University of Glasgow | School of Computing Science

Media Sentiment Analyser

Veselin Vasilev

School of Computing Science
Sir Alwyn Williams Building
University of Glasgow
G12 8QQ

Level 4 Project — March 27, 2018

Abstract

Media is how we, as a society, get informed of the news. It is our main source of knowledge about current events in the world. This dependency on media means it has an inherent power over our opinions of events and topics, by choosing what to report on, and how to do it. The goal of the Media Sentiment Analyser project is to develop a website which allows anyone to easily measure and compare how the newspaper media reports on entities, based on targeted sentiment analysis. A market and products analysis with 109 participants indicated that people think they would be able to form a better judgement of what they read in the news if they used a tool like this, and that there is a market gap for it. The website was evaluated in a usability study and the results indicate that it is highly usable and brings value to its users. These findings support the argument that the Media Sentiment Analyser can fill the identified market gap.

Acknowledgements

I would like to thank my supervisor Prof. Iadh Ounis for his continued support during the project. I would also like to extend my gratitude to Dr. Craig Macdonald, whose advices were crucial for this project.

Education Use Consent

I hereby give my permission for this project to be shown to other University of Glasgow students and to be distributed in an electronic format. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Name: Veselin Vasilev Signature: Veselin Vasilev

Contents

Chapter 1

Introduction

This chapter introduces the goals of the Media Sentiment Analyser project, and the motivations behind it. The Media Sentiment Analyser allows anyone with basic computer skills to measure and compare the sentiments mainstream media has towards different entities. Later chapters show that the current products on the market are unsuitable to offer this to a wide audience - they are tailored towards data scientists and engineers, they are expensive, or lack sufficient features. This product tries to breach these gaps and be useful to the average person.

1.1 Motivations

News is how we, as a society, share and learn the most important topics and events in our time, it is our main source of knowledge [120]. News is provided through media, and different forms of media have developed over the centuries - from word-of-mouth [108], to government proclamations [54], newspaper publishing [72], radio and television [54], and in modern times - online publishing [54]. With the progress of media, more and more people have access to information and can be informed. However, this also means that the media effects - ”the social, cultural, and psychological impact of communicating via the mass media” [61] - have more power.

Media bias is one of the major ways media can influence people’s opinions. Research has shown that media bias does exist, and it can be measured in different ways - by comparing the times that a particular media outlet cites various think tanks and policy groups to the times members of Congress do it [76], or by comparing quoting patterns of newspaper outlets [113], for example. It should also not be underestimated - research by Druckman and Parkin [71] presents compelling evidence that editorial slant influences voters’ decisions during elections. This is a rather alarming implication of the influence media bias has, and only stands to show that it is important for people to be able to quantitatively measure it.

Taking into account these findings, we can argue that the media plays an important role in shaping people’s opinions of entities and events. This illustrates the need for a tool which allows people to measure and compare mainstream media’ reporting on different entities. Looking into how people inform themselves, we see that according to research from the Reuters Institute for the Study of Journalism [92], the most popular way for people in the UK to get their news nowadays is online, including social media. This allows us to argue that such a tool would be most useful if it covered the online spectrum of news, since this produces the biggest overlap with how people get informed. One way to produce measurements like this is sentiment analysis.

Sentiment analysis is defined as ”the field of study that analyses people’s opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes” [85]. It is widely used in review-based systems, such as restaurant or film

reviewing [118, 81], or in market analysis and monitoring tools [9, 41]. In terms of the media, sentiment analysis is often used to predict the movement of stock prices and indices [16, 80]. There are a few solutions, however, which focus on the general media. Since news outlets have the ability to influence people's thoughts and feelings, estimating sentiment in newspapers' reporting is crucial to understanding the biases media outlets might have. Such knowledge can be very beneficial to people who want to inform themselves objectively on current topics. Moreover, the current political and social climate has been infested with various claims about media bias, most notably originating from the current president of the United States - Donald Trump [102]. Statements like this make it even more important for everyone to be able to quantitatively analyse and investigate such claims, so that important decisions are not based on emotional responses.

1.2 Aims

The aim of this project is to develop a web-based application, which allows anyone with basic computer skills to review the sentiment of various news outlets towards a user-defined topic, for a variable time period specified by the user. The website offers visualisations of the results of the sentiment analysis. It also allows the user the ability to compare how a single news outlet reports on two different entities, and how two different outlets report on the same entity. The current market scene will be investigated for similar products and it will be determined whether there is a market gap for this project. The software will be evaluated on its usability and its ability to deliver the previously mentioned aims.

1.3 Summary

This chapter introduced the motivations and aims behind the project, and they were linked to existing research on how people inform themselves, and on media bias. The remainder of this dissertation is structured as follows:

- Chapter 2 contains a market and product analysis and an estimate of whether there is a market gap for a product like the Media Sentiment Analyser.
- Chapter 3 builds upon Chapter 2 by introducing several scenarios designed to illustrate some potential users of the system, and a list of functional and non-functional requirements for the product.
- Chapter 4 describes the project's architecture and the system design decisions based on the requirements.
- Chapter 5 describes the approach taken to solve the challenges with sentiment analysis in the news.
- Chapter 6 outlines the implementation of the aforementioned requirements, according to the architecture.
- Chapter 7 outlines the evaluation steps taken to ensure that the product has met its requirements.
- Chapter 8 identifies opportunities for future work, and provides reflections on this project.

Chapter 2

Related Products

A broad investigation of the market shows that there are two main branches of sentiment analysis products. The first one focuses on opinion mining in financial news - trying to predict the movement of stocks on the financial market by analysing financial news articles. The other branch encompasses products which use social media as a data source. This makes sense intuitively - there is a tangible financial gain to better predicting stocks movement, and as a brand, what people say on social media is often very indicative of your public image. Example products from these two branches are introduced in the first two sections: Section 2.1 introduces FinSentS [16] - a product which targets financial news, and Section 2.2 - Sentiment Viz [40] and Crimson Hexagon [9] - products which perform social media analysis and brand analysis respectively. News sentiment analysis is of main interest, where the market is much less crowded, therefore we will see examples of products which offer sentiment analysis of the general media in Section 2.3.

The main strengths of these products will be presented, as well as their weaknesses, with respect to the expected target audience - average people with no particular expertise in data science, media studies, finances or sociology, who simply want to measure and compare how the mainstream media reports on different entities.

2.1 Financial News Analysis

2.1.1 FinSentS

FinSentS is a news analytics information portal. It provides a plethora of features, most notably - real-time sentiment analysis of financial news and various sources related to the stock market. It uses this data to detect market changes and trends, and to offer insights to investors or traders, asset managements companies, hedge funds or financial institutions. Some of its features include multiple visualisations of the results, comparison options between entities, and date range selection for the analysis.

It is a very powerful tool, however, it is aimed at a niche selection of customers with special knowledge and interests in the financial markets. This makes it unsuitable for the expected target audience. Moreover, the only entities it has data for are companies, whose stocks are traded on the financial market.

2.2 Social Media Analysis

2.2.1 Sentiment Viz

Sentiment Viz offers various tweet sentiment visualisations. It has a very simplistic user interface split into three sections - the first produces the visualisations, the second allows the user to type a keyword and perform a query, and the third is an explanation section, which provides tips on how to use the system and information on how the sentiment is estimated. The visualisation section is highly interactive - it allows zooming, clicking and hovering over elements to get more information. Overall, was it not for the fact that this system only works with data from tweets, it would have been very close to what the Media Sentiment Analyser aims to offer. An example result using Sentiment Viz is shown in Figure 2.1.

Figure 2.1: Sentiment Viz sample output

2.2.2 Crimson Hexagon

Crimson Hexagon is a representative of a suite of companies which offer brand analysis and monitoring services to their clients (another very similar example being Netbase [31]). It delivers value to its clients by tracking sentiment towards their brands on social media. This helps to identify customer motivations and industry trends, track market success, and allows for better planning of public relations campaigns.

Crimson Hexagon targets the corporate sector - its clients are companies which have a product or a service to sell, and are willing to pay in order to receive the necessary insights to improve their brand. It is true that a service like this can potentially be used to measure online sentiments towards different entities, but is highly unlikely that the identified target audience will be willing to pay enough for this. Moreover, the analysis performed is on social media, not on newspaper media.

2.3 Newspaper Media Analysis

2.3.1 Signal Media

Signal Media [41] offers media monitoring and market intelligence solutions to its customers. It uses a variety of sources for its insights - radio and broadcast channels, print newspapers, online media, even legislation and regulatory documents. The service is geared towards corporate clients who are looking to manage and improve their brand.

Although the Media Sentiment Analyser does not have such broad-sweeping capabilities, unlike Signal Media, it aims to offer a service which is accessible to anyone, not just companies which have the means to pay for it. In addition, instead of focusing on one particular company or brand, the Media Sentiment Analyser's objective is to be as general as possible - sentiment towards people, organisations and locations from all sorts can be measured and compared.

2.4 AYLIEN News API

The AYLIEN News API [5] allows companies, data scientists and software developers to access thousands of pieces of newspaper content from all over the world. In addition, this content has been analysed using AYLIEN's Text Analysis Engine [48] and the insights produced are also available through the News API. Users can query for sentiment breakdown, search for trending stories, or track social impact. They can specify the newspaper outlets they are interested in, filter by categories or locations, sort results by recency or relevance, or only look for stories with particular sentiment (positive, neutral or negative). It is a fully-featured, very powerful news analysis product.

That being said, the News API is mostly geared towards software developers and data scientists. The best way to interact with it is through the RESTful API it exposes, which returns JSON formatted results. The API is also exposed on a user interface in a Demo page on AYLIEN's website, but it does not offer all possible features - for example, you can only receive insights up to 60 days in the past. It is also impossible to compare how different newspapers reported on the same entity, or how the same newspaper reported on two different entities. The interface is easy to work with and provides useful visualisations, but it is not the main product. It is an advertisement of what the main product can do. Figure 2.2 shows a sample output from the Demo page.

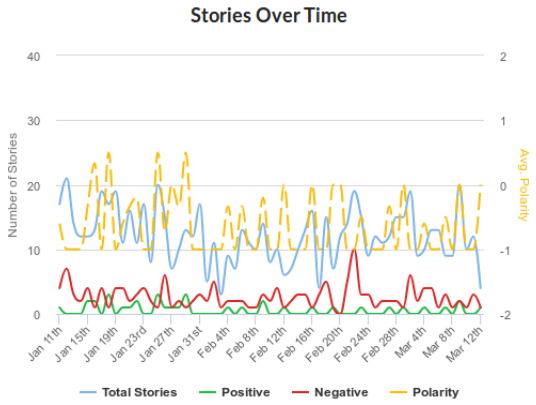


Figure 2.2: AYLIEN News API sample output

It is hard to imagine a regular person using this, mostly because it is geared towards software professionals. In addition, excluding the 14 day trial plan, the cheapest pricing begins at $US79 per month, which is not something an average person looking for a simple tool would be willing to pay.

This is contrasted with the Media Sentiment Analyser's aims, which are to provide the ability to anyone, regardless of expertise or knowledge, to measure and compare the way newspaper media reports on different entities, which could be people, locations, or organisations. Moreover, accessibility being of first-class importance, there are no price restrictions on the system.

2.5 Opinion Crawl

Opinion Crawl [34] provides sentiment analysis for current events, companies, products, and people. It allows its users to specify an entity they are interested in, and the system will produce the current web sentiment for it, calculated using a small number of recent news items. Key concepts related to the entity are also listed, since they can affect the sentiment. An example of the results produced is illustrated in Figure 2.3.

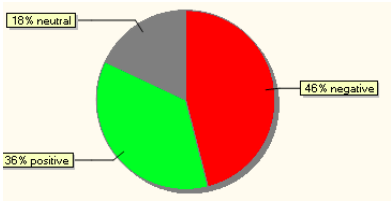


Figure 2.3: Opinion Crawl sample output for Bulgaria

Opinion Crawl also has a blog [33], where popular entities' data is updated daily. It analyses a much larger set of news items and provides more features than the ad-hoc sentiment analysis - for example, results could be produced for a day, a week, or a month. Charts illustrating sentiment trends or positive-to-negative ratio are also produced.

However, Opinion Crawl has several disadvantages. If the user searches for an entity which is not already tracked in their blog, the insights are informative of only the current media sentiment. It is impossible to do historical analysis. Moreover, even the in-depth analysis available in their blog only goes back a month. Another thing to note is that the charts produced are not interactive. The user cannot see which articles were tagged as negative, positive or neutral. Finally, comparison between the sentiment towards different entities, and selection of particular newspaper to analyse are both unavailable as options.

All of these disadvantages are addressed by the Media Sentiment Analyser, which aims to offer historical analysis with data range manipulation opportunity, fine-grained analysis with individual article scores being accessible, as well as sentiment comparisons and newspaper selection.

2.6 Summary

In this chapter, products offering similar features to the Media Sentiment Analyser's goals were investigated. It was seen that while it is definitely possible to analyse the news sentiment towards different entities, the tools available to do this have some faults. The most powerful products listed were either geared towards a specific audience with specific knowledge, or were behind a paywall. The free and accessible ones lacked certain features like historic analysis, sentiment comparisons, and newspaper selection. These shortcomings mean that the insights news sentiment analysis could offer are lost to the common user. This leads to the argument that there is a market gap for a product like the Media Sentiment Analyser, which aims to provide specialised sentiment analysis software to the average person by offering a free service, accessible to anyone with basic computer skills. It also focuses on the identified missing features - the ability to analyse historical data, and to compare how a newspaper reports on two entities, or how two newspapers report on a single entity. The need for a tool like this is further validated with a user study in Chapter 7.

