Prominent Profiles: A Web App Delivering Target-Dependent Sentiment Analysis in News Articles

Anthony Meek

Supervised by Professor Mark Lee

DA B.Sc. Computer Science with Digital Technology Partnership

School of Computer Science

College of Engineering and Physical Sciences

University of Birmingham

2023-24

Student ID: 2167216

Word Count: 22764

Credits: 40

Abstract

ProminentProfiles.com is a web app created to provide a unique way of discovering news that is free from social media algorithms or the selection bias of news outlet editors.

It delivers diverse perspectives by utilising state-of-the-art target-dependent sentiment analysis, which identifies the sentiment of article text towards a given target, and a variety of Natural Language Processing frameworks to manipulate the data into suitable forms for processing. By analysing over 25,000 articles since November 2023 via an automated process, Prominent Profiles can provide targeted insights into the representations of individuals, typically political figures we call profiles.

Three classifications are shown that reflect the predominant sentiment expressed towards a profile in an article. For transparency, a breakdown of scores and other profiles present in the article is provided. This enables greater exploration of current affairs that are portrayed differently to existing news aggregators. Users can also subscribe to profiles, which allows them to build a personalised dashboard that encourages them to return for the latest insights.

Prominent Profiles has successfully met its aims, with users praising its unique approach to offering unbiased news coverage, ease of use, clean design, and a comprehensive collection of articles concerning influential figures. The ability to track news about specific figures and gain insights from various perspectives has been especially valued.

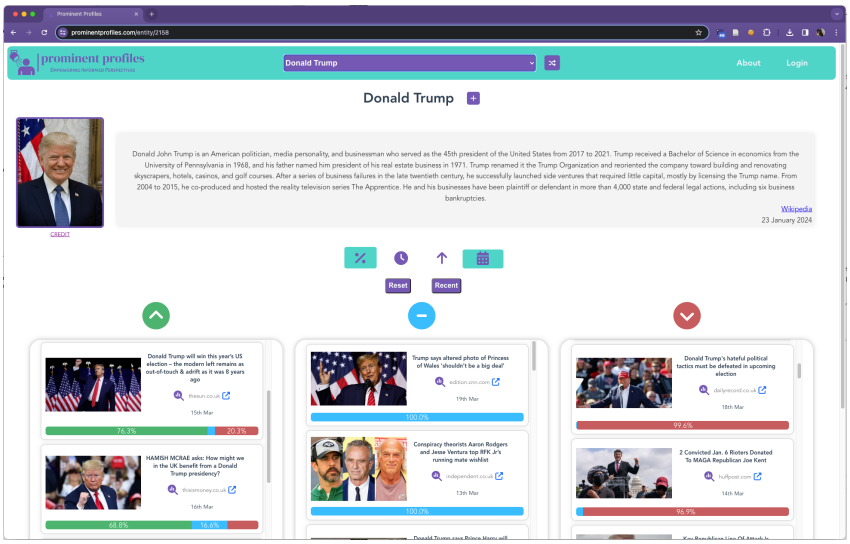


Figure 1: Prominent Profiles - Profile View

Prominent Profiles can be accessed at https://prominentprofiles.com. The Appendix provides credentials for the frontend login and the admin portal.

Acknowledgements

I would like to extend my sincerest thanks to:

- **Professor Mark Lee**, for the support he provided to the evolution of my own concept and his guidance around the approach, including completion timelines for the key aspects. Additionally, his confidence in granting me the autonomy to steer the project's direction has fostered a profound sense of accomplishment within me as this work comes to a close.
- **Dr Mirco Giacobbe**, for providing detailed feedback on my project proposal and arranging a productive group demonstration with other students. This raised beneficial areas for me to address in this report and promoted valuable conversations around Prominent Profiles.
- **Professor Matthew Leeke**, for his ad-hoc project support sessions that were most helpful in shaping my project proposal, demonstration, and composition of this report.
- **My family and friends**, for understanding when my commitment to this work prevented me from spending as much time with them as I would have desired over this academic year. I warmly look forward to making up for the missed moments.

Abbreviations

CI/CD	Continuous Integration and Continuous Development
Coref	Coreference
DFD	Data Flow Diagram
DOB	Date of Birth
Ext.	External
GRU	Gated Recurrent Unit
(L)LM	(Large) Language Model
LSTM	Long Short-Term Memory
NE(R)	Named Entity (Recognition)
NLP	Natural Language Processing
ORM	Object Relational Mapper
PP	Prominent Profiles
Rej.	Rejection
RSS	Really Simple Syndication
SPA	Single Page Application
ToS	Terms of Service
TSC	Target-dependent Sentiment Classification
UI	User Interface
UX	User eXperience
WCAG	Web Content Accessibility Guidelines

Contents

List of Figures

List of Tables

Glossary

Target-dependent sentiment classification (TSC): 'A sub-task of sentiment analysis that aims to identify the sentiment of a text, usually on sentence-level, towards a given target, such as named entities (NEs) or other semantic concepts'[8, 9]

Cohen's Kappa: 'A statistical measure used to quantify the level of agreement between two raters (or judges, observers, etc.) who each classify items into categories. It's especially useful in situations where decisions are subjective [e.g. sentiment analysis] and the categories are nominal'[10].

Cluster Text: A list of the coreference mentions of the entity name attributed with the cluster represented as strings.

Cluster Positions: A list of the coreference mentions of the entity name attributed with the cluster represented as integer $[start\_char, end\_char]$ position pairs relative to the extracted article text.

Bound Mention: A section of an article text body defined by start and end character values. Typically, it represents a sentence, but it may encompass more or less text if the tokenization of the article text is imperfect.

Stop word: Commonly used words in a language that have low semantic value as they contribute little to the overall meaning of a sentence or document, typically serving grammatical functions instead. Examples include 'he', 'the', 'is', 'are'.

Profile: Refers to a prominent person, for example 'Boris Johnson' hence the web app name: Prominent Profiles. Often used interchangeably with Entity but tends to refer to resolved entities on the frontend.

Entity: Refers to a finalised NE from the pipeline, which is expected to be a person, for example, 'John Smith'. Often used interchangeably with Profile but tends to refer to entities (regardless of their resolved status) internally within the database, administrator activities and the backend.

Resolved Entity: An entity that is well-formed and clearly identifiable, such that we can refer to it directly. Examples include 'Alexey Navalny', 'Rishi Sunak', and 'Prince William'.

Unresolved Entity: An entity that is not well-formed so is not always clearly identifiable, due to failed resolution in the NLP pipeline. Examples include 'Rip Alexey Navalny', 'Bruised Sunak', and 'Brother William'.

Authenticated User: A user with a valid access token in the app and is, therefore, logged in.

Paywall: A block on content restricting access to users who have paid to subscribe to a website.

Chapter 1

Introduction

The landscape of news consumption has undergone a radical transformation, driven by the ever-expanding reach of the internet. This evolution is evident in the long-term decline of physical print newspapers, contrasting with the rise of online sources as the second most popular platform for news among UK adults. Notably, 8 in 10 individuals aged 16-24 predominantly engage with news through digital channels; therefore, it's reasonable to anticipate that digital access will become the predominant means of news consumption as this generation matures[11].

Simultaneously, the public's perspective on how news should be selected and delivered is evolving. The 2023 Digital News Report indicates a growing scepticism toward the algorithms employed by social media platforms, with less than a third of respondents believing that the selection of stories based on previous consumption is a good way to get news[12].

Trust in the press has been linked to comparing multiple sources, offering a sense of control and critical analysis. Yet, many find this time-consuming and frustrating[13]. Making it easier for the public to compare sources is crucial, as the media can significantly reinforce pre-existing attitudes in an audience. This reinforcement power can have meaningful consequences, particularly in shaping public misperceptions of various issues[14].

The language writers use, which can predispose readers to certain opinions, is among a plethora of biases in the press. These biases lead to an even greater distrust of articles chosen by editors or journalists compared to that of algorithms[12]. The unfair representations of individuals, e.g., Jeremy Corbyn[15], likely contribute to formulating these suspicions. While those who attempt to circumvent algorithmic and outlet political biases could turn to impartial sources like the BBC, they may perceive these outlets as 'boring' due to the limitations imposed on the expression of opinions[16].

This paper proposes a different approach to discovering and selecting news stories by taking advantage of the advancements in sentiment analysis, which has been one of the fastest-growing research areas since the mid-2000s. It is a series of methods, techniques, and tools for detecting and extracting subjective information, such as opinions and attitudes, from language[17]. Traditionally, it has primarily focused on analysing reviews and social media content, where opinions are explicitly expressed.

However, recent advancements, such as the development of deep language models like BERT (Bidirectional Encoder Representations from Transformers), have enhanced sentiment analysis capabilities. BERT is adept at understanding the context of a word within search queries by examining the words around it, allowing it to capture the subtleties of language. RoBERTa (Robustly Optimised BERT Pretraining Approach) builds upon BERT by applying further pretraining optimisations and is trained on a larger corpus, including news texts[18]. This makes it more suitable for determining sentiments expressed in news articles that present a more challenging data domain due to the implicit nature of sentiments within them[8].

Additionally, the packaging of fine-tuned models through platforms and open-source libraries like Hugging Face and spaCy have made state-of-the-art Natural Language Processing (NLP) more accessible, particularly for academic projects lacking access to high-end hardware[19, 20]. For instance, the 'NewsSen-

timent' package can output a classification and confidence when supplied with sentences featuring key individuals, identified through named entity recognition (NER) and coreference resolution to extract other sentences referencing them[21, 22, 23, 24].

By utilising the above, the web app introduced by this work, Prominent Profiles, will collect, analyse and display articles related to prominent individuals, particularly in politics, segmented by the sentiments expressed toward them. The anticipated benefits are:

1. **Enhanced News Discovery**: Users can explore news stories about prominent individuals without the influence of social media algorithms or selection bias, making it ideal for those who value multiple source comparisons while simplifying the process for those who find it time-consuming.
2. **Diverse Perspectives**: The app offers access to a spectrum of sentiments, challenging biases, promoting open-mindedness and keeping news more interesting than 'boring' impartial outlets.
3. **Transparency**: Users are provided with 'sentiment scores' to make informed decisions about the articles they read, enhancing transparency in news consumption. This may encourage those with prejudice to specific news sources to explore them.
4. **Targeted Insights**: The app identifies sentiments directed at specific individuals, aiding users in understanding how prominent figures are portrayed by different journalists.

Evaluation of this app will determine the extent to which the NLP techniques available, chosen, and applied can successfully deliver these benefits.

Chapter 2

Literature Review

2.1 Paper Research

2.1.1 Origins of Target-Dependent Sentiment Classification

The evolution of sentiment analysis in news can be traced back to papers published in 2007, e.g.,[25]. During that time, sentiment analysis primarily relied on sentiment lexicons to annotate sentiment words and entities within their text corpus. This approach was necessitated by the absence of advanced language models like RoBERTa, which provide more sophisticated capabilities. Early work made use of the co-occurrence of an entity and a sentiment word within the same sentence to infer that the sentiment is linked to that specific entity[25]. However, it was acknowledged that this methodology was not always accurate, and today it would be considered too simplistic.

In 2010, a research paper raised an important motivation for focusing on specific entities in the news. This focus is crucial, as automatic opinion mining systems would often record negative outcomes for entities if they were given a news report that covered a negative news event, even if the entity was actually reflected neutrally or positively by their inclusion[26]. To address this challenge,[26] implemented a strategy to exclude specific category-defining words related to negative news events, such as crises and natural disasters. However, in 2024, this exclusion would not be necessary as modern language models can learn and infer from the training data and the context in which words appear. Additionally, it's worth noting that[26] utilised (1592) **quotes** from articles, which are more subjective and therefore easier to analyse compared to the articles this work will be examining[8].

Efforts made in 2017 to incorporate large-scale news entity sentiment analysis into a multilingual news analysis system, Europe Media Monitor, encountered a significant setback due to the low precision achieved (only 50%)[27]. The poor performance was attributed to the relatively small and unbalanced 'gold standard' dataset used, which contained just 1,274 sentences, with 72% of them being neutral and polar (positive or negative) words often not referring to the nearby entity (a word window was defined). This highlights the critical need for a more extensive and more diverse dataset.

Additionally, in the same research, the Stanford Sentiment Annotator[28], trained on a movie review sentiment tree-bank, was applied, resulting in a performance drop from 80% on the training data to 26% on their dataset[27]. This observation encourages care when designing the NLP processes for Prominent Profiles.

2021 is arguably revolutionary for target-dependent sentiment classification (TSC) research due to two papers **NewsTSC** and **NewsMTSC** both by Hamborg et al.[29, 8]. The latter is discussed here as it introduced improvements, including increased dataset size (from 3,000 to 11,000 annotated examples), better class balance and refined labelling criteria that captured more realistic instances of sentiment expression influenced by subtle word choices. Additionally, the NewsMTSC model demonstrated superior classification performance. 'MTSC' stands for Multi Target-dependent Sentiment Classification, signifying the presence of sentences in the dataset containing two or more entities. This brings an additional challenge of effectively distinguishing the (possibly varied) sentiments expressed concerning each

entity.

In the creation of their dataset, the researchers highlighted several considerations that could be applied to articles obtained for this project, depending on the complexity of its final pipeline:

- Use coreference resolution instead of relying solely on named entity recognition, as using the latter alone would result in missing approximately 30% of relevant target mentions.
- Discard entities if their presence comprises less than 20% of all article sentences.
- Exclude non-person clusters.
- Prioritise the most meaningful mention of a person if they are mentioned multiple times in a sentence.

Measures and rules applied in their annotation process ensured the creation of a high-quality dataset, as evidenced by a Cohen's Kappa (κ_C) of 0.74. These measures included instructing crowdsourced annotators to adopt the authors' perspective to mitigate personal biases, emphasising word choice, utilising a 7-point scale for identifying weak polarity cases, establishing qualification requirements for participants, and filtering out poor-quality or quickly answered responses (e.g., from bots).

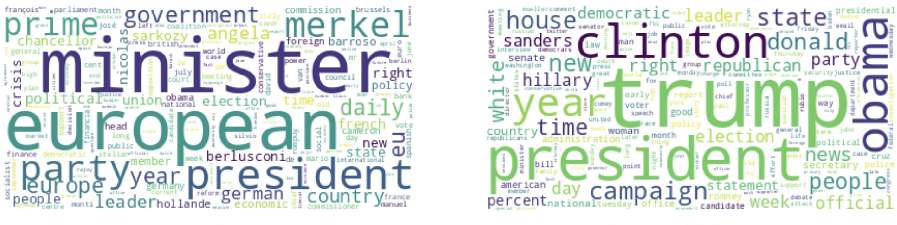
During the consolidation phase, strict criteria were applied, requiring at least four annotator agreements on a 3-class sentiment polarity. As a result, only 50.6% of the data was retained, resulting in a new κ_C of 0.93. Expert annotators further confirmed the dataset's high-quality nature.

The authors explained their model, which utilised a pre-trained language model (RoBERTa), a representation of external knowledge sources (EKS), and a target mention mask in the embedding layer. They conducted an ablation study covering each of these components, leading to the development of a model building on the embedding layer with a bidirectional gated recurrent unit (BiGRU) and a pooling and decoding layer. RoBERTa's strong performance was credited to its pre-training on news and ability to resolve in-sentence relations involving multiple entities better. The final trained and fine-tuned model achieves F1 outcomes of 83.1 and 82.5 on real-world sentiment distributions and multi-target sentences respectively[8].

In July 2023, the paper 'MAD-TSC: A Multilingual Aligned News Dataset for Target-dependent Sentiment Classification was published[1]. While the exploration of various languages and the effort to translate articles into English[1] is intriguing, the primary consideration for this work is the dataset's greater complexity compared to NewsMTSC's.

NewsMTSC only drew its content from American newspapers, resulting in most of its linkable entities (to Wikipedia) being American or, to a lesser extent, British. In fact, 39.5% of mentions in NewsMTSC pertain to the top three entities: Donald Trump, Hillary Clinton, and Barack Obama[1]. In contrast, MAD-TSC sources its content from European-related news, resulting in a top three that comprises only 18.6% of mentions[1]. This distribution is less skewed, and these mentions also link back to entities from across Europe (and America). Furthermore, the average example length in MAD-TSC is 192.3, while MTSC averages 152.2[1].

Overall, MAD-TSC exhibits greater complexity and diversity than MTSC, as the results show that it underperforms in all their experiments. Although the paper mentions some methods achieving results similar to a GRU approach, it does not employ it. Notably, the 'NewsSentiment package by the MTSC team has received multiple updates since 2021, with the latest in December 2023[21]. Hence, investigating its performance on the MAD-TSC dataset could be insightful in gauging its current effectiveness in analysing articles for Prominent Profiles.



(a) MAD-TSC dataset

(b) NewsMTSC dataset

Figure 2.1: Frequent word clouds for MAD-TSC and NewsMTSC corpora (20+ occurrences) from[1]

[1]NLP models typically perform better on English texts due to their geographical origins.

2.1.2 Algorithmic News Consumption

In 2023, online sources were the second most popular means for UK adults to access news, with a usage rate of 68%. Social media accounted for roughly half of this. Meanwhile, online is the most popular means for 16-24-year-olds (83%), with 7 in 10 accessing via social media. Consequently, they prefer to access news via social media rather than visiting news websites directly. In political news, X (formerly Twitter) served as the primary social media source for both groups[11].

X as a primary social media source is of particular concern as [30] uncovered a recurring pattern on the platform, where users with specific political leanings were more likely to be shown information from other similarly leaning users; this favours the emergence of echo chambers, where individuals predominantly in-teract with like-minded content, ultimately constraining information diversity and reinforcing existing perspectives.[30].

The algorithmic woes outlined above have been reflected in public opinion in recent research by Newman et al.[12]. Their findings indicate that people in many countries express dissatisfaction with the selection of content by algorithms. Notably, there has been a significant decline, from 50% to 31% over the past 7 years, among individuals under the age of 35 who agreed with the idea that having stories automatically tailored to their past consumption habits is a beneficial way to receive news.

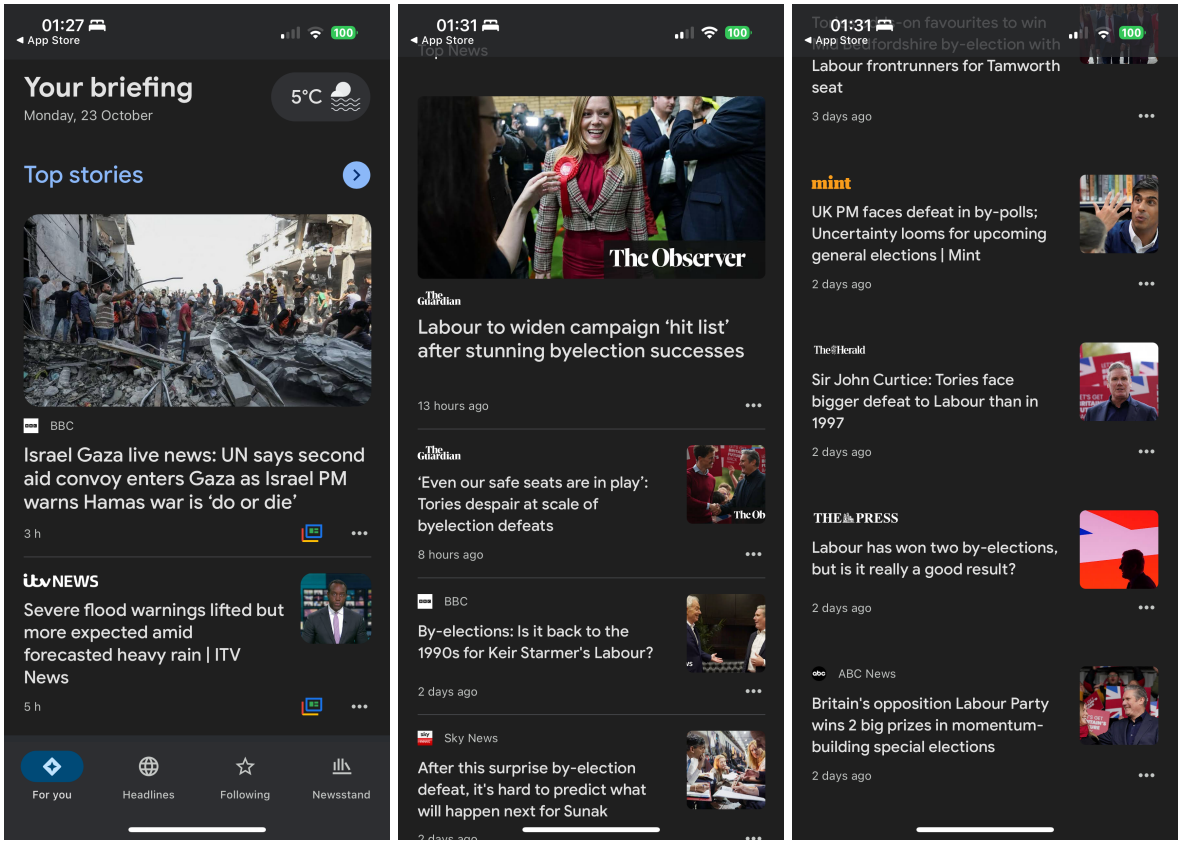
Additionally, their respondents voiced their apprehension regarding the potential for social media platforms to steer them toward information rabbit holes. This unease is more pronounced among young individuals who often find themselves overwhelmed by the predominantly negative nature of news in their social media feeds. Meanwhile, across all countries studied, nearly half of the respondents expressed concern that personalised news might cause them to miss out on diverse viewpoints[12].

