile. However, the inverse of this is that due to its rigid structure, it allows for very little flexibility, and the lack of iterative testing means that problems may arise largely undetected. Which may mean that once you reach the testing stage and discover the issue, you may need to then rewrite large swathes of code to fix the problem.

1.3.5.3 Conclusion

For this project the author chose to utilise an Agile methodology. There exist far too many systems within the project that the author is unfamiliar with for the author to feel comfortable pre-planning everything in such a definite manner as waterfall.

Our sprints were broken into the following iterations.

Iteration 1: NLP

The first iteration will focus on the development of the Natural Language Processing module, ensuring that a piece of text can have its key figure and sentiment readily detected

Iteration 2: Web crawler

The second iteration will focus on the development of the web crawler module, ensuring that the required data can be retrieved from the chosen website

Iteration 3: Completed Data

The third iteration will focus on compiling data in a manner where it can be readily saved and used by the final program, and storing it in a database

Iteration 4: Displayed Data

The final iteration will focus on developing the front end to be able to display the data for an end user.

Chapter 2: Software Design

The pre-production design documents created to plan out the construction of the project.

2.1 High Level Design

2.1.1 System Architecture

For this project we chose to develop the software using the Python programming language. The choice of this language was relatively easy due to a few key factors

2.1.1.1 Choice of tools

2.1.1.1.1 Languages:

Python: The python programming language was selected because

1. it is the primary language of machine learning research
2. It's extremely easy to use being both strongly typed and interpreted
3. The authors have a significant amount of experience with it
4. The library support is spectacular, particularly for machine learning applications.
5. It plays well with Flask

JavaScript: A vital language for dynamic web page content, used for generating charts and connecting front end and backend.

HTML: Hypertext markup language, used to create the basic webpages.

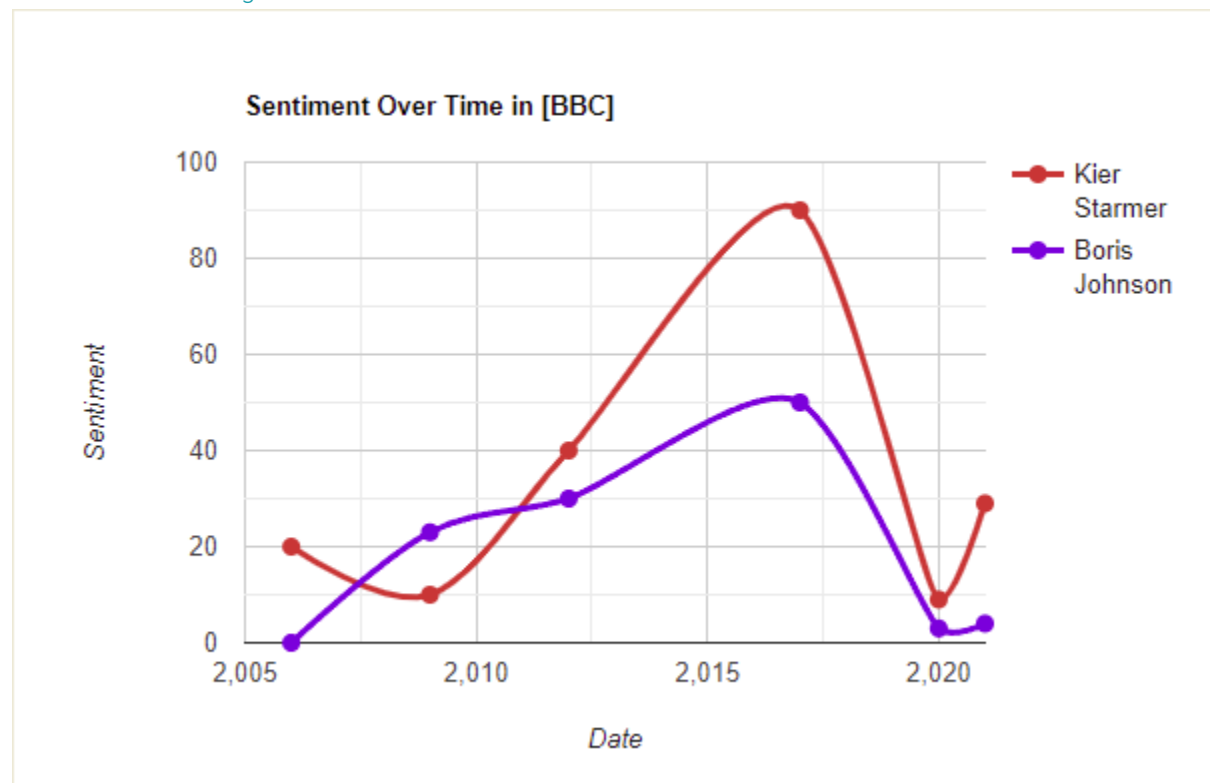
2.1.1.1.2 External Programs

Integrated Development Environment: Visual Studio Code was used, a lightweight easy to use IDE that plays well with all the required languages.

DB Browser: Database browser, designer and editor. Useful for checking the database and modifying it where needed.

2.1.1.2 User Interface

2.1.1.2.1 Chart Designs



6 Figure 2.1.1.2.1: Chart Design

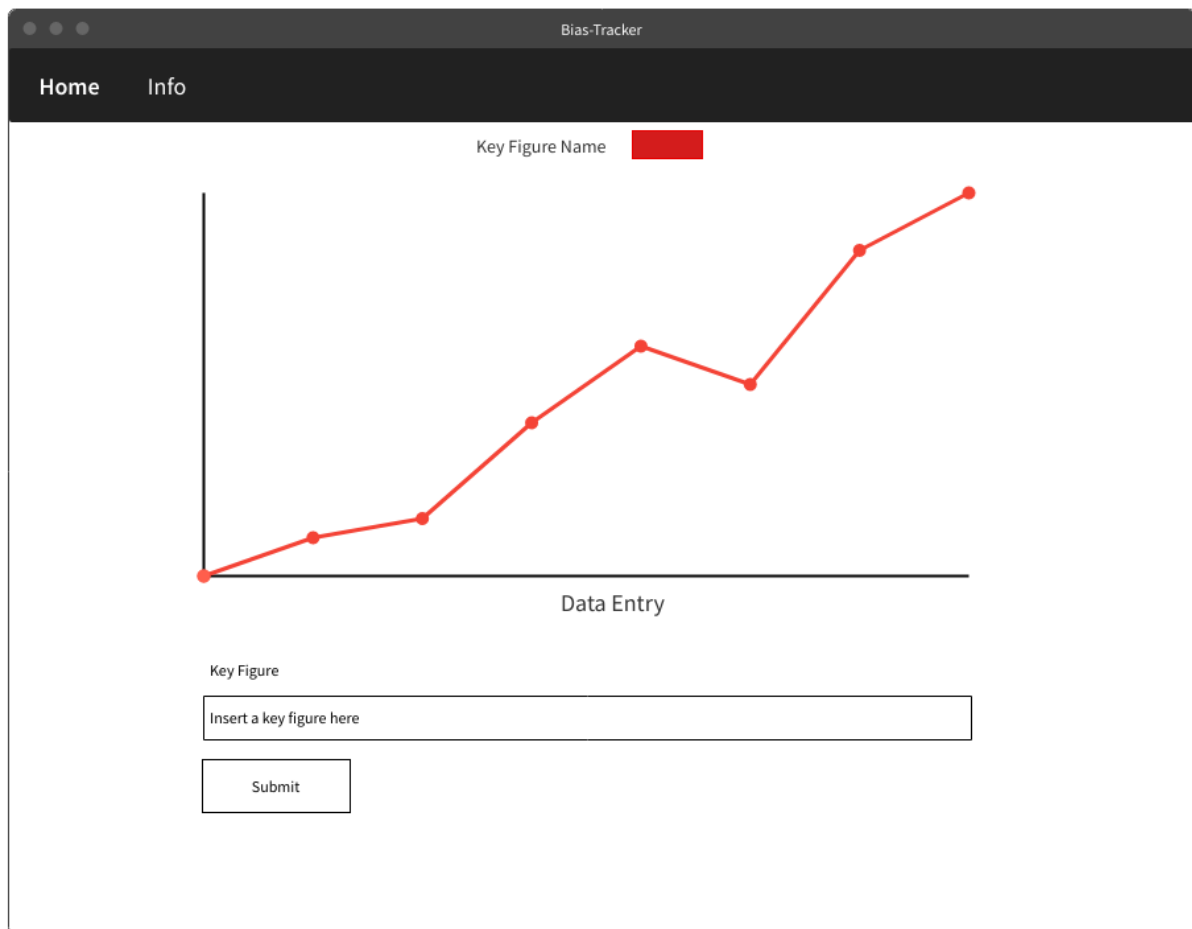
Here we have the basic design for our charts. See Figure 2.1.1.2.1

From a software side the Y-axis will be maintained as a float variable ranging from -1.0 to 1.0, we may choose to use this positive/negative scale to make it more visually apparent where the “neutral sentiment” lies and communicate that idea better to the user.

Additionally, we may elect to use a more granular dating system to support shorter time periods better

Ideally we would like to utilise graphs which have some sort of “line tension” feature such as this, to provide curved graphs, as we feel this better communicates the concept of trends than straight lines between data points.

2.1.1.2.2 Minimum Viable Product Whitebox



7 Figure: 2.1.1.2.2: MVP Whitebox Design

This is the M.V.P design for the website portion of this project, it lacks all of the features not present in the “Must” portion of the MoSCoW assessment of the project but retains the core functionality See Figure: 2.1.1.2.2: MVP Whitebox Design.

We have a simple bootstrap based Nav Bar, with the base page and a separate Info page. This is ideal as an effective starting point for expanding the project later, while providing a simple comfortable to use interface

Below is the chart which will be based on Chart.js, this will display the sentiment of the key figure over time.

Below the chart is a bootstrap text box and label showing where to enter the key figure you wish to search, and a submit button to enter that and make a request from the server.

2.1.1.3 Database Design

When designing our database, we first had to select out the type of database that was best suited for our project. There are a wide variety of options each with their own pros and cons which greatly affected our choice of system.

Our initial decision was as to whether we wanted to rely on a relational or non-relational database. We chose to opt for a relational database for a handful of key reasons.

The shape of the data we are storing comfortably fits within the constraints of columns and rows, making one of the primary advantages of NoSQL type databases irrelevant (Sareen, 2015)

The data we are attempting to store, a single table of 5 rows, estimating roughly 1500 entries a day, is likely to only occupy a relatively small amount of storage medium space. A test document of 1500 records following the rough design occupies only 95kilobytes, or 35MB of storage a year. This is perfectly sustainable for the long term; however, we may wish to migrate to a non-relational database if the addition of multiple news sources starts to increase file size beyond practical amounts.

Finally, the developers of this project have existing experience with certain relational-database models, which will speed up development times and reduce issues if it continues to be a good framework.

Next, we needed to select a particular relationship database management system. Our most likely candidates are

- MySQL
- PostgreSQL
- SQLite

Our eventual choice out of this list was SQLite, there are a few key factors which lead to this decision.