On the basis of these findings, the next chapter identifies the functional and non-functional requirements for the Media Sentiment Analyser.

Chapter 3

Requirements

Most of the requirements were identified during the initial stage of the project. They were later continuously revisited and updated, in response to feedback from the client (the project supervisor). The requirements aim to address the shortcomings of the related products listed in Chapter 2, while incorporating their positives.

3.1 Scenarios

These scenarios were created to help identify the functional and non-functional requirements for the project. They serve as a representation of the imagined end user audience and illustrate the potential tasks it might be interested in performing with a system like the Media Sentiment Analyser.

Joanne is a Politics student in the University of Glasgow. She has an assessment due, in which she has to write an analysis on the media's influence on public opinion before elections. She has found studies and research papers related to this, but she wants to include contemporary examples of media bias in her report. She also wants to be able to pinpoint sentiment analysis on particular days or weeks surrounding past events. Joanne will be happy if she can include visual representations of the results. She has basic computer skills, and is proficient in using the web, so she is looking for a web-based solution she can use without the need to download and install anything.

Gregg is a middle-aged man, who is making his choice about who to vote for in the next General Elections. He has been getting his news primarily from a handful of news sources, so he wants to see if these news sources have a bias towards certain people. He also wants to compare those to other news outlets, in order to find newspapers with a different point of view. He wants to be able to make an informed choice, without being constrained by the opinions of a particular newsgroup. He is not very proficient with computers, so he is looking for a simple to use website, which follows a standard form-based layout.

Patrick is a journalist working for a major newspaper. He is writing an article about the rise of nationalistic movements throughout Europe. He wants to see whether media reporting on entities related to this subject has changed in the recent years, so he can draw conclusions whether this has affected people's thoughts on the matter. He would like to use a sentiment analysis tool he can easily link to in his article, so people can verify his results and do queries on their own. He would also like to be able to access the news articles analysed, so he can quote some of them in his own work. He is proficient with web technologies, and would like to use a website, since it is easy to access from his workstation.

3.2 Functional Requirements

The functional requirements are concerned with the application's features. They are partitioned by significance using the MoSCoW method [96] - "Must Have", "Should Have", "Could Have", and "Won't Have".

3.2.1 Must Have

These requirements are critical for the successful implementation of the project. They provide the building blocks around which extra features can be added.

M.1 Sentiment analysis - a set of newspapers, each with a corpus of analysed articles. This is the underlying feature of the system, since this is where the results will be coming from.

M.2 Web interface - the system will be deployed as a website, so that it is easily accessible. We can see this as a requirement in **Joanne**'s and **Patrick**'s scenarios.

M.3 Entity input - input field for 1 entity, on which the sentiment analysis results will be returned.

M.4 Ability to have more than one entity - for comparing how the same news outlet reports on two different entities. This is a key feature of the Media Sentiment Analyser, which is missing from products like Opinion Crawl and the AYLIEN News API.

M.5 Ability to have more than one newspaper - for comparing how different news outlets report on the same entity. This is the other major feature related to sentiment comparison, and it is missing from Opinion Crawl and the AYLIEN News API.

M.6 Time frame input - the ability to define a date range for the results, with flexibility to choose any date range. Allows for fine-tuning the analysis and for historical comparisons. It is present as a feature in most of the related products, but in the AYLIEN News API demo for example, it is capped at 60 days. The aim to provide access to the entire corpus available.

M.7 Newspaper input - a drop-down or a selection field, which contains the list of the supported news outlets. This provides a typical and easy to use interface, suitable for people like **Gregg**.

M.8 Retrieval of results - functionality to retrieve sentiment results based on the user input.

M.9 Visualisation - after the analysis is finished, a graph will illustrate the sentiment trend over the time frame specified by the user. This is a token feature of all of the related products, since it is the main way to report the results back to the user. It is also identified in the scenario for **Joanne**.

M.10 "How to" section - description of the features the application offers, and how to work with it. This feature is available in the Sentiment Viz website, and it goes a long way to help the users work with the product. Moreover, users like **Gregg**, who are not very proficient with computers, will benefit from it.

3.2.2 Should Have

These requirements are assigned a high level of importance and will be defining features, but they are not necessary for the application to be able to perform its main tasks.

S.1 Ability to click on a particular day and see links to the articles written on this day, along with their respective scores. This feature provides interactivity and better insights to the user. It is missing from the charts produced from Opinion Crawl or the AYLIEN News API demo page.

S.2 Popular entities list. Offered by Opinion Crawl, in order to guess what the user will look for, and to improve the interaction by offering a quick entity selection mechanism.

S.3 Trending entities list. Like **S.2**, offered by Opinion Crawl. Both of them will also help users like **Gregg** to quickly and confidently handle the form.

S.4 Autocomplete feature. Found in the AYLIEN News API demo page and the FinSentS' dashboard, it helps the user with entity selection and makes the entire interaction faster and more pleasant.

3.2.3 Could Have

These requirements are desirable, because they will improve customer satisfaction by enriching the feature set of the application. They are not necessary to be delivered though, and are to be implemented if time permits.

C.1 Selection of news outlets based on country of origin. Offered in the AYLIEN News API demo page, this feature will help users like **Gregg** or **Patrick** - Gregg is interested in forming a better opinion on who to vote for, so he will likely look for newspapers based in his country, and Patrick will be interested in newspapers from specific countries he will include in his article.

C.2 Ability to produce a tabular report over a timeframe, containing all the articles analysed, along with their respective scores. Available in FinSentS, this is a powerful feature which could help people like **Joanne** or **Patrick**.

C.3 Sentiment calendar - visualising the results in a calendar format, with each day coloured accordingly to its sentiment score. Available in FinSentS, this is another form of visualisation which will help the users get better insights from the information, especially if they are interested in small time frames.

C.4 Suggested list of entities, which are currently popular in the news. Very similar to **S.2** and **S.3**, this feature will help users like **Gregg**, since it is likely that around election times there will be a spike of reporting on the entities involved.

3.2.4 Won't Have

These are requirements which were agreed with the client to be out of scope. They could potentially be implemented in a future stage.

W.1 Geolocation analysis and topic suggestion based on it. An expansion of **S.2**, **S.3**, and **S.4**, but perhaps more suitable for a project with a larger scope, with news corpora from many different countries.

W.2 Identification of different users (with cookies or by user registration) and personalised lists of previous searches. Such personalisation is associated with products such as Crimson Hexagon or Signal Media, which cater to specific clients. While definitely useful, it is not a feature which characterises what the Media Sentiment Analyser aims for.

W.3 Small example topics with small graphs next to them, which represent sentiment fluctuations in recent time (a week) for a sample newspaper. Should be clickable - when clicked, will lead to a page where the user will modify the timeframe and the news outlets analysed. This feature is an expansion o **S.2**, **S.3**, and **S.4**. It is a cosmetic feature which would not provide enough benefits to be considered desirable at the current stage.

3.3 Non-Functional Requirements

The non-functional requirements address things like usability and affordance of the application. While these requirements do not describe the behaviours of the application, they are vital for the successful delivery of the aims described in Section 1. As we will see in the following chapters, architectural and implementation decisions were often based on these non-functional requirements.

NF.1 The application should be intuitive and easy to use, even for people with limited computer skills. No specialised knowledge should be required. This requirement satisfies users like **Gregg**.

NF.2 It should be fast to respond to user queries - response times of more than 3 seconds will be considered slow. All of the related products have response times less than 3 seconds, so this is a suitable benchmark.

NF.3 It should produce accurate sentiment results, corresponding to the target entity. As the main feature of the system, care should be taken to make it as accurate as possible.

NF.4 It should have the ability to ingest new data efficiently and without affecting the performance of the live application. All of the examined related products continuously ingest new data and analyse it without affecting the user experience.

NF.5 It should communicate to the user about what it is doing. This will help users with basic computer skill, like **Gregg**, to easily learn how to use the system.

NF.6 It should support concurrent access by different users. Since it will a web-based system (M.2), it it likely that multiple users will access it at the same time, and this should be catered to.

NF.7 It should be maintainable and easily extendable. This is an important requirement for software products, since ”we cannot do maintenance on a system which was not designed for maintenance” [105]. If this project is to be extended and supported in the future, it needs to be built with maintainability in mind.

NF.8 It should be usable on range of devices like computers and mobile phones. Research shows that smartphone usage to read news matches the computer usage in the UK in 2017 [92]. This statistic means that potentially a lot of people will use their phones to access the Media Sentiment Analyser, so it should be usable on mobile platforms as well.

3.4 Summary

In this chapter, the functional and non-functional requirements identified for the project were outlined. In the next three chapters, we are going to discuss the steps taken towards incorporating these requirements into the application.

Chapter 4

Architecture

The architecture of the system is a product of two approaches. The first one is Minimalist Architecture. It favours taking care of the highest priority architectural requirements, and then doing the least possible to achieve them [86]. The other approach is to adhere to the principles of component-based software development [77]. The system was designed to use small, reusable components, which are decoupled. This allowed for easier changes of individual components, without affecting the rest of the system.

This combination of approaches ensured that requirement **NF.7** is adhered to, while at the same time prevented the author from spending too much time on hypothetical solutions to problems which did not exist.

The resulting architecture of the The Media Sentiment Analyser comprises of 7 components:

- Ingestion Component - takes care of ingesting the newspaper articles from various sources.
- Sentiment Analysis Component - performs the sentiment analysis.
- Database Component - the placeholder for the corpus of articles, their metadata, and any other application data.
- Indexing Component - indexes the ingested articles to allow for retrieval of the most relevant ones for a particular entity.
- Search Component - exposes the indices produced by the indexing component.
- Web API Component - handles the business logic of the Media Sentiment Analyser website.
- UI Component - the user interface the client sees and interacts with.

This was the overarching architectural plan for the system since the beginning of the project, but initially the system was designed with the goal of real-time sentiment analysis in mind. Section 4.1 describes this initial design and the problem with it, and Section 4.2 outlines the solution to this problem and the revised system architecture.

In order to satisfy requirement **NF.4**, both of the designs had an offline component for ingestion, which allowed for ingestion and indexing of the corpus without affecting the running application.

4.1 Initial Architecture

4.1.1 Offline Processing

Figure 4.1 illustrates the offline steps in this initial design. We see that after newspaper articles are ingested from various sources, they are indexed. This is done in order to provide the ability to retrieve identifiers for the most relevant articles to an entity.

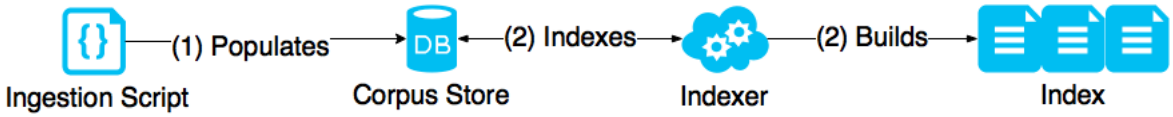


Figure 4.1: Offline - ingestion and indexing of the news articles.

We can identify 3 of the system's components here - the ingestion, the database and the index.

4.1.2 System Architecture

Initially, the system was architected to handle sentiment analysis in real-time, in response to user queries. The system architecture is displayed in Figure 4.2. The rest of the components can be identified in this diagram. We see that the client interacts with the system through the web interface, which makes calls to the web API. In this case, it acts as nothing more than a router for the request to the sentiment analyser, where the actual analysis happens. In order to decide which articles to process, the sentiment analyser requests a list of relevant article identifiers from the search service. The search service has access to the previously produced indices for the articles of each newspaper, so it is able to quickly return the requested list. The sentiment analyser uses this list to query the database for the relevant articles, and then continues to perform sentiment analysis on them. The results are sent back to the user through the web API.

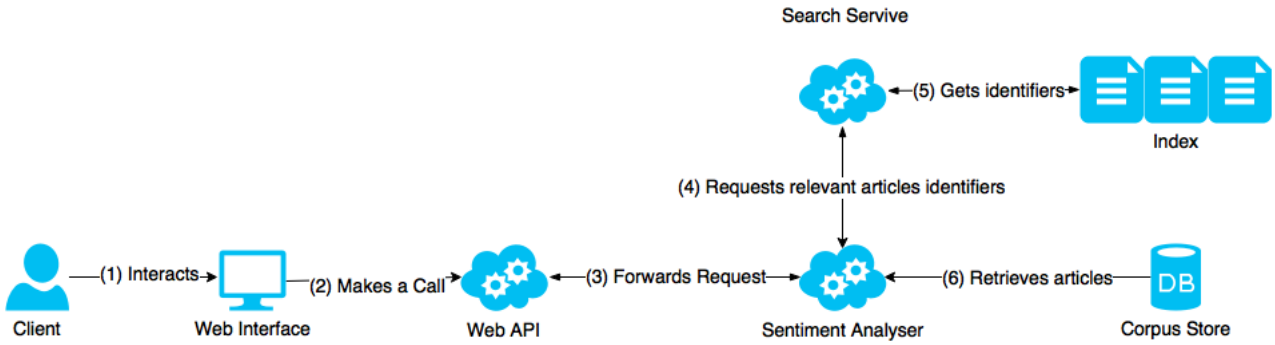


Figure 4.2: Initial system architecture.

There was one major issue with this approach, however - latency. For some entities, which had a large number of relevant documents, it would take up to 5 minutes before the sentiment analysis was done. This was by no means responsive enough for a modern web application, and it also violated requirement **NF.2**. The initial architecture had to be revised to work around this issue.