Collectively, these findings underline the need for a new web app that aggregates news and provides diverse viewpoints, particularly in politics. This desire is further supported by younger age groups, which show a weaker connection with news brands' own websites and apps compared to previous generations, instead favouring access to news through side-door routes, such as news aggregators[12]. Young people also like having access to information from a variety of sources to help them form a consensus or highlight where there are disagreements, so the design of Prominent Profiles should consider how to achieve this best[13].

2.2 Software Research

2.2.1 Google News

The landing page of Google News (2.2a) initially displays headlines, followed by local news and 'Picks for you.' While the headlines are consistent across user accounts, other sections differ, indicating the presence of personalisation algorithms. To access a 'full coverage' view (2.2b) of any major event[1], select the icon in the bottom right. This feature compiles a variety of coverage from multiple sources. These options offer users a choice of many outlets and headlines, potentially helping them diversify their news diet. However, there is no analysis of the sentiments expressed by the articles or the entities within them. You may find different perspectives if you manually scroll through the headlines (2.2c).



(a) Landing Page

(b) Full Coverage Dashboard for a Selected Story

(c) 'The Press' offers a Different Perspective

Figure 2.2: Google News iOS Screenshots

[1] Minor events do not appear to have the full coverage feature, likely due to a lack of news sources to provide a worthwhile overview.

2.2.2 Awario

Awario is one software solution among several subscription-based commercial platforms, such as Talkwalker, Meltwater, and Mention. These platforms are not designed for the general public; instead, they cater to brands and prominent individuals. Since they are private companies with closed-source systems, it can be challenging to evaluate their feature sets comprehensively. Is coreference resolution applied? What about the state-of-the-art target-dependent sentiment classification models mentioned earlier? Consequently, a comprehensive evaluation of their capabilities is complex.

However, these platforms are included for the sake of completeness, as their concept shares similarities with Prominent Profiles, albeit targeting a different audience. Some insights into Awario's features were revealed in one of their blog posts[31].

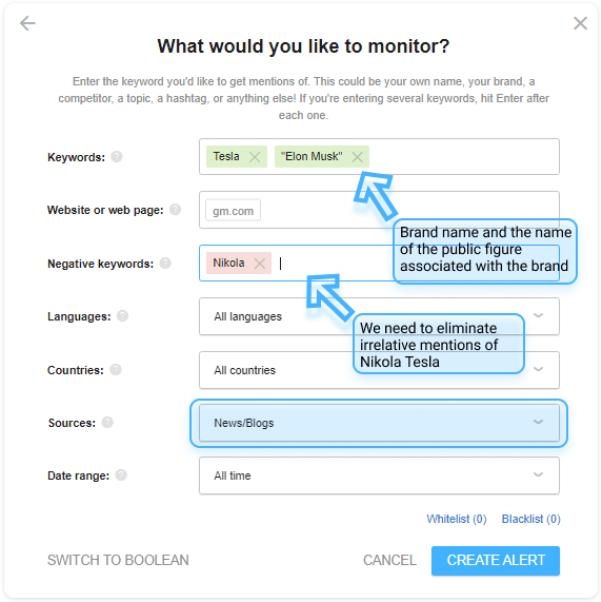
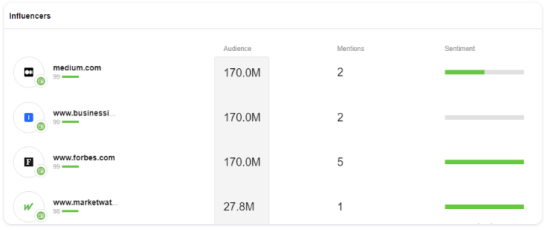
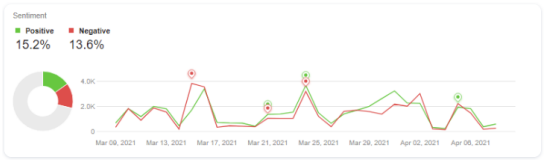


Figure 2.3: Setting up a news monitoring alert[2]

Users are responsible for specifying the terms to search for and identifying any similarly named entities to exclude themselves. A particularly good feature is the scope of mentions it can collect across the web - not just news articles: blog posts, online forums, news stories, Reddit, X, Instagram, Facebook, etc.



(a) List of news outlets where alert was triggered



(b) Sentiments of media mentions for the last 30 days

Figure 2.4: Extracts from the Awario blog

The list of mentions and the accompanying sentiment bars may imply a relatively straightforward background process. Nonetheless, this could be an intentionally simplified example or a deliberate choice to provide a clean and user-friendly interface. A review by Blogging Wizard highlighted that, in their tests, Awario incorrectly categorised a substantial number of mentions, leading to the conclusion that its brand sentiment analytics were 'fairly inaccurate'[32].

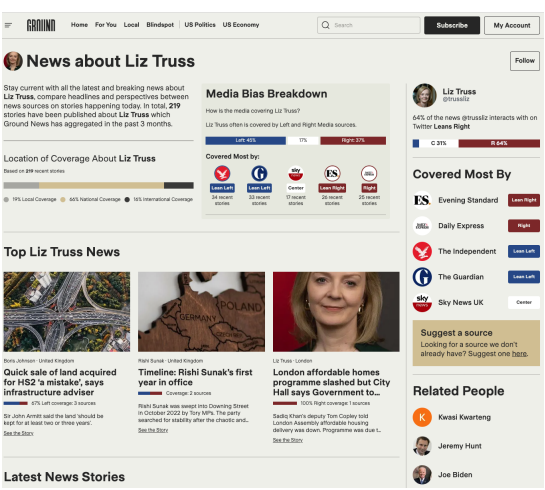
[2]The 'website or webpage' option enables users to discover media that links to their brand or personal website.

2.2.3 Ground News

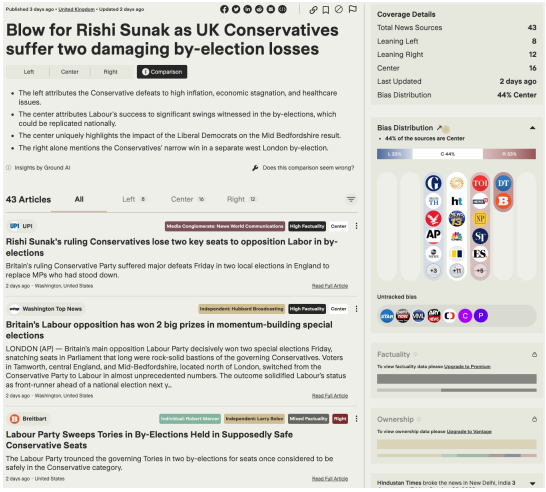
Ground News stands out as a software solution geared towards the general public, providing users with access to news articles from a diverse range of sources across the political spectrum. The creators of Ground News state: 'threading multiple perspectives from thousands of publications through one, reliable information platform, Ground frees people from algorithmic restraints, illuminates blindspots, and makes media's implicit biases explicit.' [33]

Notable features that make Ground News a valuable platform include the aggregation of multiple articles on the same topic in one location, the provision of political bias, factuality and ownership categorisations for news sources, the generation of AI-based story summaries, and an extension that identifies if other sources cover an article you are reading. One standout feature is its 'blindspot™' formula, which identifies news stories with political undertones disproportionately covered by media sources on one side of the political spectrum.

However, Ground News does not appear to include sentiment analysis capabilities for articles or the entities mentioned within them. Consequently, its approach to diverse perspectives relies on the assumption that left-leaning sources will often present right-leaning perspectives negatively and vice versa. While this dynamic commonly occurs, it does not universally hold (Fig: 2.6). For instance, right-leaning publications may criticise conservative political leaders they once supported, leading to a more nuanced media landscape than what Ground News is equipped to handle effectively.



(a) Person Overview



(b) Story Overview

Figure 2.5: Extracts from the Ground News website

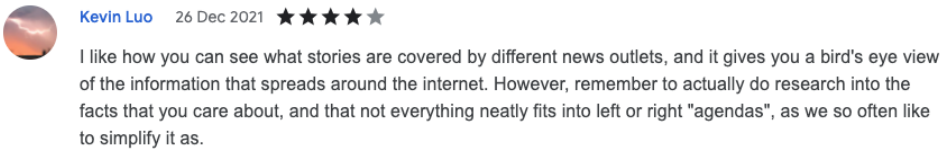


Figure 2.6: A review of the Ground News chrome extension cautioning the assumptions made

Chapter 3

Background

3.1 Pipeline Tools

3.1.1 Article Acquirement

Prominent Profiles is at its most useful and engaging when it is applied to recent articles so there was a need to source these. Two methods to achieve this were compared.

RSS Feeds

News websites commonly offer Really Simple Syndication (RSS) feeds to deliver the latest headlines and associated articles via a feed URL, often to help attract more users[34]. A library of feeds could be used by Prominent Profiles as input for its article analysis.

Bing News Search API

The Bing News Search API is a service offered by Microsoft Azure. It provides developers access to real-time news search results and is marketed as ideal for creating a news aggregator experience similar to Bing News. Ideally, Prominent Profiles is a news aggregator split by prominent individuals that has the unique functionality of applying TSC. It could target political and current affairs by using a regular set of search queries with Microsoft's API[35].

Selection

Utilising the decisional balance sheet (Table. 3.1), Bing was chosen due to the significant resources that Prominent Profiles would have needed to find RSS feeds for a diverse collection of news websites to support a successful news aggregator. Even then, coverage could be missed on certain events covered by less well-known publishers. Moreover, its uniform data structure lends itself to a dependable and repeatable process for retrieving articles. This process can be automated to run regularly and on demand at specific times, unlike the more complex and continuous monitoring required for various RSS feeds. Fears about cost were relieved by the free 1000 requests per month and the $200 student credit available for the first year of use towards paid tiers with greater request and rate limits.

Table 3.1: Decisional Balance Sheet: RSS Feeds vs. Bing News API

	RSS Feeds	**Bing News API**
Pros	– Free to use – Simple to subscribe to (No API keys) – Wide availability – Real-time updates	– Real-time access (send a query!) – Advanced search capabilities – A change in search query provides variation of articles – Provides rich metadata – Consistent data structure and quality
Cons	– Limited search capabilities – Varied data structure and quality – Many feeds required for publisher variety	– Cost for high-volume usage – Rate limits may apply – Reliant on Bing's ranking of articles – So not every article on every topic will be returned

Trafilatura

Trafilatura is a text discovery and extraction tool. It enables main text, comments, metadata extraction, and functionality to start web crawling. However, in this work, only the extraction features are utilised with article URLs sourced from Bing API (3.1.1)[36]. The tool was chosen because it performs significantly better than other open-source solutions in the creator's evaluation and external benchmarks. It was a 2021 ACL contribution and has been used, for instance, to create the RefinedWeb dataset for Falcon Large Language Model (LLM), which released an extract of 600 billion tokens[37] and in an official SemEval-2023 Task 3 submission aiming to Detect the Category, the Framing, and the Persuasion Techniques in Online News in a Multi-lingual Setup[38]. Its performance and successful application in significant papers made it the ideal candidate for Prominent Profiles.

3.1.2 Named Entity Recognition

A key task in preparing the article text to eventually perform TSC requires obtaining the named entities in the text and in the case of Prominent Profiles those that refer to people exclusively[39].

1. Sentence: 'The Birmingham School of Medicine and Surgery was founded in 1825 by William Sands Cox.'
    - The Birmingham School of Medicine and Surgery: Organisation
    - 1825: Date
    - William Sands Cox: Person
2. Sentence: 'The Elizabeth Tower is located in London, UK, and was completed in 1859.'
    - Elizabeth Tower: Location/Organisation
    - London: Location
    - UK: Location
    - 1859: Date
3. Sentence: 'Jennifer bought 100 shares of Vodafone at 69.04p each.'
    - Jennifer: Person
    - 100: Number

- Vodafone.: Organisation
- 69.04p: Monetary Value

4. Sentence: 'The conference will be held on March 20, 2023, at the Edgbaston Park Hotel in Birmingham.'
   - March 20, 2023: Date
   - Edgbaston Park Hotel: Organisation
   - Birmingham: Location

A free open-source library for NLP in Python, spaCy, was selected for this purpose over alternatives (e.g. Natural Language Toolkit (NLTK), Flair etc.) due to its straightforward API, performance optimisation, and in particular its direct support for providing entity mentions via an array of character positions (useful for stitching to other data) allowing work to progress on the more complex areas of development with pace.

3.1.3 Coreference Resolution

For Prominent Profiles to provide TSC across all mentions of an entity across an article, coreference resolution would need to be employed. Coreference resolution consists of identifying textual mentions that refer to the same entity in a given text[23, 40].

1. Mark lost his coat. He searched for it everywhere but couldn't find it.
   Explanation: 'His' and 'it' both refer to 'Mark's coat'.
2. The cake was left on the counter, which disappointed Susan because she wanted to eat it later.
   Explanation: 'It' refers to 'the cake', and 'which' refers to 'the fact that the cake was left on the counter'. 'Susan' is also referred to by 'she'.
3. The students asked the professor when they could submit their assignments. He told them the deadline was Friday.
   Explanation: 'They' refers to 'the students', 'their' refers to 'the students' assignments', and 'He' refers to 'the professor'.

Several approaches were considered. As spaCy was already being used for NER, **NeuralCoref** a coreference resolution module based upon it by Hugging Face that could simply be added to the pipeline appeared ideal; however, it was based on a neural net scoring model developed in 2016, and since then the state-of-the-art has moved on[41, 42]. Further research uncovered **LingMess**, 'a coreference model that significantly improves accuracy by splitting the scoring function into different categories and routing each scoring decision to its category based on a deterministic, linguistically informed heuristic'[24].

Exploring further, **FastCoref** was discovered, which can process 2.8K documents in 25 seconds vs 6 minutes by LingMess with only a tiny drop in accuracy[23]. More importantly, it achieves a lower memory footprint than LingMess by utilising an approach where each span is represented as a function of its start and end tokens. As a result, the model avoids storing vector representations for each of the $O(n^2)$ spans in memory, opting instead to store only $O(n)$ vectors.

Minimising memory requirements appealed in the development of Prominent Profiles as the project's low budget meant a server equipped with plentiful memory or dedicated GPU was unlikely. FastCoref was also conveniently available as a package that could utilise the metal performance shaders (MPS) framework to take full advantage of the GPU on the development device to speed up research in a notebook environment[43]. Finally, it could return arrays of character integers of coreference's that could be used to map them to the tokenized sentences found in article text later in the NLP pipeline. These factors taken together led to opting for it as the coreference resolution library for Prominent Profiles.

[1]This paper used CoNLL 2012 English Test data rather than the English OntoNotes 5.0 dataset used by the others, so it is not directly comparable. Still, the lead in the newer approaches is convincing.

Method	Average F1 Score (%)	Memory Use (GB)
NeuralCoref	65.4[1]	Unpublished
LingMess	81.4	4.6
FastCoref	78.5	3.3

Table 3.2: Average F1 Scores and Memory Usage of Coreference Resolution Methods

3.1.4 Sentence Tokenization

To obtain the appropriate broader context for entities and their coreference mentions the sentence they featured in was required. Sentence tokenization, also known as sentence segmentation, is dividing a text into its constituent sentences[44]. The goal is to identify the boundaries that separate sentences—typically punctuation marks like periods (.), exclamation marks (!), or question marks (?), along with other cues in written language that indicate sentence breaks. NLTK was first given consideration but did not have a simple built-in way to access sentence start and end positions.

TextBlob, builds upon NLTK's foundation and incorporates features from Pattern, provides a neater solution with its `blob.sentences` method[45]. This method not only segments text into individual sentences but also supplies positions in the original text alongside, minimising additional coding, and providing an easy way to link the broader sentence context to the results of FastCoref.

3.1.5 Interval Overlap Identification

In the implementation, it will be necessary to wed the start and end character positions of sentences to mentions of selected entities. A brute-force approach could have been used, where every sentence interval was checked against the query interval to determine whether they overlap. The time complexity of this approach would have been O(n), where n is the number of intervals to check against. This would have been inefficient, especially in the case of longer articles with more intervals.

Interval Tree

An interval tree stores intervals as nodes in a balanced binary search tree, typically organised by the start points of intervals[46]. Each node in the tree also keeps track of the maximum end point in its subtree. This structure allows the tree to efficiently eliminate large portions of the dataset from consideration when searching for overlapping intervals, significantly speeding up query times compared to the brute-force mentioned above approach. It has an initial construction cost of $O(n \log n)$ but a more efficient $O(\log n + m)$ query time complexity, making it well-suited to the many queries involved in checking each entity's coreference cluster position pairs against the sentence boundaries of an article[46]. `intervaltree` provides a mutable, self-balancing interval tree for Python and is therefore utilised in implementing Prominent Profiles[47].

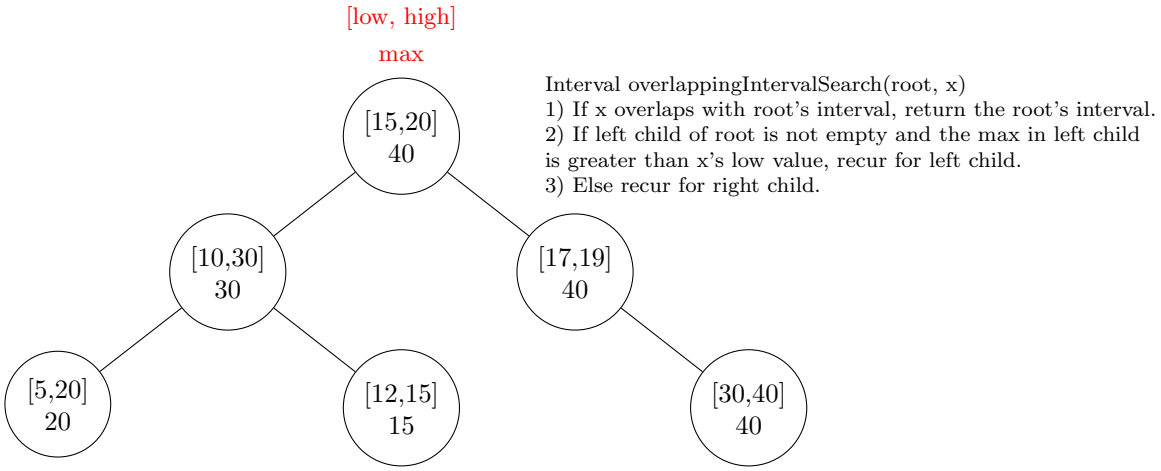


Figure 3.1: Interval Tree Example from [2]

3.1.6 NewsSentiment

This project aimed to deliver a full-stack web app incorporating TSC. In the project timescales, creating a TSC model with worthwhile performance alongside fulfilling other requirements would have been unrealistic. The developer was more enthusiastic about making a full-stack app as it aligns with their future career goals, and they did not believe they would be able to replicate the results of the referenced papers. Indeed, project resources would have been an issue: In 2023, Dufraisse et al. used the Nvidia A100, which costs approx $10,000, and despite this, still used 780 GPU-hours for training and fine-tuning their models[1].

Thankfully, during project research, NewsSentiment was discovered. It is 'an easy-to-use Python library that achieves state-of-the-art performance for TSC on news articles', and it was reassuringly created by the authors behind the ACL 2021 paper NewsMTSC (2.1.1)[21]. On their GitHub page, there is a strong recommendation to use the package, delivering their GRU-TSC model over training one. The inputs must be in a (left, target, right) form hence the requirement for NER, coreference resolution, sentence tokenization and an interval tree.

Table 3.3: Example Use of TSC on a Multi-Target Sentence

	Target 1	**Target 2**
Sentence	Peter's excellent plan has been impacted by Paul's dreadful delivery of it.	
Left Segment		Peter's excellent plan has been impacted by
Target Segment	Peter's	Paul's
Right Segment	excellent plan has been impacted by Paul's dreadful delivery of it.	dreadful delivery of it.
Class Label	Positive	Negative
Class Probability	0.576	0.982

As this package is fundamental for determining the scores displayed on Prominent Profiles, we will evaluate its performance on a range of annotated datasets.

3.2 NLP and Artificial Intelligence

We explored the underlying AI approaches in our selected packages to understand how they support our work.

3.2.1 Token Embedding

Tokenization is breaking down a text into smaller meaningful lexical units that are given numerical representations for input into architectures, e.g. transformers. The choice of lexical units is consequential. Word-level can 'blow up' the vocabulary if lots of unknown words are encountered, which will increase model complexity in later tasks[48]. While char-level won't encounter unknown words, a character itself lacks meaning[48]. Subword-level provides a compromise by splitting a word into 2+ parts. We studied two approaches to sub-word tokenization:

WordPiece

Training Phase:

1. Get all unique words from text and count occurrences.
2. Get all unique characters.
3. Select pairs of consecutive characters that when combined, most improve the overall likelihood of the training data (by reducing entropy).
4. The picked pairs become single tokens and are added to the vocabulary.
5. Repeat until a predefined vocabulary size is met[49].

Inference Phase: When word piece is applied to new text, it tokenizes the input using the learned subword vocabulary by selecting the longest subword for each part of the input text[49].

Byte Pair Encoding

Byte pair encoding differs from WordPiece because it considers the most frequent pairs of characters rather than joint probability. The inference is by following the order of merges applied in its training process rather than simply using the longest subword in the vocabulary[49, 50].

3.2.2 Transformers

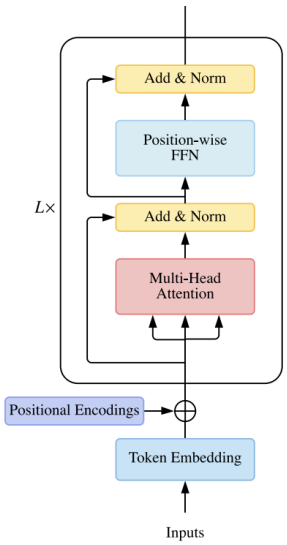


Figure 3.2: Encoder from [3]

Transformers are a type of neural network architecture that addresses the sequentiality requirement in long short-term memory networks, which are inefficient as they prevent use of the parallelisation capabilities offered by today's hardware[51].