Lack of an external server: As an experimental piece of software that was produced locally on a single machine, this project doesn't have access to an external server on which to store the data. It was therefore preferably to choose SQLite which is designed around storing its database as a single file on the same server as the program itself. SQLite has built in functionality for facilitating this that the other options lack. Additionally, the other options require the set-up of a virtual server if you want to use them on the same machine. (Hossain, 2019)

Ease of use: SQLite is widely regarded as the easiest and fastest database model to learn making it ideal for rapid development of an untested concept. (Hossain, 2019) Additionally, the developers are experienced with it already

Low Write Volumes: The major limitation of SQLite is that it is only able to handle a sequential write operations. Any application that needed to make multiple write connections to the database would find it unsuitable, however the application we developed only needs to make approximately 1500 writes a day from a single connection. As the limit of SQLite is approx. 50,000 insert statements a second, this is well within the constraints of the model. Additionally, SQLite is capable of handling multiple reads simultaneously, meaning it is still capable of serving the database content to multiple users. (Hossain, 2019)

Lightweight: SQLite is lightweight, taking relatively little memory, using relatively small amounts of storage, and having no external dependencies. As this program is going to be running on a personal desktop for the majority of its development time, this is extremely advantageous. (Hossain, 2019)

Support: SQLite is both well supported with the external web-development framework Flask which will be connected to the database using the SQLAlchemy Library, it is still within active development and widely used and supported. (Hossain, 2019)

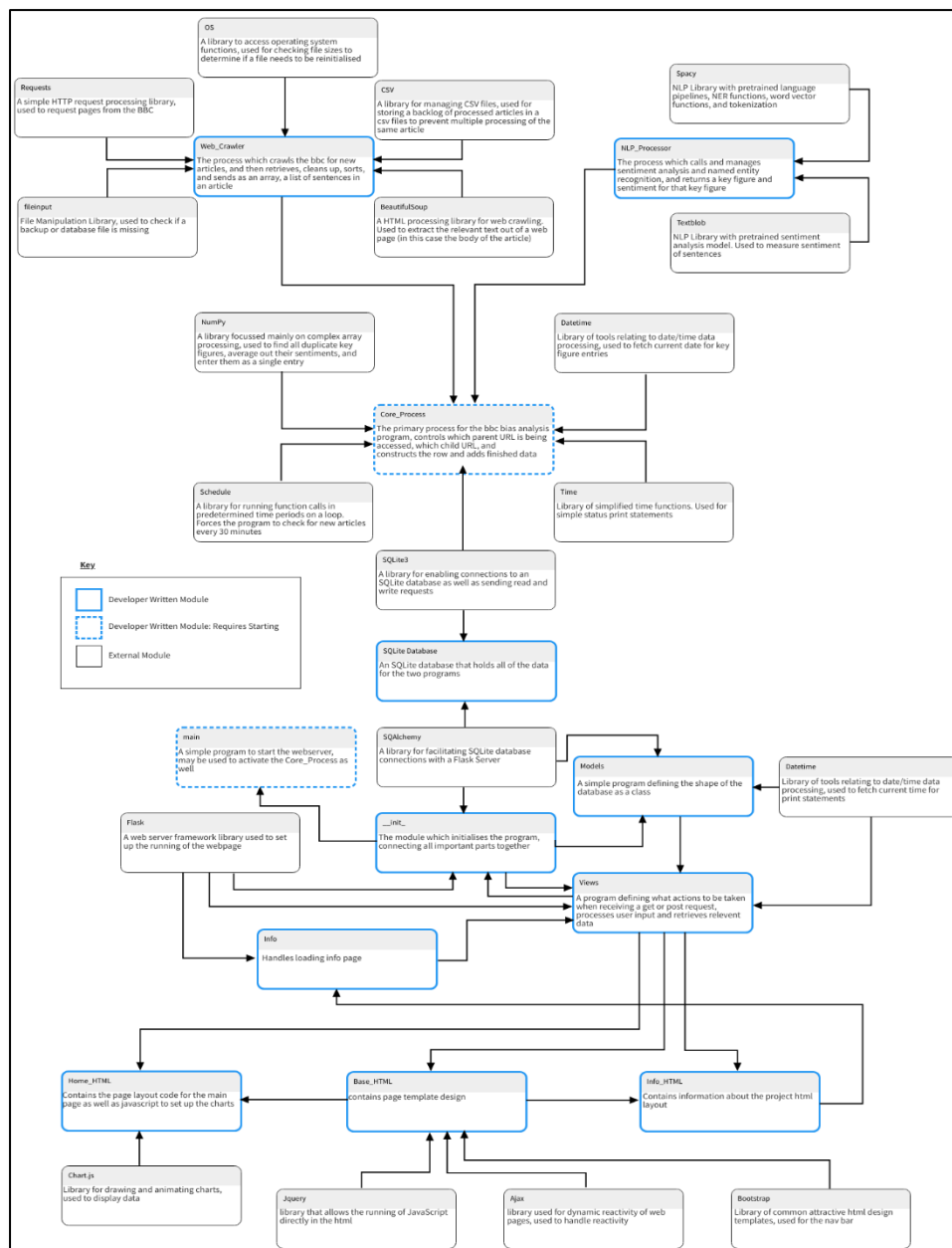
Table 9: Database Example

Primary Key				
Id	KeyFigure	Sentiment	Date	URL
1	chris	-0.1	19/12/1995	bbc.com/1234124
2	chris	0.9	20/12/1995	bbc.com/1234124
3	chris	-0.5	21/12/1995	bbc.com/1234124
4	chris	-0.1	21/12/1995	bbc.com/1234124

Here you can see an example design of the table for the production environment. As the initial design of the software only focusses on a single website there is no need for multiple tables with foreign keys, however in the future development section we will discuss how this could be designed.

2.1.1.4 Environment design

Here we have a diagram displaying the developer made components and the imported components and how they relate to each other. See Figure 2.1.1.4: Environment Component Diagram



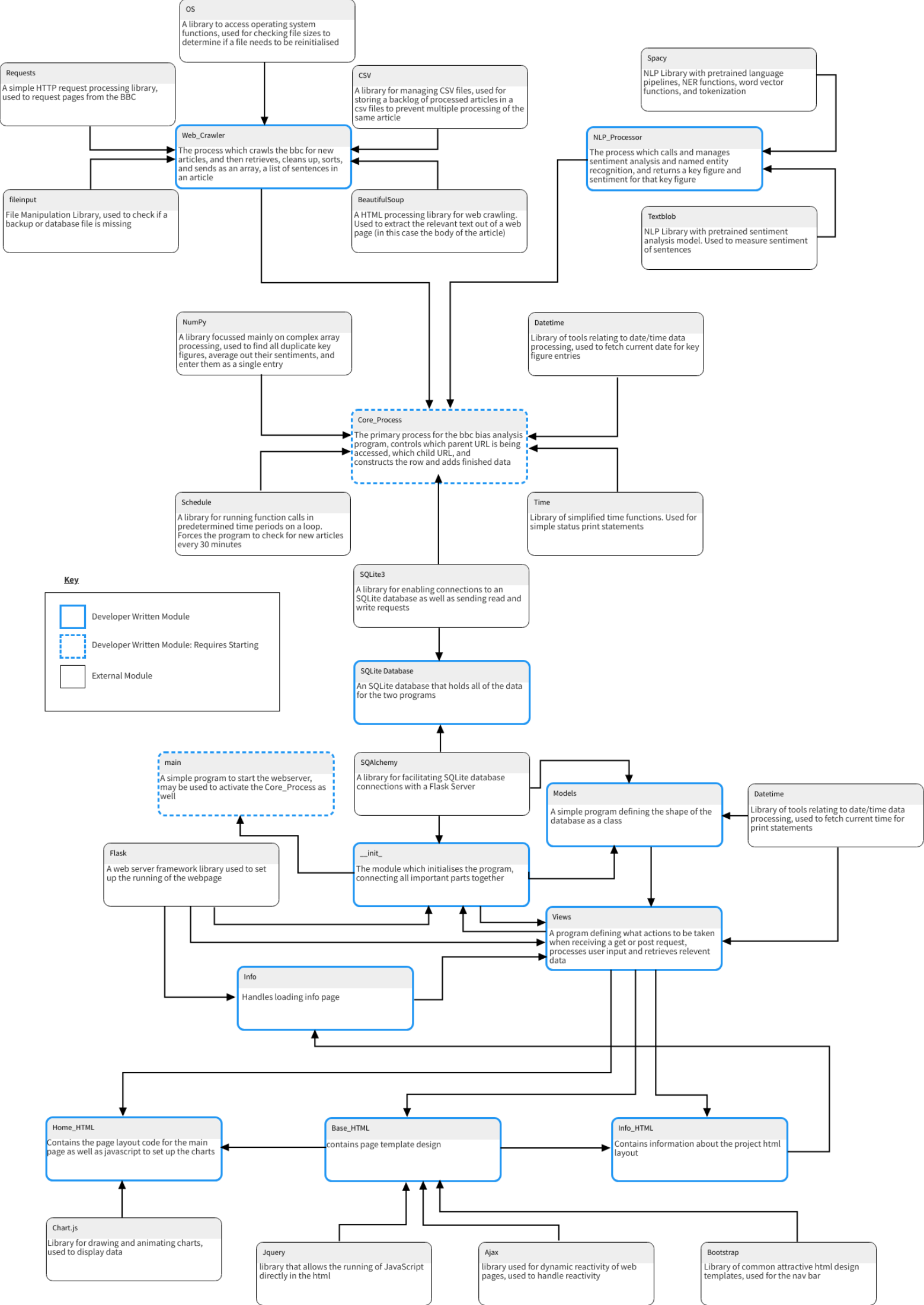
As you can see most of the external modules are called by the portion of the program dedicated to generating the data that the website will use.

The largest external dependency is, by far, the Spacy and Textblob libraries used for sentiment analysis and NER.

These are quite large libraries. Which is likely to increase the time to initialise the program significantly

8 Figure 2.1.1.4: Environment Component Diagram: Fullscreen version on next page

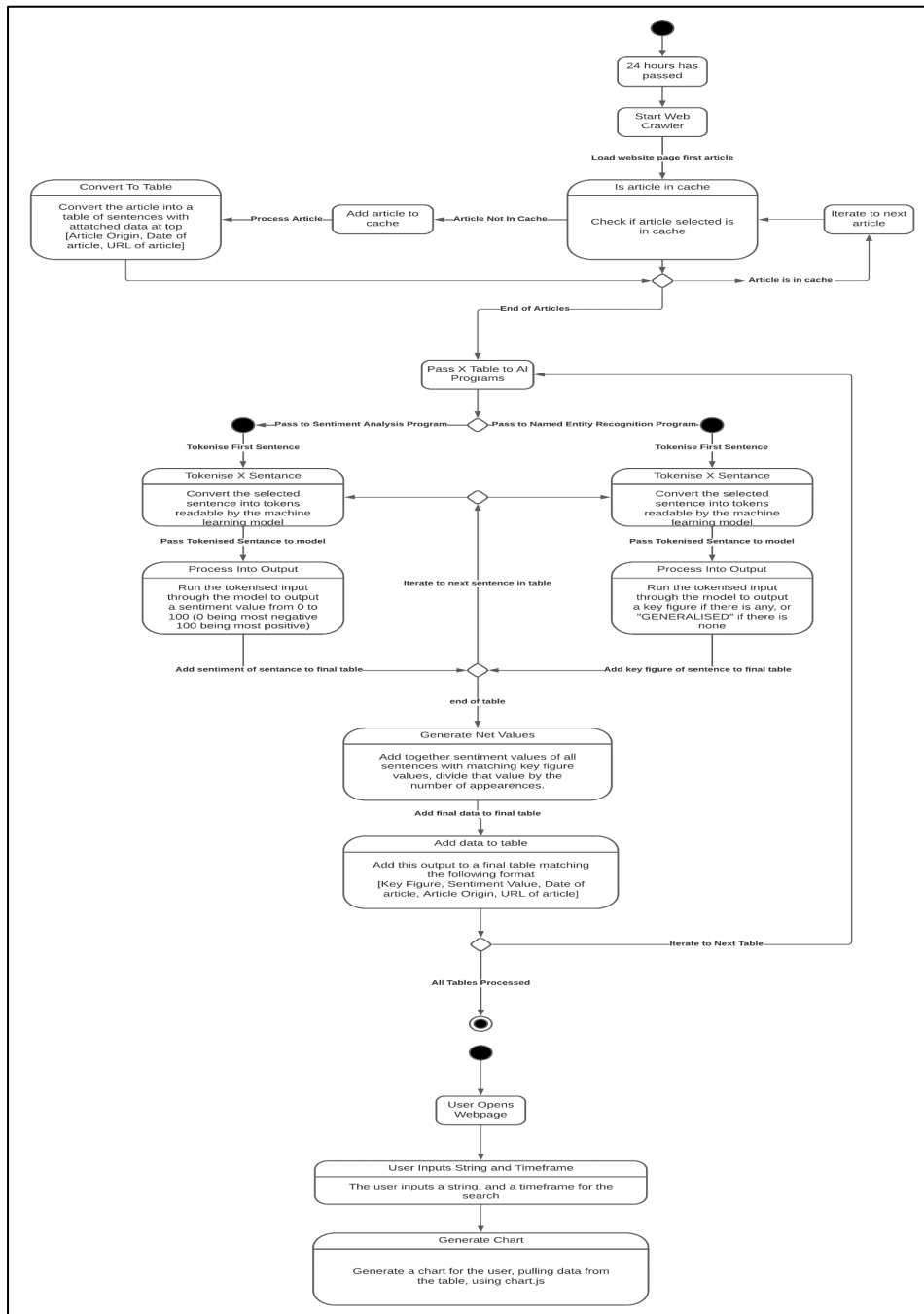
This informs the choice to have the program run endlessly on a schedule, as opposed to being run periodically when the web server instructs it to.