4.2 Revised Architecture

4.2.1 Offline Processing

Figure 4.3 displays the revised offline processing steps. We see that the sentiment analysis was added to this stage. There were several benefits to this approach. First of all, storage is cheap and plentiful. Storing the extra sentiment scores did not cost anything. Moreover, computing resources are widely available. The sentiment analyser component was easily deployed to an AWS [1] cloud instance, and it could run until it finished analysing the entire corpus. This move to batch processing meant that the sentiment scores towards every entity in every article could be stored in the database, and it would take only a database call to fetch the results for the user. This change in the processing and retrieval model introduced a significant reduction of response times, which satisfied the **NF.2** requirement.

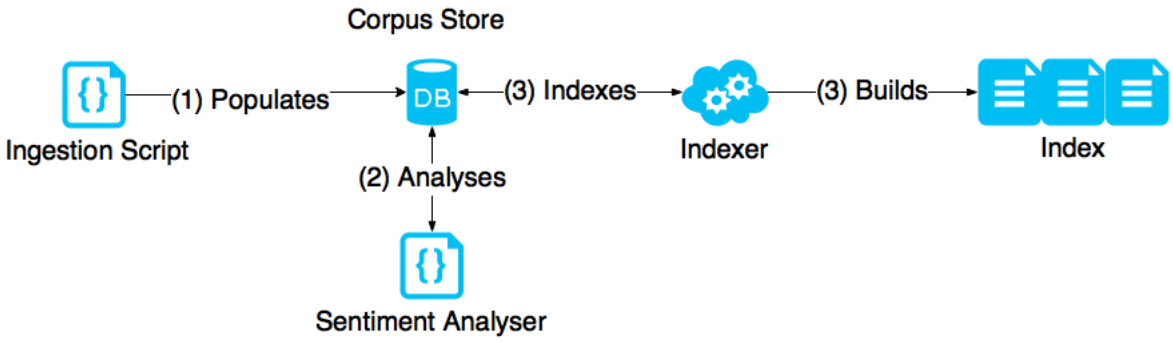


Figure 4.3: Offline - ingestion, sentiment analysis and indexing of the news articles.

4.2.2 System Architecture

Under the previously described changes, the system took a more simplified look. In Figure 4.4, we see that the web API communicates directly with the search service, and uses the list of identifiers of the most relevant documents to retrieve the precomputed sentiment scores from the database.

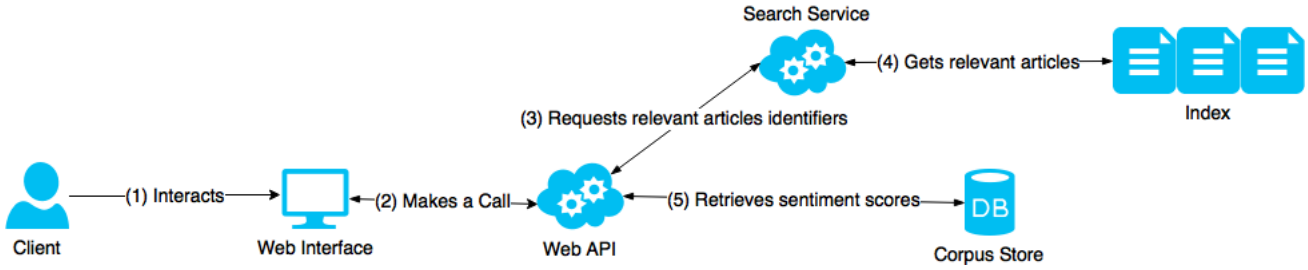


Figure 4.4: Revised system architecture.

4.3 Summary

This chapter presented the architecture of the Media Sentiment Analyser and highlighted how requirements **NF.2**, **NF.4**, and **NF.7** were met.

Chapter 5

Sentiment Analysis

The main challenge this project was presented with was the actual sentiment analysis of newspaper articles. Sentiment analysis is a branch of Natural Language Processing (NLP) which is concerned with the identification of opinions in textual information [79]. Bing Liu defines sentiment analysis as ”the field of study that analyses people’s opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes” [85]. Sentiment analysis is a challenging problem with a significant body of research, with regards to entire documents [53], sentences [119, 94] (including tweets [97, 83]), and phrases [117]. Research has also focused on tasks like aspect sentiment - the identification of opinions towards multiple aspects of the same entity [101, 100].

The particular problem for this project was targeted sentiment analysis - ”the task of identifying the sentiment (positive, negative) or lack thereof (neutral) that a writer holds toward entities mentioned in a statement” [115]. Research on targeted sentiment is still ongoing, with examples including an application-driven task - understanding students’ sentiment towards courses and instructors [115], or target-dependent Twitter sentiment classification [69, 114]. The statements analysed in both of those cases were either sentences or tweets, i.e. short spans of text with an explicit mention of the various targets. Analysing news articles is different for several reasons.

A news article is in most cases a relatively large document, which has multiple target entities. Moreover, it is likely that a paragraph is concerned with one or two entities, but the next paragraphs are concerned with different entities. It is often the case in news that the first several paragraphs are related to the topic of the heading, but the ones following them branch out in relation to other stories or events, or serve as a reminder for past news. This back-and-forth between topics and events requires a careful identification of shorter spans of text (sentences or paragraphs), which are about particular entities, and then doing analysis only on those snippets.

An example of this is the following paragraph, taken from the news piece “Kremlin says accusing Putin of ordering spy attack is ’unforgivable’ ” published in The Guardian [104].

> “Russia has sought to undercut the British argument that the nerve agent used in the Salisbury attack was only available to Moscow. Its envoy to the Organisation for the Prohibition of Chemical Weapons claimed on Friday that the novichok nerve agent used in the attack could have come from the US or UK. The two countries are not known to have produced the nerve agent, but international chemical weapons experts did help dismantle Soviet chemical weapons factories after the collapse of the Soviet Union.”

We can observe that the main entity reported in this paragraph is “Russia”. It was directly followed by this paragraph, which reports on what the Labour leader Jeremy Corbyn commented:

> "Jeremy Corbyn has also said the government should not rush to blame Moscow. Politicians must not "rush way ahead of the evidence being gathered by police," the Labour leader wrote in a column in the Guardian on Friday. He said that Russian "mafia-like groups" could be responsible, rather than the Kremlin."

It is evident that these two paragraphs target different entities, even though the main topic might be the same. If sentiment analysis was performed on the entire article, it is likely that it would not be accurate with respect to all the entities in it, some of which might appear in limited parts of the text. Another important observation is the different ways the author used to refer to Jeremy Corbyn. After using his full name, the second time he was referred to as "the Labour leader", and the third time - simply as ''he". This presents yet another challenge for estimating sentiment in a news article (or in large pieces of text) - the same entity might be referred to in many different ways.

5.1 Algorithm

In order to tackle these problems, Named Entity Recognition (NER), Coreference Resolution (CR), and Word Embeddings are used. NER allows recognition and labeling of entities (Person, Organisation, Location) [110]. CR finds mentions of the same entity in a text, such as when "Theresa May" and "she" refer to the same person [93]. Word Embedding is a way to measure similarity between words by looking at common contexts the words appear in, and producing a vector space model - words which appear in proximity in this vector space are considered more similar [90]. More details on the approach taken to word embedding are outlined in Section 5.2. The algorithm for identifying targets and performing sentiment only on sentences about them is outlined in Algorithms 1, 3, and 2. Some of the functions are implementation dependent, so they are not presented, but their names are descriptive enough to illustrated their purpose.

Let us apply the algorithm to an example excerpt, taken from a slightly modified article in the Washington Post [98] (one instance of "Trump" was changed to "Donald Trump" to help illustrate the case):

> "The situation remains fluid, and Donald Trump has previously in his presidency backed off economic threats at the last minute. But he has shown a recent willingness to unilaterally impose tariffs — even amid objections from advisers who fear starting a global trade war and economists who warn such actions could ultimately hurt U.S. businesses."

This paragraph was followed by:

> "Trump was particularly determined to follow through on tariffs on China, as criticism of U.S.-China relations was at the center of his presidential campaign, according to the administration officials, who spoke on the condition of anonymity to discuss the president's plans."

We start our analysis by initialising two maps - *sentimentStats* and *articleSentiments*. *sentimentStats* will contain analysed entities as keys, and the values as tuples containing the cumulative total of sentiment score for the entity and the total number of sentences analysed. *articleSentiments* will be the final result - each entity will be a key to its overall sentiment score for the article.

We start with the first paragraph and extract the named entities from it. In this case, these are "Donald Trump" and "U.S.". We set these as keys in the *sentimentStats* with default values. Afterwards, using Coreference Resolution, we extract the sentence "chains" - lists of sentences which are about the same entity, but refer to it in

different ways [62]. In this case, there is only one chain, containing both of the sentences in the first paragraph. The representative mention for it is "Donald Trump", and we can see that it is in our set of named entities. This check is needed because the coreference resolution sometimes produces chains which to not have a named entity.

We proceed to analyse and score every sentence in the chain. A majority vote between three sentiment classifiers is used. Empirical analysis showed that this produced the best results on sentences extracted from news articles. Details on this evaluation of sentiment classifiers are located in Section 7.2 in Chapter 7. The possible sentiment scores are 0 for negative sentiment, 0.5 for neutral, and 1 for positive - more on this in Section 5.3. Once we have the sentiment scores, we update the *sentimentStats* about "Donald Trump" with the results. Let us say that the *totalChainSentiment* is 1.0, and we know that there are two sentences in the paragraphs.

We continue to the next paragraph and we extract its entities. In this case, they are "Trump" and "China". We proceed to set them as keys in *sentimentStats*, but because of word embedding we see that "Trump" and the already existing key "Donald Trump" are very similar. Next, we find that "Trump" is contained within "Donald Trump". We make an assumption that the longer entity is the more important one, so we keep "Donald Trump" in the map and discard "Trump". We extract the chains and we see that there is only one chain - the only sentence in the paragraph. The representative mention is "Trump". We know that it is in the entities for the paragraph, so we proceed to perform sentiment analysis on the sentence.Let us say that we compute the *totalChainSentiment* to be 0.5. We update the sentiment stats, and we use the same similarity determination as before, which tells us to update the score for "Donald Trump".

Finally, we are done with the analysis. We proceed to compute the average sentiment score for the entities in the *sentimentStats*. There is only one entity which was analysed in this case - "Donald Trump". Its *totalSentiment* is 1.5 and *totalSentences* is 3. Hence, the overall sentiment in the article for "Donald Trump" is 0.5.

5.2 Word Embedding

The intuition behind word embedding is that "a word is popularised by the company it keeps" [74]. The idea is to compute word vectors which are positioned in a vectors space such that if they share common contexts in the corpus, they will be located close to one another in the vector space [89]. Word embedding is a way to represent words as dense vectors that are derived by various training methods inspired from neural-network language modeling, which enables computation of word similarities through low-dimensional matrix operations [84]. Word2Vec is an implementation of the continuous bag-of-words and skip-gram architecture for computing vector representations[90], and is the implementation of word embedding which is used for the purposes of this project.

Word embedding offers a lot of possibilities [106, 107], but in this project it is only used to measure entity similarity, so that "Jeremy Corbyn" and "Corbyn" are identified as the same entity, for example. This is done to avoid the case when an article refers to an entity in different ways, so scores are computed for each of them, when it should be one aggregate score.

The words of interest in this project's case are only the named entities which appear in a text. To produce the training dataset, the corpus of Guardian articles was used, since it is the largest one. The dataset is created by discarding the words which are not named entities in an article and producing a list of the named entities in the order in which they appear. Empirical analysis showed that for computing named entity similarity, such corpus is superior to a dataset which contains the entire article text. More details on this analysis can be found in Section 7.2 in Chapter 7.

Algorithm 1: Sentiment Analysis

```
List sentimentAnalysis(List paragraphs)
begin
    Global sentimentStats ← ∅
    Global articleSentiments ← ∅
    for paragraph ∈ paragraphs do
        entities ← extractEntities(paragraph)
        sentimentStats ← setSentimentStats(entities)
        chains ← getChains(paragraph)
        for chain ∈ chains do
            representativeMention = getRepresentativeMention(chain)
            for entity ∈ entities do
                if entity == representativeMention then
                    totalChainSentiment ← 0
                    for sentence ∈ chain do
                        totalChainSentiment ← totalChainSentiment + getSentiment(sentence)
                    updateSentimentStats(entity, totalChainSentiment, size(chain)
                    break
    for entity ∈ keys(sentimentStats) do
        totalSentiment ← getTotalSentiment(get(sentimentStats, entity))
        totalSentences ← getTotalSentences(get(sentimentStats, entity))
        if totalSentences > 0 then
            entitySentiment ← totalSentiment/totalSentences
            put(articleSentiments, entityentitySentiment)
    return articleSentiments
```

Algorithm 2: Similarity Determination

```
Tuple determineSameEntities(String existingEntity, String entity)
begin
    if similarity(existingEntity, entity) < 0.55 and inTopTenSimilar(existingEntity, entity)
     then
        levenshteinDistance ← getLevenshteinDistance(existingEntity, entity)
        if contains(existingEntity, entity) and levenshteinDistance > 2 then
            return Tuple(true, false)
        else if contains(entity, existingEntity) and levenshteinDistance > 2 then
            return Tuple(true, true)
        else if isAbbreviation(existingEntity, entity) then
            return Tuple(true, true)
        else if isAbbreviation(entity, existingEntity) then
            return Tuple(true, false)
        else if levenshteinDistance ≤ 4 then
            return Tuple(true, determineBasedOnPopularity(existingEntity, entity))
```

Algorithm 3: Set and Update Sentiment Stats

```
1  Void setSentimentStats(Set entities)
2  begin
3      for entity ∈ entities do
4          if containsEntity(sentimentStats, entity) == false then
5              setDependingOnWordSimilarity(entity)

6  Void setDependingOnWordSimilarity(String entity)
7  begin
8      existingEntities ← keySet(sentimentStats)
9      foundSimilar ← false
10     for existingEntity ∈ existingEntities do
11         entitySubstitute ← determineSimilarity(existingEntity, entity)
12         if isSimilar(entitySubstitute) then
13             if isSubstitute(entitySubstitute) then
14                 put(sentimentStats, entity, get(sentimentStats, existingEntity))
15                 remove(sentimentStats, existingEntity)
16             foundSame ← true
17             break
18     if foundSame == false then
19         put(sentimentStats, entity, Tuple(0, 0))

20 Void updateSentimentStats(Set entity, float totalSentiment, int numSenteces)
21 begin
22     if containsKey(sentimentStats, entity) then
23         updateEntityStats(entity, totalChainSentiment, numSentence)
       else
24         updateDependingOnWordSimilarity(entity, totalChainSentiment, numSentences)

25 Void updateDependingOnWordSimilarity(String entity, float totalSentiment, int numSenteces)

26 begin
27     existingEntities ← keySet(sentimentStats)
28     foundSimilar ← false
29     for existingEntity ∈ existingEntities do
30         if isSimilar(entitySubstitute) then
31             updateEntityStats(entity, totalChainSentiment, numSentence)
32             if isSubstitute(entitySubstitute) then
33                 put(sentimentStats, entity, get(sentimentStats, existingEntity))
34                 remove(sentimentStats, existingEntity)
35             foundSame ← true
36             break
37     if foundSame == false then
38         entityStats ← Tuple(totalChainSentiment, numSentences)
39         put(sentimentStats, entity, entityStats)
```

5.3 Implementation

As previously mentioned in Section 5.1, three sentiment classifiers are used to determine sentiment based on a majority vote. One of them is StanfordNLP's sentiment classifier [107], and the other 2 are custom classifiers, trained using Apache OpenNLP [3].