They come in 2 varieties. An encoder processes input data, converting raw text into a fixed-sized vector representation capturing some feature of the text. A decoder is typically applied after an encoder to return the vector representation to text. We focus on encoders since the answers to problems we encounter in our work, like sentiment analysis, are represented by a number (class) and some probability or, in coreferences, the character positions, not by text generation or translation.

As transformers process in parallel, they require some way of understanding the positions of each word in the sequence. A positional encoder adds a vector to each input embedding.

[51] used sine and cosine functions of different frequencies for even and odd token positions respectively to do this:

$$\mathrm{PE}(i, 2k) = \sin\left(\frac{i}{10000^{2k/D}}\right)$$

$$\mathrm{PE}(i, 2k+1) = \cos\left(\frac{i}{10000^{2k/D}}\right)$$

By choosing functions that ensure orthogonal representations, the positional encoding for one position does not correlate with the positional encoding for another, supporting distinct representations.

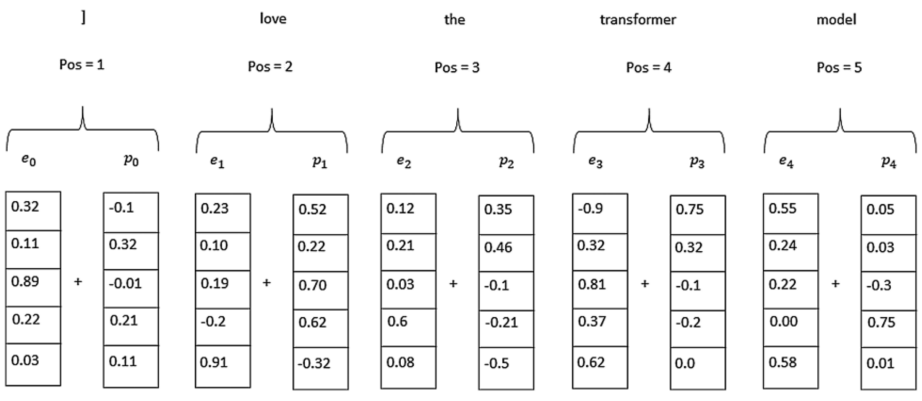


Figure 3.3: Positional Encoding from [4]

Next, is the Attention Layer. Every word can be given an attention vector to capture its contextual relationships with other words in the sentence (including itself).

For a scaled dot-product attention we have vectors[51]:

- Q: Queries - Representations to 'ask' how relevant is each other word to this one

- K: Keys - Queries are compared against these representations of each word via dot product.
- V: Values - These values are reweighed according to the relevance determined by the comparison of Q and K.

The initial representations of (Q, K and V) are determined by applying a trained matrix of weights to the initial word embeddings. The dot products of queries and keys are divided to ease the gradient vanishing problem that softmax is vulnerable to[3].

$$Z = \text{softmax}\left(\frac{QK^T}{\sqrt{\text{dimension of vector Q,K or V}}}\right)V$$

Consider this sentence: 'The bank of the river'. The query (Q) for 'bank' will interact with the keys (K) for all the words in the sentence. But, the relevance scores resulting from the dot product will be higher for 'river' due to the semantic connection, which influences the contextually adjusted output (V) for 'bank.' This enables the transformer to avoid interpreting 'bank' in the financial sense.

The problem is that each word rates its attention with itself much higher, so we compute several attention vectors for each word. Hence, this is called the multi-head attention block[51]. To ensure each attention vector is computed from a different perspective, multiple weights are held in matrices W_v, W_k, W_q. We take a weighted average using another weight matrix (W_z) to get a single attention vector per word.

A feed-forward net is then applied to each word's attention vector to transform outputs into a form the next encoder block can understand. Each attention net is independent of the others, allowing parallelisation[51].

Normalisation is utilised after the multi-head attention block and after the feed-forward block. A batch normalisation could be used, which smooths the loss surface to make it easier to optimise while using larger learning rates (other normalisations can be beneficial though[52]).

Overall, in a traditional RNN, if we were encoding words to vectors, we would pass **one word** at a time, but with a transformer encoder, we can pass **words** in and determine embeddings simultaneously.

3.2.3 Large Language Models

BERT

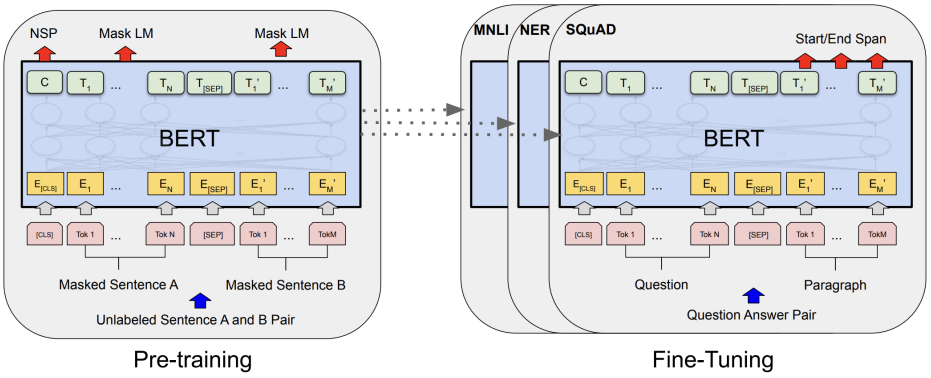


Figure 3.4: Positional Encoding from [5]

Bidirectional Encoder Representations from Transformers (BERT) took multiple transformer encoders and stacked them together[5].

BERT leveraged masking. It works by replacing 15% of the words in a given text and tasks the model with predicting the correct word that has been masked [53]. As it is forced to use the surrounding words to predict the original word, it learns the context of the words in a sentence (or more) in both directions, unlike a traditional embedding, where each word has a static representation regardless of its context [53]. Masking is convenient since it does not require annotation.

Once each input token has passed through the multiple layers of encoders, it receives a probability distribution across the entire vocabulary. The goal is to predict the missing word by minimising the cross-entropy loss between the predicted probability distribution and the actual word. Backpropagation and Adam[1] updates parameters. Iteration of this process then improves BERT's ability to predict masked words[5].

RoBERTa

As the Robustly Optimised BERT Pretraining Approach (RoBERTa) is involved in our selected FastCoref and NewsSentiment packages (3.1.3, 3.1.6), we will highlight their approaches.

Three downstream tasks were used to evaluate their optimisations, including SQuAD, which provides a paragraph of context and a question that must be answered using a relevant span from the context. The particulars of the downstream tasks are not essential; they just need to be consistent.

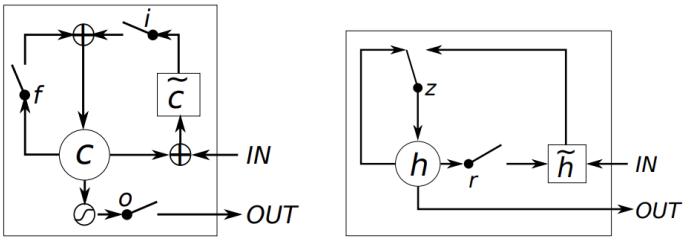
Firstly, the training data used is 10x that of BERT, and interestingly, for our purposes, CC-News, a dataset of English News of 63 million articles, represented 76GB of the total 160GB of uncompressed text leveraged. Second, when training, they find dynamic masking where the pattern of hiding words is generated each time a sequence is fed to the model, and results find this is crucial when using more steps or larger datasets. Thirdly, they find that BERT's other pre-training task of next sentence prediction where given 'two concatenated document segments' the model must predict if they are from the same or different source, is not beneficial to improving downstream task performance over simply filling the input with contiguous full sentences. Fourthly, larger batch sizes with the same amount of data per epoch result in better outcomes. Finally, they use byte pair encoding due to its universal nature[2]. RoBERTa achieves approx. 4% points higher F1 than BERT[18].

3.2.4 Target Mask

NewsSentiment utilises a target mask in its embedding layer to draw the model's attention to the particular person we are seeking sentiments towards by marking 'each word piece i in the sentence that belongs to the target, mi = 1, else 0'[8]. Notably, adding coreferences to the mask when using RoBERTa did not help performance (yet it did with BERT)[8].

3.2.5 Gated Recurrent Units

A traditional RNN can't retain long term memory and it is highly vulnerable to the vanishing gradient issue. Long short-term memory (LSTM) networks where the original solution but they have a more complex structure and require more resources than a newer solution: Gated Recurrent Units (GRU)[6].



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 3.5: Structures of LSTM and GRU from [6]

r: Reset Gate:

- Behaves like logistic regression.
- Determines the amount of past information to be kept or forgotten in the new candidate hidden state.

[1] An extension of stochastic gradient descent (SGD)[54]

[2] 'Google never open-sourced its implementation of the training algorithm of WordPiece' [55])

- Produces values between 0 and 1 through a sigmoid function:

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$

where σ is the sigmoid function, and a value close to 0 signals to drop previous information.

z: Update Gate:

- Behaves like logistic regression.
- Decides what information to keep or discard, providing new inputs for the final hidden state:

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$

$\tilde{h}$: Candidate Hidden State:

- If the reset gate output is close to zero, then the candidate hidden state will disregard more information.
- Calculated using the tanh function, providing values between -1 and 1:

$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \cdot h_{t-1}) + b_h)$$

h: Hidden State:

- A weighted sum, where the update gate values scale the contribution of the new candidate state:

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$$

- This shapes the GRU's memory at each time step.

NewsSentiment combines two independent GRUs to create a Bi-directional GRU where the input sequence is fed in forward order for one GRU and reverse order for the other. Two hidden states are obtained, one for each direction, and they are concatenated before the final prediction is made[56, 8]. This preservation of information from both past and future helps understand the context better (recall aforementioned masking to train BERT), which is essential for their target-dependent sentiment classification model to perform well since the words leading up to the target, e.g. 'Rishi' and those that follow can both impact the reader's feelings towards him.

3.3 Web App Technologies

3.3.1 Backend Frameworks

The back-end of a web app handles server-side logic, database management, and authentication and serves as the backbone for processing business logic; for instance, in this work, fetching and processing articles through scraping and applying NLP to evaluate sentiment towards key entities. It will receive requests from the front end, process them by interacting with databases or other services, and then send the appropriate responses back to the front end. This includes managing user data, ensuring security through authentication and authorisation, and enabling dynamic content generation based on user interactions or changes in the database.

Django

Django, a high-level Python web framework, is designed for rapid development and is 'fully loaded' so everything needed for a typical web app is already installed[57]. It features database agnosticism through its object relational mapper (ORM), enabling easy database switches without altering Python code. Additionally, it emphasises security with protections against SQL injection, cross-site scripting, cross-site request forgery, and uses strong hashing for passwords, and supports HTTPS[58]. It is ideal for developers prioritising efficiency, flexibility, and robust security in web applications.

Flask

Flask is a Python web framework that provides tools, libraries, and technologies to help build a web app. However, unlike Django above, it is a micro-framework, so it has little to no dependencies on external libraries[59]. Positively, this makes it lighter with few dependencies to update that can introduce security flaws, but negatively, it means there are features missing that would require developer time to implement manually or to source plugins to provide the desired functionalities[60]. For example, Flask lacks a built-in ORM; developers often use SQLAlchemy. Depending on developer preference, Flask's minimalist and extensible philosophy allows for flexibility in adding capabilities.

Fast API

Fast API is another micro Python web framework with a focus on building APIs and microservices. It leverages standard Python-type hints for validation and serialisation, contributing to its speed and efficiency. FastAPI automatically generates API documentation and supports asynchronous programming, making it highly scalable. Its simplicity and performance advantages make it ideal for projects requiring fast API development, with a growing community and evolving documentation. On the other hand, it lacks ORM, the extensive built-in-security measures of Django, and its community is small (albeit growing)[61].

Selection

Considering each framework, Django became Prominent Profiles' backend framework. Django's built-in ORM for database interactions, robust security features, and extensive libraries for handling tasks like creating RESTful APIs, user authentication, session management, and customisable admin interfaces were ideal for this project, where project requirements would split development time between the NLP process and the full stack web app development[60]. Plus, we aimed to schedule processes, and Django offered several third-party packages, such as Celery, that integrate well for this purpose. Lastly, the developer had some prior experience with Django in a module, so its learning curve did not put him off, and the scale of documentation and community to support the implementation and any errors that arose were also appealing to them.

3.3.2 Frontend Frameworks

The front end of a web app is responsible for everything the user experiences directly: the layout, the interactivity, and the visual presentation. It involves the design of user interfaces, the structuring and styling of content, and the responsiveness to different device screens. Technologies like HTML for structure, CSS for styling, and JavaScript for interactivity create a UI that allows users to interact with the web app's data and services in real time.

It is only in the 2010s that web development frameworks emerged for building dynamic and complex web applications, wrapping the technologies above into a collection of pre-written code and tools enabling a set of reusable components, libraries and utilities to improve development processes[62, 63]. React, Angular and Vue are the most popular today, so they were considered for Prominent Profiles' front-end[64].

React

Facebook released React in 2013. It is a declarative, component-based JavaScript library for building user interfaces. It allows developers to create large web applications that can change data without reloading the page. Its key features include a virtual DOM (Document Object Model) for efficient updates and JSX for writing UI components[65].

Angular

Google released Angular in 2010. It is a platform and framework for building single-page client applications using HTML and TypeScript[66]. It offers a comprehensive solution with tools for routing, forms, and more, following the MVC (Model-View-Controller) architectural pattern.

Vue

Vue was created by Evan You, who worked at Google on AngularJS before creating Vue in 2014. It is a progressive JavaScript framework used for building UIs and single-page applications[67]. It is designed to be incrementally adoptable, focusing on simplicity and flexibility. Vue's core library focuses on the view layer only, which is easy to pick up and integrate with other libraries or existing projects.

Selection

Vue became Prominent Profiles' frontend framework. Vue tends to be more suited to smaller, less complex apps than React or Angular. Furthermore, Vue offered the developer a gentler introduction to advanced JavaScript concepts yet still supported their future career aspirations to work with the similar, more complex React. Opting for Vue also ensured that project resources could be more effectively allocated towards developing the NLP pipeline, Django backend, and deployment strategies, avoiding the overcommitment that might have resulted from selecting the much larger Angular for this project's scope.

Development Methodology

Feature-driven Development (FDD) embraces Agile software development and feature-driven design, which aligns well with the project's aim of incrementally adding functionalities like article acquisition, text extraction, NER, and coreference resolution to build the pipeline and later deliver the frontend capabilities, including trending profiles, article analysis overviews, and user accounts, in a more manageable fashion.

To incorporate FDD, we utilised a Kanban board on Monday.com, structured into five swim lanes:

- Not Started: A backlog of tasks to complete.
- Working On It: Tasks currently in progress.
- Stuck: Tasks hindered. We always provide a note when moving a task here so we can easily recall the problem in the future.
- Done: Finished tasks to keep track of our completion rate.
- Nice to have: Non-essential, low-priority features out of scope until the backlog is clear.

Each feature was typically assigned a due date and priority to organise the workflow and ensure adherence to milestones.

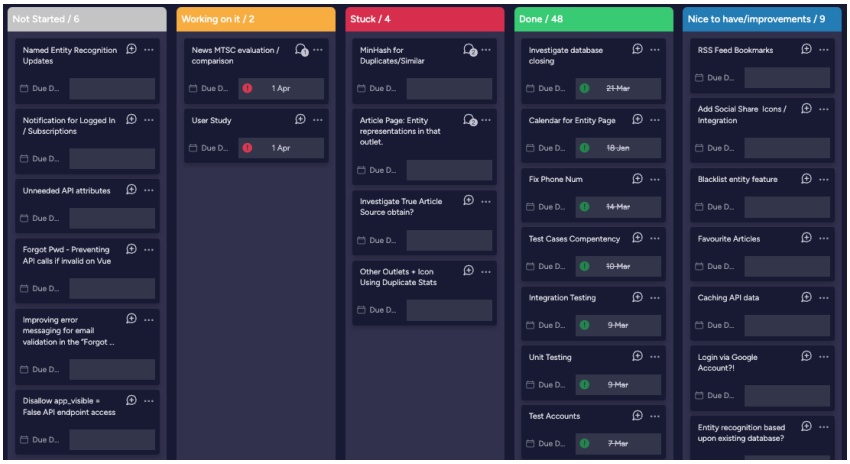


Figure 3.6: Kanban Snapshot

Chapter 4

System Requirements

We established an initial set of functional (what the system should do) and non-functional (how the system should do it) system requirements to guide the design and development of Prominent Profiles and help it achieve its anticipated benefits (1). We utilised MoSCoW, an acronym, to structure them [68]:

- **Must have:** Non-negotiable requirements; any failure is a failure of the entire project.
- **Should have:** Important, valuable features to include if possible.
- **Could have:** Nice to have less critical features than should's.
- **Won't have:** Items that have some significance, but they will not be implemented in the current software, reserved as ambitions for future work.

In line with our Agile approach, these requirements were reviewed and adjusted in response to unexpected development challenges and user feedback. This iterative process ensured the project aligned with user needs and practical constraints. As new requirements emerged, we tended to categorise based on their priority: Low-priority items were classified as 'Could have', Medium as 'Should have', and High or Critical as 'Must have'. Those that became blocked tickets became 'Won't have'. This approach enabled a flexible and responsive development environment that was ideal for shaping Prominent Profiles into a robust and user-centric software solution.

Requirement Type	MUST	SHOULD	COULD	WON'T
Functional	34	25	18	2
Non-Functional	19	18	7	0

Table 4.1: Summary of System Requirements

The full categorised list of requirements can be found in the Appendix (12.5).

Chapter 5

Design

5.1 Architecture

The Agile framework, Feature-driven Development, is the core design strategy for Prominent Profiles as the list of features to implement is extensive. It provides a systematic approach to managing and delivering features efficiently, and allows the app to provide increasing value to users with each iteration. Github Actions, a continuous integration and continuous development (CI/CD) platform that allows the automation of building, testing and deployment, compliments this approach working in the background to seamlessly implement new functionality in the production environment. A local development and production environment have been established. The local configuration differs only slightly for simplicity, resource efficiency and a faster startup by using a lightweight database and excludes the docker containers supporting automation of regular jobs. This means code across the stack behaves near identically in either setup. Docker-compose supports this implementation.

- **Development Environment:** Utilises SQLite3 for database management and it where the initial coding, testing, and iteration occur driven by the Kanban board. It is more lightweight and flexible, allowing for rapid changes and simpler setup for developers.
- **Production Environment:** Relies on a robust PostgreSQL database for handling queries in production. The finalised code runs here, serving the end users via the web.
- **CI/CD Pipeline:** Github Actions builds fresh images of the software, performs automated testing, and deploys the latest changes to the production server. Each stage is dependent on its predecessor to ensure only successful code overwrites the webs servers previous deployment.

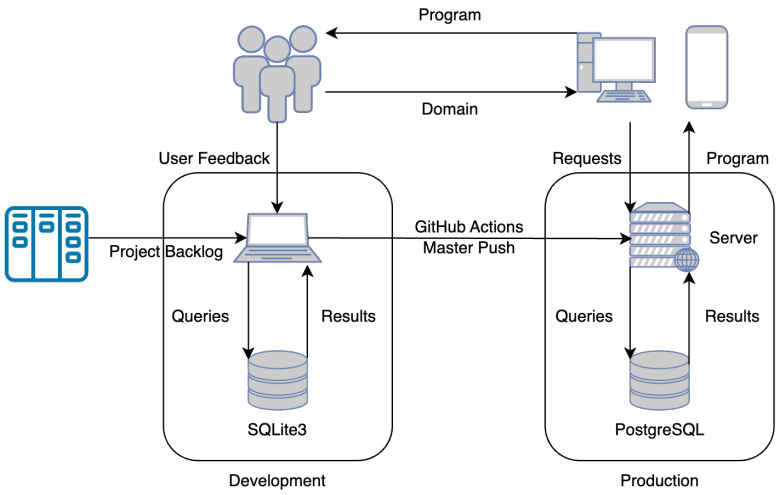


Figure 5.1: Prominent Profiles' High Level Architecture

5.2 Data

5.2.1 Database

The database design phase was continuously re-visited as part of agile development or when unexpected challenges arose. Figure 5.2 shows the initial concepts from the Project Proposal before any coding took place.

Articles Table

ArtID	HeadLn	Image	URL
739	Rishi Sunak defends scepticism of Boris Johnson's lockdown strategy during covid inquiry	enquiryc19.png	www.GenericNews.com

Entities Table

EntID	EntityName	Type
1	Boris Johnson	Politician
2	Rishi Sunak	Politician
3	Nicola Sturgeon	Politician
4	Keir Starmer	Politician

Mentions Table

MtnID	ArtID	EntID	Class	Confidence	Sentence
1	739	1	Positive	0 to 1	'...'
1	739	1	Positive	0 to 1	'...'
2	739	1	Negative	0 to 1	'...'
			⋮		
8	739	2	Positive	0 to 1	'...'

Figure 5.2: Database Concepts - Project Proposal

Contrast this with Figures 5.3 to 5.5 showing the current database schema. Both aforementioned environments share this structure, which is crucial as the web app content relies on daily data population jobs and API calls that specify database model attributes. When changes are made migrations are automatically applied to successful production builds to ensure consistency.