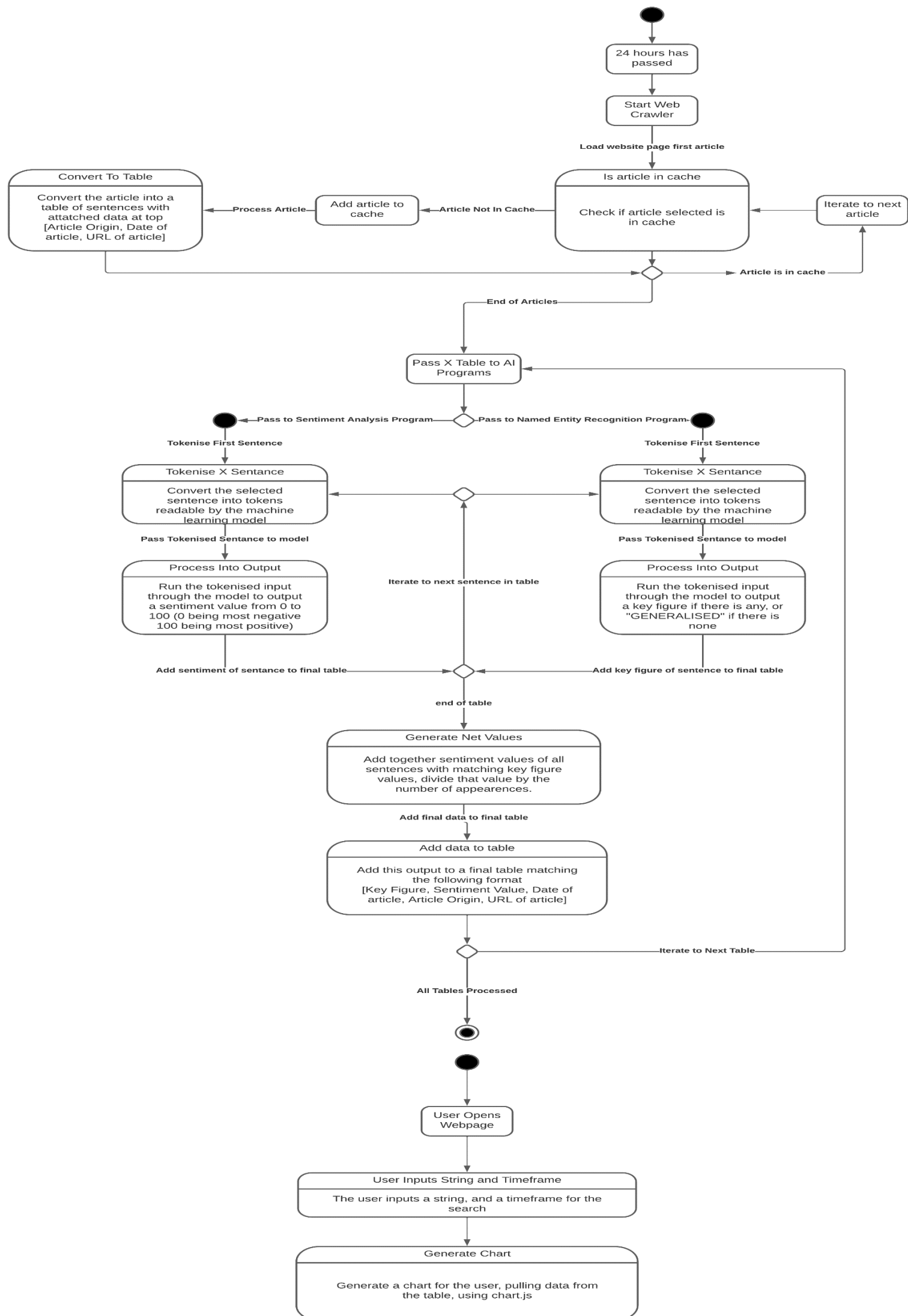
2.2 Low Level Design

2.2.1 State Machine Diagram

Here we have a diagram displaying the state machine diagram for the project. A simplified explanation of programs logic, showing all key states, and the steps required for acquiring, processing, and displaying data See 2.2.1: State Machine Diagram

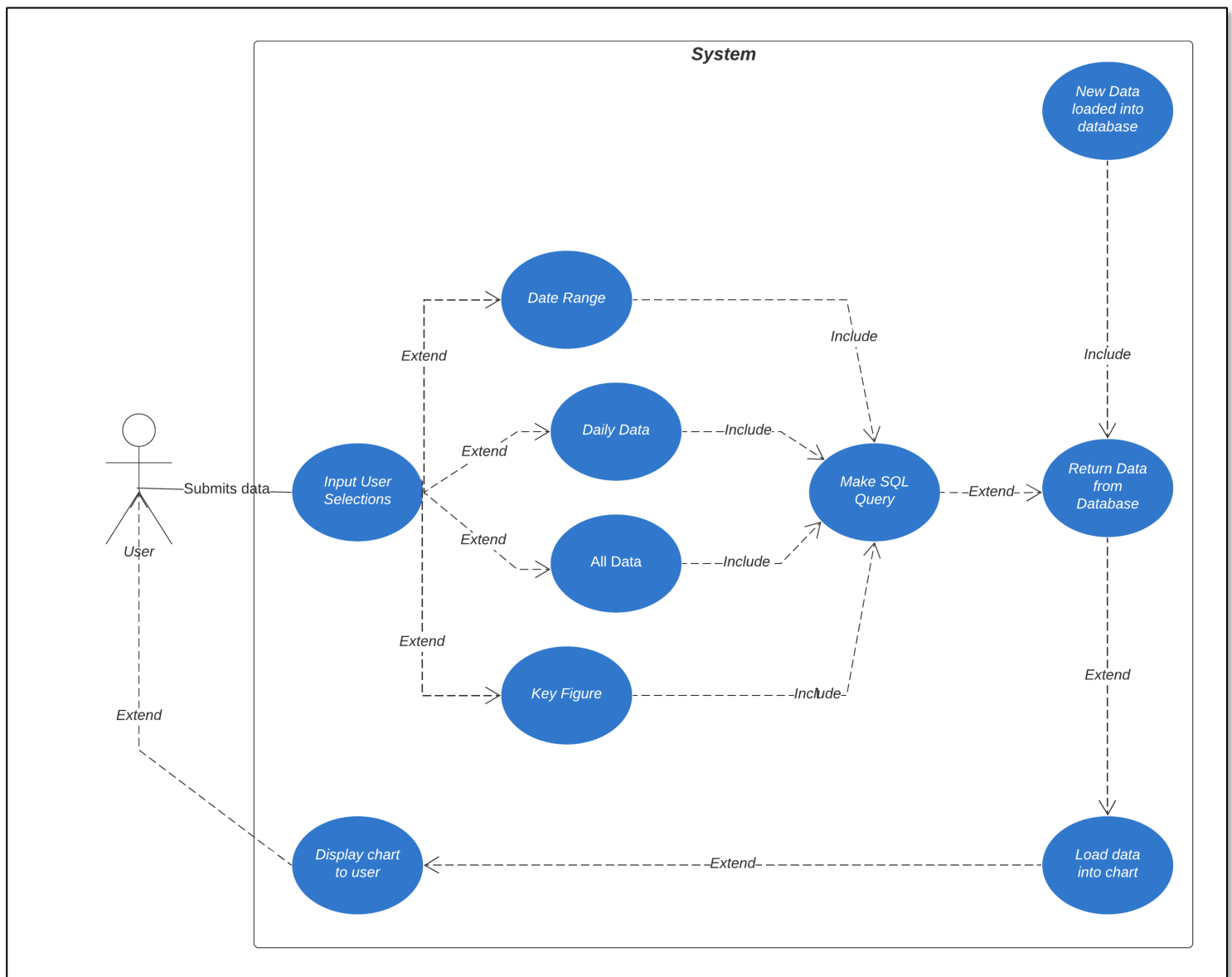


9 Figure 2.2.1: State Machine Diagram: Fullscreen version on next page



2.2.2 Use Case Diagram

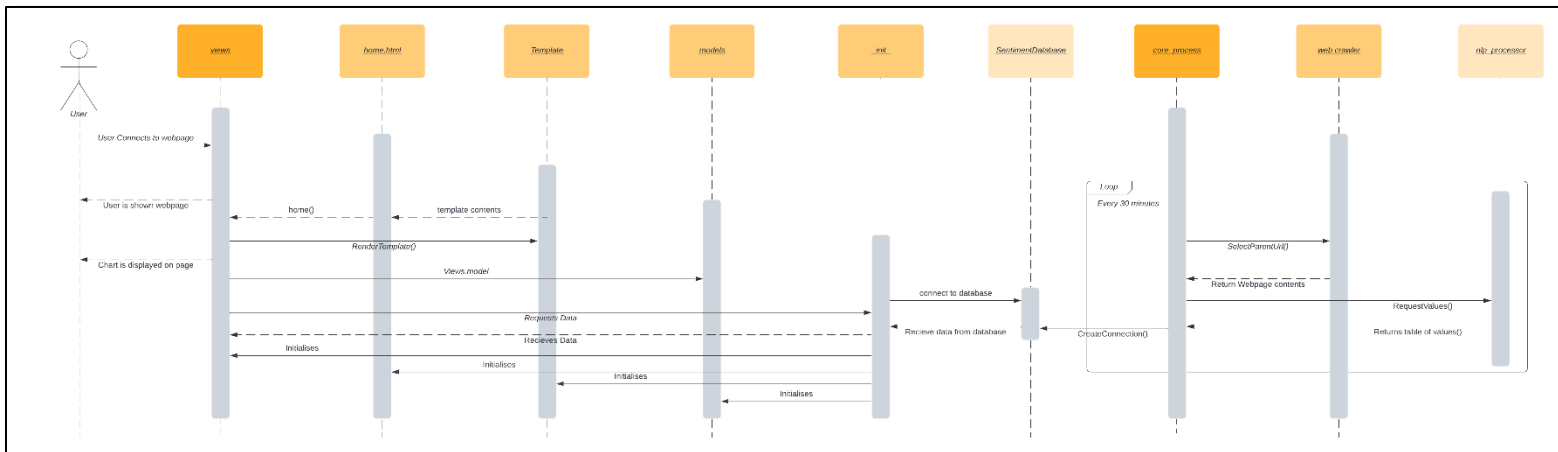
Here we have a user case diagram, this is valuable for visualising how the data is likely to be processed by the end program and the behaviour of the web server. See Figure 2.2.2: User Case Diagram



10 Figure 2.2.2: User Case Diagram

2.2.3 Sequence Diagram

Here we have a sequence diagram. This is useful for understanding the order in which events are likely to occur in the final program, and how modules interact with one another. *see Figure 2.2.3: Sequence Diagram*



11 Figure 2.2.3: Sequence Diagram

2.2.4 Choice of tools for Natural Language Processing

Natural language processing is a very broad field of machine learning, but for this project we needed to rely on two specific subsets of it, Sentiment analysis, and Named Entity Recognition.

The major initial choice when developing this project is, should we develop a model from scratch, select or create datasets to train it, and then hope it functions appropriately. Or should we rely on easy to implement, tried and true, pretrained models, with the concession that they will likely provide less accurate results.

Within this project we would also like to compare the viability of machine learning models within a production environment, in comparison with simpler AI solutions such as rule-based systems.

2.2.4.1 Sentiment Analysis

Beginning with Sentiment Analysis, this is the ability of a computer algorithm to measure sentiment within a piece of text, in this case an article sentence by sentence.

There are a handful of viable options for developing a Sentiment Analysis AI using Python

These are:

NLTK: VADAR: Rule-based

The first choice we considered was to utilise the pretrained bag-of-words style rule-based VADAR model that is included as a part of the python Natural Language Tool Kit (or NLTK).

Not machine learning: As a rule-based model, it makes a valuable comparison point for the validity of machine learning in measuring bias in media.

Slower: NLTK's pretrained VADAR model unfortunately suffers from the issues associated with NLTK, as a library designed more for experimental work, it is relatively slow compared to other options, making live application use impractical.

Wide Options: NLTK has a wide array of potential pretrained models and components to choose from

Better for informal language: NLTK's VADAR model is specifically trained off data sets that utilise informal language such as tweets, it is not as applicable for formal language such as that found in news media

Harder to use: The added flexibility of NLTK does also unfortunately entail a degree of difficulty by increasing the complexity of the program, this makes it harder to develop for in short time frames.

Textblob: VADAR: Rule-Based

Better for formal language: While based on the same VADAR model as NLTK's prebuilt VADAR model, text blob utilises a data set that is more geared towards formalised language. This makes it ideal for our application.

Easier to use: Textblob is extremely easy and simple to use, even easier when utilised through the Spaceyblob pipeline to connect it to the Spacy library toolset

Restrictive: This library is only capable of using, out of the box, this single model. It is difficult for testing but ideal for user facing code.

Ignores words it doesn't know: Textblob, unlike the NLTK model, is designed to ignore words it doesn't recognise. While this may seem like a deficiency it is likely to cause it to ignore names which are often difficult to distinguish from sentiment data. This is ideal for sentiment analysis where names might recur frequently and develop an attached negative sentiment. Which would be extremely counter-intuitive for detecting changes in sentiment.

Textblob: Naïve Bayes Classifier: Machine Learning

Not appreciably more accurate: Experiments with the pretrained Naïve Bayes Classifier available within Textblob revealed it to generally only partially more accurate than a rules-based approach. We suspect there is a greater amount to be gained with informal language.