StanfordNLP [46] was chosen because it offers all the features required for this project's use cases, including Named Entity Recognition [73] and Coreference Resolution [62]. The API it exposes is easy and intuitive to work with, and there are helpful tutorials and pointers online. It has an active community on GitHub [45], where the code is hosted, which means that it is easy to ask for help if problems occur with the software. Most importantly, its sentiment classifier offers state of the art accuracy, with 85.4% accuracy in single sentence positive/negative classification [107].

Apache OpenNLP, which is a machine learning based toolkit for the processing of natural language text [3], was chosen, because it allows to easily train a custom sentiment classifier. The author trained two classifiers:

- A classifier trained using the Sentiment Labelled Sentences Dataset [82] (manually annotated positive and negative sentences extracted from the Amazon, Yelp and IMDB websites) and the objective sentences (used as a neutral class) from the Subjectivity Dataset (sentences from IMDB plot summaries), introduced by Pang/Li [99]. From hereinafter, this classifier will be addressed as the Reviews Classifier.
- A classifier trained using a custom dataset. The decision to build a custom dataset resulted from the unavailability of an existing set with similar features. The MPQA dataset [64] is the closest to what is required, but it proved impossible to use without the GATE [18] software. The custom dataset was created from manually annotated sentences extracted from news articles. Annotators were split into groups of 2, and each group was allocated a set of sentences. Independently, they had to annotate them for sentiment. Their results were then cross-referenced. Sentences on which both annotators agreed were put in the dataset. The sentences on which they disagreed were artbitrated by the author, and then included in the dataset. The resulting dataset consisted of only 234 sentences, but this nevertheless produced strong results, as it is shown in Chapter 7. From hereinafter, this classifier will be addressed as the News Classifier.

When analysing a sentence, the majority vote between the three classifiers decides what the sentiment will be. If there was no agreement, i.e. every classifier assigns a different score, then the score from the News Classifier was taken. Details of the empirical analysis which led to this decision are outlined in Section 7.2 in Chapter 7. The StanfordNLP classifier produces sentiment scores in the range 0-4: 0 - very negative, 1 - negative, 2 - neutral, 3 - positive, 4 - very positive. The classifiers trained using OpenNLP produce three scores: 0 for negative sentiment, 0.5 for neutral, and 1 for positive. In order to compare these scores, a conversion of the StanfordNLP scores is done - 0 and 1 are converted to 0, 2 to 0.5, 3 and 4 to 1.

In terms of word embedding, two Word2Vec implementations are used. Gensim [19] is used to train the Word2Vec model on the extracted named entities, and DeepLearning4J [11] is used to load the model and use it alongside the sentiment analysers. Gensim was chosen because of Python's inherent simplicity and its easy interaction with MongoDB. Moreover, the cloud environment Colaboratory [8], which currently only supports Python, allowed the author to write and run the training code in the cloud free of charge.

5.4 Summary

In this chapter we saw the approach taken to targeted sentiment analysis in the system, and the implementation choices made for the sentiment analyser component. This component implements the core **M.1** requirement.

Chapter 6

Implementation

6.1 Software Engineering Process

Throughout the duration of this project, the following software engineering practices were adhered to.

6.1.1 Agile Development

The Agile Manifesto [58] was followed as much as possible. The product development was iterative in nature, with very close communication with the client (supervisor). The author met with the client once every week and presented the project's progress. The week span between meetings served as the iteration time, or sprint. The aim was to have new features delivered every week, and to identify new requirements, or refine existing ones during the meetings. Other Agile methodologies such as refactoring, spike solution [43] or Continuous Integration [75] were also employed.

The Agile principles were followed, but without adhering to a particular Agile framework like Scrum or Kanban. Only several documentation artifacts were used - one document captured the minutes of each client meeting, another one was concerned with logging the time spent on various tasks, and a third one contained notes, observations, and details for each task. This was done for several reasons. Firstly, the author of the project was only one. This meant that there was no communication overhead between different team members, and tracking tasks was not a problem. Rigorously maintaining a Scrum board or a Kanban board was not going to add any value. Secondly, because of the continuous face-to-face communication with the client, updates on tasks were easy to be communicated without employing a formal method.

6.1.2 Minimalism

As previously mentioned in Chapter 4, Minimalist Architecture was one of the approaches taken to architecting and developing the system. In terms of the implementation of different components, a recurring concept was the aim to do the least possible to achieve the required goals. This pragmatic approach resonates with teachings from the wide Agile community:

- YAGNI - "You aren't gonna need it" - "Always implement things when you actually need them, never when you just foresee that you need them." [103]

- "Do The Simplest Thing That Could Possibly Work" - recommended by Ward Cunningham, a pioneer in Extreme Programming [112].
- Pain Driven Development - "waiting to apply principles, patterns, and practices to your code until there is some pain the current approach is causing that must be addressed" [65].

These are all based on one of the principles behind the Agile Manifesto: "Simplicity – the art of maximizing the amount of work not done – is essential" [58].

This minimalist approach led to this project's software development process to look like the image displayed in Figure 6.1. We can see its inherent iterative nature, which is focused only on features needed at the moment of development. The next sections provide evidence of this affecting project decisions.

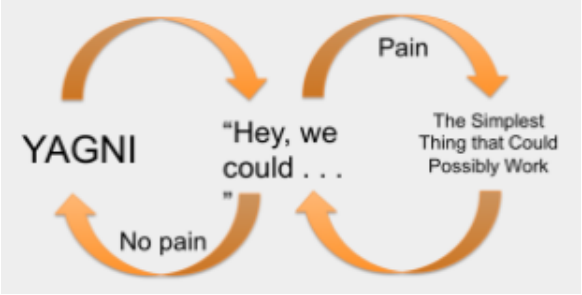


Figure 6.1: Pain Driven Development, image taken from the blog post "Pain Driven Development" [109]

.

6.1.3 Version Control

Git [20] is a distributed Version Control System (VCS) which was used during the development of this project. It is one of the most popular source control systems. It was chosen because of past experience with it, as well as for its simplicity and ease of use.

The remote repository is hosted on GitLab [21]. This service was chosen because it allows for unlimited private projects, and also has a free integrated Continuous Integration and Continuous Delivery pipeline.

6.1.4 Continuous Integration and Delivery

Continuous Integration (CI) is the process of frequently integrating recent work into the project, and having an automated script run tests and build the software, in order to verify that the integration was successfull [75]. CI is used for the majority of the system's components - the indexing component, the search service, the sentiment analyser, the client interface, and the web API. These components are often updated and well tested, since they implement the major functional requirements. The ingestion component is comprised of scripts running rarely, and their code hardly changes, so a decision was made not to set up a CI pipeline for this component.

Continuous Delivery (CD) works on top of CI by automatically deploying the software if the build is passing without errors. CD is set up for the website's code - the UI component and the web API. This is the portion of the system that the user sees and interacts with. An update on the *master* branch triggers the CI script, and on success the application is deployed to Heroku, a PAAS service [22], where it is publicly accessible. It can be found at http://media-sentiment-analyser.herokuapp.com/.

6.2 Ingestion Component

The ingestion component was used to build a static corpus of newspaper articles. Further work could be done to ensure that ingestion happens continuously, so that the system is up to date with newly published articles.

Article ingestion was done using several Python scripts. Two data sources were used - The Juicer [50], developed by BBC News Labs [6], and The Guardian Open Platform [49].

The Juicer aggregates and extracts news articles from the web, and exposes them through a REST API. It was used to procure news articles from The Washington Post, The Daily Mail, and BBC News. Unfortunately, the articles available were constraint to the year 2016, which prevented the collection of a large corpus of articles.

The Guardian Open Platform is a public web service, developed by the Guardian, which exposes all the content The Guardian creates. It allows access to over 1.9 million pieces of content, which is exposed through a REST API. The system's largest corpus was built with articles extracted from there. Every article since 2010 was extracted - 856264 articles in total.

Other sources were considered as well, such as Event Registry [15] or the AYLIEN News API, which was described in Chapter 2. Both of these would have allowed for the creation of a much larger article corpus, with multiple newspapers. However, there was a pay wall on both of them, even after contact was made to request for academic access.

6.3 Database Component

The database technology chosen for the database component is MongoDB [30]. It is a NoSQL document database solution which stores data in JSON-like documents, such as the example in Figure 6.2. An advantage of documents is that they map easily to native data types in programming languages like Python or JavaScript, and MongoDB provides native language drivers for the most popular programming languages.

Figure 6.2: Fields and values in MongoDB. Image taken from the MongoDB website [30]

.

More importantly, this choice was made because of the nature of the data which is stored. The entire article text is stored in the database, along with metadata about the articles including publishing date, heading, description and URL. This data is semi-structured, and would often change, depending on the ingestion source. Freedom is required to change the schema of the database easily anytime the architecture or the requirements change. An example of this is the decision to take the sentiment analysis offline, described in Chapter 4. Initially, the persistent storage of the sentiment was not envisioned, but a new field was easily added, to contain the sentiment scores as an array of documents. Using a traditional RDBMS would have presented more difficulties for a change like this, since such nested structure would have to be normalised.

6.4 Indexing Component

The indexing component is implemented mainly using the Terrier search engine [47]. It is an open-source Information Retrieval platform, developed at the Univeristy of Glasgow. Terrier follows an easy to extend plugin architecture, which allowed the author to easily adjust it to the system's use case. By default, Terrier does not support indexing of documents store in a MongoDB database, so a custom implementation of the *Collection* interface used in Terrier to read documents from a corpus was developed.

6.5 Search Component

The search component exposes a REST API interface to Terrier's retrieval functionality. The interface has only one endpoint: */search/{newspape}/{entity}*. The application sending the request only needs to specify the newspaper to retrieve from, and the entity of interest.

Since Terrier is implemented in Java, the easiest way to expose it to the web is to use a Java-based web framework. There are plenty of options available for this - Spring Boot [44], Apache Struts [4], Play Framework [36], Spark Framework [42], Pippo [35]. Pippo was chosen for several reasons.

Frameworks like Spring Boot and Apache Struts are fully featured frameworks aimed at the enterprise. It would be a waste of time and resources to use them in a project with just a single endpoint. This fact prompted the author to search for a micro web framework, in accordance to the minimalist approach to development. Micro frameworks are small and lack some of the features enterprise frameworks offer out of the box, like a dependency injection container, authentication handling, or ORM features, but offer a clean and easily extensible core [66]. If the need arises, these frameworks can be extended with the missing features by adding plugins.

Pippo and Spark Framework are examples of such frameworks in the Java ecosystem. For this use case, Pippo was chosen, because of its minimal dependencies and quick learning curve. It's core is rather small (140KB [35]), which means it can easily be deployed anywhere. It can be very easily customized by a simple to use plugin architecture. Working with it proved easier than working with Spark framework. Moreover, Spark framework's embedded server - Jetty [24], had conflicts with the Jetty instance which comes bundled with Terrier. Adjusting Spark Framework to handle this proved harder than using Pippo with a Tomcat [51] embedded server.

In terms of deployment, the entire search service is packaged as an executable JAR, and is deployed to a DigitalOcean instance. Continous Delivery was not set up, since the code does not change often, and once deployed, the search component rarely needs to be redeployed. This only happens when a new newspaper is added, and the search service needs to be notified of the extra index.

6.6 Web API Component

The web API component, along with the UI component, implement the **M.2** requirement. They are essential to the product, since they expose the product's feature's to the users.

The web API is implemented with Python and the Flask micro framework [17]. The reasoning behind choosing a micro framework is similar to the one for the search component's web technology - the web API exposes only a handful of endpoints, and the results returned are in a simple JSON format. Also, despite the fact that it is a micro framework, Python's Flask has a large community behind it, with a lot of support. The Media Sentiment Analyser has no authentication, upload handling, or any other complex feature - it simply exposes the

precomputed sentiment analysis scores. Choosing a large and fully featured framework like Django[12] would not have added value. Flask was able to address all of the system's needs with the smallest amount of work.