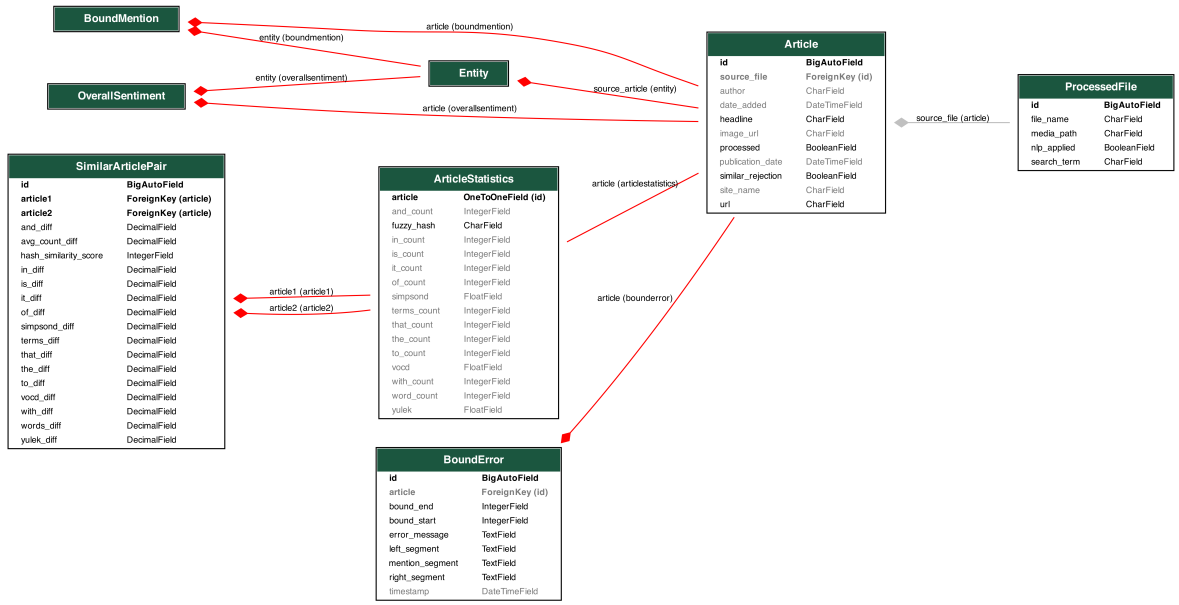


Figure 5.3: Article Centred Models

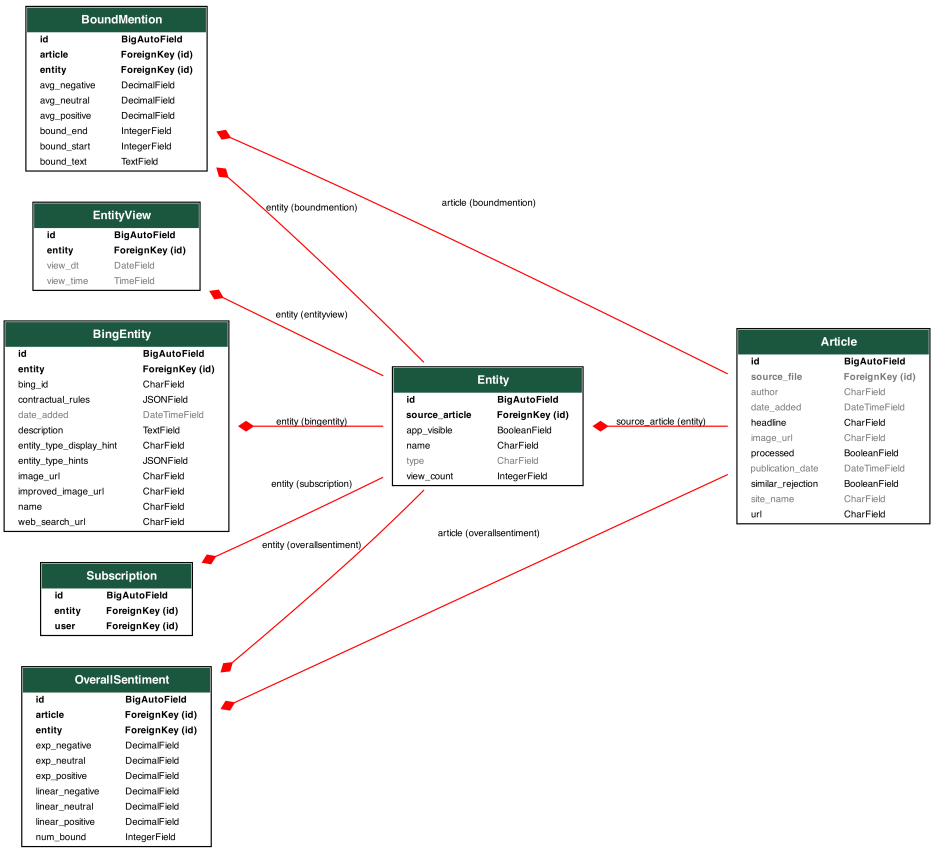


Figure 5.4: Entity Centred Models

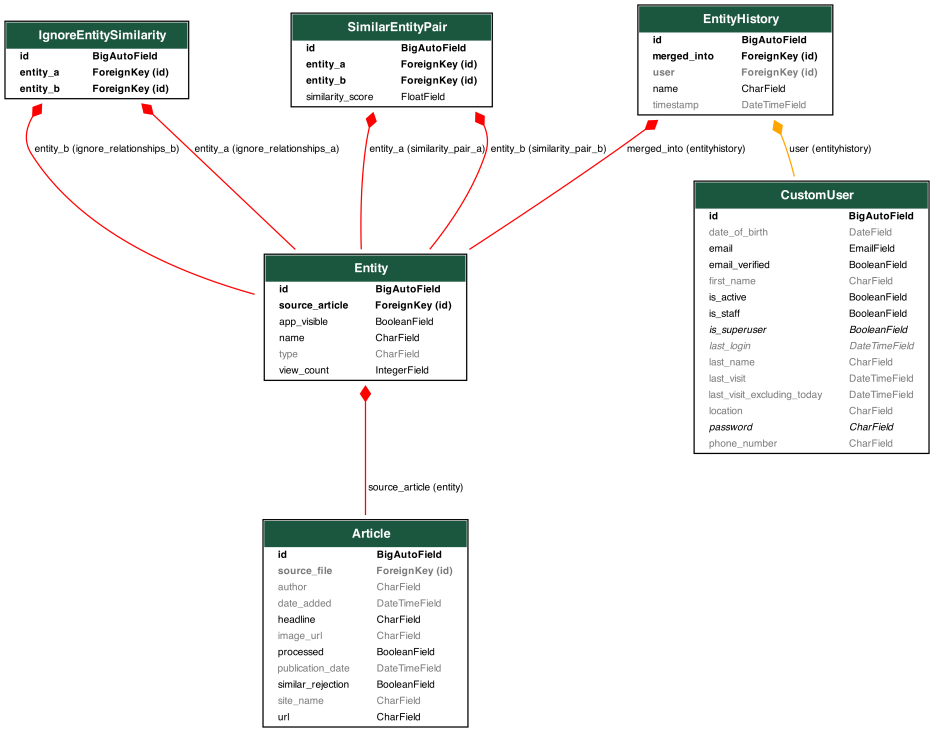


Figure 5.5: Entity Similarity Models

Django App	Model Name	Purpose	Inception
profiles_app	Article	Store key article metadata (e.g., URL) and whether it has been processed or rejected based on similarity grounds.	Early Dec 23
profiles_app	Entity	Holds name, article obtained from for debugging[1], and whether to display in the front-end app.	Early Dec 23
profiles_app	BoundMention	Tracks mentions of entities within articles with sentiment scores.	Early Dec 23
profiles_app	OverallSentiment	Stores the outcome of linear and exponential score determination of all BoundMentions for a given entity and article pair.	Early Dec 23
profiles_app	BingEntity	Stores Bing Entity API Search outcome for an Entity (name) lookup.	Early Dec 23
profiles_app	EntityView	Records views of entities to facilitate trending entities on homepage[3].	Early Jan 24
profiles_app	SimilarEntityPair	Identifies unique pairs of entities considered similar (rapidfuzz score), for admin merge review portal.	Late Jan 24
profiles_app	IgnoreEntitySimilarity	Records pairs of entities to skip in future similarity check jobs.	Late Jan 24
profiles_app	EntityHistory	Records entity merges; this means the former took all the latter's linked sentiment data. Logs admin for accountability.	Late Jan 24
nlp_processor	ProcessedFile	Provides media file locations to scrape + NLP job and marks when completely processed (enabling repeats in failures).	Early Dec 23
nlp_processor	ArticleStatistics	Records a fuzzy hash and specific linguistic stats for determination of SimilarArticlePairs.	Late Jan 24
nlp_processor	SimilarArticlePair	Created if the similarity criteria are met and will prevent wasteful NLP analysis of suspected duplicate articles[2].	Late Jan 24
nlp_processor	BoundError	Tracks exceptions raised when applying NewsSentiment by each article, for improvement of the NLP pipeline.	Late Jan 24
accounts	CustomUser	Extends Django user to store location, phone, last_visit, DOB to enable phone login (UK only), custom dashboard functionality, and GDPR compliance.	Late Dec 23
accounts	Subscription	Allows users to subscribe to an entity to view in their personal dashboard.	Mid Jan 24

Table 5.1: Summary of Key Data Models

The purpose of the earlier incepted tables grew with that of the later tables created in response to challenges in the implementation. Examples include:

- **Article**: Expanded to include a boolean, `similar rejection`, once **ArticleStatistics** and **SimilarArticlePair** were created, and their respective logic implemented. Additionally, **Article** was modified to add a `processed` boolean so if the pipeline was interrupted, it could be picked up in the next job run.
- **BingEntity**: Modified to add the field `improved_image_url` once a fix for low-quality images was identified. Also, **BingEntity** was modified to add `date_added` to be displayed next to the entity description on the front end and provide future ability to auto-expire.
- **CustomUser**: Gained `last_visit` and `last_visit_excluding_today` fields to support registered user subscription dashboard functionality.

[1] E.g. If an entity is unresolved, examining the article text and NLP process in the notebook may help make incremental improvements to entity resolution.

[2] For duplicate articles already analysed, before the Scrape Articles + NLP pipeline integrated ArticleStatistics creation and SimilarArticlePairs checks, only the oldest article of the suspected duplicates will be returned in API calls.

Entity	**Related Entity**	**Fields**	**Relationship**	**Reasoning**
Article	ProcessedFile	source_file	Many-to-One	Many articles will be obtained from each Bing API json.
Entity	Article	source_article	Many-to-One	An entity is associated with its origin article. Entities can originate from the same article.
IgnoreEntitySimilarity	Entity	entity_a, entity_b	Composite Many-to-Many	Constrained to unique pairs of entities marked by admins.
EntityHistory	Entity	merged_into	Many-to-One	A entity record can have many past merges of other entities data into itself.
EntityHistory	CustomUser	user	Many-to-One	One admin user can make many entity merges (that create these records).
SimilarEntityPair	Entity	entity_a, entity_b	Composite Many-to-Many	Pairs of entities considered similar are unique.
EntityView	Entity	entity	Many-to-One	An entity can have many web app hits/views.
BingEntity	Entity	entity	Many-to-One	A bing entity can be linked to several entities (with good admin practice this is always one-to-one)
BoundMention	Article, Entity	article, entity	Many-to-Many	Many mentions of entities within many articles and their bound mentions.
OverallSentiment	Article, Entity	article, entity	Composite Many-to-Many	Many articles will be linked to entities but each (entity, article) will have only one set of sentiment values.
ArticleStatistics	Article	article	One-to-One	Each article has only one text body so there will only be one collection of fingerprinting statistics.
SimilarArticlePair	Article -Statistics	article1, article2	Composite Many-to-Many	Since ArticleStatistics is one-to-one this will always be unique pairs of articles.
BoundError	Article	article	Many-to-One	Many exceptions can be raised when applying NewsSentiment[21] to an article bound.
Subscription	CustomUser, Entity	user, entity	Composite Many-to-Many	Many users can subscribe to many entities but a users subscription to an entity is binary and therefore unique.

Table 5.2: Schema Relationship Reasoning

A large overview of the entire database design can be found on the repository here.

5.2.2 Data Flow Diagrams

With the schema outlined, data flow diagrams (DFD) were utilised to provide a simple graphical representation that defines the system boundaries, communicates the current system knowledge, assists in explaining the data flow logic, and for documentation purposes [69]. There are two main notations used, Yourdon-Coad and Gane-Sarson, and our approach tends to the former, but for the avoidance of doubt, Fig. 5.6 is provided[70].

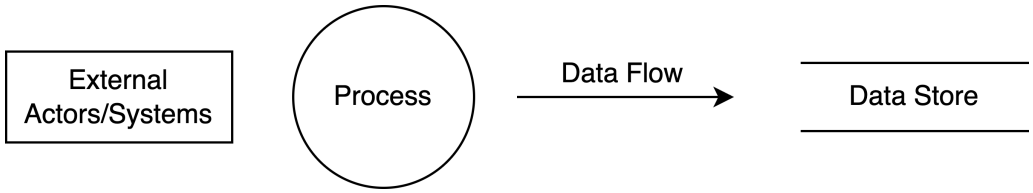


Figure 5.6: DFD Symbol Key

Level-0 DFD

Fig. 5.7 is a context diagram, which is also known as a level 0 data-flow-diagram so it only identifies flows of data or items between the Prominent Profiles' system, which is represented as a single process, and external actors/systems [71].

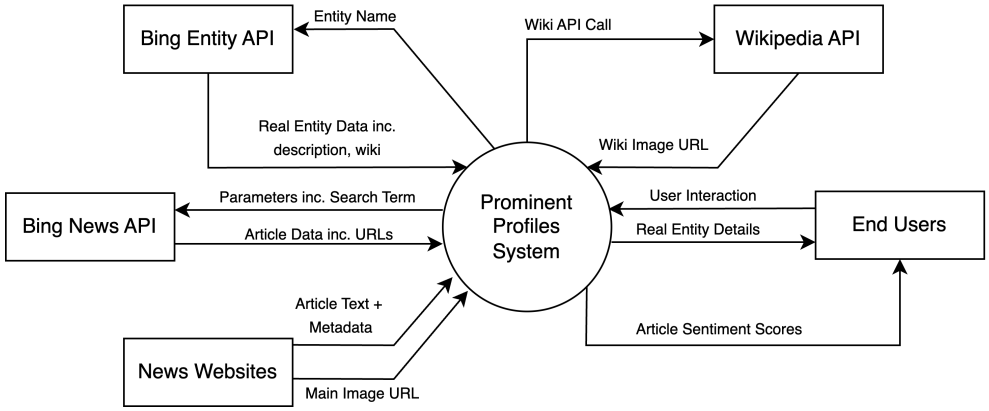


Figure 5.7: Prominent Profiles' Level 0 DFD

Level-1 DFD

Level-1 DFDs have also been created for aspects of the Prominent Profiles system to provide a more insightful view of the system by showing the main sub-processes and data stores that make it up as a whole[71]. While it is possible to create a single level-1 diagram as an 'exploded view' of the context diagram, this becomes unwieldy and struggles to fit within the bounds of the page, so key processes where data stores are relied upon to provide functionality within the Prominent Profiles system were chosen to create various separate DFDs. Many of these align with scheduled automated Django tasks in production (6.4.2).

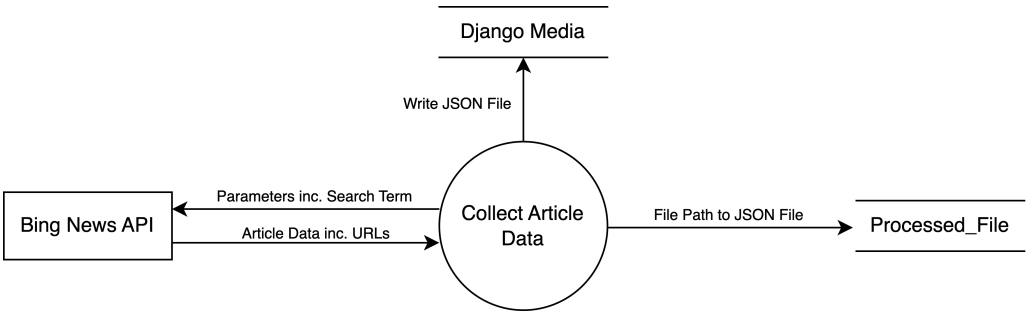


Figure 5.8: Prominent Profiles' Level 1 DFD - Collect Article Data

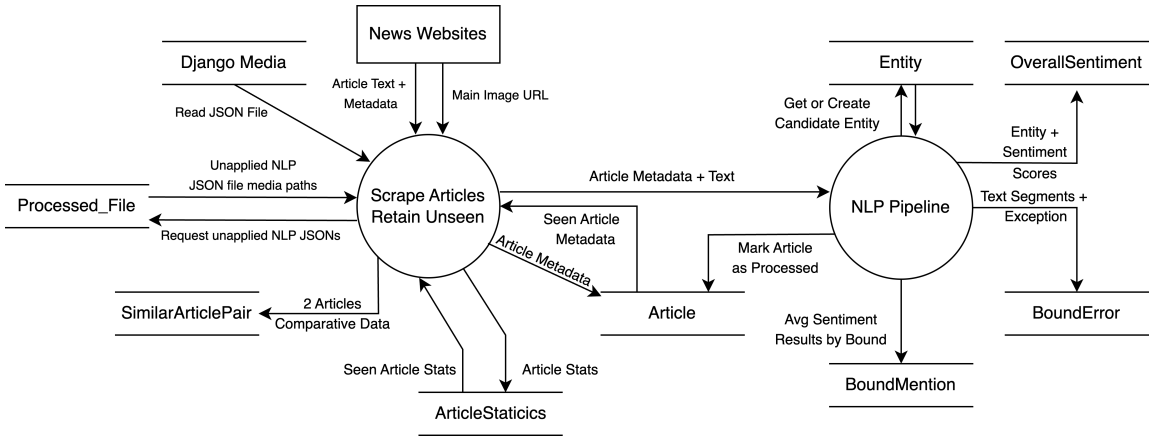


Figure 5.9: Prominent Profiles' Level 1 DFD - Scrape Articles & NLP Pipeline

5.8 and 5.9 depict the essential operations: without these, Prominent Profiles would have no 'Profiles' (Entities) and no articles associated with them. The datastores and flows have increased as logic evolved from the proposal definition and experimental work within the Jupyter notebook to the production variant they depict now (refer to 5.1 for their inception dates).

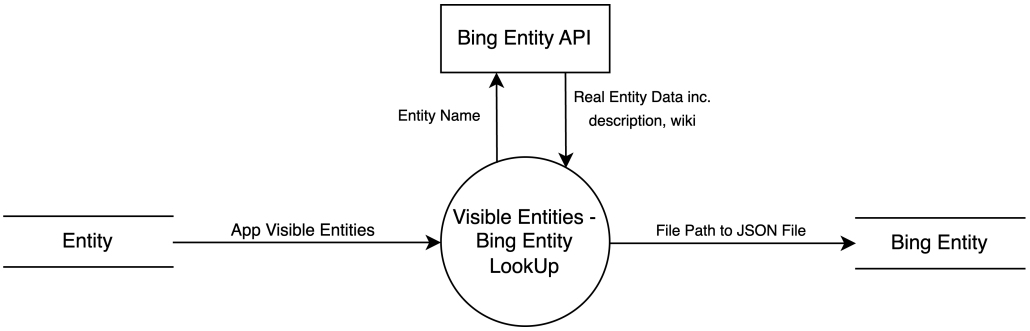


Figure 5.10: Prominent Profiles' Level 1 DFD - Visible Entity Bing Lookup

5.10 is an example of an unanticipated flow, but when design work began on the frontend content, the desire to obtain meaningful entity data became apparent to enrich the UI/UX through the provision of profile images, descriptions and Wikipedia links.

5.11 & 5.12 also serve as examples of unanticipated flows but deemed valuable to provide a semi-automated way to resolve unresolved entities for Prominent Profiles' administrators.

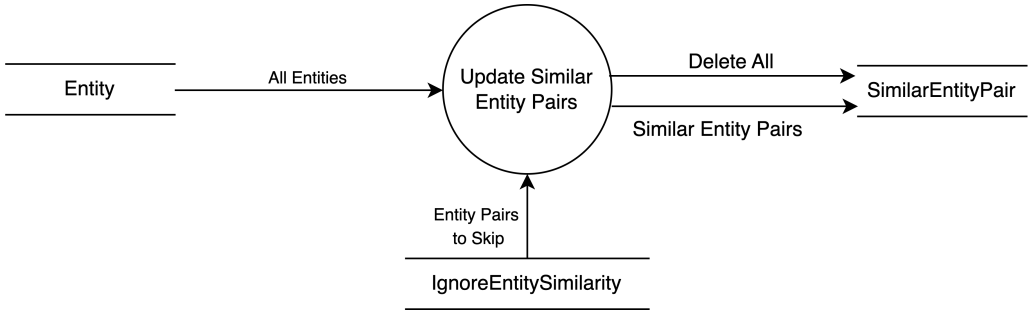


Figure 5.11: Prominent Profiles' Level 1 DFD - SimilarEntityPair Recreation

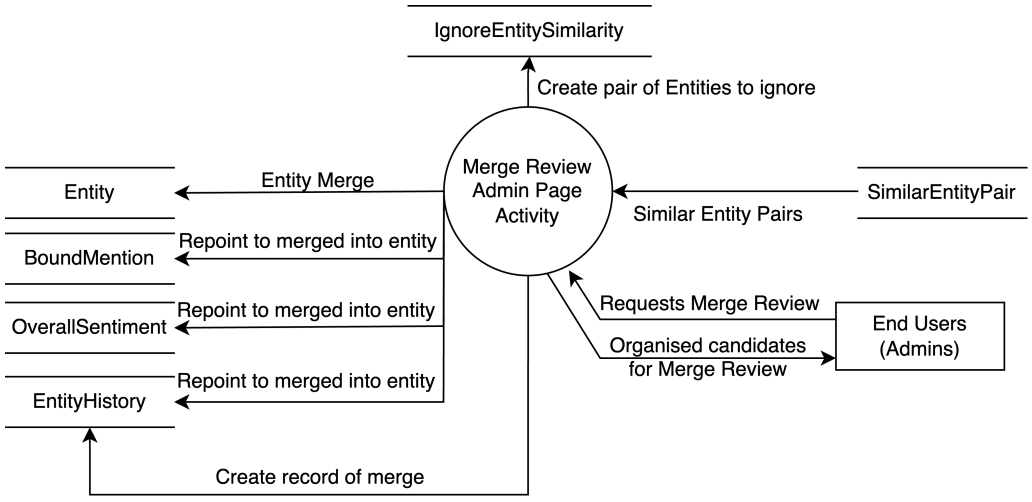


Figure 5.12: Prominent Profiles' Level 1 DFD - Delivery of Merge Review Admin page

5.3 Backend

5.3.1 NLP Pipeline

Conceptual Pipeline

The initial design for the NLP pipeline was informed by preliminary investigations conducted within a Jupyter notebook environment. This practical exploration was worked on alongside and driven by the research carried out for the literature review.