Much Slower: This model is much slower to run, which is likely to be problematic when it comes to running in an active environment with lots of data.

2.2.4.2 Named Entity Recognition

EntityRecognizer: part of spacy: deep convolution neural network

Faster: Of the available options this one was the fastest by far

Easier to implement: Implementing spacy NER is extremely quick and easy, excellent for testing and development purposes

More accurate: This model has a general higher accuracy than the other available alternatives.

Stanford NER: Convolution neural network

Much slower: Run times with Stanford NER are often several times slower than the alternative provided by Spacy

No more accurate: While the accuracy of Stanford NER is competitive with spacy, it is also not significantly faster, making its negative aspects more significant.

More cautious: Produces generally more conservative estimates that are not as strongly negative or positive. Probably more accurate but also less useful as data displayed for general purpose users.

Developer trained Model

The arguments for and against a developer trained model are functionally the same as for sentiment analysis, so we have included them here at the end.

Too time consuming: Constructing your own model and training it is an extremely time consuming and complex task. While we believe this would be an excellent approach to take in future development on the project, it is unlikely that this could be implemented at this stage.

Unreliable: There is no guarantee that with a developer made model the functionality would be as good as with a pretrained model. It is likely that selecting existing models to compare is more viable in the short term.

Potentially better performance: While it is possible the performance may be worse; it is likely that with sufficient development time and the correct choice of datasets the resulting outputs may be an improvement over the existing models.

Conclusion

From this work we can discern that there are best choice modules for both the sentiment analysis and the named entity recognition. While training a bespoke model is the preferable option for both, due to the constraints of production times it is best that this is only developed in future additions.

For the sentiment analysis it is best to set up and compare both the non-machine learning rules based, as well as the naive bayes classifier, Textblob modules. This provides a good background to test the viability of the machine learning model for this application whilst also providing a fallback if the machine learning techniques prove to be in some way inadequate.

For the Named entity recognition, it is likely only practical to attempt implementation of the Spacy NER model. As an example of machine learning it is ideal, and it is reliably considered to be very performative. Ideally, we would also like to implement the Stanford NER model, but it takes considerably more set up and is not particularly well regarded in comparison to Spacy.

2.2.5 Why Chart.js

For generating the required charts, the primary choice of graphic library was Chart.JS, this was selected based on a few key criteria.

A large, comprehensive documentation, as well as a high user count leading to a large volume of tutorials and troubleshooting utilities.

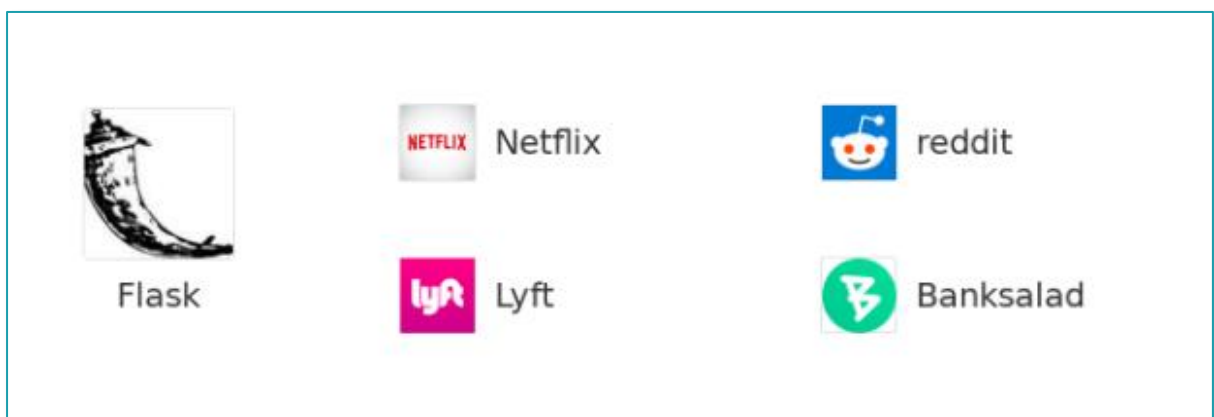
A user-friendly object property-based programming infrastructure to the library.

Wide selection of impressive, attractive animations and appearance controls.

2.2.5.1 Why Flask

As the backend of our web server, we've selected to use the Flask Framework. Flask is a web server framework used to quickly build web servers in an expandable and logical way. It was primarily chosen because

It is extremely widely used, being utilized in multiple high end consumer facing applications see *Figure 2.2.5: Usages of Flask in Major Software Companies*



12 *Figure 2.2.5: Usages of Flask in Major Software Companies*

It also has extremely good documentation, and a broad community making troubleshooting rapid.

It is also particularly easy to use for a web framework, in comparison to Django it is much faster to develop with the trade-off of being slightly more restrictive, which is unlikely to be an issue with the scale of this project.

2.2.5.2 Why Beautiful Soup

There was a degree of internal debate between using Scrapy and Beautiful soup. Scrapy is generally considered to be much easier to use, being a complete web framework, as opposed to BS4 which has multiple external dependencies. However, it is also quite restrictive, and given the peculiar design of the BBC news website, as well as the variety of other websites we want to implement, it is more practical to use a flexible library as opposed to an entire out of the box framework. BS4 is also much more widely used, making troubleshooting easier.

2.2.6 Test Design

Key Terms:

Unit Test (UnTest): Testing a specific component directly to check for functionality integrity

User Test (UsTest): Testing end functionality by sitting in front of the running program and using it.

For testing we chose to utilise a Test Casing methodology.

2.2.6.1 NLP Testing

Table 10: NLP Testing table

Test ID	Requirement Reference	Description	Expected Result	Pass/Partial/Fail
1	KFNN-01	UnTest: Run 5 test articles to compare for accuracy	Average accuracy 70%+	
2	SENN-01	UnTest: Run 5 test articles to compare for accuracy	Average accuracy 70%+	
3	SENN-02	UnTest: Attempt to average sentiment of all key figure occurrence	Accurate Average	
4	SENN-03	UnTest: Run sentiment analysis again after retraining	Improved Accuracy	

2.2.6.2 Web Scraping Testing

Table 11: Web Scraping Testing Table

Test ID	Requirement Reference	Description	Expected Result	Pass/Partial/Fail
5	WC-01	UnTest: pull article from BBC website	Html stored in variable	
6	WC-02	UnTest: Article must be stored as a normalised, cleaned list of sentences.	List of sentences generated	
7	WC-05	UnTest: Database must contain data from multiple sourced	Multiple Tables Filled	
8	WC-06	UnTest: Must return valid search result and load into DB	DB Contains new rows	

2.2.6.2 Core Process Testing

Table 12: Core Process Testing Table

Test ID	Requirement Reference	Description	Expected Result	Pass/Partial/Fail
9	DB-01	UnTest: Check for finished row in DB	Row is constructed properly	
10	DB-03	UnTest: Attempt SQL Injection	SQL injection Fails	
11	WC-04	UnTest: Check for entire days articles	No Articles Missed	
12	DB-02	UnTest: Check Daily File size	File size grows under 1mb	
13	DB-04	UnTest: Check Table is Sorted	Table should be alphabetical	

2.2.6.3 Front End Testing

Table 13: Front End Testing Table

Test ID	Requirement Reference	Description	Expected Result	Pass/Partial/Fail
14	DWS-01	UsTest: Ask user to enter details and submit	Should Return valid chart	
15	DWS-03	UsTest: Ask used to change background colour	Background should change	
16	WC-03	UnTest: Feed WebCrawler Unusual Formats	Crawler should discard data	
17	DWS-02	UsTest: consider if the program is easy to use	Should be relatively effortless to understand	
18	DWS-04	UnTest: Load Multiple Key Figures	Two lines should appear	
19	DWS-05	UsTest: Load Multiple Sources	Two lines should appear	
20	DWS-21	UsTest: Load Specific Timeframe	Dates should be within specific timeframe	
21	DWS-06	UsTest: Statement must be present	Statement is present	

22	DWS-10	UnTest: Watch uploads to check articles aren't double updated	No double results should appear in database	
23	DWS-17	UnTest: Load multiple graph types	Multiple graph types should be present	
24	DWS-18	UnTest: Export as image	Image should be downloaded	
25	DWS-19	UnTest: Export as .txt	Text format table should be downloaded	
26	DWS-20	UsTest: Debate aesthetic appraisal	Should be subjectively attractive	

Chapter 3: Implementation

This chapter details the process of implementing the project and the various challenges and testing that was involved in its conception.

In assistance to understanding this chapter, the author makes frequent reference to the module names under the the Environment Component Diagram on page 22

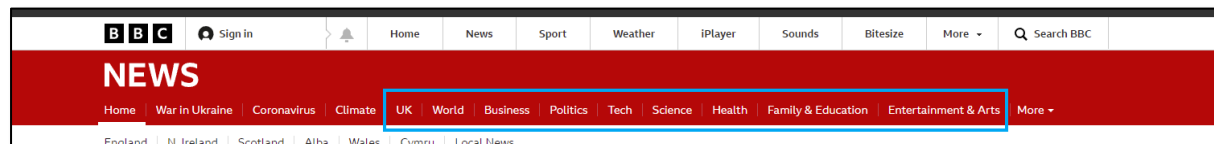
3.1 Understanding the BBC News Website.

The first major challenge the author initially had with this project, was how to programmatically process a comprehensive collection of recently updated articles.

Accessing individual pages is trivial but there is no comprehensive list on the BBC news website of all of their articles, chronologically, and their search engine both lacks that functionality and is disallowed by their web scraping policies (BBC, n/a) likely to reduce excessive impact on their servers.

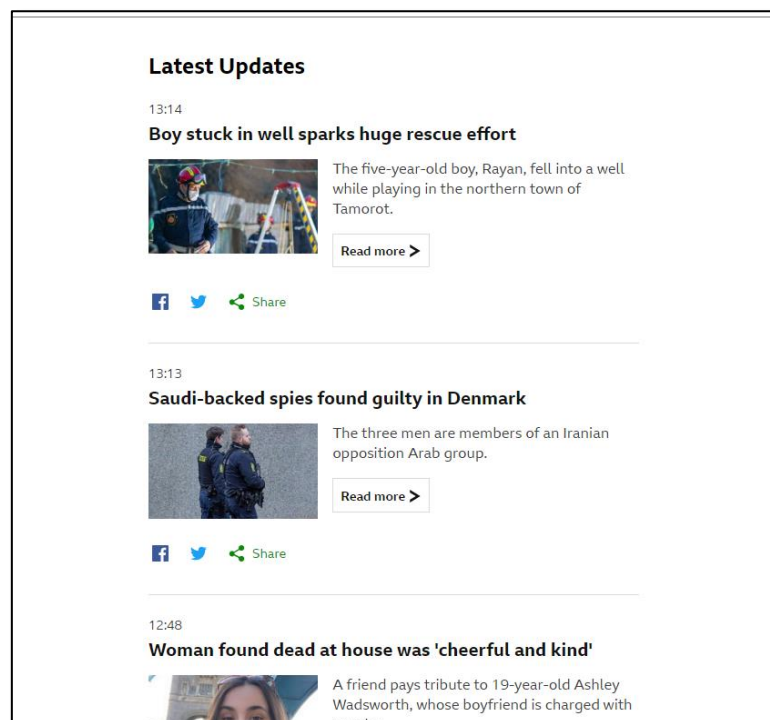
At first viewing, the BBC News page is quite complex, with a variety of sub-headings and sub-sub-headings. However, it follows a few simple rules.