Python was chosen as the implementation language because of preexisting experience with it. Moreover, working with MongoDB is intuitive with Python, because of the natural mapping between Python's dictionaries and MongoDB's documents.

6.7 UI Component

6.7.1 Front-end Framework

The client interface was implemented using the Vue.js JavaScript library [52]. The other possible alternatives were frameworks like Angular and React, or simply plain JavaScript.

It was decided against using plain JavaScript, even in the context of the system's simple client interface. Doing front-end development without a stable framework for guidance and structure means that it is easy to produce inflexible code.

The alternatives for a front-end framework are many, but the strongest candidates were React [38], Angular 2 [2], and Vue. These are the most popular front-end frameworks at the current market, and each comes with different advantages.

React is built and supported by Facebook and has been growing in following in the recent years. However, it works with a syntax extension to JavaScript - JSX [67], which meant that the author would have to learn new syntax and concepts just to be able to use it. Moreover, React has a steep learning curve [68], which would increase the time to get proficient in it.

Angular 2 is developed and supported by Google, and is currently the most popular front-end framework [111]. It uses the TypeScript programming language, which transpiles to JavaScript. Like React, it also has a steep learning curve, since it is designed to target large, complex applications [68].

Vue is not as popular as the React and Angular, but it nevertheless has a significant following. It is designed to be lightweight and easy to learn, since it embraces classic web technologies - HTML, CSS and JavaScript. It provides reactive and composable components. Its speed is comparable to React and Angular. It is not as opinionated as Angular, and lets the developer make the choices of how to develop the application. Vue was designed to be incrementally adoptable and extendable with supporting libraries and tools [68].

In the end, Vue was chosen because of the approach to minimalism. It required less time to learn and be productive with, and did not constrain the user interface layer with a particular design approach.

6.7.2 Styling Framework

A decision on how to present and style the application needed to be made as well. The first option was to use pure CSS, but this was decided against in a favour of using a styling framework. The author did not have significant prior experience with front-end technologies, and using pure CSS to style the application would have proven unnecessarily difficult. A styling framework in most cases provides a lot of built-in classes, an easy to use grid layout, which allows the developer to position elements in rows and columns, and is often responsive out of the box, which was needed to satisfy requirement **NF.8**.

Examples of such frameworks are Bootstrap [7] and Element [14]. Bootstrap is very popular and provides all the previously mentioned features, but using it with Vue presents certain problems. Bootstrap offers not only CSS styles, but also features implemented using jQuery [25], which is an unnecessary dependency for this project. Element, on the other hand, is built specifically for Vue, which means that it is easy to integrate with it. Since it offers the same features as Bootstrap, it became the styling framework of choice for this project.

6.7.3 Individual Interface Components

Header

The header is designed to contain the navigational logic of the website, along with the logo. It is displayed in Figure 6.3. The most important page is the "Analyser", which is the landing page of the application. The "About" page is a simple textual page stating the purpose of the system. The "FAQ" page answers the questions "What data do you currently have?", "How do you estimate sentiment?", and "How do you choose which articles to display sentiment from?". The "Acknowledgements" section references the research and tools the system is based on.

Figure 6.3: The website header.

Main Form

The main form allows the user to query the system. It is the main source of interaction, and implements most of the major functional requirements. It is comprised of several components:

- Selection of the type of analysis. There are three types - producing results for one entity and one newspaper, comparing results for one entity and two newspapers (which adds the ability to select two newspapers), and comparing results for two entities and one newspaper (which ads an extra input field for the second entity). This satisfies requirements **M.4**, and **M.5**.
- Helper features ("Popular Entities" and "Trending Entities") - these implement requirements **S.2** and **S.3**.
- Input of the entity of interest - to satisfy **M.3**. This input field is also equipped with an autocomplete functionality, which list suggested entities based on the current input, in order to satisfy requirement **S.4**. The list of suggested entities is compiled by extracting all the entities for which sentiment analysis was performed, and sorting them by popularity (number of articles they appeared in).
- Date range selection - to satisfy **M.6**.
- Newspaper Selection - to satisfy **M.7**.

After the form is submitted and while waiting for the response, the user is presented with a loading spinner, to indicate that the system is working. This relates to requirement **NF.5** for clear communication of what the system is doing. The layout of the form is illustrated in Figure 6.4, along with examples of the popular entities feature and the autocomplete feature.

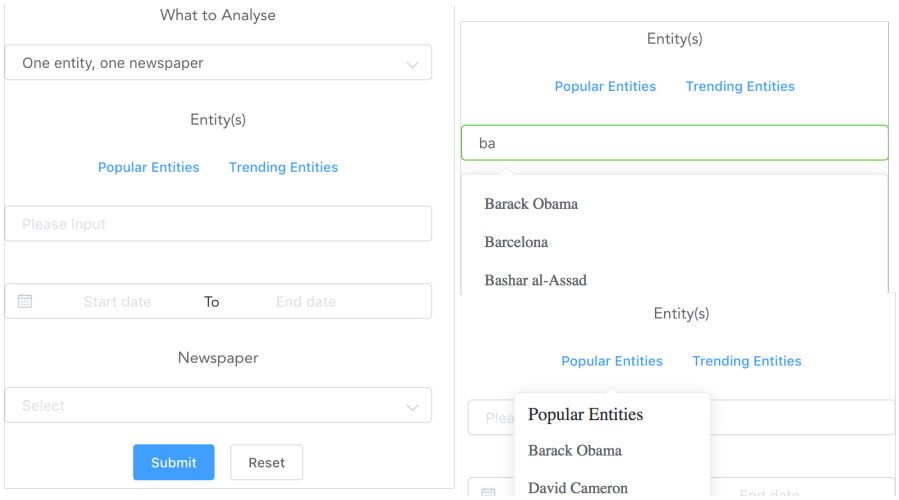


Figure 6.4: The main form, with examples of the autocomplete feature and the popular entities feature.

Explanation Section

Similar to the Sentiment Viz product we looked at in Chapter 2, an explanation section was included in the Media Sentiment Analyser. It outlines the main features of the application, what is expected from the user, and an overview of how sentiment is estimated. This satisfies the **M.10** requirement, and contributes to the **NF.1** requirement.

Visualisation

This component satisfies the **M.9** requirement. There was a careful consideration of what technology to use for it. The initial choice was the charting library Dygraphs [13], but it proved hard to integrate with Vue. This led to the consideration of the D3 charting library [10], which offers a very powerful API for building charts and graphs. However, it does not provide a predefined set of charts which could be used, instead it offers the low-level building blocks. This finally led to the choice of Highcharts [23]. Highcharts is a powerful charting library which exposes a simple API, with available modules for Vue. The graphs produced are clickable and zoomable, which are necessary features to satisfy requirement **S.1**. An example of a resulting visualisation is displayed in Figure 6.5. A point on the graph represents an average score for a particular day. For each day, one or more newspaper articles have been analysed.

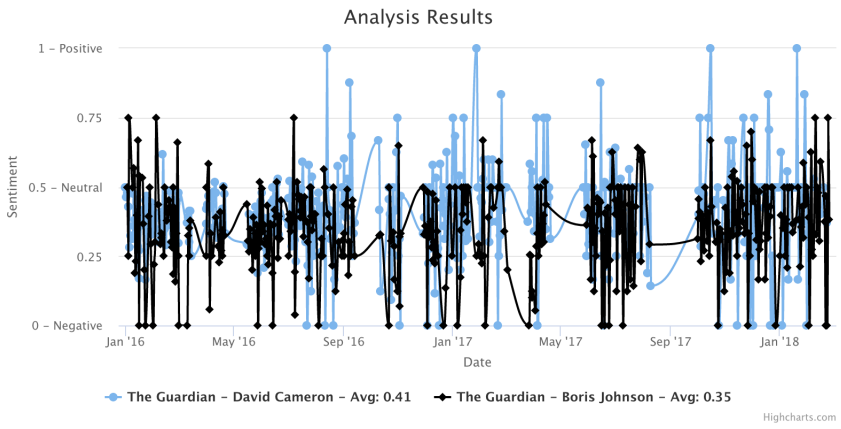


Figure 6.5: Visualisation of how The Guardian reported on David Cameron and Boris Johnson.

Daily Report

The daily report feature satisfies requirement **S.1**. The daily report is produced by clicking on a point on the chart. It lists the articles used for this day's analysis, along with their respective scores and a link to the article's webpage. A sample daily report is illustrated in Figure 6.6.

Figure 6.6: Sample daily report output.

6.7.4 Responsive Design

According to requirement **NF.8**, the Media Sentiment Analyser needs to be usable on mobile devices as well. Extra care was taken to style the application in a way to make this possible. This effort was helped by the built-in responsiveness of Elements. Where this was not enough, custom style directives called media queries, which allowed the layout of the system to be changed depending on the screen size[91], were used. Figure 6.7 illustrates how the website looks when viewed on an iPhone 6, 7, or 8.

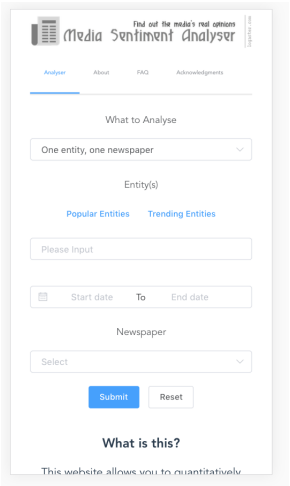


Figure 6.7: Responsive design on iPhone 6, 7 or 8.

6.8 Summary

This chapter outlined the implementation details for all of the system's components, except for the sentiment analyser, which was described in Chapter 5. It was demonstrated that the Media Sentiment Analyser implements all of the "Must Have" and "Should Have" requirements outlined in Chapter 3. While it is true that the "Could Have" requirements were not addressed, the system is nevertheless able to provide value to its users by successfully delivering on its most important requirements. This claim is validated in the next chapter.

Chapter 7

Evaluation

This chapter discusses the methods used to evaluate the system. The main purpose of the evaluation is to answer the following questions:

1. Is there a gap in the market for this product?
2. Were sufficient steps taken to ensure that the system produces accurate results?
3. Did users find the system to be usable?
4. Did the system help the users compare sentiments of different newspapers?
5. What impact on the usability of the system did the autocomplete and entity-suggestion features have?
6. Do weaker sentiment analysis results change the user's view of the accuracy of the system?
7. Were sufficient steps taken to ensure that the system is well tested and executes its functions as designed?

7.1 Validation of the Market and Products Analysis

In order to answer the first question outlined above, a questionnaire was distributed among members of the Computing Science School and acquaintances the author has on social media. The main purpose of this questionnaire was to validate the conclusion made in Chapter 2 that there is indeed a market gap for a product like the Media Sentiment Analyser. It was completed by 109 people.

The first question, "What are the ways you inform yourself of the news?", which allowed multiple choices, indicated that people inform themselves mostly through online newspapers (78.9% of respondents) and social media (76.1%). Television, radio, and print newspapers all had less than 30%.

These results are in accordance with the findings by the Reuters Institute for the Study of Journalism, where we see that the majority of people in the UK in 2017 inform themselves online, including social media [92].

Aiming to investigate the people's trust in the media's integrity, we see in Table 7.1 that a staggering percentage of people believe that the mainstream newspaper media is biased.

Question	Yes	No
Do you think mainstream newspaper media is biased in their reporting?	78.9%	21.1%

Table 7.1: Media Bias Analysis.

In Figure 7.1 we can see that the majority of people showed that they think about bias by saying they either always consider it, or most of the time.

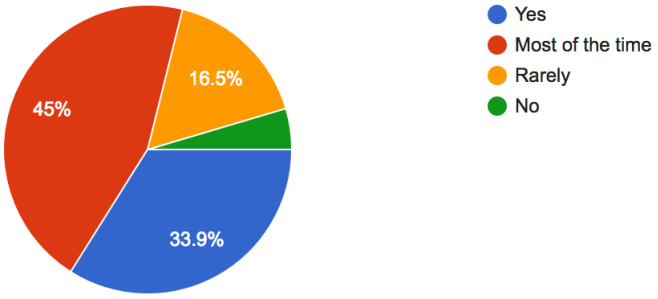


Figure 7.1: When reading a newspaper article, do you consider the potential bias of the author?

The questionnaire also tried to evaluate people's desire for features like those offered by the Media Sentiment Analyser. In Table 7.2 we see that a little over three quarters of the respondents have wanted to compare the reporting of different outlets, and a little over two thirds of the respondents have wanted to compare the reporting of the same newspaper to different entities. These questions aim to evaluate whether people have ever wanted to do what the system is offering them. The results clearly show that this is true. We also observe that despite this fact, the vast majority of them did not know of a tool which would allow them to do that.

Question	Yes	No
Did you ever want to compare how different newspapers report on the same entity?	78.9%	21.1%
Did you ever want to compare how the same newspapers reports on different entities?	69.7%	30.3%
Do you know of a tool which will allow you to do what the previous 2 questions asked?	95.4%	4.6%

Table 7.2: Core features market analysis.

In Figure 7.2 we see that 80.6% of the participants indicated that they would be able to form a better judgement of the media's coverage if they used a tool like this. In light of the previously highlighted distrust in the media's objective coverage, this stands to show that people will benefit from a tool like the Media Sentiment Analyser.