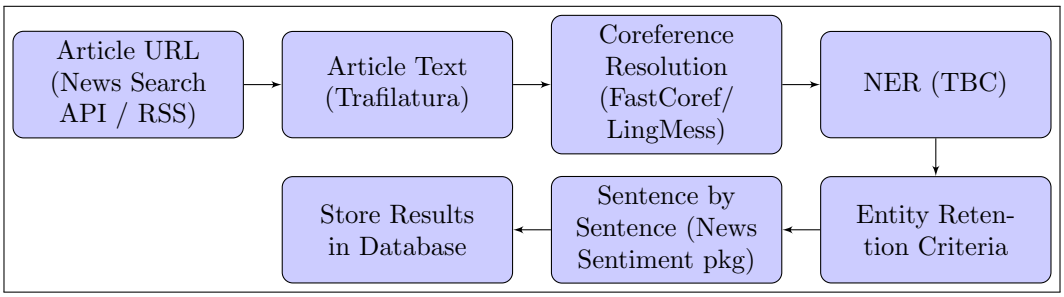


Figure 5.13: NLP Pipeline Diagram - Project Proposal

Iterative Pipeline Desgin

Here, we introduce the production NLP Pipeline in flowcharts, as these provide a graphical means of representing its control structures and decision logic. Consequently, they helped ease the iterative development and supported the improvement of the process flow, as we could refine each step to address its problem areas before being translated into code blocks with decision points becoming selection or, if

they involved looping, forms of iteration[72]. Moreover, they serve as a communication tool to explain to those unfamiliar with the codebase to understand visually before reading code comments.

To ensure developers and stakeholders can interpret these diagrams consistently the symbols we used are in keeping with the ISO standard 5807:1985. A key is provided in 5.14.

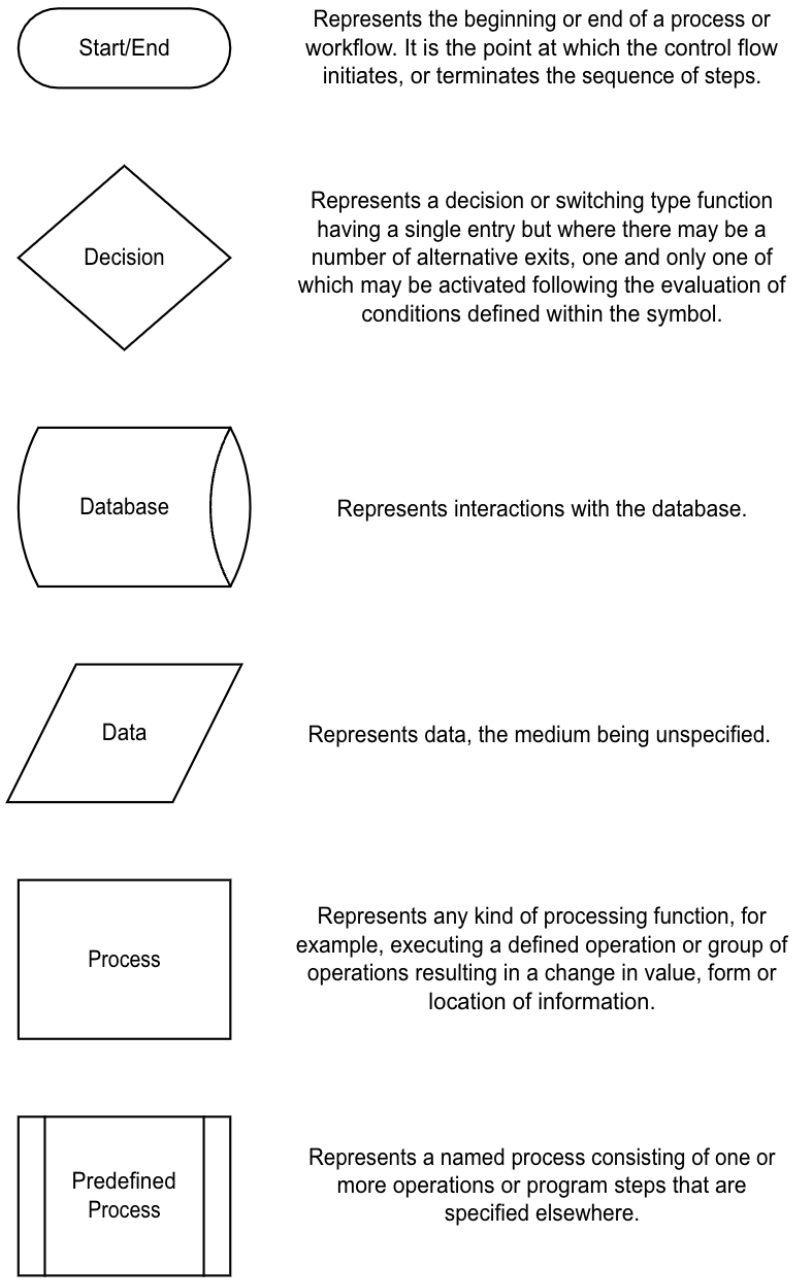


Figure 5.14: Flowchart Symbol Key based upon ISO 5807:1985[7]

The colour coding (Fig. 5.15) is indicative of the design stage a particular step was added to the pipeline as code was migrated from the notebook to a Django environment. Later, we will revisit crucial logic, especially that created owing to implementation issues that were not anticipated.

Fig. 5.16 is the headline flowchart for the task of obtaining and processing article JSON files that have been saved in the Django media directory. It compliments the DFD shown in 5.9 by providing the inner-workings of its processes. The flow chart consists of two key subprocesses the downloading of articles including selective retention (Fig. 5.17) and the processing of articles with their coreference clusters (Fig. 5.18). These flowcharts also contain sub-processes (Figs. 12.3 - 12.6) to keep them a reasonable length

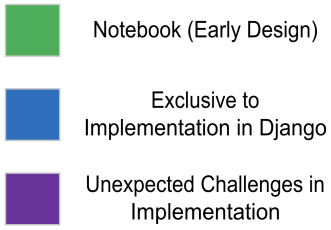


Figure 5.15: Flowchart Colour Code Key

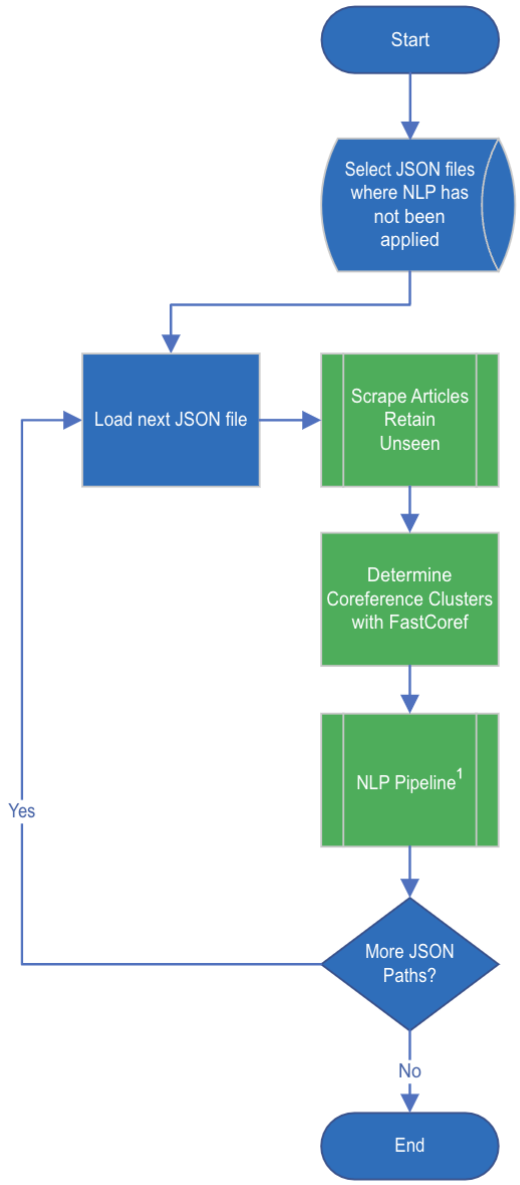


Figure 5.16: Scrape Articles & NLP Pipeline Main Flowchart

with the aim of maintaining clarity in the depiction of each stage of the article acquisition and NLP pipeline. The subprocess Similar Article Checks will be covered in Implementation Challenges rather than as a subprocess flowchart here to cover the problem and solution in greater detail.

Higher Level Subprocesses

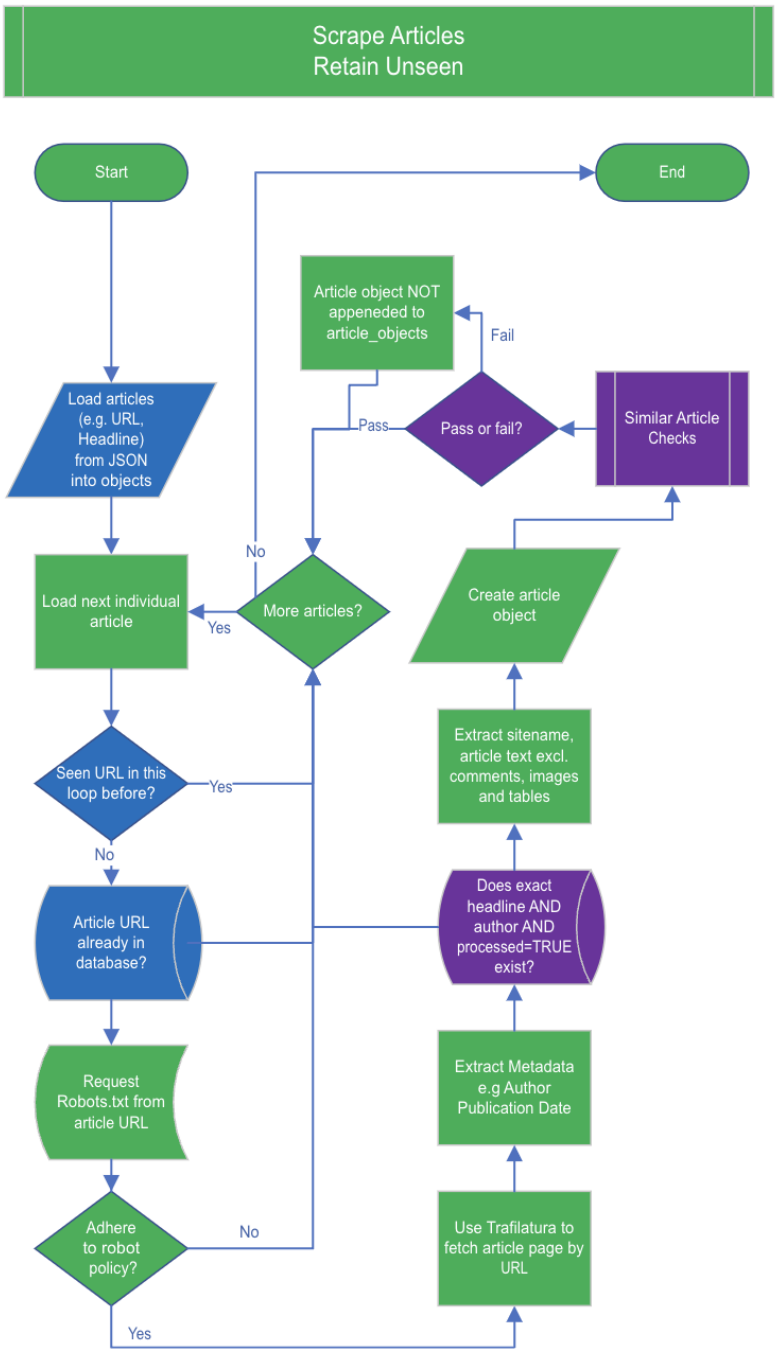


Figure 5.17: Scrape Articles - Retain Unseen Subprocess Flowchart

[1]Locally this operates sequentially but in production it supports threading (i.e multiple NLP pipelines can be ran in parallel to reduce time taken to process articles).

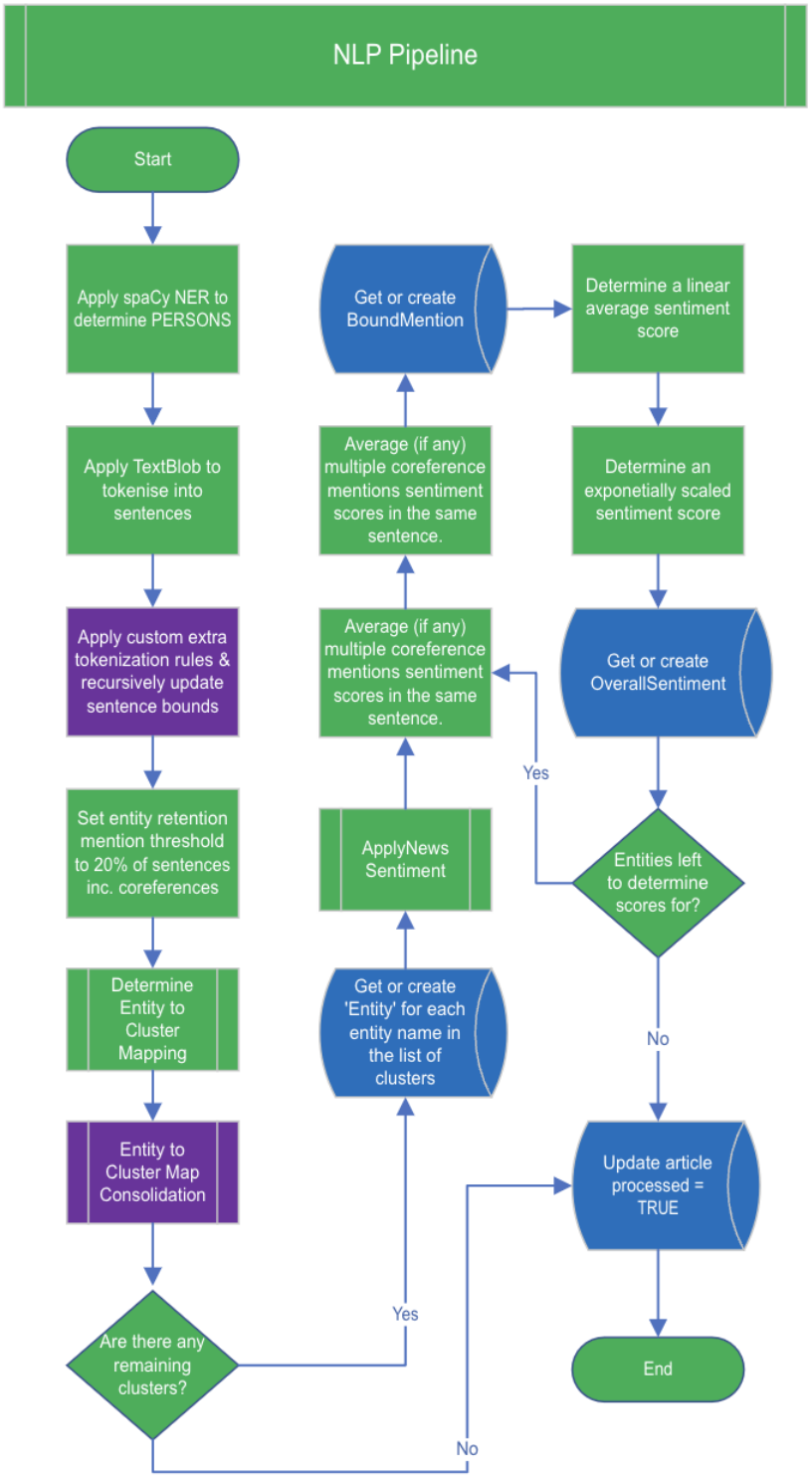


Figure 5.18: NLP Pipeline Subprocess Flowchart

Lower Level Subprocesses

Please refer to Appendix:12.6.1

5.3.2 Django Applications

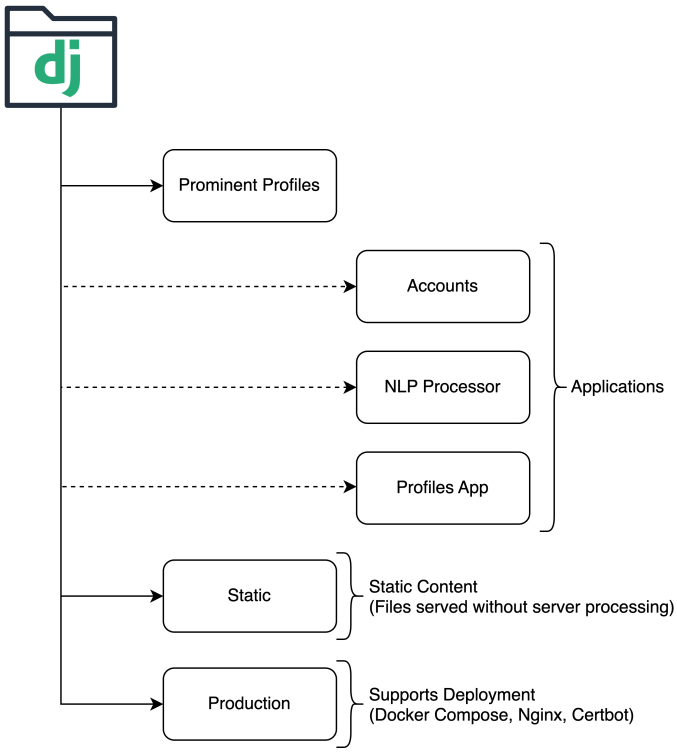


Figure 5.19: Folder Structure of Django App

Django encourages modular development through its application system. 'A Django app is a small library representing a discrete part of a larger project' [73]. This enables a separation of concerns, making code more organised and manageable. It also helps reduce the risk of bugs, simplifies maintenance by isolating changes to specific apps, and allows easy reuse of functionality in related projects. Prominent Profiles has three apps that group related functionalities together (Fig. 5.19).

`accounts` takes care of JSON web login tokens and models to support the extension of Django's default user (custom fields) and the subscription model.

`nlp_processor` is responsible for all activities involved in obtaining article URLs, processing them via the NLP Pipeline, and finally enriching resolved entities with the Bing and Wikipedia APIs. It is the only app that connects to external systems to obtain the data crucial for app functionality. Models residing within `nlp_processor` locate and keep track of outstanding JSONs to be processed, help prevent duplicate articles, and log exceptions thrown by NewsSentiment for later debugging. In summary, all models in this app are for internal use only, and the end users of the frontend app will not need access to them, so it has no defined API views.

`profiles_app` oversees models that contain data resulting from NLP pipeline stages or modify it in a way that could impact the end-user experience. The former includes articles, entities and their sentiment scores, while the latter covers administrator merges of entities and their foreign key references to remove unresolved entities from the database. Most API views that feed data to the Vue app are located here.

Notably, this structure does mean models residing in the `profiles_app` are utilised by the `nlp_processor`, but these interdependencies between apps are standard practice in Django projects. Moreover, there is still a clear separation, even though this is subjective, so other developers might use different criteria. Refer to table 5.1 for a full breakdown.

5.4 Frontend

5.4.1 Vue Overview and App Structure

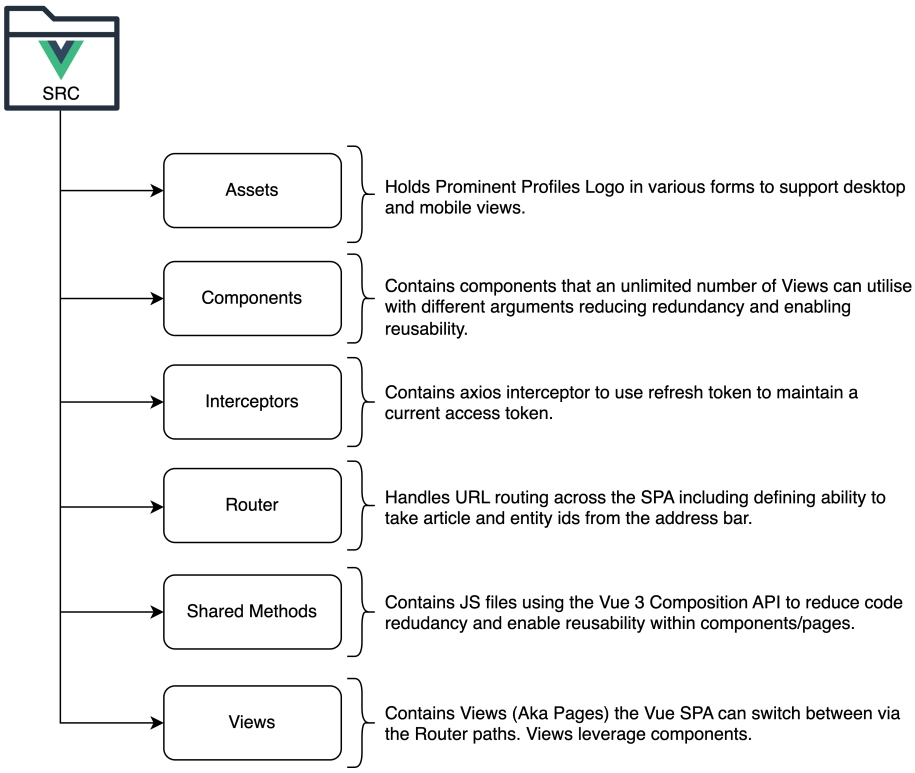


Figure 5.20: Folder Structure of Vue App

Vue promotes a component-based architecture that facilitates the construction of sophisticated user interfaces through small, self-contained, and often reusable parts[74]. A Vue component is an encapsulated reusable unit that can maintain its own state, templates, and behaviour. They enable us to split Prominent Profiles page content into multiple, more manageable chunks that can be reused across views or on the same view with different input data from the API (e.g. entity in URL changes).