All new articles are posted within the headings highlighted in blue, headings to the left, right and below the blue box are all sub-headings pertaining to specific tagged topics within those highlighted categories. *See Figure 3.1.1: BBC News Header*



13 Figure 3.1.1: BBC News Header

Within each of these parent headings exists a “latest updates” section which contains a .react component showing the most recently updated news articles for that parent URL. *See Figure 3.1.2 BBC News Latest Update Section*



14 Figure 3.1.2 BBC News Latest Update Section

Now that we understand how the website was constructed, we can direct our web scraper to the appropriate part of the sight to collect data. See Figure 3.1.3: Checking for New Articles

```
Checking for new articles now! :D
https://www.bbc.co.uk/news/uk
[]
https://www.bbc.co.uk/news/world
[]
https://www.bbc.co.uk/news/business
[]
https://www.bbc.co.uk/news/politics
[]
https://www.bbc.co.uk/news/technology
[]
https://www.bbc.co.uk/news/science_and_environment
[]
https://www.bbc.co.uk/news/health
[]
https://www.bbc.co.uk/news/education
[]
https://www.bbc.co.uk/news/entertainment_and_arts
[]

Checking for new articles now! :D
https://www.bbc.co.uk/news/uk
['https://www.bbc.co.uk/news/uk-61105169', 'https://www.bbc.co.uk/news/uk-wales-61106879']
https://www.bbc.co.uk/news/uk-61105169
[['Boris Johnson', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/uk-61105169'], ['African', -0.04, '14/04/2022', 'https://www.bbc.co.uk/news/uk-61105169']]
https://www.bbc.co.uk/news/uk-61105169
https://www.bbc.co.uk/news/uk-wales-61106879
[['Kaylea Titford', -0.2, '14/04/2022', 'https://www.bbc.co.uk/news/uk-wales-61106879'], ['Alun Titford', -0.25, '14/04/2022', 'https://www.bbc.co.uk/news/uk-wales-61106879'], ['Mold Crown Court', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/uk-wales-61106879']]
https://www.bbc.co.uk/news/uk-wales-61106879
https://www.bbc.co.uk/news/world
['https://www.bbc.co.uk/news/uk-61105169', 'https://www.bbc.co.uk/news/world-middle-east-61105492', 'https://www.bbc.co.uk/news/world-us-canada-61105168', 'https://www.bbc.co.uk/news/world-europe-61105959']
https://www.bbc.co.uk/news/uk-61105169
[['Boris Johnson', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/uk-61105169'], ['African', -0.04, '14/04/2022', 'https://www.bbc.co.uk/news/uk-61105169']]
https://www.bbc.co.uk/news/uk-61105169
https://www.bbc.co.uk/news/world-middle-east-61105492
[['Palestinians', 0.009999999999999998, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['the West Bank', -0.17, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Israeli', 0.009999999999999998, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Naftali Bennett', 0.25, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Palestinian', -0.07, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Reuters', 0.03, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Hamas', -0.05, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Jewish', 0.09, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492'], ['Jews', 0.03, '14/04/2022', 'https://www.bbc.co.uk/news/world-middle-east-61105492']]
https://www.bbc.co.uk/news/world-middle-east-61105492
https://www.bbc.co.uk/news/world-us-canada-61105168
[['Michigan', -0.17, '14/04/2022', 'https://www.bbc.co.uk/news/world-us-canada-61105168'], ['Patrick Lyoya', 0.5, '14/04/2022', 'https://www.bbc.co.uk/news/world-us-canada-61105168'], ['Lyoya', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/world-us-canada-61105168'], ['Grand Rapids Police Department', 0.25, '14/04/2022', 'https://www.bbc.co.uk/news/world-us-canada-61105168']]
https://www.bbc.co.uk/news/world-us-canada-61105168
https://www.bbc.co.uk/news/world-europe-61105959
[['Marine Le Pen', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/world-europe-61105959'], ['Le Pen', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/world-europe-61105959']]
https://www.bbc.co.uk/news/world-europe-61105959
https://www.bbc.co.uk/news/business
[]
https://www.bbc.co.uk/news/politics
['https://www.bbc.co.uk/news/uk-61105169']
https://www.bbc.co.uk/news/uk-61105169
[['Boris Johnson', 0.0, '14/04/2022', 'https://www.bbc.co.uk/news/uk-61105169'], ['African', -0.04, '14/04/2022', 'https://www.bbc.co.uk/news/uk-61105169']]
https://www.bbc.co.uk/news/uk-61105169
https://www.bbc.co.uk/news/technology
[]
https://www.bbc.co.uk/news/science_and_environment
['https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643']
https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643
[['the Berks, Bucks and Oxon Wildlife Trust', 0.01, '14/04/2022', 'https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643'], ['Oxfordshire hire', 0.01, '14/04/2022', 'https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643'], ['Colin Williams', 0.68, '14/04/2022', 'https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643'], ['BBC South', 0.01, '14/04/2022', 'https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643']]
https://www.bbc.co.uk/news/uk-england-oxfordshire-61103643
https://www.bbc.co.uk/news/health
```

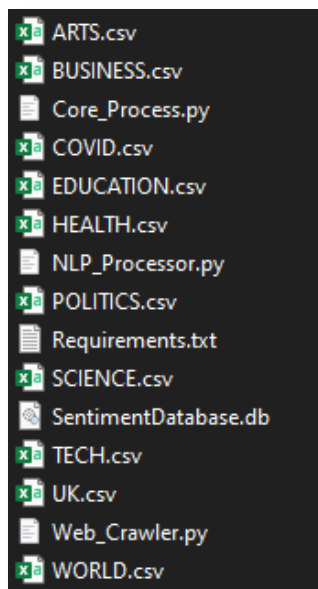
15 Figure 3.1.3: Checking for New Articles

3.2 Web_Crawler

The Web_Crawler proved a surprisingly challenging component to build. Prior to this project the author had never used or studied web crawling techniques.

Because of this lack of experience, we ran into numerous roadblocks during development. The first was trying to extract the article URLs from the “Newest Update” section. This section is a React.js object, meaning the proceeding pages are not easily extractable as hyperlinks in the same way the parent headings were. Only the top 20 most recently updated articles were readily accessible. While it is possible to access react contents with some web scraping libraries, it is extremely difficult to implement with BS4.

This was problematic because articles would occasionally pass out of the first page of results, and then be edited, and then reappear at the top of the list. This regularly caused articles to be processed multiple times.

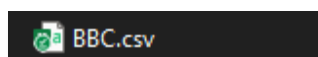


As a preferable alternative solution, we instead implemented a series of backlog files for each parent heading as single line CSV files, chosen because single line CSV files easily convert to Python lists. *See Figure 3.2.1: Multiple Update Backlogs*

Unfortunately, it became apparent during testing that, while all articles are posted under the parent headings, they are not always exclusively posted to one of them. This meant the original approach of saving to multiple discrete CSV files was itself causing certain articles to be processed multiple times.

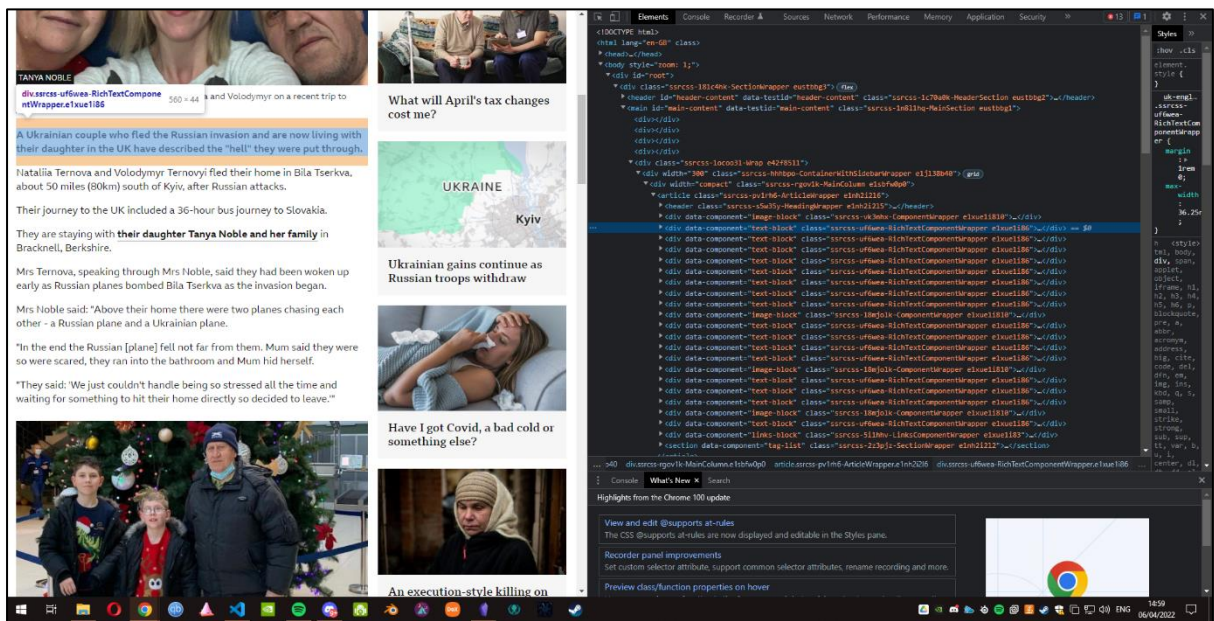
This had to be replaced with a single CSV containing a backlog of all parent headings, which contains 400 entries, or slightly over two pages worth of articles for each parent heading. *See Figure 3.2.2: Single Updates Backlog*

16 Figure 3.2.1: Multiple Update Backlogs

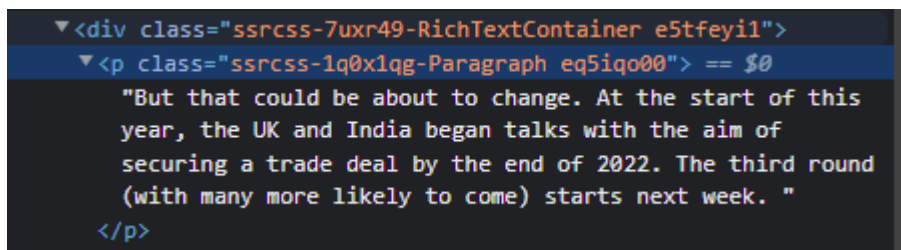


17 Figure 3.2.2: Single Updates Backlog

The final issue encountered was how to select out the desired text. It eventually became apparent that only sections tagged like so contain actual body text. See Figure 3.2.3: BBC HTML View & Figure 3.2.4: BBC Text Body Class



18 Figure 3.2.3: BBC HTML View



19 Figure 3.2.4: BBC Text Body Class

This could be singled out using what would turn out to be a surprisingly opaque series of commands see Figure 3.2.5: BS4 Body Search Code

```
for link in doc.find_all("p", class_="ssrcss-1q0x1qg-Paragraph eq5iqo00"):
```

20 Figure 3.2.5: BS4 Body Search Code

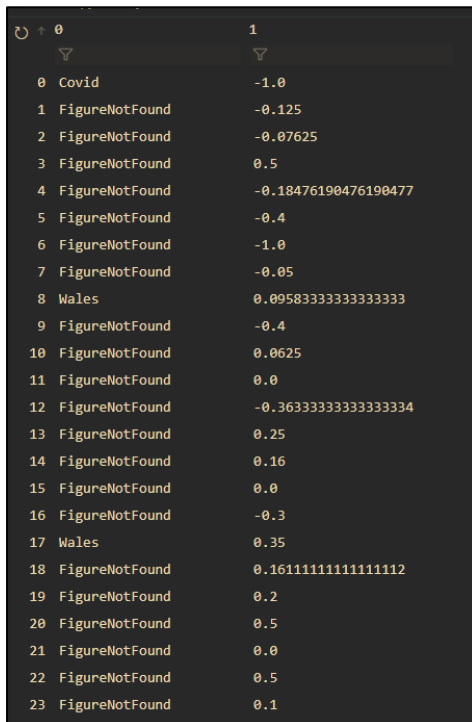
Despite this there still needed to be a surprising amount of clean up on the data. Removing certain key phrases see Figure 3.2.6: Content Cleaning Code

```
website_content_cleaned = website_content.replace("This video can not be played", "")
```

21 Figure 3.2.6: Content Cleaning Code

As well as formatting the data to make it easier to work with later, removing whitespaces, and putting it into lower case.

3.3 Post Processing of the Data



	0	1
0	Covid	-1.0
1	FigureNotFound	-0.125
2	FigureNotFound	-0.07625
3	FigureNotFound	0.5
4	FigureNotFound	-0.18476190476190477
5	FigureNotFound	-0.4
6	FigureNotFound	-1.0
7	FigureNotFound	-0.05
8	Wales	0.09583333333333333
9	FigureNotFound	-0.4
10	FigureNotFound	0.0625
11	FigureNotFound	0.0
12	FigureNotFound	-0.36333333333333334
13	FigureNotFound	0.25
14	FigureNotFound	0.16
15	FigureNotFound	0.0
16	FigureNotFound	-0.3
17	Wales	0.35
18	FigureNotFound	0.16111111111111112
19	FigureNotFound	0.2
20	FigureNotFound	0.5
21	FigureNotFound	0.0
22	FigureNotFound	0.5
23	FigureNotFound	0.1

After the Web_Crawler has provided the necessary data to process, and the NLP_Processor has generated sentiment and key figure values for all of it, there remains a big list of every Key Figure and associated sentiment. *See Figure 3.3.1: Pre-Refactoring Article Data*

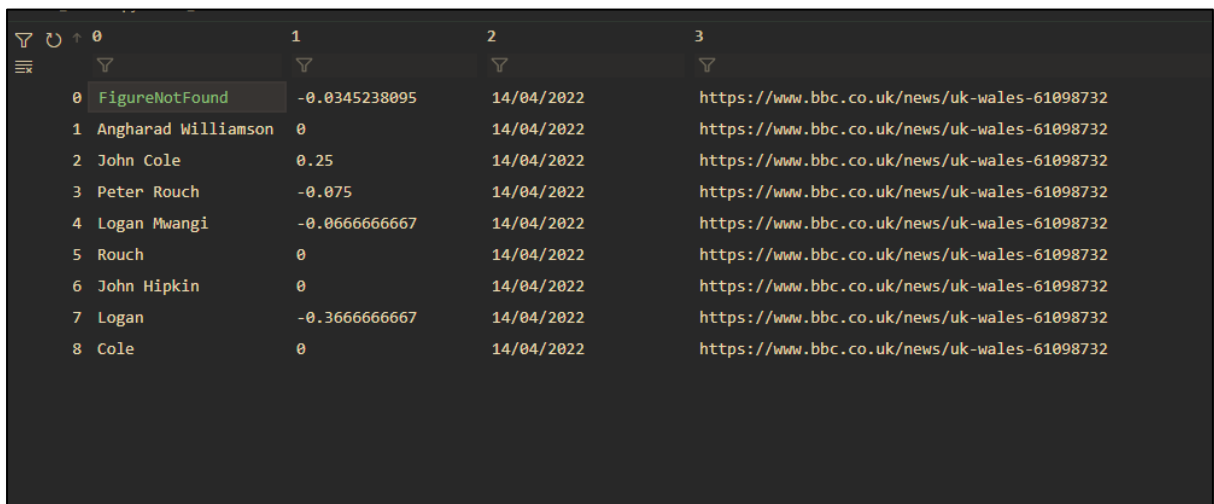
This data then needed to be processed down into something more usable.