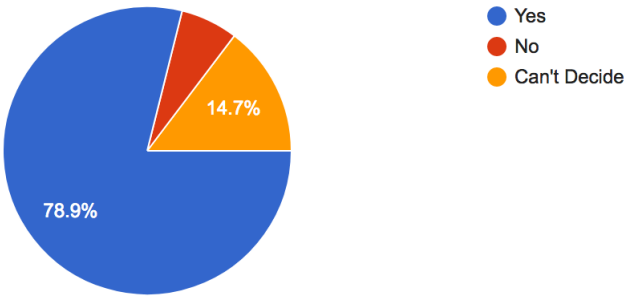


Figure 7.2: Do you believe you would be able to form a better judgement of what you read in the news if you used a tool like this?

A little over three quarters of the respondents indicated that they would use a tool like this. This only reinforces the fact that people distrust the media and would like to be able to quantify its bias. Results are in Table 7.3.

Question	Yes	No	I am using a similar tool
Do you see yourself using a tool like that?	78.9%	21.1%	0%

Table 7.3: Would a tool like the Media Sentiment Analyser be used?

The results from this questionnaire show that there is indeed a market gap for this system. People are distrustful of mainstream media's objectiveness, and online newspapers are the most popular way people inform themselves. The respondents strongly indicated that they are interested in the features the Media Sentiment Analyser offers and that they would use a tool like that. This confirms the findings presented in Chapter 2.

7.2 Sentiment Classifiers Evaluation

This section aims to provide an answer to the second evaluation question, by outlining the steps taken to ensure the best possible sentiment analysis results.

As previously mentioned in Chapter 5, a majority vote between three sentiment classifiers determines the sentiment score for a sentence. This configuration grew organically from observations into the effectiveness of the initial sentiment classifier - the StanfordNLP one. It was observed that it tends to negatively score most of the sentences. Qualitative analysis was performed, and it was noticed that sentences typical to news articles, which were expected to be tagged as neutral, were tagged as negative. An example of this is the sentence "Trump met with di Genova in person on Thursday after his hire was announced.", taken from an article in the Washington Post [70]. One explanation for this could be the fact that the corpus the StanfordNLP sentiment classifier was trained on was made of movie review excerpts from the Rotten Tomatoes website [107]. Intuitively, because of the difference in domains, it is expected that sentences from news articles will have different structure and will consist of words which are unlikely to be seen in the movie reviews dataset.

This prompted the author to train two more classifiers - the Reviews Classifier and News Classifier described in Chapter 5. This would allow the evaluation of the different classifiers on some of the manually annotated sentences from the corpus used to train the News Classifier, which were set aside for evaluation purposes. The validation set is comprised of 48 sentences. The results of the classification accuracy for each classifier are displayed in Table 7.4.

StanfordNLP	News Classifier	Reviews Classifier
0.54	0.56	0.43

Table 7.4: Majority vote performance on the training dataset.

An interesting observation is the fact that the News Classifier, despite the small dataset it was trained on, performs comparatively to the StanfordNLP classifier, even slightly better. Moreover, we see that the Reviews Classifier performs the worst. These two observations provide evidence to support the claim that a sentiment classifier trained on a review-based dataset might not be suitable for analysis of newspaper articles. The author believes that further research on this topic is required, but it is out of the scope of this project.

Testing was also done to determine whether a majority vote produced better results. Three cases were tested, one for each of the cases when a sentiment classifier had the strongest vote (the deciding vote if there was no agreement). The results are displayed in Table 7.5

StanfordNLP deciding vote	News Classifier deciding vote	Reviews Classifier deciding vote
0.56	0.625	0.54

Table 7.5: Majority vote performance on the training dataset.

We can clearly see that the majority vote produces the best results when the News Classifier has the deciding vote. This is why it was decided to use this strategy in the sentiment analyser component.

Finally, statistical analysis of annotator agreement indicates that people do not agree much about sentiment expressed in the news. Table 7.6 illustrates the results. Cohen's Kappa [63] was used to measure annotator agreement in each group.

Group1	Group2	Group3	Group4	Group5	Group6	Group7	Group8	Group9	Group10
0.33	0.46	0.39	0.39	0.23	0.32	0.25	0.18	-0.02	0.53

Table 7.6: Cohen's Kappa for each annotator Group

Scores lower than 0.4 are interpreted to mean lower agreement [55]. We can see that most of the scores are in this range. The cause for this low agreement between annotators can be attributed to the lack of context, since the annotators only saw individual sentences, not entire articles. Nevertheless, this is an interesting observation, and the author recommends more research into the topic.

7.3 Word2Vec Models Evaluation

This section supplements the previous one, by illustrating the evaluation steps taken to ensure that the word embedding model used produces the best results. Word embedding helps the sentiment analyser aggregate the scores for an entity, regardless of how it was referred to, instead of producing separate scores for each mention. This adds to the overall accuracy of the sentiment analysis.

In order to make sure that the best results are achieved with word embedding, two Word2Vec models were evaluated. One of them was trained only on lists of named entities, in the order in which they appear (from hereinafter referred to as the Named Entities Model), while the other was trained on the entire articles (from hereinafter referred to as the Full Articles Model). This was done in order to investigate which model produced better similarity suggestions in the top 10 similar entities for an entity.

Empirical evaluation was performed with several sample entities. The suggested similar entities for these entities by the two models are listed in Table 7.7.

Entity	Full articles model	Named entities only model
Barack Obama	Obama	Obama, Barack
Donald Trump	Trump	Trump, Mr Trump, Donald J Trump, Donald J. Trump
Borish Johnson	Boris	Boris, Mr Johnson, Johnson, London Mayor, Mayor of London
Jeremy Corbyn	Corbyn	Corbyn, Mr Corbyn
David Cameron	Cameron	Cameron, Mr Cameron
Britain	UK	UK, Brexit Britain
Apple		Apple Inc

Table 7.7: Word2Vec models similarity suggestions (only relevant ones are listed).

We can clearly see that the Named Entities Model outperforms the Full Articles model, by producing more relevant similarity suggestions. The author argues that the approach using training data comprised of just named entities is superior for the task of identifying embeddings of named entities. The Named Entities Model takes significantly less time to train, and its size is 4 times less than the Full Articles Model. While out of this project's score, more research is recommended, in order to identify other areas where this would prove useful.

7.4 Usability Evaluation

In order to evaluate the usability of the product and provide answers to questions 2 and 3 from above, 10 people were asked to participate in a think-aloud study. They had to complete a set of tasks using the website and while doing this they were asked to share their thoughts aloud. After finishing the tasks, the participants filled a usability form based on the System Usability Scale (SUS) [60]. The tasks the participants were asked to complete are listed below.

1. Find out how The Guardian reported on the former U.S. President Barack Obama.
2. Compare how The Washington Post and The Daily Mail reported on David Cameron between the 15th May and 30th September 2016.
3. Find out The Guardian daily report on Ted Cruz on the 12th March 2016 and navigate to the web page of one of the articles.

These tasks were chosen because they encapsulate the main features the product provides. They aim to evaluate whether the system was successful in implementing its "Must Have" requirements and "Should Have" requirements from Chapter 3. Moreover, they were designed to look like potential queries the people in the scenarios outlined in Chapter 3 would make.

7.4.1 System Usability Questionnaire

There are 10 questions in the SUS, all of them with answers ranging from 1 to 5, where 1 means "Strongly Disagree" and 5 means "Strongly Agree". The results from a SUS can be transformed to a score ranging from 0 to 100. It can be seen in Table 7.8 below, that the average from all the SUS scores is 81. From the 10 participants, there is only one whose score was particularly low.

User1	User2	User3	User4	User5	User6	User7	User8	User9	User10	**Mean**
85	92.5	77.5	80	85	80	95	82.5	75	57.5	81

Table 7.8: Individual SUS scores and an overall average.

Empirical analysis has shown that a SUS score of 68 can be considered as an average score[59]. Taking this into account, we conclude from Table 7.8 that the Media Sentiment Analyser is above average in terms of usability.

7.4.2 Think-aloud Study

The purpose of the think-aloud study was to qualitatively evaluate the application and to indicate whether there were any usability issues which a large number of participants noticed, as well as identify features which a lot

of users liked. This would allow us to measure the Gulf of Execution - the difference between the intentions of the users and what the system allows them to do or how well the system supports those actions[95]. The most common feedback follows.

- *"I thought I have to select a single day instead of a date range when doing the third task"*
  Participants were expected to choose a suitable range and then use the chart to select the specific day they were interested in. However, all of the participants proceeded to select the single day 12th March 2016 in the form, but got no results, since the system was designed to expect a date range, instead of a specific day. Choosing a specific day makes the system return results from 00:00 on this day to 00:00 this same day, which produces no results. Most of the users did eventually use a date range, but this nevertheless shows a flaw in the system design's assumptions, which needs to be addressed in a future iteration.
- *"What is the difference between "trending" and "popular" entities?"*
  Several users pointed out that they were not immediately clear on the meaning of those terms. It was suggested that a more suitable naming convention should be followed, or just one of those entity suggestion lists should be used.
- *"Why can't I move the datepicker a year back?"*
  When trying to adjust the date range, some participants tried to move the end date part of the datepicker to a time before the start date, so the datepicker did not allow them. They had to adjust the date by changing the month, which was allowed, since it did not produce an end date before the starting date. This was not immediately obvious though, so it was suggested that a helper message be displayed to indicate why the act of going back a year is forbidden in some cases.
- *"I like the loading spinner, but I would like to be informed of how much time is left until it loads."*
  While the loading spinner was well-received by participants, it was suggested that it could use a percentage metric to indicate how much more the user will have to wait for results.
- *"The autocomplete feature is really helpful and looks very nice."*
  Everyone took advantage of the autocomplete feature and found it helpful. A detailed investigation on its usefulness is performed in Section 7.5.1.
- *"The chart is beautiful, it is easy for me to distinguish what is on it, and to compare the results."*
  Several people exclaimed that they enjoy working with the chart. They stated that the colour-coding was good, and that even with a lot of points of the graph, it was still easy to examine the data and do comparisons. This provides a clear answer to the third evaluation question, listed in the beginning of the chapter.
- *Most of the users took advantage of the help text provided. They were mostly interested in the "How do I use this?" section, and did not pay much attention to the rest.*

The information gathered from this think-aloud study shows that the system manages to provide its users with the required features, but there is definitely space for improvement. Evidence for this is the flaw discovered in the system's date handling, and the confusion around the selection of a date range. This is where the widest Gulf of Execution can be observed. None of this was critical for the users however, and there was a lot of positive feedback, which reinforces the findings in Section 7.2.1.

7.5 Evaluation of Specific Features

Two experiments were also performed during evaluation. The first investigated whether the autocomplete and entity suggestion features increase the usability of the system. The second experiment looked into whether a

weaker sentiment analysis results change the user's view of the system. The two experiments tried to answer evaluation questions 4 and 5 respectively.

7.5.1 Did the autocomplete and entity suggestion features increase the usability of the system?

These features correspond to requirements **S.2**, **S.3**, and **S.4**. and represent the majority of the "Should Have" requirements. They were included in order to help the users with a quick entity selection mechanism, and to streamline the experience for people like **Gregg**, from the scenarios in Chapter 3.

Hypothesis

The following variables are identified:

- Dependent variable: usability (measured using SUS).
- Independent variable: the autocomplete and entity suggestion features.

In order to test the independent variable's effects on the dependent variable, we vary it. The autocomplete and entity suggestion features can either be available to the user or not. Following from this, the experiment's hypothesis is the following:

H_1 The system's usability is increased by the autocomplete and entity suggestion features.

In order to prove this hypothesis, the following null hypothesis needs to be disproved:

H_0 The system's usability is not increased by the autocomplete and entity suggestion features.

Experimental Procedure

The 10 users mentioned in the previous section did the evaluation while having all features enabled. Five more people were asked to evaluate the system with the autocomplete and entity suggestion features missing. They were also asked to think-aloud. Their results are shown in the Table 7.9.

User1	User2	User3	User4	User5	**Mean**
90	75	80	47.5	77.5	74

Table 7.9: Individual SUS scores and an overall average when autocomplete and entity suggestion was turned off.

Quantitative Analysis

In order to analyse the results, a Mann-Whitney[87] statistical test is performed. The sample data sets are both less than thirty, which is not enough to assume normal distribution[78]. This means that a non-parametric test needs to be performed[78]. The reasona a Mann-Whitney test is chosen is because it handles the case when there are two independent samples. The confidence level is 0.05. The test performed is also a two-tailed test, in order to make sure that we do not miss the case when instead of improvement, there was actually a decrease in the usability.

The Mann-Whitney test calculates a statistic, called U, whose distribution under the null hypothesis is known. Using the data above, we calculate that U is 16 and that the critical value of U at $p < .05$ is 8. Hence, we can conclude that the result is not significant at $p < .05$. This means that we cannot not reject the null hypothesis and we have to conclude that the system's usability is not increased by the autocomplete and entity suggestion features.

It is worth noting that there is a difference in the means between the two samples which indicates better results when the user help functions were available. As evident in the next section, this is also reinforced by the qualitative analysis.

Qualitative Analysis

The main point of confusion was the decision of what to put as an entity. Some people put only "Obama" and did not get all the results, or "U.S. President Barack Obama" and got none of the results. Some of them also made a typo, typing "Barak Obama", instead of "Barack Obama", which produced just 1 result. The participants spent extra time double checking their form because of mistakes like this, and were unsure of what they did wrong. This is in stark contrast to the observations from the group who had all the features - they would take immediate advantage of the autocomplete feature, which eliminated their mistakes in spelling. Some of them also used the popular entities and the trending entities lists to select an entity, which removed the need for typing altogether. These observations prove that the autocomplete and entity suggestion features were of use to the participants and helped them to work with the system.