The Vue app is divided into several key directories, each with a specific purpose that contributes to the overall functionality (Fig. 5.20).

5.4.2 UI Design

The user interface (UI) design of a web app is the bridge between users and the app's functionality. It is crucial as it shapes the user's first impression, facilitates ease of use, and enhances user satisfaction. In the creation of Prominent Profiles, the UI was crafted to prioritise clarity and encourage logical navigation to other prominent profiles on the site, supporting user exploration of current affairs.

Our design approach was iterative, allowing for refinement and adaptation based on user feedback and technical considerations. We progressively incorporated elements that would enhance the user experience (UX), starting with basic wireframes.

Vue is a single-page application (SPA), which means it loads a page and then rewrites it with new content instead of loading a new page for every interaction, providing a more responsive UI. So here, we refer to each wireframe as a view rather than a page.

Designs shown here are often not the final design, as we revisited them once Vue programming began. The refinements made during the implementation phase (6) will be more compelling to discuss as the evolution will be more distinct, and some tweaks were only made as the developers' confidence with Vue grew.

Finally, these designs limited the scope to computer and laptop use. Project resources would then determine if we could make optimisations for mobile later.

Entity View

Entity View is the core of Prominent Profiles. The initial concepts (Fig. 5.21) served as the starting point for Prominent Profiles and were presented to the project supervisor along with other project ideas.

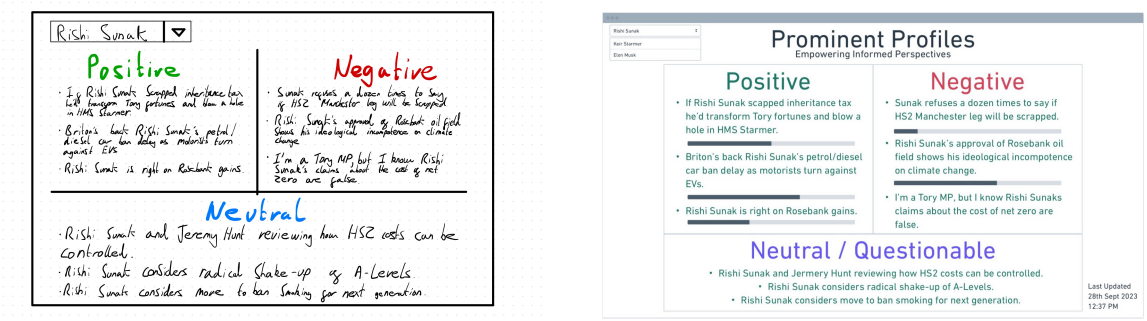


Figure 5.21: App Concepts

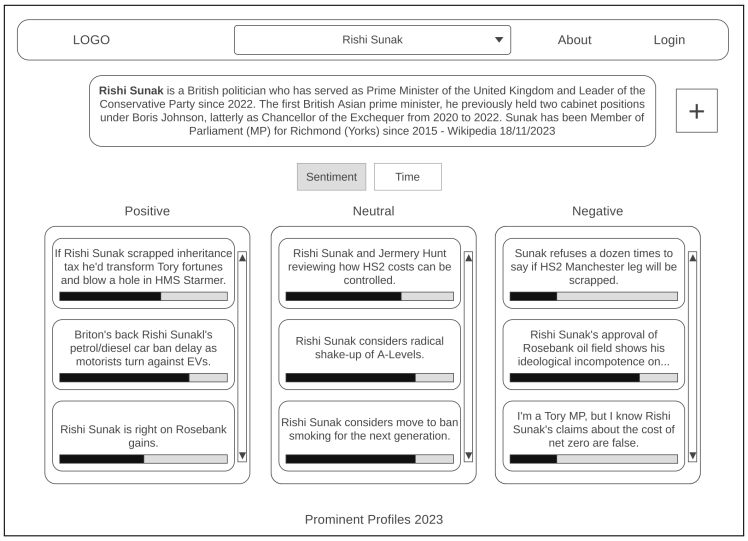


Figure 5.22: Entity View Wireframe

Users can reach the Entity View by selecting a 'Prominent Profile' from the dropdown box within the header shown across the SPA. It gives the user a description of the entity obtained from the Bing Entity API, with fair attribution and an acquirement date. While users may already have some knowledge of the profile, this additional context could expand it. It also enables users who want to become more familiar with current affairs to gain a greater understanding of them.

Below this, articles are presented in one of three containers depending on the largest % category of negative, neutral and positive the NLP pipeline has determined. The benefit of containers over one list is a clear separation of viewpoints. Additionally, buttons for sorting by sentiment or time allow the user to prioritise ordering by scores to identify the articles with the strongest sentiment outcomes quickly or by time to focus on the latest headlines in each classification.

To present the data in a visually engaging way, bars are used instead of raw percentage figures. When a user clicks a headline is clicked, the Vue router will direct them to the Article Analysis view to access extra information.

From the concepts to the later design iterations, a worthy change was moving towards three columns of category containers rather than the wider layout for neutral articles because it was not entirely clear how the app would function with this layout on a narrow screen, for example, when multitasking on a

desktop or accessing via tablet and mobile devices. By moving to three columns, we can vertically stack the article card containers and provide a better UX.

Another development that appeared in later iterations once the account features were defined was the subscription (+ button) to the right of the entity description container. When clicked by a logged in user, this entity will begin to appear in their dashboard. In contrast, if an anonymous user clicks it, they will be redirected to the login page to sign up or authenticate so that they can use this feature.

Now that we have branded the web app, we can also refer to the 'Entity View' as a 'Profile View'.

Article Analysis View

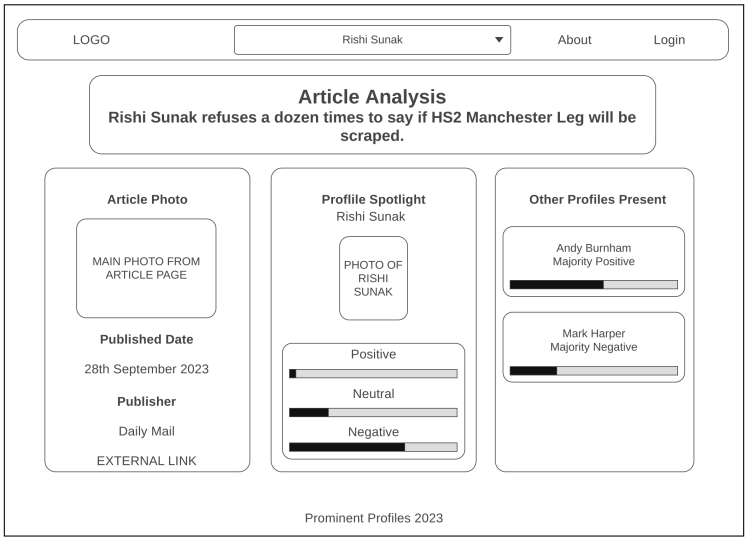


Figure 5.23: Article Analysis View Wireframe

Users can reach the Article Analysis View by selecting an article headline in the Entity View. First, it gives the user the complete article headline, as Entity View may have truncated it.

Below this, three containers deliver a variety of data associated with the article stored in the database. The leftmost provides key article metadata and, most importantly, the external link for the user to leave the app to read the article text; entire article text bodies are not retained by Prominent Profiles for legal reasons (8.1). The central container provides the entity photo[1] and a breakdown of each sentiment category; these scores will add up to 100%. Finally, the rightmost container lets the user see other profiles' presence in the article and overall sentiment category. Moreover, they can click on one of the 'other profiles present' to visit their Entity View, further encouraging our users to learn and explore more prominent profiles and diverse perspectives towards them.

Home View

[1]This may seem repetitive; however, users may choose to share Article Analysis Views on social media, and this way, any inbound user coming directly to this view will benefit from this context.

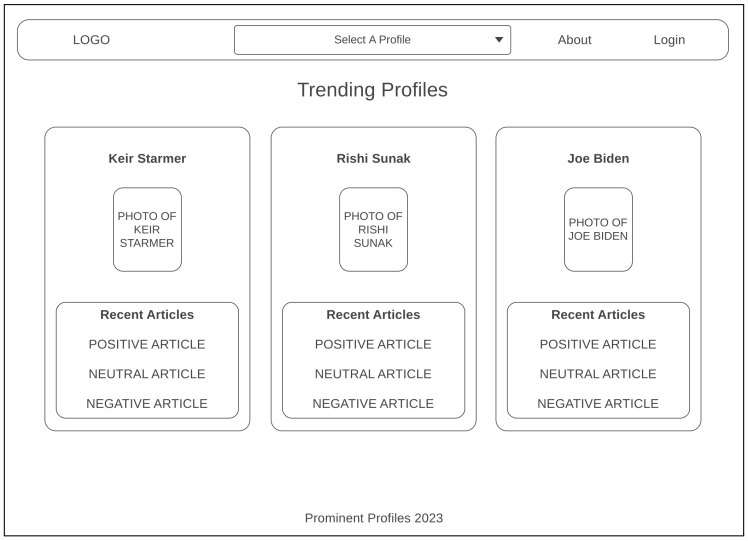


Figure 5.24: Landing View Wireframe

Users will reach the Home View by default or from anywhere in the app by clicking the logo in the header.

Both Google News and GroundNews (2.2) incorporate top stories or trending topics/people functionality on their respective home pages. As Prominent Profiles' main functionality is centred around the Entity (/Profile) Page, it was fitting to make Trending Profiles the main content for containers on HomeView. Later in the implementation, we will discuss the dilemma of determining this data. Currently, the wireframe introduces the idea of showing the name, photo, and one most recent negative, neutral, and positive article for each. An ambition exists to introduce some animation to display multiple articles here, although this is contingent on project timelines and capabilities within Vue.

We initially considered a carousel of Trending Profiles; however, few users interact beyond the first item or even find it in tests when they are tasked with performing an action that resides in the carousel. Plus, companies say they perform poorly in terms of promotional click-thru. Finally, they pose a significant challenge to screen readers, hindering accessibility[75, 76]

Sign Up / Login Views

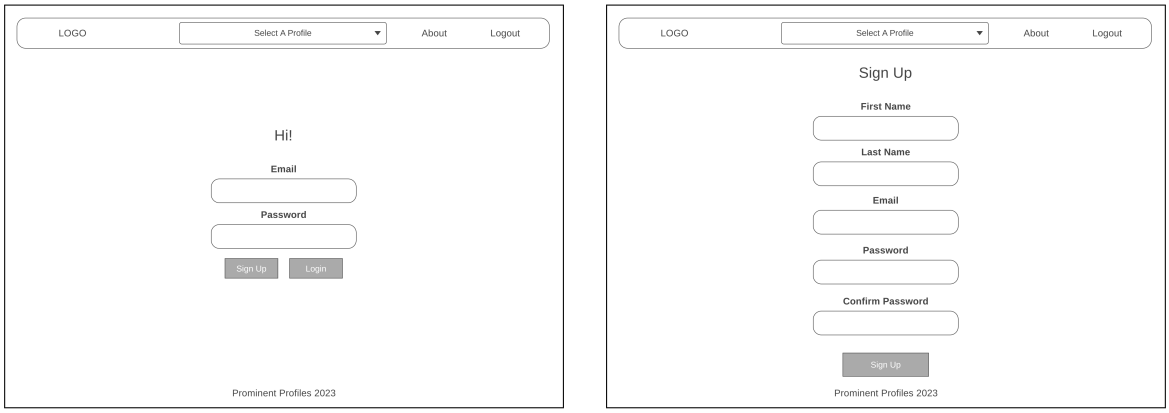


Figure 5.25: User Account Views Wireframes

Most Prominent Profiles functionality is not hidden behind a login but to use the subscription features an account is required. The Login View is deliberately minimalist to reduce cognitive load and focus the users attention on the action of logging in [77]. This view can be reached from the header at any time by unauthenticated users by clicking login rightmost in the consistent SPA header or via clicking a subscription button on an Entity View which creates a frictionless path to membership, encouraging users to engage further by signing up for a personalised UX.

The Sign Up View collects essential information and explicitly validates with a password confirmation step to assure users their credentials are correctly set.

Dashboard Page

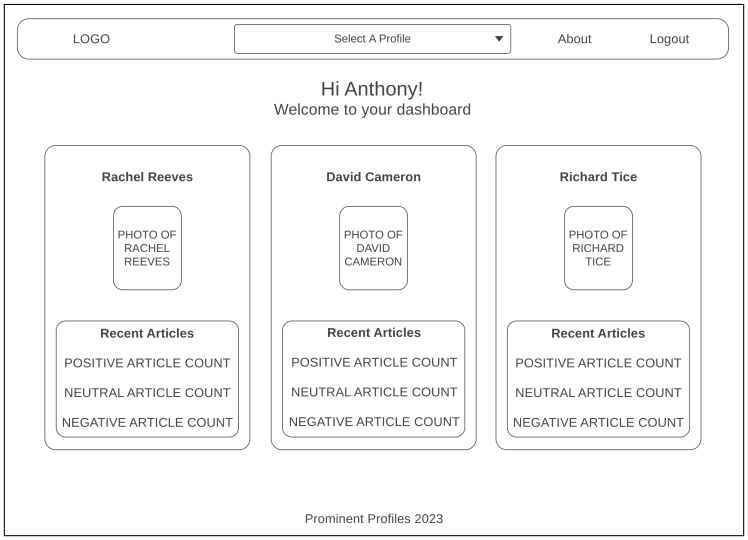


Figure 5.26: Dashboard View Wireframe

Following a successful login, the Vue router will redirect users to the Dashboard View. Meanwhile, users already authenticated can reach it by clicking 'Your Dashboard', which appears in the header on all views. This view starts by offering a personalised welcome, implicitly recognising their signed-in status and also revealing if they are signed into the correct account (e.g., in the case of a shared computer).

Below this, users encounter as many containers as entities they are subscribed to, providing a one-stop shop for any profile they wish to follow the news stories of more closely. Each shows the name and photo, as well as the key benefit of a subscription: a count of articles by sentiment category that have been published since the user last visited the site. This approach is inspired by other social media platform notification systems, such as YouTube, which gives a count of new videos by accounts a user is subscribed to and is appealing as it keeps them coming back to the app[78]. Later, we will implement a custom user model in Django to store users' last visit to the site, enabling this functionality. Finally, users can then click through to visit the Entity Pages of their respective subscriptions.

About Page

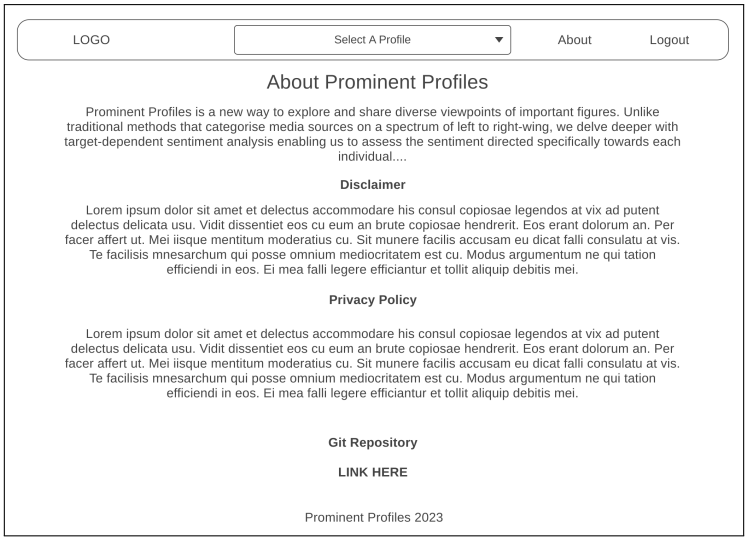


Figure 5.27: About View Wireframe

Users can easily reach the About View from any view via the header. As its content includes a description of our novel approach (most users will require an explicit explanation) and an important legal disclaimer (8.2), we have placed it here rather than more subtly in the footer, as many websites tend to. Additionally, links to the privacy policy and git repository are included here, providing users with all key references in one convenient location. Lastly, the presentation of this vital information is intentionally simplified, centred on the page without the distraction of containers, to guarantee accessibility and compatibility across various devices.

Chapter 6

Implementation

6.1 Article Collection and Processing

We use code snippets to evidence the following commentary. Note that many comments, docstrings and some lines of less important code have been removed for brevity. The best-commented production variant of this code is found within the `nlp_processor` module. Although the `demo_book.ipynb` helpfully provides outputs alongside the code, while the very experimental, sprawling `main_book.ipynb` is where much of this logic was first determined.

6.1.1 Collect Article Data

A Bing Search API key was obtained via Azure. The original intention was to take the freshness of a week or month to populate the initial database quickly; however, most results beyond the first 2-3 pages returned begin to repeat and so only give a subset of articles published in a longer freshness period (others face this issue: Stackoverflow). The solution was to restrict freshness to one day and run the job daily. Even though we obtain more unique articles over a given period this way, duplicates will still occur within each run, so there is logic to skip and count duplicates to halt the API calls (to save quota) if we see more than 25. Separately, we found that news articles from outlets delivered via msn.com were never successfully extracted by Trafilatura, so it makes sense to ignore these proactively rather than fail to scrape them later on.

Listing 6.1: API Parameters

```
    for offset in range(0, results_to_fetch, count):
        if skip_future_calls:
            break
        time.sleep(1)
        params = {
            "q": search_term,
            "count": count,
            "offset": offset,
            "freshness": "day",
            "textDecorations": True,
            "textFormat": "HTML",
            "sortBy": "Date"
        }
```

Listing 6.2: Proactively Ignore MSN

```
    # There have been no successful trafilatura extractions of MSN text
    if not url.startswith("https://www.msn.com"):
```

Listing 6.3: Duplicate URL Handling

```
        urls_seen = set()
```

```
        duplicate_count = 0
        max_duplicate_count = 25
        skip_future_calls = False

        # Later in code...
        if url in urls_seen:
            duplicate_count += 1
            if duplicate_count >= max_duplicate_count:
                skip_future_calls = True
```

Following this, a JSON file for each search query is stored in a directory (created if non-existent) with a timestamp, and the file paths are stored in the processed file table. The JSON file itself contains a URL, title and description for each article, which provides us with everything required for the next stage. In January, we expanded the number of search queries from ['Rishi Sunak', 'Keir Starmer', 'UK Politics', 'Prince Harry', 'Meghan Markle'] to include 'General Election' (for 2024) and the largest UK political parties to obtain more coverage to analyse.

Figure 6.1: Left: Django Media Directories, Right: JSON File Contents

6.1.2 Article Text Extraction

We start by establishing the outstanding JSONs yet to have been through the pipeline. Each article read from the JSON file is subjected to increasingly stricter and more expensive checks to ensure it has not seen before. First we check if its URL has been seen in this job already, then if it already exists in the database and is marked as processed or rejected (unprocessed means the job was interrupted previously so we should retry).

Listing 6.4: Determine JSONs to Read

```
    unapplied_files=ProcessedFile.objects.filter(nlp_applied=False)
```

Listing 6.5: Duplicate Checks Prior To Extraction

```
    if url in processed_urls:
        skips += 1
        continue

    # Check if the URL already exists in the database
    existing_article = ArticleModel.objects.filter(url=url).first()
    if existing_article:
        if existing_article.processed or existing_article.similar_rejection:
            skips += 1
            continue  # Skip this article as already processed / rejected.
```

Next, we extract the article text using Trafilatura to obtain the author to check the headline **and** author pair for an existing match since we identified headline equality as an overly strict rejection criterion (6.5.2).

The parameters aim to obtain the article text alone by excluding comments, images, and tables. The decision to favour recall over precision means extraction is more opportunistic[36]. However, as we will only take sentences with mentions above a particular threshold, if other page contents, e.g., advertisements/related news panels, are obtained, it is unlikely to impact the overall sentiment determination later. If some text was returned, an article object is created and appended to a list[1].

Listing 6.6: Trafilatura Extraction Parameters

```
article_text = trafilatura.extract(downloaded, favour_recall=True,
               include_comments=False, include_images=False, include_tables
                   =False)

if article_text and len(article_text) > 249:
    source_file = ProcessedFile.objects.get(id=source_file_id)

    article_obj = Article(url, headline, article_text, ner,
        publication_date,
                          author, site_name, source_file)
    article_objects.append(article_obj)
```

Listing 6.7: Duplicate Checks After Extraction

```
if ArticleModel.objects.filter(headline=headline, author=author,
                               processed=True).exists():
    print('Exact headline and author already exists in db and has been '
          'processed')
```

6.1.3 Coreference Determination

Now we have a list of article objects to process to take advantage of Fastcoref's (3.1.3) batch processing support; we obtain the coreference for each article text first.