The first job was to iterate through the entire table to find duplicates. This is where we began relying on the NumPy library for its excellent dynamic array processing capabilities.

Most of this post processing occurs within the Analysis_Calculator function within the Core_Process module. Here the data goes through a series of key steps.

22 Figure 3.3.1: Pre-Refactoring Article Data

The first step is that the list is loaded into a NumPy array, each occurrence of a particular key figure is combined into a single entry, with their sentiments averaged out. *See Figure 3.3.2 Post-Refactoring Data*



	0	1	2	3
0	FigureNotFound	-0.0345238095	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
1	Angharad Williamson	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
2	John Cole	0.25	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
3	Peter Rouch	-0.075	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
4	Logan Mwangi	-0.0666666667	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
5	Rouch	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
6	John Hipkin	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
7	Logan	-0.3666666667	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
8	Cole	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732

23 Figure 3.3.2 Post-Refactoring Data

Next the FigureNotFound value, which relates to sentences in which no key figure is detected, is removed and its sentiment is assigned as a weight.

The purpose of using this value is that it helps to tackle a common problem in published media. Say that an article is published in which the single appearance of a named entity is perceived as relatively positive, but the rest of the article, which is ostensibly about that named entity, is

extremely negative. This would produce data in which, despite the article being a negative discussion about the key figure, the data would perceive it as positive about the key figure. Thus, taking the generalised sentiments in the article and using it to weight the rest of the values helps combat outlier data. See *Figure 3.3.3: Post Weighting Data*

0	1	2	3
0 Angharad Williamson	-0.0276190476	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
1 John Cole	0.25	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
2 Peter Rouch	-0.075	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
3 Logan Mwangi	-0.0666666667	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
4 Rouch	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
5 John Hipkin	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
6 Logan	-0.3666666667	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
7 Cole	0	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732

24 *Figure 3.3.3: Post Weighting Data*

This unfortunately did introduce the issue, where once it became apparent the text needed to be stored in lower case, the program stopped recognising the now lowercase “figurenotfound” data as weight data and entered incorrect data see *Figure 3.3.4: Incorrectly formatted data*

622	scotland	
623	figurenotfound	
624	jason leitch	
625	figurenotfound	
626	figurenotfound	
627	misty ardouin	
628	wales	

25 *Figure 3.3.4: Incorrectly formatted data*

3.4 Measuring Sentiment.

Initially within the development of this program, we attempted to implement a machine learning model for sentiment analysis. We implemented the Naive Bayes Classifier model pretrained as a component of the Textblob library. However, experiments with the model proved to be impractically slow.

Each individual sentence we attempted to appraise took up to 5 minutes to process, when facing a typical daily workload of 2400 article sentences, this would take approximately 200 hours to complete a days’ worth of articles. This is evidently impractical.

Ultimately this meant that we had to settle for the non-machine learning based NLP processing technique utilised by the default Textblob model, which utilises a Rule-based approach instead.

3.5 Choosing a Named Entity

When designing the Named Entity Recognition system around detecting the key figure in a sentence, it at no point occurred to the author that a single sentence might contain multiple named entities. Unfortunately, when this self-evidently turned out to be true, a system had to be designed to handle this, this system design is justified under *1.2.6 Understanding Data*

The first aspect of this was selecting the type of named entities we were trying to measure. Spacy has out of the box support for a wide variety of labels, the majority of which we do not want. These include

```
['ORG', 'CARDINAL', 'DATE', 'GPE', 'PERSON', 'MONEY', 'PRODUCT', 'TIME', 'PERCENT',  
'WORK_OF_ART', 'QUANTITY', 'NORP', 'LOC', 'EVENT', 'ORDINAL', 'FAC', 'LAW', 'LANGUAGE']
```

As you can imagine, quite a few of these are useless to us. Instead, we produced the following reduced list

```
['ORG', 'GPE', 'PERSON', 'NORP']
```

Or

“Organisation”, “Geopolitical Entity”, “Person”, “Nationalities or religious or political groups”

Our hierarchy is:

- 1) Person: If a person is named, they are invariably the topic of the sentence
- 2) Organisation: Organisations are often labelled as an attribute of a person, but are never the main topic if a person is present i.e., “MRLP party leader Jim Henson”
- 3) Norp: Norp are generally discussed in the context of how a person or organisation has treated or responded to the Norp, hence the topic of the sentence is typically not the Norp if they are not the only aspect discussed. i.e., “Bob Walker was in the news today for denigrating comments aimed towards the Kettle Crisp Association”
- 4) GPE: Geopolitical entities are often cited as an attribute of the Person, Org, or Norp to which they relate to, but are only very rarely the main topic. Hence, they are at the bottom of the hierarchy. i.e., “Professional Lego builder Pepper Roni, resident of Lego Island, was in the news today”

Finally, we needed to devise a methodology to tackle instances where two key figures in the same place in the hierarchy were present. Again, as a general rule it appeared that the key figure in a sentence was usually the first key figure mentioned, so the program defaults to the first named entity who is highest up the list.

Unfortunately, these methodologies, are still reliant on the results of the models used. There existed a strange propensity to mis-label certain data which the author suspects was a result of the training data used being historically and nationally biased, for example the contemporary topic of covid being mislabelled *see Figure 3.3.5: Mislabelled Data 1*

```
The "dreadful" impact of Covid on schools is casting doubt on fairness for exam pupils, headteachers have warned
Key Figure: Covid
Type of Key Figure: GPE
[]
```

26 Figure 3.3.5: Mislabelled Data 1

And the specifically British topic of the GCSE exams being mislabelled see Figure 3.3.6: Mislabelled Data 2

```
A-level and GCSE exams will begin in May after they were cancelled for two years due to the pandemic
Key Figure: GCSE
Type of Key Figure: WORK_OF_ART
```

27 Figure 3.3.6: Mislabelled Data 2

Observing the data being processed and entered did also reveal an interesting issue, certain strings have a huge amount of overlap, for example the “office for national statistics” and “The office of national statistics” see Figure 3.3.7: Fetched Article Data. While we have not implemented a solution for this at this point, in the future changes we discuss a library that would potentially solve see 5.7 Fuzzy Wuzzy

```
https://www.bbc.co.uk/news/uk-51768274
[['office for national statistics', 0.334, '22/04/2022', 'https://www.bbc.co.uk/news/uk-51768274'], ['uk', 0.20400000000000001, '22/04/2022', 'https://www.bbc.co.uk/news/uk-51768274'], ['england', 0.354, '22/04/2022', 'https://www.bbc.co.uk/news/uk-51768274'], ['wales', 0.004, '22/04/2022', 'https://www.bbc.co.uk/news/uk-51768274'], ['scotland', 0.004, '22/04/2022', 'https://www.bbc.co.uk/news/uk-51768274'], ['scotland', 0.004, '22/04/2022', 'https://www.bbc.co.uk/news/uk-51768274']]
Parent URL: https://www.bbc.co.uk/news/education
```

28 Figure 3.3.7: Fetched Article Data

3.6 Sentiment Database/SQLite Database

The production of the database for storing the data was relatively straight forward. While it was practical to generate the table programmatically, as we were experimenting with its design regularly, we instead relied on the software “DB Browser (SQLite)”. *see Figure 3.6.1: Test Data*

Id	KeyFigure	Sentiment	Date	URL
Filter	Filter	Filter	Filter	Filter
1	test	1.0	18/04/2022	n/a
2	test	0.9	19/04/2022	n/a
3	test	0.4	20/04/2022	n/a
4	test	0.1	23/04/2022	n/a
5	test	-0.2	01/05/2022	n/a
6	test	-0.4	02/05/2022	n/a
7	test	0.9	03/05/2022	n/a

29 Figure 3.6.1: Test Data

From the design documents we added an Id tag to help iterate through the data more effectively

However as established under 3.3 *Post Processing of the Data*. A problem did arise with key figure data being mis-formatted as upper and lower case. This meant that for practical purposes this database had to be emptied out and rebuilt late in the lifespan of this project. *See Figure 3.6.2: Real Data*

Id	KeyFigure	Sentiment	Date	URL
Filter	Filter	Filter	Filter	Filter
1	Angharad Williamson	-0.00345238095238095	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
2	John Cole	0.246547619047619	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
3	Peter Rouch	-0.078452380952381	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
4	Logan Mwangi	-0.0701190476190476	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
5	Rouch	-0.00345238095238095	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
6	John Hipkin	-0.00345238095238095	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
7	Logan	-0.370119047619048	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732
8	Cole	-0.00345238095238095	14/04/2022	https://www.bbc.co.uk/news/uk-wales-61098732

30 Figure 3.6.2: Real Data

3.7 Flask Web Server

The development of the flask server was hampered slightly by a lack of experience with the tools. However, when dealing with the Chart.JS library, a significant number of difficulties arose.

There were two primary issues with the Chart.JS library that caused significant delays in the production of the last part of this project.

The first, and most significant, was related to the versioning of Chart.js. The 2.9 update in 2019 that was pushed to the Chart.JS library entailed a complete overhaul of the class interface used to interact with the library. This meant that all tutorials and documentation pointing to the 2.9 or earlier version, were effectively unusable. As it would turn out, this is most of them. This made searching for solutions to problems extremely difficult. In addition to this the documentation for Chart.JS, while initially appearing relatively comprehensive, has some major flaws such as an

```
datasets: [  
  {  
    label: 'Dataset 1',  
    data: Utils.numbers(NUMBER_CFG),
```

insistence on using backend data generation to fill variables for data, making it needlessly difficult to discern the format the data is supposed to follow. See Figure 3.7.1: Chart.JS Documentation

31 Figure 3.7.1: Chart.JS Documentation

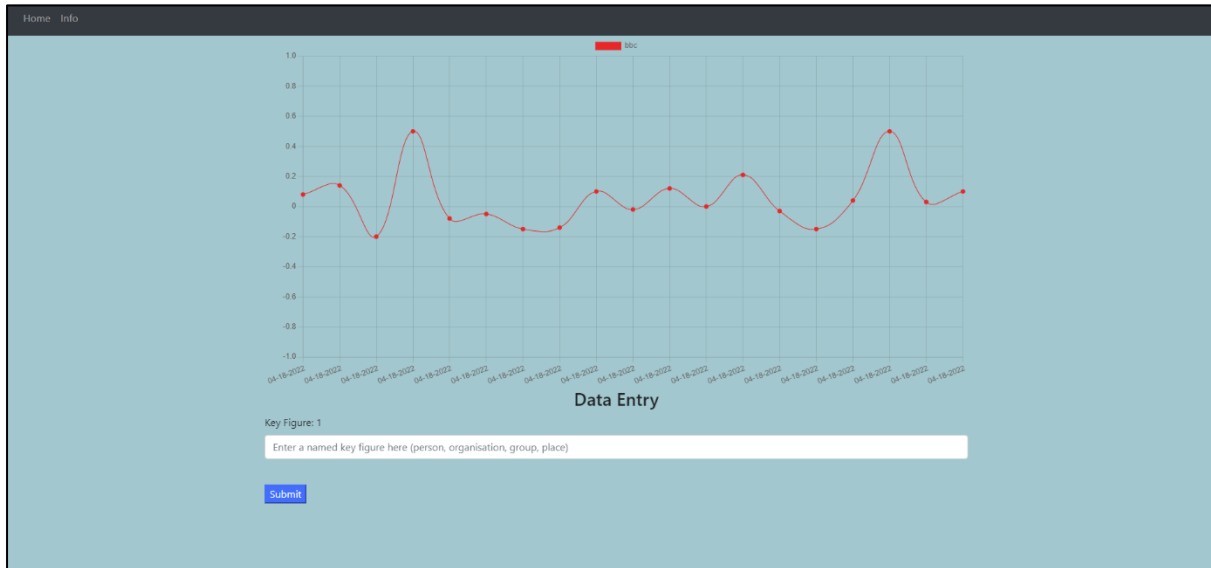
Even reaching this simple point, took multiple days of work. See Figure 3.7.2: Early Graph Attempt



32 Figure 3.7.2: Early Graph Attempt

Chart.js also turned out to be more limited in ability than originally anticipated.