Discussion

We see conflicting results from the qualitative and quantitative analysis. However, the sample sizes are small, so we cannot have full confidence in the statistical results. Moreover, we see that there is a difference in the means for the two samples in favour of increased usability, which is supported by the qualitative analysis. In the case of this experiment, it can be argued that the qualitative analysis provides more meaningful insights, which leads to the conclusion that the autocomplete and entity suggestion features did improve the system's usability by shortening the time it takes to make a request and eliminating potential typos made by the users.

7.5.2 Do the results from a weaker sentiment analysis classifier decrease the users' perception of the accuracy of the system?

Hypothesis

The following variables are identified:

- Dependent variable: the users' perception of the accuracy of the system.
- Independent variable: the sentiment classifier used.

As in the previous experiment, the value of the independent variable is varied. The sentiment results can either be taken from the majority vote between the three sentiment classifiers, or from the weakest classifier. Following from this, the experiment's hypothesis is the following:

H_1 The users' perception of the sentiment accuracy is decreased when a weaker sentiment classifier is used.

In order to prove this hypothesis, the following null hypothesis needs to be disproved:

H_0 The users' perception of the sentiment accuracy is not decreased when a weaker sentiment classifier is used.

Experiment Procedure

In Section 7.2 the analysis of the different sentiment classifiers was illustrated and it was shown that the majority vote between the three classifiers produces the best results. The worst results are produced by the Reviews Classifier. For this experiment, we alternate between the results from the two.

Five of the 10 people who used the system with all its features enabled were presented with results from the majority vote, and the other five - with results from the reviews classifier. In addition to the SUS questionnaire, they also had to answer an extra question - how they rated the system's sentiment accuracy between 1 and 5, where 1 meant "Very Innacurate" and 5 meant "Very Accurate".

It is worth noting that the 5 people who did the evaluation with the autocomplete and entity suggestion features turned off were not included, because for some of the tasks some of them did not manage to produce a chart which they could analyse, hence their impression of the sentiment accuracy was limited.

The results are shown in the two tables below.

User1	User2	User3	User4	User5	**Mean**
3	1	2	2	3	2.2

Table 7.10: Sentiment accuracy estimate of the majority vote sentiment results.

User1	User2	User3	User4	User5	**Mean**
3	2	1	2	5	2.6

Table 7.11: Sentiment accuracy estimate of the custom reviews model sentiment results.

Quantitative Analysis

It can easily be observed that the change in the sentiment classifier does not influence the users' perception of the sentiment accuracy. For the sake of completeness, we perform a Mann-Whitney test on the results. The reasons for choosing a Mann-Whitney test are the same as the ones in the previous experiment.

In this case, we calculate that U is 11.5 and that the critical value of U at $p < .05$ is 2. Hence, we can conclude that the result is not significant at $p < .05$ and that we cannot reject the null hypothesis. This confirms the initial observation in the beginning of this subsection.

Discussion

It is worth pointing out that the three tasks the users did were not enough for them to form a strong opinion on the sentiment accuracy. While all the users spent time investigating the results they got, none of them actually compared what the system returns to what the actual articles say. This means that they compared the results they saw to what their prior expectations were.

In order to confidently answer this research question, a study focused just on sentiment analysis should be performed. An experiment could split participants into two groups. The first group would be asked to rate the sentiment accuracy of the system on the results from two tasks, where one task would produce results using the majority vote, and the other task - using the Reviews Classifier. The second group would be required to do the same, but the sentiment scores would be swapped. Statistical analysis could then be performed on the results in order to provide an answer to the research question.

7.6 Unit and Integration Testing

Unit tests were developed for the majority of the components - the UI, the web API, the sentiment analyser, the search service, and the indexing component. These tests make sure that the lower-level functions of these components work as expected.

Most of the testing effort was spent on the sentiment analyser, since it provides the sentiment scores required for the entire system to be useful. In order to test it, 50 unit tests and several integration tests were developed, resulting in 91% coverage of the lines in the sentiment analysis code, which increases the confidence in the quality of the software.

For the Java based components, JUnit [26] was the testing framework used, along with Mockito [28] for the mocking of dependencies. For Python, Pytest [37] along with MockupDB [29] was used. For JavaScript, Karma [27] was used.

7.7 Acceptance Testing

Acceptance tests are black box system tests - each acceptance test represents some expected result from the system [116]. They are identified from the project's requirements, and written in order to verify that the system is behaving as the clients have specified [88]. Acceptance tests for the Media Sentiment Analyser were developed using the Nightwatch testing framework[32]. It is a Node.js based End-to-End (E2E) testing solution for websites and uses a Selenium[39] server for browser automation of tasks. Tests were written to examine most of the ways a user could interact with the application, including the main form submission, analysis chart display, and website navigation. Acceptance tests for the daily report feature implementing requirement **S.1** were not produced, because of difficulties in interacting with the Highcharts library in browser automation mode. Further work could be done to ensure that chart interaction and daily report production are tested as well. An example of how the Nightwatch API works is shown in Figure 7.3.

```
'submitting the form with an entity produces analysis': function(browser) {
  const devServer = browser.globals.devServerURL
  browser
    .url(devServer)
    .waitForElementVisible('#app', 5000)
    .setValue('input[name=entity1-input]', 'Obama')
    .waitForElementVisible('#newspaper-input', 1000)
    .pause(1000)
    .click('#newspaper-input')
    .pause(1000)
    .click('#thebbc')
    .click('button[name=submit-main-form]')
    .waitForElementVisible('.analysis', 5000)
    .assert.elementPresent('#graphdiv')
    .assert.containsText('.highcharts-title', 'Analysis Results')
    .assert.elementCount('.highcharts-point', 4)
    .end()
```

Figure 7.3: Nightwatch acceptance tests API

7.8 Summary

This chapter attempted to answer the evaluation questions listed in the beginning of the chapter. A Summary of the answers is listed below.

1. *Is there a gap in the market for this product?*
   Using the Market and Products Analysis questionnaire, it was proven that there is indeed a gap in the market for this product. The findings were discussed in Section 7.1.

2. *Were sufficient steps taken to ensure that the system produces accurate results results?*
   This question relates to requirement **NF.3**. Sections 7.2 and 7.3 addressed this question and provided evidence to answer it positively, by detailing the careful evaluation steps taken to ensure the best results.

3. *Did users find the system to be usable?*
   It was showed both through qualitative (SUS) and quantitative (Think-aloud study) analysis that the system was considered usable by the participants in the evaluation. The average SUS score was 81, which is 13 points more than the industry standard of 68, and except for one design flaw, no serious issues were discovered with the system. These findings were discussed in Section 7.4.1 and 7.4.2. The high usability score allows the author to argue that requirement **NF.1** is satisfied. Moreover, even though there was some slight criticism towards the loading spinner used to indicate that the system is doing an action, it was well-received by the evaluation participants, and indicates that the system satisfies requirement **NF.5**.

4. *Did the system help the users compare sentiments of different newspapers?*
   Evaluation participants' feedback in the think-aloud study indicated that the system provides them with everything they need to investigate and compare sentiments in the media. In-depth analysis was performed in Section 7.4.2. This indicates that the "Must Have" features **M.1** and **M.8** were well executed.

5. *What impact on the usability of the system did the autocomplete and entity-suggestion features have?*
   Section 7.5.1 described an experiment aiming to answer this question, where we saw that there was a conflict in the quantitative and qualitative analysis. However, it was observed that in the context of the evaluation, the results from the qualitative analysis are more meaningful, and concluded that there was indeed an improvement in the system's usability. These findings can be related to requirement **NF.1**, and to reaffirm the argument that the system satisfies it.

6. *Do the sentiment analysis results change the user's view of the accuracy of the system?*
   Section 7.5.2 described an experiment where the users' perception of the accuracy of the system was evaluated. The results showed that a change in the sentiment analysis results do not affect the users' perception of the accuracy of the system. It was also pointed out that further experiments are needed in order to fully prove this.

7. *Were sufficient steps taken to ensure that the system is well tested and executes its functions as designed?*
   Sections 7.6 and 7.7 provided a positive answer to this question by outlining the unit, integration, and acceptance tests developed for the system's components.

Chapter 8

Conclusion

The Media Sentiment Analyser aims to provide an easy to use web-based application, which allows anyone to measure and compare the newspaper media's sentiment towards different entities. It was argued that there is a market gap for a product like this, and appropriate requirements were identified for the product.

We saw in Chapters 5 and 6 that the system delivered on the most important functional requirements (the "Must Have" and "Should Have"), and in Chapters 4 and 7 that the non-functional requirements were met as well. This allows the conclusion that the Media Sentiment Analyser is successful in meeting its requirements and delivering value to its users. While further work is required, it can be argued that the Media Sentiment Analyser has a clear potential to fill the identified market gap and help combat media effects, by providing people with the ability to investigate media bias using sentiment analysis.

8.1 Future Work

Should this project be extended in the future, the following subsections identify suggestions for future work.

8.1.1 Continuous Data Ingestion

The current approach of one-time ingestion and batch processing is not viable in the long run. This ties the system to a set of articles for a particular time period, which will eventually become outdated. All of the related products described in Chapter 2 either perform continuous ingestion, or do analysis in real-time. In order for the Media Sentiment Analyser to be up to date with its data, the corpus should be constantly updated by a continuous ingestion of new articles. A data source different than the BBC Labs' The Juicer needs to be identified, which contains a larger corpus of articles. In terms of real-time ingestion, this can be achieved by using RSS to ingest articles directly from newspapers' feeds, index them, and analyse them.

8.1.2 "Could Have" Requirements

It was pointed out in Chapter 6 that the "Could Have" requirements of the system were not implemented. While they do not add to the core features, addressing them in a future iteration would improve the system by introducing features which will make it even more powerful. Moreover, features like **C.1** and **C.4** will help users like **Gregg** and **Patrick** to easily fulfill their needs.

8.1.3 Browse Plugin

In order to make the Media Sentiment Analyser really standout, a browser plugin can be developed, which would allow the user to measure and compare historical and current media sentiment on the fly, while browsing the internet, or reading news online. Because of the decoupled architecture of the system, the only new code to be developed would be the plugin's front end, which could make calls to the existing web API.

8.1.4 UI Improvements and Date Hanlding

Several flaws of the Media Sentiment Analyser were identified in Chapter 7. The date range picker could be improved to allow for the selectino of specific year by choosing from a predefined menu. In terms of date handling, the logic of the application should be updated to reflect the possibility that the user might select a single day, instead of a date range.

8.1.5 Further Research

Even though this project required the development of a software product, the nature of the requirements demanded significant research into topics such as sentiment analysis and word embedding. A number of interesing observations were made in Chapter 7, like the bad performance of sentiment classifiers, which were not trained on news articles, when classifying sentiment in sentences extracted from news articles. It was demonstrated that the News Classifier, which was trained with just a small set of manually annotated sentences performs comparably to the StanfordNLP classifier. Further research into this could constitute of building a larger dataset of manually annotated sentences extracted from news articles, and performing statistical experiments to determine if a classifier trained on this dataset is superior to a classifier like the StanfordNLP one. This would not only help this project by providing the opportunity to use a more accurate classifier, but it would also advance the entire field of sentiment analysis.

Finally, further work on the sentiment analysis of news articles should be done, in order to distinguish between negative sentiment towards the entities, and simply negative news. Some work on this has been done by reseachers [56], and applying it could prove benefitial to the sentiment analyser's success.

8.2 Reflection

This project represents the most comprehensive piece of work I have done so far. I was presented with a lot of challenges and learned a lot along the way. The pragmatic principles of Minimalist Architecture and Pain Driven Development made a lasting impression on me, and I will continue to apply them in my software development career.

Looking back at the project, there are two things I would change if I had to redo it. Firstly, I would have implemented the offline sentiment analysis from the start. I spent a lot of time trying to make it work in real-time, before I eventually decided to take it offline. Secondly, I would make use of the Test Driven Development [57] approach. It would have given me more confidence during development that the code is clean and modular, and would have reduced the need for big refactoring session in order to make sure that this is the case.

Overall, I am very happy to have worked on this project, and I believe it has the real potential to be of use to a lot of people.

Appendices

Appendix A

Introduction and Debrief Scripts For The User Evaluation

A.1 Introduction Script

The aim of this experiment is to investigate the usability of this website. I want to quantitatively and qualitatively measure the usability. In order to to this, I need to ask people to perform some tasks on the website, so I can collect usability data, which I can then analyse. I have prepared a set of tasks for you to complete using this website. They should be simple and should not take much time. I will be observing you while you perform the tasks. I will ask you to think aloud and say anything that crosses your mind, which is related to your use of the website - any frustrations you have, anything you particularly enjoy, etc. Please ask questions if you need to and please let me know when you are finished. I will be sitting next to you and will not bother you. I will note down what you say when you are thinking out-loud.

After you are done, I will ask you to complete a short questionnaire, when you will be able to provide feedback on your experience using this website. Please remember that it is the system, not you, that is being evaluated. You are welcome to withdraw from the experiment at any time. Do you agree to taking part in this evaluation? Do you have any questions before we start?

A.2 Debrief Script

The main aim of the experiment was to investigate the usability of this website. However, I was particularly looking to see whether you figured out to use the daily report feature easily or not, and whether you appeared confused at any time about how to fill in the form.

Do you have any comments or questions about the experiment? Please take a note of my email address and the email address of the module co-ordinator, and please let us know if you have any further questions about this experiment. Thank you for your help.