Listing 6.8: Fast Coref Batch Processing

```
# Within Function
predictions = model.predict(texts=article_texts, max_tokens_in_batch=
    batch_size)

article_text_clusters = []
for prediction in predictions:
    clusters_text = prediction.get_clusters()
    clusters_positions = prediction.get_clusters(as_strings=False)
    combined_clusters = [(text, positions, len(text)) for text, positions
        in zip(clusters_text, clusters_positions)]
    sorted_combined_clusters = sorted(combined_clusters, key=lambda x: x
        [2], reverse=True)
    article_text_clusters.append(sorted_combined_clusters)
return article_text_clusters

# Later in Main body
for article, clusters in zip(article_objects, article_text_clusters):
    article.set_coref_clusters(clusters)
```

```
Cluster 1 : ["Labour's", 'the Opposition party', 'Labour's', 'Labour', 'Labour', 'Labour', 'his Party', 'Labour', 'Labour', 'Labour', 'Labour']
Cluster Text Count: 11
Positions: [(78, 86), (146, 166), (230, 238), (373, 379), (439, 445), (963, 969), (1042, 1051), (1240, 1246), (1767, 1773), (1895, 1901), (2163, 2169)]
Cluster 2 : ['Tory', 'the Tories', 'the Conservatives', 'Tory', 'the Conservatives', 'Tory', 'Tory', 'Tory', 'the party's']
Cluster Text Count: 9
Positions: [(47, 51), (249, 259), (403, 420), (2005, 2009), (2046, 2063), (2107, 2111), (2557, 2561), (2579, 2583), (2712, 2723)]
Cluster 3 : ['Hoyle', 'Speaker Sir Lindsay Hoyle', 'the Speaker', 'Hoyle', 'Sir Lindsay', 'he', 'Sir Lindsay', 'he', 'his']
Cluster Text Count: 9
Positions: [(224, 229), (1064, 1089), (1197, 1208), (1324, 1329), (1734, 1745), (1751, 1753), (1824, 1835), (1866, 1868), (1961, 1964)]
Cluster 4 : ["Sir Keir Starmer's", 'Keir Starmer', 'Sir Keir Starmer', 'his', 'his', 'the Labour leader', 'his', 'Sir Keir']
Cluster Text Count: 8
Positions: [(0, 18), (168, 180), (574, 590), (600, 603), (900, 903), (959, 976), (1042, 1045), (1385, 1393)]
Cluster 5 : ['Lee Anderson MP, who lost the Tory whip', 'The former Tory deputy chairman', 'Mr Anderson', 'his', 'the MP's']
```

Figure 6.2: Coreference Clusters in String & Character Position Form

[1]See Article Duplication (6.5.2) for extra checks added prior to appending an article object.

6.1.4 Threading

Articles were processed sequentially in the notebook research environment and initial Django implementation (see legacy `scrape_articles.py`). Our dev device CPU was less than 30% utilised, so given the application of the NLP pipeline does not depend on other's results or modify shared resources in a way that could cause conflict, we parallelised the operations that follow with the processed file marked as processed iff the article list has been exhausted, and threads have completed. Max concurrent threads was set 3-4 depending on passive or active use of the Macbook and in production this is set to 2 due to Digital Ocean server resources.

Listing 6.9: Thread Pool NLP Pipeline Execution & Marking JSON file

```
    with concurrent.futures.ThreadPoolExecutor(
        max_workers=self.max_concurrent_threads) as executor:
    list(executor.map(self.process_article_wrapper,
                      ((article, i + 1, len(article_objects)) for i, article in
                       enumerate(article_objects))))

# Update the ProcessedFile object so future jobs don't reanalyse.
file.nlp_applied = True
file.save()
```

6.1.5 spaCy NER

While the extract of coreference clusters shown in Fig. 6.2 seems ideal, in fact, fastcoref returns any cluster of coreferences (corefs) it finds, including those of locations, organisations, dates, etc, and **does not** categorise them so spaCy NER (3.1.2) is applied to the article text to obtain named entity of type PERSON.

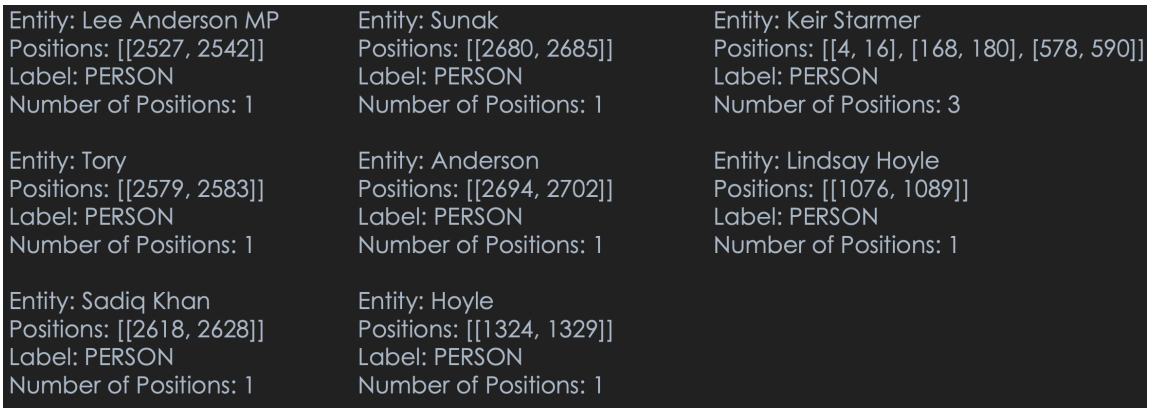


Figure 6.3: spaCy NER outcome

6.1.6 Sentence Tokenization

NewsSentiment was trained on sentence level data (and the underlying Pytorch model has character limits) so it is preferable to deliver our mentions of NEs in this form too. Textblob (3.1.4) is used to obtain the bounds of the sentences required to achieve this form.

In the notebook phase we identified that TextBlob's tokenization was not always satisfactory. The article introduction in Fig. 6.4 was tokenized as one complete sentence. We identified Trafilatura was converting the bullet points to '-' (the cause) so wrote a functional expression and code to appropriately update the TextBlob intervals.

Listing 6.10: Custom Sentence Tokenization

```
hyphen_nl_pos = [pos for pos, char in enumerate(article_text) if article_text[
    pos:pos+2] == '\n-']
extra_split = hyphen_nl_pos
```

Figure 6.4: TextBlob Sentence Tokenization Issue Example

```
sentence_bounds = [(int(sentence.start), int(sentence.end)) for sentence in
    sentences]
updated_list = insert_intervals(sentence_bounds, extra_split)
```

Listing 6.11: Recursive Interval Update

```
def insert_recursive(intervals, values):
    if not values:
        return intervals  # Base case: Return the intervals when there are no
            more values to insert.

    value = values[0]
    result = []
    for interval in intervals:
        if interval[0] <= value <= interval[1]:
            # If the value falls within an existing interval, split into 2
                parts.
            # The first part goes from the interval's start to the value.
            # The second part goes from the value+1 to the interval's end.
            if interval[0] < value:
                result.append((interval[0], value))
            if value < interval[1]:
                result.append((value + 1, interval[1]))
        else:
            # If the value doesn't fall within the interval, keep the interval
                as is.
            result.append(interval)

    # Recursively process other values.
    return insert_recursive(result, values[1:])
    updated_list = insert_recursive(initial_list, new_values)
    return updated_list
```

Listing 6.12: Terminal: Before and After Custom Tokenization

```
TextBlob tokenization: [(0, 489), (490, 626), (627, 764), (765, 874), ...]
hyphen_nl_pos: [138, 223, 301]
After custom sentence tokenization: [(0, 138), (139, 223), (224, 301),
    (302, 489), (490, 626), (627, 764), (765, 874), ...]
```

For now, only this case is manually implemented. Still, with the method defined, adding further custom tokenization rules is straightforward, and the unexpected addition of the BoundErrors table (Fig. 5.3) to log TooLongExceptions thrown along with the offending text segments will support the identification of new rules.

Finally, we set a minimum mention threshold for the article to follow the approach: 'On coreference cluster level, we discard a cluster c in a document d if $|M_c| \leq 0.2|S_d|$, where $|\ldots|$ is the number of mentions of a cluster (M_c) and sentences in a document (S_d)'[8].

6.1.7 Identifying Relevant Coreferences

Entity ↔ Coref Cluster Mapping

We then had to consider an appropriate way to wed NER and corefs. The obvious would be to use the same IntervalTree (3.1.5) approach we do for corefs and sentences later. However, this is challenged. For instance, spaCy NERs' returned positions may only match with one of the cluster positions, and it is tricky to be confident that we are accurately pairing based on just one position intersection.

Take for instance this hypothetical example:

- A coref cluster: ['Tory', 'the Tories', 'Rishi Sunak's party' ,'the Conservatives', 'Tory', 'the Conservatives', 'Tory', 'Tory', 'Tory', 'the party's'] with the 3rd at positions [2674, 2693]
- An NER: 'Rishi' at positions [2674, 2678].

We would later determine TSC for the Conservative Party and wrongly tag that as sentiments towards (at the time of writing) the Prime Minister.

To consider the context around each mention better, we devised a custom approach. For each cluster with more than four cluster text strings, we remove stop words to cleanse the cluster text. Then the entity name detected by spaCy NER is split into words, all the cleansed cluster strings are concatenated into one string and a % rate between occurrences of each of the entity words in the long string, and the total number of coreferences found is calculated. 30% was identified as an appropriate pairing threshold.

Notebook research revealed titles were dampening percentage match rates, so we also removed those before concatenation.

Listing 6.13: spaCy NER ↔ Cluster Mapping Extract

```
def cleanse_cluster_text(cluster_text):
    undesired_words = ["i", "he", "his", "she", "they", "it", "this", "that",
                       "these", "those", "the", "a", "an", "of"]

    return [word.strip() for word in cluster_text if word.lower().strip() not
            in undesired_words]

def remove_titles(text):
    title_pattern = r"^(Mr|Mrs|Ms|Miss|Dr|Prof|Rev|Capt|Sir|Madam|Mx|Esq|Hon|
        Gen|Col|Sgt|Fr|Sr|Jr|Lord|Lady)\s"
    text = re.sub(title_pattern, "", text)

    # Pattern for titles at the end
    title_pattern_end = r"\s*(KC|QC)\s*$"
    text = re.sub(title_pattern_end, "", text)
    return text

for index, (cluster_text, cluster_positions, _) in self.coref_clusters:
    cluster_id += 1

    cluster_text = cleanse_cluster_text(cluster_text)
    cleaned_cluster_text = [remove_titles(text) for text in cluster_text]

    total_coref_words = " ".join(cleaned_cluster_text)
            entity_parts = entity_name.split()
            max_percentage = 0.0

            for entity_part in entity_parts:
                entity_count = total_coref_words.count(entity_part)
                #  How well the entity name matches the cluster.
                percentage = entity_count / len(cleaned_cluster_text)

                # Is this a better match than the previous best match?
                # Yes, make this the new best match.
                if percentage > max_percentage:
                    max_percentage = percentage
```

```
36
37              if max_percentage >= ENTITY_THRESHOLD_PERCENT:
```

Listing 6.14: spaCy NER ↔ Cluster Mapping - % Match Rate Terminal Output

```
Cluster ID: 3, Original Text: ['Hoyle', 'Speaker Sir Lindsay Hoyle', 'the
    Speaker', 'Hoyle', 'Sir Lindsay', 'he', 'Sir Lindsay', 'he', 'his']
Matching 'Lindsay' in cluster 3: 50.00% match
Matching 'Hoyle' in cluster 3: 50.00% match
Accepting cluster 3 for 'Lindsay Hoyle' with 50.00% match on part 'Lindsay'

Cluster ID: 3, Original Text: ['Hoyle', 'Speaker Sir Lindsay Hoyle', 'the
    Speaker', 'Hoyle', 'Sir Lindsay', 'he', 'Sir Lindsay', 'he', 'his']
Matching 'Hoyle' in cluster 3: 50.00% match
Accepting cluster 3 for 'Hoyle' with 50.00% match on part 'Hoyle'

Cluster ID: 4, Original Text: ["Sir Keir Starmer's", 'Keir Starmer', 'Sir
    Keir Starmer', 'his', 'his', 'the Labour leader', 'his', 'Sir Keir']
Matching 'Keir' in cluster 4: 80.00% match
Accepting cluster 4 for 'Keir' with 80.00% match on part 'Keir'

Cluster ID: 3, Original Text: ['Hoyle', 'Speaker Sir Lindsay Hoyle', 'the
    Speaker', 'Hoyle', 'Sir Lindsay', 'he', 'Sir Lindsay', 'he', 'his']
Matching 'Lindsay' in cluster 3: 50.00% match
Accepting cluster 3 for 'Lindsay' with 50.00% match on part 'Lindsay'

Cluster ID: 5, Original Text: ['Lee Anderson MP, who lost the Tory whip', '
    The former Tory deputy chairman', 'Mr Anderson', 'his', 'the MP's']
Matching 'Lee' in cluster 5: 25.00% match
Matching 'Anderson' in cluster 5: 50.00% match
Matching 'MP' in cluster 5: 50.00% match
Accepting cluster 5 for 'Lee Anderson MP' with 50.00% match on part '
    Anderson'
```

Entity ↔ Cluster Consolidation

With a mapping established between clusters, it became apparent that further processing would be required to obtain a sufficient number of resolved entities to make the Prominent Profiles data robust enough for its purpose. For example, pairs may exist between an entity's first name and a second name due to the spaCy NER output, or multiple NER outcomes may have met the 30% criteria, consequently linking them to the same cluster. Additionaly, the name may not be representative e.g. 'Rishi' instead of 'Rishi Sunak' so we try to resolve this.

The consolidation process can be split into 2 parts:

1. **Intra-cluster Consolidation:**
   - NER names linked to the same same cluster are examined for substring relationships between them and names in the cluster text.
   - (NER, Coref) pairs are merged by removing the entity with the shorter name or by creating a new entity entry if a combined entity name is found in the cluster text.
2. **Inter-cluster Consolidation:**
   - Checks across different clusters for potential merges based on name substring relationships.
   - Clusters with related entities are merged by updating cluster IDs to a combined format and consolidating cluster texts and positions, aiming to create a truer representation of entities for processing.

The above process repeats between all entities linked to a given cluster (1) and between all the clusters and their respective entities (2) if **any** change occurs during an iteration **until** no further changes are made: the cluster consolidation is almost complete. The final step uses the minimum mention threshold

set in Section 6.1.6 to discard any final clustered coreferences that do not contain enough mentions to aid the provision of a reliable, fair sentiment score.

Listing 6.15: Intra-Cluster Consolidation Extract

```
if is_substring(entity_1_name, entity_2_name):
    if len(entity_1_name) > len(entity_2_name):
        if entry2 in entries:
            entries.remove(entry2)

    elif entry1 in entries:
        entries.remove(entry1)
    consolidation_done = True

elif combined_entity in cluster_text:
    combined_entry = {'Entity Name': combined_entity, 'Label': entry1['Label'],
        'Cluster Info': entry1['Cluster Info']}
    entries.remove(entry1)
    entries.remove(entry2)
    consolidation_done = True
```

Listing 6.16: Inter-Cluster Consolidation Extract

```
for cluster_id1, entries1 in cluster_dict.items():
    for cluster_id2, entries2 in cluster_dict.items():
    # Prevents comparing entries within the same cluster
        if cluster_id1 != cluster_id2:
            for entry1 in entries1:
                for entry2 in entries2:
                    entry1 = update_entity_name(entry1)
                    entry2 = update_entity_name(entry2)
                    entity_1_name = entry1['Entity Name']
                    entity_2_name = entry2['Entity Name']
                    # Check if an entity name is a substring of another.
                    if entity_1_name in entity_2_name or entity_2_name in
                        entity_1_name:
                        # Create one combined cluster of text and positions...
```

Listing 6.17: Results of applying Cluster Consolidation to Earlier Mapping: 6.14

```
Removing Hoyle as substring of Lindsay Hoyle
Removing Anderson as substring of Lee Anderson MP
Removing Lindsay as substring of Lindsay Hoyle
Number of entities before mention threshold application: 4
number of entities after mention threshold application: 4
Entity Cluster Map Consolidation Time: 0.0009729862213134766 seconds
```

Listing 6.18: Finalised Clusters for Sentence Interval Matching

```
{'Entity Name': 'Keir Starmer', 'Label': 'PERSON', 'Cluster Info': {'
    Cluster ID': 4, 'Cluster Text': ["Sir Keir Starmer's", 'Keir Starmer', '
    Sir Keir Starmer', 'the Labour leader', 'Sir Keir'], 'Cluster Positions'
    : [(0, 18), (168, 180), (574, 590), (600, 603), (900, 903), (959, 976),
    (1042, 1045), (1385, 1393)]}}

{'Entity Name': 'Lindsay Hoyle', 'Cluster Info': {'Cluster ID': 3, 'Cluster
     Text': ['Hoyle', 'Speaker Sir Lindsay Hoyle', 'the Speaker', 'Hoyle', '
    Sir Lindsay', 'Sir Lindsay'], 'Cluster Positions': [(224, 229), (1064,
    1089), (1197, 1208), (1324, 1329), (1734, 1745), (1751, 1753), (1824,
    1835), (1866, 1868), (1961, 1964)]}}

{'Entity Name': 'Lee Anderson Mp', 'Cluster Info': {'Cluster ID': 5, '
    Cluster Text': ['Lee Anderson MP, who lost the Tory whip', 'The former
    Tory deputy chairman', 'Mr Anderson', 'the MP's'], 'Cluster Positions':
    [(2527, 2566), (2568, 2599), (2691, 2702), (2740, 2743), (2822, 2830)]}}
```

```
{'Entity Name': 'Tory', 'Cluster Info': {'Cluster ID': 2, 'Cluster Text': [
    'Tory', 'the Tories', 'the Conservatives', 'Tory', 'the Conservatives',
    'Tory', 'Tory', 'Tory', 'the party's'], 'Cluster Positions': [(47, 51),
    (249, 259), (403, 420), (2005, 2009), (2046, 2063), (2107, 2111), (2557,
     2561), (2579, 2583), (2712, 2723)]}}
```

6.1.8 Determining Bounds and Applying NewsSentiment

Using an IntervalTree (3.1.5) the sentences that contain the coref mentions are identified. We then send these positions with the article text so NewsSentiment can be applied to each. The BoundError table (Fig. 5.3) will receive logs of the mention and sentence if there is an error to help improve future tokenization development. Multiple mentions in the same sentence are stored identifiably so an average score for the sentence can be calculated later. Applying NewsSentiment is the most time-consuming single operation in the pipeline, so for NERs sharing the same cluster results are reused for efficiency.

Listing 6.19: IntervalTree Application

```
bounds_tree = IntervalTree(Interval(start, end) for start, end in
                                    sentence_bounds)

# Seen cluster in a previous pair? Reuse results.
if cluster_id in cluster_id_mapping:
    for entry in cluster_id_mapping[cluster_id]:
        bounds_sentiment[bounds_key][entity_name][entity_db_id].append(entry['
            result'])
else:
    cluster_id_mapping[cluster_id] = []
    for mention_start, mention_end in cluster_positions:
        overlap = bounds_tree.overlap(mention_start, mention_end)
        if overlap:
            for interval in overlap:
                sentence_start, sentence_end = interval.begin, interval.end
                bounds_key = (sentence_start, sentence_end)
                result = self.bounds_sentiment(mention_start, mention_end,
                                               sentence_start, sentence_end,
                                               article_text, database_id)
                bounds_sentiment[bounds_key][entity_name][entity_db_id].append(
                    result)
                cluster_id_mapping[cluster_id].append({
                    'bounds_key': bounds_key,
                    'result': result
                })
```

Listing 6.20: NewsSentiment TSC Application Extract

```
left_segment = article_text[sentence_start:mention_start]
mention_segment = article_text[mention_start:mention_end]
right_segment = article_text[mention_end:sentence_end]
sentiment = self.tsc.infer_from_text(left_segment, mention_segment,
    right_segment)
```

6.1.9 Determining Overall Sentiment

Linear Scoring Method

The linear scoring method involves multiplying the average sentiment score for each bound by 100, summing each category, and calculating the percentage share of the totals.

```
NewsSentiment Candidateappearance1
Keir Starmer - Mention (959, 976) is within bounds (939, 1164)
MPs have called for the Labour leader to be reported to the Privileges Committee over allegations that his Party intimidated
Speaker Sir Lindsay Hoyle ahead of the SNP's opposition day debate on a ceasefire in Gaza last week.
NewsSentiment Candidateappearance1
Keir Starmer - Mention (1042, 1045) is within bounds (939, 1164)
MPs have called for the Labour leader to be reported to the Privileges Committee over allegations that his Party intimidated
Speaker Sir Lindsay Hoyle ahead of the SNP's opposition day debate on a ceasefire in Gaza last week.
NewsSentiment Candidateappearance2
Keir Starmer - Mention (1385, 1393) is within bounds (1379, 1614)
While Sir Keir has "categorically" denied the claims, reports suggested Commons leader Penny Mordaunt believes there could
have been a "breach of privilege" and an investigation is one of a number of potential options being considered.
```

Figure 6.5: Main Notebook Terminal Output - Target Mentions in Blue

Table 6.1: Sentiment Analysis Results for Keir Starmer

Entity	**Bound Start**	**Bound End**	**Avg (Neutral/Positive/Negative)**	**Bound Text**
Keir Starmer	0	167	0.00 / 0.00 / 0.94	Sir Keir Starmer's popularity drops while some Tory voters vow to vote Reform Labour's popularity drops following a rocky week in Westminster for the Opposition party.
Keir Starmer	168	315	0.00 / 0.00 / 0.90	Keir Starmer denies threatening to withdraw support for Hoyle Labour's lead over the Tories has dropped by three points, according to a fresh poll.
Keir Starmer	564	659	0.00 / 0.00 / 0.91	Meanwhile Sir Keir Starmer recorded his lowest approval rating since May last year at plus two.
Keir Starmer	859	938	0.00 / 0.62 / 0.00	Some 35% of people approved of his performance while 33% disapproved.
Keir Starmer	939	1164	0.00 / 0.00 / 0.95	MPs have called for the Labour leader to be reported to the Privileges Committee over allegations that his Party intimidated Speaker Sir Lindsay Hoyle ahead of the SNP's opposition day debate on a ceasefire in Gaza last week.
Keir Starmer	1379	1614	0.00 / 0.00 / 0.98	While Sir Keir has "categorically" denied the claims, Commons leader Penny Mordaunt believes there could have been a "breach of privilege" and an investigation by the standards committee is being considered.