While the program has excellent support for data which shares an identical horizontal axis, if the data has two differing horizontal axis it is unable to combine them. All Labels must have data on all lines. This prevented implementation of the intended functionality of comparing two key figures



33 Figure 3.7.3: Graph with All Data

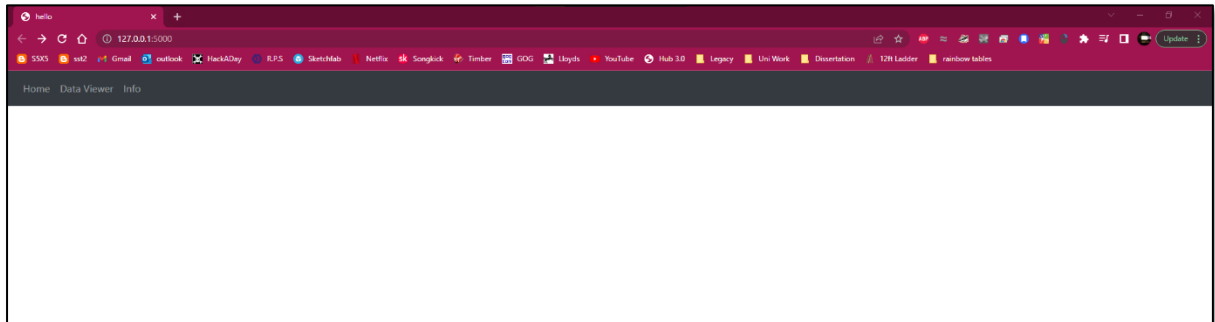
Finally, it was frustrating to find the library selected did not have out of the box support for Date values. Instead, it relied on one of three separately connected adapters, one of which was deprecated, another mis-documented, and the third unable to support day/month/year format time objects. This unfortunately makes cleanly displaying date data somewhat difficult, so we had to rely on simply using the date range as a label. *See Figure 3.7.3: Graph with All Data*

Ultimately, we settled on having the daily values averaged as there was no way to dynamically collapse between daily data average and all values

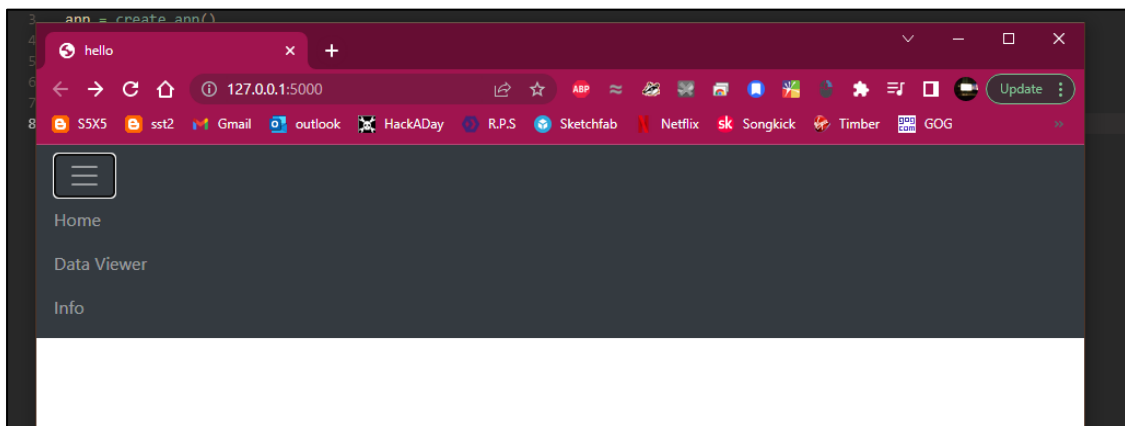
3.8 Base_HTML

Implementing the basic overarching design of the web server proved to be interesting

For purely practical reasons we utilised bootstrap to make sure it was fully reactive (including the graphs) to make the project usable on both mobile and desktop devices. *see Figure 3.8.1: Bootstrap Navbar Wide & Figure 3.8.2: Bootstrap Navbar reactive*



34 Figure 3.8.1: Bootstrap Navbar Wide



35 Figure 3.8.2: Bootstrap Nav Bar Reactive

3.9 Testing Results

NLP Testing

Table 14 NLP Testing Results

Test ID	Requirement Reference	Result	Pass/Partial/Fail
1	KFNN-01	Accurate to 73%	Pass
2	SENN-01	Accurate to 65%	Partial
3	SENN-02	Successfully averaged Result	Pass
4	SENN-03	not implemented	Fail

Web Scraping Testing

Table 15 Web Scraping Testing Results

Test ID	Requirement Reference	Result	Pass/Partial/Fail
5	WC-01	loaded data into bs4 data object	Pass
6	WC-02	filled array with processed data	Pass
7	WC-05	Not Implemented	Fail
8	WC-06	Not Implemented	Fail

Core Process Testing

Table 16 Core Process Testing Results

Test ID	Requirement Reference	Result	Pass/Partial/Fail
9	DB-01	Row Present	Pass
10	DB-03	;DropTables attack succeeded	Fail
11	WC-04	Articles Present	Pass
12	DB-02	File size Small	Pass
13	DB-04	Sorting breaks other functionality	Partial

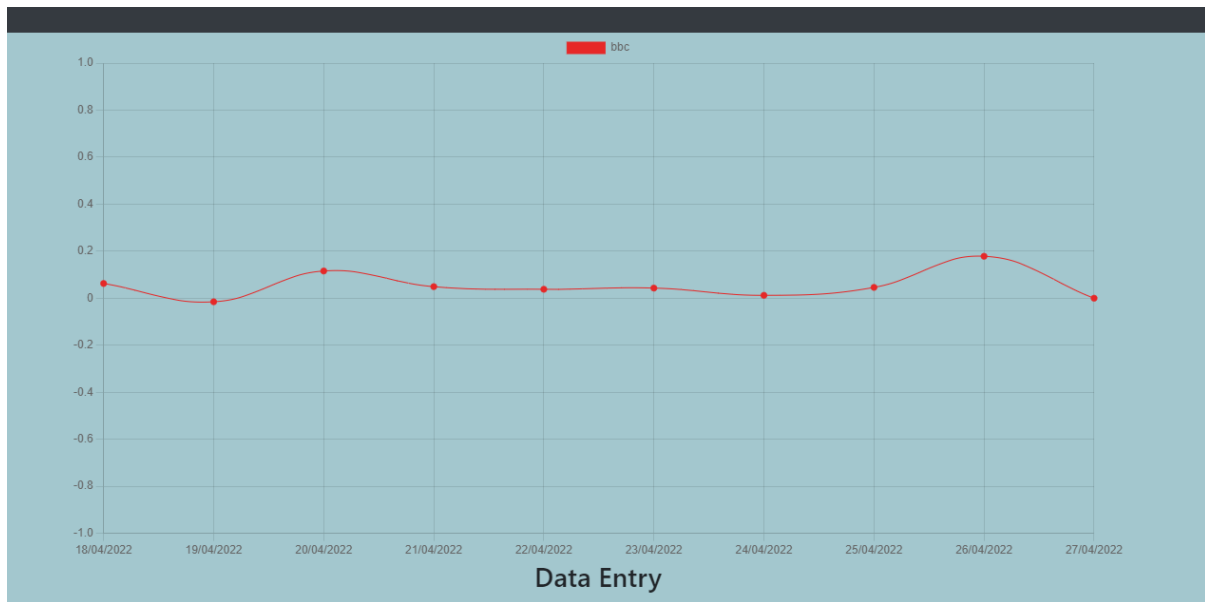
Front End Testing

Table 17 Front End Testing Results

Test ID	Requirement Reference	Result	Pass/Partial/Fail
14	DWS-01	Graph Displayed	Pass
15	DWS-03	Not Implemented	Fail
16	WC-03	Ignored anything outside correct class tag	Pass
17	DWS-02	User stated they did	Pass
18	DWS-04	Not Implemented	Fail
19	DWS-05	Not Implemented	Fail
20	DWS-21	Not Implemented	Fail
21	DWS-06	Is Present	Pass
22	DWS-10	Edited Articles are not double updated	Pass
23	DWS-17	Not implemented	Fail
24	DWS-18	Not Implemented	Fail
25	DWS-19	Not Implemented	Fail
26	DWS-20	CSS and bootstrap attractive, but could be more	Partial

Chapter 4: Project Evaluation

This section details an assessment of the final product and how well it achieved its goals



36 Figure 4.1.1: Final Website

Overall, the author of this project feels the product produced a moderate, but extremely limited success. While the core functionality of the project was realised as an effective proof of concept, there was a large amount of unrealised potential.

4.1 Functional Limitations

As we can see from our test results, a significant portion of the final functionality has not been implemented. The majority of this was due to limitations of Chart.JS, which in retrospect we would not have utilised for this project. Its requirement that there be data on every chart label for every data set created serious problems with the project which was designed around presenting two datasets with different time frames side by side, because of this the project had to be scaled back to showing a single key figure. This was ultimately the cause of failing test 18

Additionally, Chart.JS reliance on external adapters to enable date functionality proved quite limiting, as two of the three date adapters were deprecated, and the last somewhat difficult to use and limited to American style MM/DD/YYYY formatting styles. Ultimately the date formats were left as simple strings without any date processing functionality. This meant that it was impractical to show all date data in a scalable way, meaning that the program had to rely on just showing daily averages. This caused the test 20 Failure

4.2 Research

Overall, we feel the research was relatively good, it successfully identified the problem, identified a further area of study, narrowed down a methodology, and provided a solid basis for design. In retrospect we do wish further study had been done into the minutia of machine learning design, as a custom model would have been a better proposition.

4.3 Design

Generally, we feel the design aspect was good, however we cannot be too confident in its quality as this is also where most of the core difficulties with this project began to arise. Many of these issues could also arguably be blamed on poor research into the limitations of the tools.

In further work, we would like to do a better job of breaking down the functional requirements of each individual tool. From here we would proceed to search their documentation to ensure that they are capable of fulfilling the functionality required, as opposed to assuming they will support what seems like basic functionality which they do not.

4.4 Implementation

Barring problems that occurred as a consequence of failures of design and research, we feel the implementation aspect of the project went relatively well. Production was speedy and had relatively few roadblocks beyond the limitations of the tools libraries and frameworks utilised.

4.5 Time Management

Irrefutably, the largest failure of this project was time management. Both in terms of the scope of the project being unreasonable within the time frame and the failure to make sufficient progress throughout the timeframe provided.

Over this six-month period the majority of the development of this project occurred during the last 3 months, with only preliminary research, design and conceptualisation steps being taken prior. While these preliminary steps were valuable, more time would have enabled the implementation of the various features that weren't implemented.

This time management is the primary cause off the failure to implement multiple key features such as those featured in testing ID's 4, 7, 8, 10, 15, 18, 19, 20, 23, 24, and 25. These features were completely feasible developments, but a lack of sufficient development time made them impractical to implement.

Additionally, we feel that part of this was a by-product of the choice of Agile methodology for development, as agile being an iterative design process is typically bad at measuring out scope.

4.6 Supervisor

Throughout the development of this project, we did not utilise the supervisor enough. The few meetings we did organise were generally insightful and helped a great deal with focussing the project and detecting misinformation and oversights.

In particular the insight from the project in progress day was most useful, as we had during the process of developing the project misunderstood certain natural language processing methodologies to be machine learning when they in fact were not.

Ultimately, we feel that had more of an effort been made to organise meetings with we would have been able to produce a more insightful, complete project.