Appendix B

Introduction and Debrief Scripts For The Market and Products Analysis

B.1 Introduction Script

The aim of this survey is to investigate whether there is a market gap for the product my project is based on - Media Sentiment Analyser. In order to do this, I need to ask people general questions about their preferred way of getting informed, and other questions about their opinions on media bias. These questions follow in the survey. They should be simple and should not take much time. Your response will be anonymously recorded.

You are welcome to withdraw from filling in the survey at any time. Do you agree to taking part in this evaluation? If you have any questions before you start, please direct them to me.

B.2 Debrief Script

The main aim of the experiment was to investigate whether there is a market gap for the product my project is based on - Media Sentiment Analyser.

Do you have any comments or questions about the experiment? Please take a note of my email address and the email address of the module co-ordinator, and please let us know if you have any further questions about this experiment. Thank you for your help.

Appendix C

Introduction and Debrief Scripts For The Sentiment Annotation Task

C.1 Introduction Script

The aim of this task is to collect a dataset for training a sentiment classifier. This is done in order to investigate whether sentiment analysis classifiers trained on manually annotated sentences extracted from news articles perform better than classifiers who were trained on different data. In order to do this, I need to ask people to manually annotate sets of sentences extracted from newspaper articles for sentiment. I have prepared a set like that for you to annotate. This should be a simple task and should not take a lot of time. Your annotations will be anonymously recorded.

You are welcome to withdraw from this task at any time. Do you agree to taking part in this evaluation? If you have any questions before you start, please direct them to me.

C.2 Debrief Script

The main aim of this task was to collect a dataset of manually annotated sentences for sentiment, in order to investigate whether sentiment analysis classifiers trained on manually annotated sentences extracted from news articles perform better than classifiers who were trained on different data.

Do you have any comments or questions about the experiment? Please take a note of my email address and the email address of the module co-ordinator, and please let us know if you have any further questions about this experiment. Thank you for your help.

Appendix D

Signed Assessment Ethics Checklist Forms

Three forms were signed, one for each evaluation/task which required human participants. They appear in the following order: Sentiment Annotation Task, Market and Products Analysis Survey, User Evaluation.

School of Computing Science
University of Glasgow

Ethics checklist form for assessed exercises (at all levels)

This form is only applicable for assessed exercises that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, or getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Department Ethics Committee for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your assessed work.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and PDAs is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.

12. *All the data collected from the participants is stored in an anonymous form.*
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Module and Assessment Name Level 4 Individual Project

Student's Name Veselin Vasilev

Student's Registration Number 2132561

Student's Signature 

Date 15.02.2018

School of Computing Science
University of Glasgow

Ethics checklist form for assessed exercises (at all levels)

This form is only applicable for assessed exercises that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, or getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Department Ethics Committee for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your assessed work.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and PDAs is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

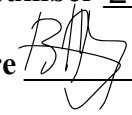
11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.

12. *All the data collected from the participants is stored in an anonymous form.*
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Module and Assessment Name Level 4 Individual Project

Student's Name Veselin Vasilev

Student's Registration Number 2132561

Student's Signature 

Date 12.03.2018

School of Computing Science
University of Glasgow

Ethics checklist form for assessed exercises (at all levels)

This form is only applicable for assessed exercises that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, or getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Department Ethics Committee for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your assessed work.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and PDAs is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.

12. *All the data collected from the participants is stored in an anonymous form.*
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Module and Assessment Name Level 4 Individual Project

Student's Name Veselin Vasilev

Student's Registration Number 2132561

Student's Signature 

Date 12.03.2018

Bibliography

[1] Amazon Web Services, official website. https://aws.amazon.com.

[2] Angular, official website. https://angular.io.

[3] Apache OpenNLP, official website. https://opennlp.apache.org.

[4] Apache Struts, official website. https://struts.apache.org.

[5] AYLIEN News API, official website. https://newsapi.aylien.com/.

[6] BBC News Labs, official website. http://bbcnewslabs.co.uk.

[7] Bootstrap, official website. http://getbootstrap.com.

[8] Colaboratory, official website. https://colab.research.google.com/notebooks/welcome.ipynb#recent=true.

[9] Crimson Hexagon, official website. https://www.crimsonhexagon.com/.

[10] D3, official website. https://d3js.org.

[11] DeepLearning4J, official website. https://deeplearning4j.org.

[12] Django, official website. https://www.djangoproject.com.

[13] Dygraphs, official website. http://dygraphs.com.

[14] Element, official website. http://element.eleme.io/.

[15] Event Registry, official website. http://eventregistry.org.

[16] FinSentS, official website. http://landing.finsents.com/.

[17] Flask, official website. http://flask.pocoo.org.

[18] GATE, official website. https://gate.ac.uk.

[19] Gensim, official website. https://radimrehurek.com/gensim/.

[20] Git, official website. https://git-scm.com/.

[21] GitLab, official website. https://about.gitlab.com/.

[22] Heroku, official website. https://www.heroku.com/.

[23] Highcharts, official website. https://www.highcharts.com.

[24] Jetty, official website. https://www.eclipse.org/jetty/.

[25] jQuery, official website. https://jquery.com.

[26] JUnit, official website. https://junit.org/junit5/.

[27] Karma, official website. https://karma-runner.github.io/2.0/index.html.

[28] Mockito, official website. http://site.mockito.org.

[29] MockupDB, official website. http://mockupdb.readthedocs.io.

[30] MongoDB, official website. https://www.mongodb.com/.

[31] Netbase, official website. https://www.netbase.com/.

[32] Nightwatch, official website. http://nightwatchjs.org/.

[33] Opinion Crawl Blog, official website. http://www.opinioncrawl.net/.

[34] Opinion Crawl, official website. http://www.opinioncrawl.com/.

[35] Pippo, official website. http://www.pippo.ro.

[36] Play Framework, official website. https://www.playframework.com.

[37] Pytest, official website. https://docs.pytest.org/en/latest/.

[38] React, official website. https://reactjs.org.

[39] Selenium, official website. https://www.seleniumhq.org/.

[40] Sentiment Viz, official website. https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/.

[41] Signal Media, official website. https://signalmedia.co/.

[42] Spark Framework, official website. http://sparkjava.com.

[43] Spike, from the Agile Dictionary. http://agiledictionary.com/209/spike/.

[44] Spring Boot, official website. https://projects.spring.io/spring-boot/.

[45] Stanford CoreNLP GitHub Repository. https://github.com/stanfordnlp/CoreNLP.

[46] Stanford CoreNLP, official website. https://stanfordnlp.github.io/CoreNLP/.

[47] Terrier, official website. http://terrier.org/.

[48] Text Analysis Engine, official website. https://aylien.com/text-api/.

[49] The Guardian Open Platform, official website. http://open-platform.theguardian.com.

[50] The Juicer, official website. http://bbcnewslabs.co.uk/projects/juicer/.

[51] Tomcat, official website. http://tomcat.apache.org.

[52] Vue.js, official website. https://vuejs.org.

[53] Alekh Agarwal and Pushpak Bhattacharyya. Sentiment analysis: A new approach for effective use of linguistic knowledge and exploiting similarities in a set of documents to be classified. In *Proceedings of the International Conference on Natural Language Processing (ICON)*, pages 238–247, 2005.

[54] Stuart Allan. News culture . maidenhead, 2004.

[55] Douglas G Altman. *Practical statistics for medical research*. CRC press, 1990.

[56] Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik Van Der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. Sentiment analysis in the news. *arXiv preprint arXiv:1309.6202*, 2013.

[57] Kent Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003.

[58] Kent Beck, Mike Beedle, Arie Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, et al. Manifesto for agile software development. 2001.

[59] John Brooke. Sus: a retrospective. *Journal of usability studies*, 8(2):29–40, 2013.

[60] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[61] Jennings Bryant and Dolf Zillmann. *Perspectives on media effects*. Lawrence Erlbaum Associates, 1986.

[62] Kevin Clark and Christopher D. Manning. Entity-centric coreference resolution with model stacking. In *Association for Computational Linguistics (ACL)*, 2015.

[63] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[64] Lingjia Deng and Janyce Wiebe. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1323–1328, 2015.

[65] DevIQ. Pain driven development. `http://deviq.com/pain-driven-development/`.

[66] Flask Documentation. Foreword. `http://flask.pocoo.org/docs/0.12/foreword/#what-does-micro-mean`.

[67] React Documentation. Introducing JXS. `https://reactjs.org/docs/introducing-jsx.html`.

[68] Vue Documentation. Comparison with Other Frameworks. `https://vuejs.org/v2/guide/comparison.html`.

[69] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 49–54, 2014.

[70] Josh Dowsy, Carol D. Leonnig, and John Wagner. In another blow to Trump's efforts to combat Russia probe, diGenova will no longer join legal team. *The Washington Post*, 2018.

[71] James N. Druckman and Michael Parkin. The impact of media bias: How editorial slant affects voters. *Journal of Politics*, 67(4):1030–1049, 2005.

[72] Irving E Fang. *A history of mass communication*. Focal, 1997.

[73] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[74] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[75] Martin Fowler. Continuous integration. `https://www.martinfowler.com/articles/continuousIntegration.html`, 2006.

[76] Tim Groseclose and Jeffrey Milyo. A measure of media bias*. *The Quarterly Journal of Economics*, 120(4):1191–1237, 2005.

[77] Jon Hopkins. Component primer. *Communications of the ACM*, 43(10):27–30, 2000.

[78] Tanya Hoskin. Parametric and nonparametric: Demystifying the terms. In *Mayo Clinic*, 2012.

[79] Nitin Indurkhya and Fred J. Damerau. *Handbook of Natural Language Processing*. Chapman & Hall/CRC, 2nd edition, 2010.

[80] Vaanchitha Kalyanaraman, Sarah Kazi, Rohan Tondulkar, and Sangeeta Oswal. Sentiment analysis on news articles for stocks. In *Modelling Symposium (AMS), 2014 8th Asia*, pages 10–15. IEEE, 2014.

[81] Hanhoon Kang, Seong Joon Yoo, and Dongil Han. Senti-lexicon and improved naïve bayes algorithms for sentiment analysis of restaurant reviews. *Expert Systems with Applications*, 39(5):6000–6010, 2012.

[82] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM, 2015.

[83] Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164, 2011.

[84] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308, 2014.

[85] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[86] Ruth Malan and Dana Bredemeyer. Less is more with minimalist architecture. *IT professional*, 4(5):48–47, 2002.

[87] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18(1):50–60, 03 1947.

[88] Robert C Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall, 2002.

[89] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[90] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[91] Mozilla Developer Network. Using Media Queries. `https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries`.

[92] Nic Newman, Richard Fletcher, Antonis Kalogeropoulos, David AL Levy, and Rasmus Kleis Nielsen. Reuters institute digital news report 2017. 2017.

[93] Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 104–111. Association for Computational Linguistics, 2002.

[94] Kamal Nigam and Matthew Hurst. Towards a robust metric of opinion. In *AAAI spring symposium on exploring attitude and affect in text*, pages 598–603, 2004.

[95] Donald Norman. The design of everyday things. 1988.

[96] International Institute of Business Analysis and K. Brennan. *A Guide to the Business Analysis Body of Knowledger*. International Institute of Business Analysis, 2009.

[97] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, 2010.

[98] Damian Paletta, Steven Mufson, and Josh Dawsey. Trump prepared to hit China with $60 billion in annual tariffs. *The Washington Post*, 2018.

[99] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[100] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, AL-Smadi Mohammad, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. Semeval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 19–30, 2016.

[101] Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, 2015.

[102] Kevin Quealy. Trump is on track to insult 650 people, places and things on twitter by the end of his first term. *The New York Times*.

[103] Extreme Programming Roadmap. You arent gonna need it. `http://xp.c2.com/YouArentGonnaNeedIt.html`.

[104] Andrew Roth. Kremlin says accusing Putin of ordering spy attack is 'unforgivable'. `https://www.theguardian.com/world/2018/mar/16/kremlin-peskov-accusing-putin-ordering-spy-attack-unforgivable`, 2018.

[105] Norman F. Schneidewind. The state of software maintenance. *IEEE Transactions on Software Engineering*, (3):303–310, 1987.

[106] Richard Socher, John Bauer, Christopher D Manning, et al. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 455–465, 2013.

[107] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[108] Mitchell Stephens. From the drum to the satellite: A history of news, 1988.

[109] Steven Thomas. Pain driven development. `http://itsadeliverything.com/pain-driven-development`, 2012.

[110] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.

[111] Stack Overflow Trends. Most Popular JavaScript Framework. `https://insights.stackoverflow.com/trends?tags=jquery\%2Cangularjs\%2Cangular\%2Creactjs`.

[112] Bill Venners. The simplest thing that could possibly work. https://www.artima.com/intv/simplest.html, 2004.

[113] Justine Zhang Cristian Danescu-Niculescu-Mizil Jure Leskovec Vlad Niculae, Caroline Suen. Quotus: The structure of political media coverage as revealed by quoting patterns. In *Proceedings of WWW 2015*, 2015.

[114] Duy-Tin Vo and Yue Zhang. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*, pages 1347–1353, 2015.

[115] Charles Welch and Rada Mihalcea. Targeted sentiment to understand student comments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2471–2481, 2016.

[116] Don Wells. Acceptance Tests. hhttp://www.extremeprogramming.org/rules/functionaltests.html.

[117] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

[118] Boya Yu, Jiaxu Zhou, Yi Zhang, and Yunong Cao. Identifying restaurant features via sentiment analysis on yelp reviews. *arXiv preprint arXiv:1709.08698*, 2017.

[119] Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136. Association for Computational Linguistics, 2003.

[120] Bu Zhong. *Searching for Meaning: Multi-Level Cognitive Processing of News Decision Making Among US and Chinese Journalists*. PhD thesis, 2006.