4.7 Aims and objectives

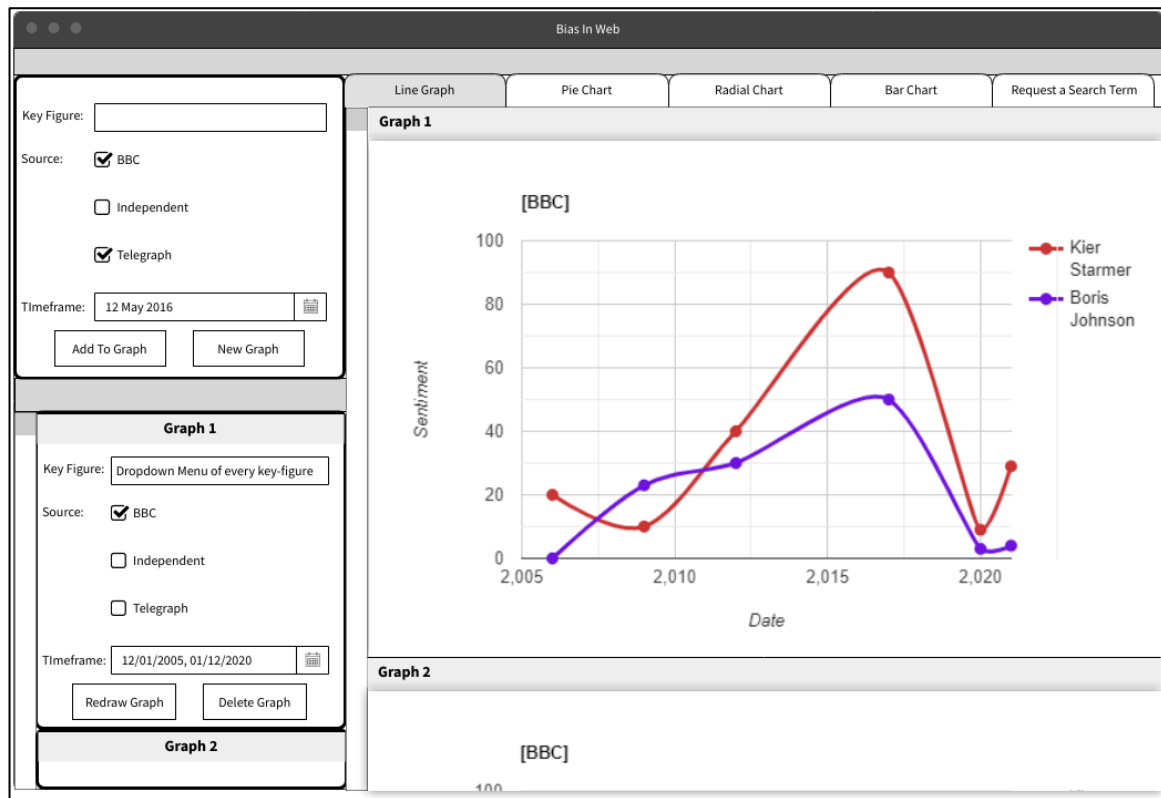
Overall, the aims and objectives laid out at the start of this project were only partially realised. While the core functionality was achieved, we feel that a greater volume of them could have been achieved within the time frame provided.

Chapter 5: Further Work and Conclusions

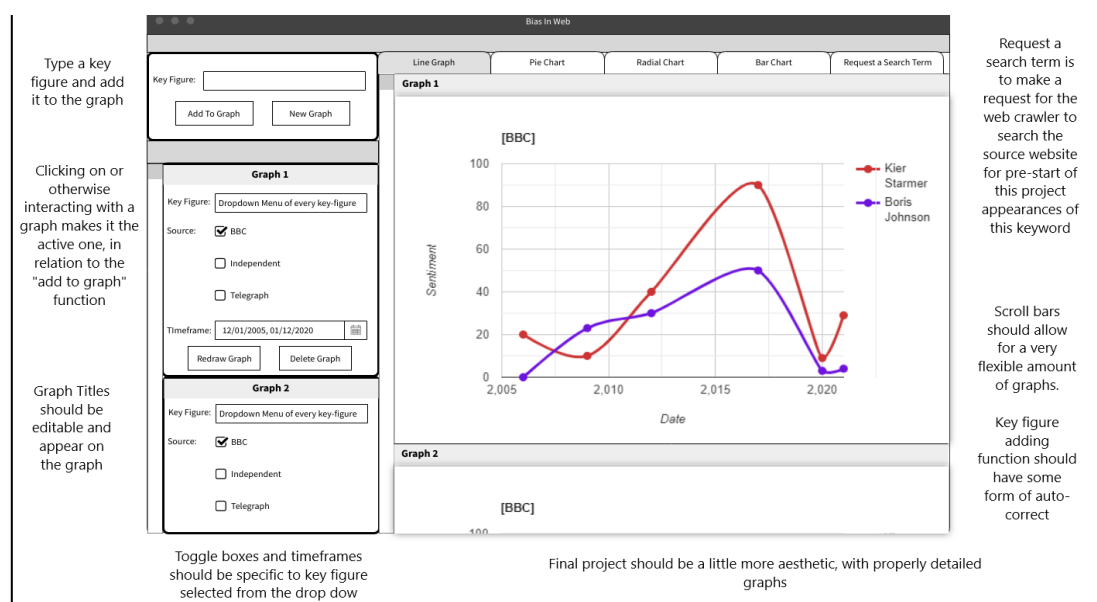
When discussing the unrealised scope of this project, there is a significant volume of further work and additions that should be introduced during further development.

5.1 Ideal Whitebox Designs

This is an ideal Whitebox design, which includes all features across the entire MoSCoW feature list. While it is unlikely that this will be achievable within the available timeframe it does help to define a future target. Ideally with further work we would like the project to approach this style. See *Figure 5.1.1: Ideal Whitebox Design* & *Figure 5.1.2: Ideal Whitebox Design Labelled*



37 Figure 5.1.1: Ideal Whitebox Design



38 Figure 5.1.2: Ideal Whitebox Design Labelled

5.2 Graph Library

Throughout development Chart.JS proved to be the main limitation, below is an image showing an earlier attempt which unfortunately had to be scaled back. Ideally, we would pivot to a different graphing library.



39 Figure 5.2.1: Two Key Figure Graph

Further research suggests that the Google Chart API might be a preferable alternative, a more difficult to use framework but significantly more flexible and with the functionality for multiple datasets the project requires.

5.3 Web Crawling

A core addition that would improve the project significantly, would be the introduction of support for multiple websites. This would require bespoke web crawling functionality for all websites as well as modifications to the database to build each websites data into its own discrete table

5.4 Data Refactoring

The original design of the project didn't consider date format rules within SQL which follow a YYYY/MM/DD format. Refactoring the format would permit functionality within the software that couldn't be implemented.

5.5 Bespoke Model

While the general accuracy of the results of the pretrained models used was acceptable, there is sufficient evidence a custom trained model using either available datasets or a bespoke dataset would be able to generate more accurate results, to suggest that building a bespoke model would be a better choice.

There do exist multiple pre-produced data sets of news media that could be used to train a bespoke model, so one of these would likely be a better choice.

5.6 Multiple Metrics

From the research taken into existing attempts to discern a metric for bias in media, it was evident that there already existed a few interesting attempts to realise this concept. *See 1.2.2 Attempts to Appraise Bias*. We feel that it would be valuable to integrate these approaches into ours, to provide multiple metrics to the user, to make a more valuable tool.

Some of the metrics we would like to include are.

Objectivity: Textblob already has an existing ability to measure subjectivity, it would be valuable to introduce this as a metric as typically a writer being more or less objective with certain key figures is a sign of bias

Word choice: Often, the choice of words used to describe someone, or something is an indicator of bias. It may be possible to recognise synonyms of key words and recognise when a writer is electing to use specifically negative words. Implementing this would be a valuable metric to add to the project

Frequency: When a writer speaks about a specific individual more than similar individuals, it generally suggests a degree of bias. It doesn't specifically label it as positive or negative bias, but it does label it as a bias, and could be used as ancillary data to the existing metrics.

5.7 Fuzzy Wuzzy

The program is currently unable to recognise and combine values where named entities are referred to differently depending on context. For example, "Cole" and "Robert Cole" *see Figure 3.3.3: Post Weighting Data*.

This has naturally led to inaccurate data in certain situations. As a solution to this we would like to propose the use of the Fuzzy Wuzzy Library for Python. This library uses Levenshtein Distance to generate a percentage value for how much of two strings overlap. Using this we could consistently, if imperfectly, detect when this has occurred and combine the two strings and their associated values.

5.8 Conclusion

In conclusion, we feel that while this project was a moderate success, there is a significant amount of unrealised potential and lessons to be learnt in virtually all aspects. Every stage of the project had some degree of avoidable failure.

From our core aims and objectives we feel that we have proven the viability of measuring bias using natural language processing AI, by implementing a novel approach that could be utilised in other applications and has research merit. However, we have not proven the validity of custom machine learning models as we had initially desired.

Additionally, we feel that the project ultimately proved the veracity of the methodology for measuring bias via changing sentiment towards key figures, as well as the utility of displaying it in a line graph style format. However, a methodology that utilises multiple forms of measurement would be more useful.

It is the hope of the authors of this project that its development be continued using the lessons learned during this initial development project, and its full potential eventually be realised.

Glossary

The Fourth Estate: A commonly used term referring to the various news media organisations and their pivotal role in government.

Bidirectional Encoder Representations from Transformers: a transformer-based machine learning technique for natural language processing

Long Short Term Memory: an artificial neural network used in the fields of artificial intelligence and deep learning.

Vader: Lexicon and Rule based NLP analyser

Table of Abbreviations

NLP: Natural Language Processing

AI: Artificial Intelligence

BS4: Beautiful Soup 4

IDE: Integrated Development Environment

DB: Database

LSTM: Long Short Term Memory

BERT: Bidirectional Encoder Representations from Transformers

NLTK: Natural Language Tool Kit

TPB: The Bipartisan Press

References / Bibliography

Bibliography

All Sides Media Bias, 2020. "Available From: <http://www.rcn.org.uk/development/learning>, Volume [Accessed 22 December 2010]..

BBC, n/a. *robots.txt*. [Online]
Available at: <https://www.bbc.co.uk/robots.txt>
[Accessed 20 03 2022].

Christian Davenport, C., (2010). *Media Bias, Perspective, and State Repression (The Black Panther Party)*.. 1st ed. New York: Cambridge University Press..

D'Alessio D.D, 2003. An Experimental Examination of Readers' Perceptions of Media Bias. *Journalism & Mass Communication Quarterly*, 80(2), pp. 282-294.

Durrheim, D. Q. M. W. K. a. K. A., (2005). Denying racism: Discursive strategies used by the South African media. *South-north Cultural and Media Studies [online]*, 19(2)), pp. pp. 167-186.

F Hamborg, K. D. B. G., 2019. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries* , 20(1), pp. 395-395.

Hossain, M. D. I. S. T., 2019. *Oracle, MySQL, PostgreSQL, SQLite, SQL Server: Performance based competitive analysis*, BANGLADESH: Daffodil International University .

Kaplan, E. a. D. S., (2007). The Fox News Effect: Media Bias and Voting.. *The Quarterly Journal of Economics [online]*., 122((3)), pp. pp. 1187-1234..

L. I. Tan, W. S. P. K. O. C. a. A. P., 2015. Rule-Based Sentiment Analysis for Financial News. *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1601-1606.

Leavy, S., 2019. Uncovering gender bias in newspaper coverage of Irish politicians using machine learning. *Digital Scholarship in the Humanities*, 34(1), pp. 5-8.

Mohamed, T. H. E. a. M. T., 2011. Discovering relations between noun categories. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 1(1), pp. 1447-1455.

New, B. S. F. R. B. M. J. L. K., 2004. The processing of singular and plural nouns in French and English. *Journal of Memory and Language*, 51(4), pp. 408-423.

Rathje, S. V. B. J. a. L. S., (2021). Out-group Animosity Drives Engagement on Social Media. *Proceedings of the National Academy of Sciences (Pnas) [online]*., 118(26), pp. 2-5.

Rish, I., 2001. August. An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence* , 3(22), pp. 41-46.

S. Albawi, T. A. M. a. S. A.-Z., 2017. Understanding of a convolutional neural network. *International Conference on Engineering and Technology* , 1(1), pp. 1-6.

Sareen, P. a. K. P., 2015. NoSQL Database and its comparison with SQL Database. *International Journal of Computer Science & Communication Networks*, 1(1), pp. 293-298.

Tung, M., 2021. *DIFFBOT*. [Online]

Available at: <https://blog.diffbot.com/a-less-biased-way-to-discern-media-bias-using-knowledge-graph-enhanced-ai/>

[Accessed 01 04 2022].

VanPatten, B., 2004. *Processing Instruction: Theory, Research, and Commentary*. 7th ed. Carelton: Mahwah, NJ: Erlbaum..

VanPatten, B. V., 1993. Grammar Teaching for the Acquisition-Rich Classroom. *foreign language annals*, 26(4), pp. 435-450.

Wang, W., 2019. *The Bipartisan Press*. [Online]

Available at: <https://www.thebipartisanpress.com/politics/calculating-political-bias-and-fighting-partisanship-with-ai/>

[Accessed 06 04 2022].

Appendix A: First Appendix