For instance, given the sentiment scores:

$$\begin{aligned} \text{Neutral:} &\ 0.62 \times 100 = 62 \\ \text{Positive:} &\ \text{N/A} \\ \text{Negative:} &\ 0.94 \times 100 + 0.90 \times 100 + 0.91 \times 100 + 0.95 \times 100 + 0.98 \times 100 = 468 \end{aligned}$$

The total score is 530, with the percentage distribution as follows:

$$\begin{aligned} \text{Neutral:} &\ \frac{62}{530} \approx 12\% \\ \text{Negative:} &\ \frac{468}{530} \approx 88\% \end{aligned}$$

Exponential Scoring Method

For the exponential scoring, the average score for each bound is cubed before proceeding as above.

Given the same categories:

$$\begin{aligned} \text{Neutral:} &\ (0.62)^3 \times 100 \approx 24 \\ \text{Positive:} &\ \text{N/A} \\ \text{Negative:} &\ ((0.94)^3 + (0.90)^3 + (0.91)^3 + (0.95)^3 + (0.98)^3) \times 100 \approx 411 \end{aligned}$$

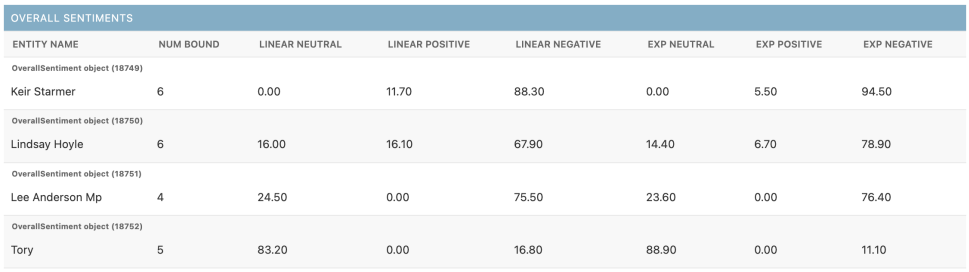
Thus, the total score is 435, with the percentage distribution:

$$\text{Neutral: } \frac{24}{435} \approx 5\%$$
$$\text{Negative: } \frac{411}{435} \approx 95\%$$

By cubing, the probability that values closer to 1 (high confidence) become even larger relative to their original scale, while values closer to 0 (low confidence) shrink further. This disproportionately scales up the impact of high-confidence predictions in the overall sentiment calculation. On the other hand, we see that predictions, where the model is less certain, have less influence now. Currently, we use the exponential variant in API calls (but the linear variant could easily be swapped in). Furthermore, the BoundMentions are stored in a form similar to the above, so any new logic, e.g., using thresholding to ignore a particular confidence, could easily be applied with a Python script run as a one-off Django Management command and update all records.

Linear or exponential, this approach brings the risk of introducing misleading results if NewsSentiment class probabilities are not well calibrated, so investigations into them can be found in 9.1. The storage of these overall sentiment values for each named entity concludes the pipeline with the article marked as processed in the database.

Figure 6.6: Article Marked as Processed (Admin Portal)



OVERALL SENTIMENTS

ENTITY NAME	NUM BOUND	LINEAR NEUTRAL	LINEAR POSITIVE	LINEAR NEGATIVE	EXP NEUTRAL	EXP POSITIVE	EXP NEGATIVE
OverallSentiment object (18749)							
Keir Starmer	6	0.00	11.70	88.30	0.00	5.50	94.50
OverallSentiment object (18750)							
Lindsay Hoyle	6	16.00	16.10	67.90	14.40	6.70	78.90
OverallSentiment object (18751)							
Lee Anderson Mp	4	24.50	0.00	75.50	23.60	0.00	76.40
OverallSentiment object (18752)							
Tory	5	83.20	0.00	16.80	88.90	0.00	11.10

Figure 6.7: OverallSentiment records linked to Article (Admin Portal)

6.2 App Views

6.2.1 Entity View

We leveraged Vue components in Entity View with the three columns declared within an Article Entries Container. Once the API call returns, each article, depending on its sentiment, is passed to a column to create new Article Entry components within that class. The entity ID used in the API call is extracted from the URL.

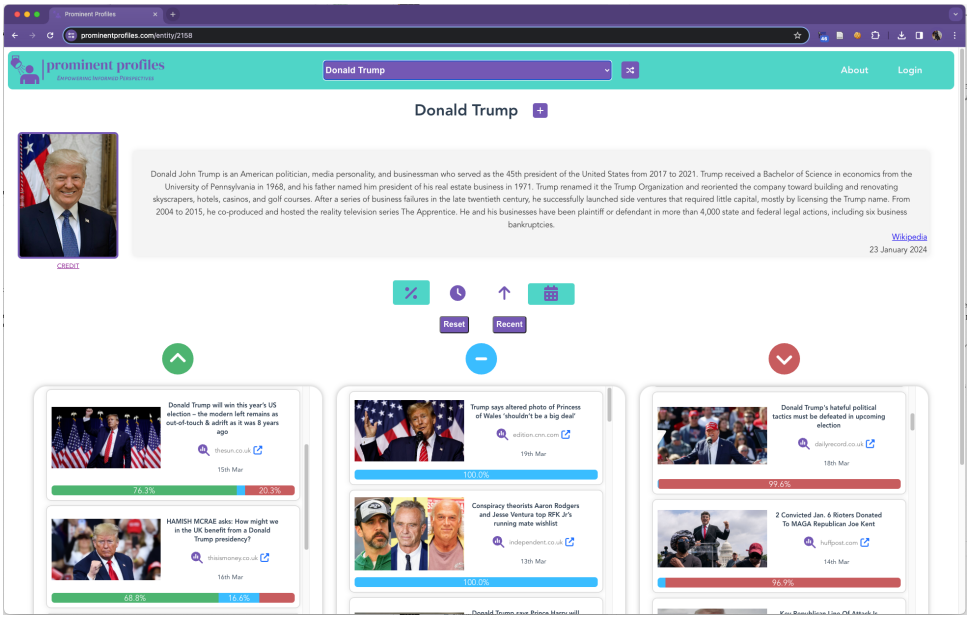


Figure 6.8: Entity View

```
<div class="entry-column">
    <!-- Positive Entries -->
    <ArticleEntry v-for="entry in positiveEntries" :key="entry.id" :entry="entry" />
</div>
```

Listing 6.21: Article Entry Component Use

```
    <div class="sentiment-bar">
        <div class="positive" :style="{ width: positiveWidth }">
      <template v-if="isWidthSufficient(positiveWidth)">{{ positiveWidth }}</template>
    </div>
```

Listing 6.22: Sentiment Bar Component Extract

Changes from Wireframe Design

Filtering functionality declared in `SortToggle.vue` has evolved to allow custom ordering direction by sentiment or time, and we have used a library, VDatePicker, to enable users to set a custom date range. We highlight active filters using the same green as the header to complement and provide a consistent UI appearance. Moreover, we incorporated reset and recent (last seven days) buttons using user feedback to speed up filter adjustment.

The Profile's name is now shown clearly, with the subscription button component appended alongside it to make it clearer that you are subscribing (following YouTube/X convention).

We declared a Sentiment Bar in `ArticleEntry.vue` to replace the single classification intensity so a full sentiment breakdown is provided without clickthru to Article Analysis View. If the bar is of sufficient width, the % value of that category to 1dp will also be displayed. We also added the publication date for completeness.

Font-Awesome icons and careful use of colour have been incorporated to replace column headings, and to clearly indicate how to reach the analysis page (magnifying glass) and the article itself (external link). Photo URLs were later added to the article extraction process, so they are shown here to improve visual appeal, and the news publisher domain has been added to fairly credit them close to the content.

6.2.2 Article Analysis View

Article Analysis View utilises EntitySpotlight and ProfileEntry components to provide a consistent appearance that dynamically adjusts to the entities linked to the article. An API call that specifies the

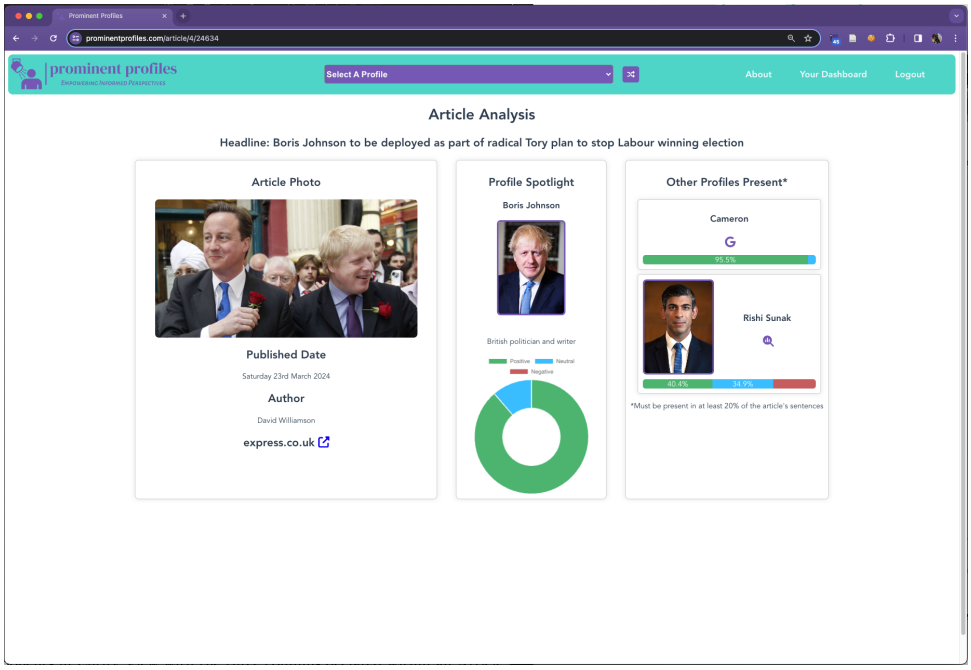


Figure 6.9: Article View

article ID and entity wanted in the spotlight from the URL. Hence, the OverallSentiment database lookup and response time are much faster than those used on Enity View. Other entities analysed in the same article are shown on the right - each entity calls a minimal Bing Entity API to obtain the photos and a short description for the entity in the 'spotlight'.

Other entities mentioned in the same article are displayed on the right, using a minimal Bing Entity API call to fetch their photos and brief descriptions for the 'spotlight' section. If an entity is not set as visible in the app and hasn't been queried on Bing, it will appear without a photo accompanied by a link for a Google search. TThis approach ensures that entities that are unresolved, lack sufficient data for a dedicated page, or are switched off by admins are not prominently featured.

```
    <h2>Other Profiles Present*</h2>
    <div>
       <ProfileEntry v-for="entry in Article" :key="entry.id" :entry="entry" />
      </div>
    <p class="bottom-paragraph">*Must be present in at least 20% of the article's
    sentences</p>
    </div>
```

Listing 6.23: Profile Entry Component Use

```
    const apiUrl =
        `${API_BASE_URL}/profiles_app/overall_sentiments/exp/article/${EntityId}/${
    ArticleId}/`
      fetch(apiUrl)
        .then((response) => response.json())
        .then((data) => {
          this.Article = data
          console.log(this.Article)

          // 0th entry is always the primary entity in the URL so it matches click
    throughs
          this.chartdata.datasets[0].data = [data[0].positive, data[0].neutral, data[0].
    negative]
        })
        .catch((error) => continues...
```

Listing 6.24: Article Analysis View API Call Extract

Changes from Wireframe Design

In the revised design, a donut chart has been implemented for enhanced visual appeal, with individual scores becoming visible upon interaction. Also, the date format has been refined to include the day, catering to user preference for specificity when reviewing recent news events delivered by our app. Where possible, the author's name is now displayed to give greater attribution. The design also incorporates the previously mentioned Sentiment Bar and Font Awesome icons for improved aesthetic and functional elements.

6.2.3 Home View

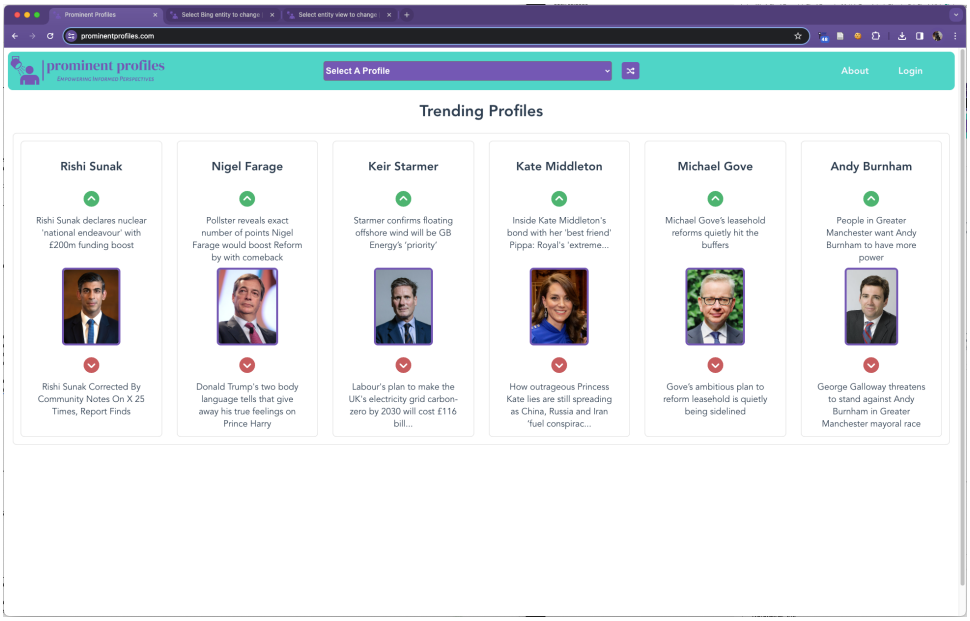


Figure 6.10: Home View

Home View delivers Trending Profiles. We use a cookie, viewedProfiles across the frontend, to determine them. The cookie means a single user cannot skew results, as a maximum of one visit per day is recorded via an API call that pushes a record into EntityView (db table) to record 'hits' anonymously. A different approach would have used user accounts, but then you would miss all anonymous visits. If a user rejects non-essential cookies, their views won't contribute (8.3.2).

```
    VueCookie.set('viewedProfiles', JSON.stringify([...viewedProfiles, entityId]), {
    expires: 1 day})

    async incrementViewCount (entityId) {
    try {
        // Check if non-essential cookies are accepted
        const accepted = VueCookie.get('non_essential_cookies_accepted')
        if (!accepted) {
        console.log('Non-essential cookies not accepted. View count not incremented.')
        return
        }

    // Check if the user has already viewed the profile in the current session
    // JSON.parse converts the string to an array
    const viewedProfiles = JSON.parse(VueCookie.get('viewedProfiles') || '[]')

    if (viewedProfiles.includes(entityId.toString())) {
        console.log(`Already viewed in this session: ${entityId}`)
        return
    }
```

Listing 6.25: Entity View Visited → Update Cookie

Listing 6.26: get_trending_entities() API extract

```
time_ranges = [24, 72, 120]
    for hours in time_ranges:
        end_time = timezone.now()
        start_time = end_time - timezone.timedelta(hours=hours)

        # Top 6 entities within the time range
        top_entities = EntityView.objects.filter(view_dt__range=(start_time,
            end_time)) \
                        .values('entity__id', 'entity__name') \
                        .annotate(total_views=Count('entity__id')) \
                        .order_by('-total_views')[:6]
        # Continues to check until a time range gives us a top six else return
            whatever was found using 120 hours...
```

Changes from Wireframe Design

We've refined each `TrendingProfileCardcomponent` by removing neutral perspectives to better align with our app's focus on positive and negative viewpoints. Additionally, we've introduced dynamic animations for displaying articles, where a smooth left-to-right transition every 5 seconds highlights the five most recent positive or negative articles provided by the API. To enhance user experience, all items are interactive: clicking on a headline navigates to the Article Analysis View while clicking on a photo leads to the Entity View for that profile. Moreover, hovering over an entity on a desktop (or long-pressing on mobile) reveals the photo's attribution.

6.2.4 Sign Up / Login View

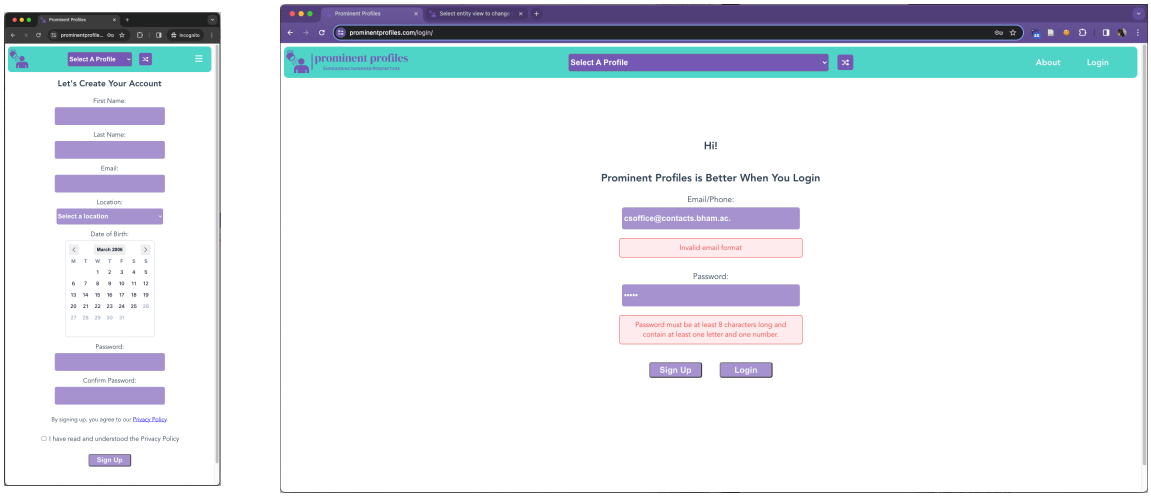


Figure 6.11: User Account Views

These views are designed to manage user authentication through the accounts API, enabling new user registrations, logging in existing users, and displaying rejection messages for scenarios like email addresses is already registered. These views and various app parts were initially built using the Vue Options API. However, we have (partially) transitioned to the (more functional & reactive) Composition API. This shift allows for the reuse of shared methods across different views—such as client-side email validation at both signup and login—thereby reducing redundancy and enhancing code maintainability. Generally, the enter key moves the user to the next input for convenience.

```
    <div class="form-group">
    <label for="lastName" class="label">Last Name:</label>
    <input type="text"
           id="lastName"
           ref="lastNameInput"
           v-model="lastName"
           class="input-field"
           @keyup.enter="focusEmailInput"/>
```

```
    <p v-if="validationMessageLastName" class="validation-message">{{
    validationMessageLastName }}</p>
  </div>
```

Listing 6.27: Sign Up View Extract

```
  const validatePhone = () => {
    clearTimeout(validationTimerPhone)
    validationTimerPhone = setTimeout(() => {
      const trimmedPhone = phone.value
      if (trimmedPhone.startsWith('+44') && trimmedPhone.length > 3) {
        const nationalFormatPhone = '0' + trimmedPhone.substring(3)
        const isPhoneNumber = validator.isMobilePhone(nationalFormatPhone, 'en-GB')
        validationMessagePhone.value = isPhoneNumber ? '' : 'Invalid UK mobile number
    format'
      } else if (!trimmedPhone.startsWith('+44')) {
        validationMessagePhone.value = '+44 is required at start here'
      } else {
        validationMessagePhone.value = ''
      }
    }, 1000)
  }
  watch(phone, validatePhone)
  return { validationMessagePhone, validatePhone }
}
```

Listing 6.28: validationUtils.js Extract

Changes from Wireframe Design

The form now requests the user's date of birth, limiting eligibility to those over 16 by turning off younger calendar dates to comply with General Data Protection Regulation (GDPR) (8.3.1).

Each field is validated in real-time to provide immediate feedback to users and facilitate error correction before form submission. This approach minimises user frustration by pinpointing errors as they occur rather than requiring users to backtrack. Mobile numbers are collected for users selecting the UK as their location, offering a more straightforward method for signing in.

Figure 6.12: UK Users Mobile Input

Figure 6.13: Mobile Validation

6.2.5 Dashboard View

To implement the desired dashboard counts since the last visit, we used Django middleware to keep track of the user's last visit before the current day. Middleware means the code runs on every request to the Django server, which is purposeful here. This last visit could then be used to filter articles published since to provide the counts. A special flag was introduced to the overall sentiment API to optimise performance. When the dashboard parameter is `true`, the database query is narrowed down before data retrieval. This approach is more efficient than fetching all records and filtering on the client side, reducing server load and improving user response times.

We tweaked Entity View to read a start and end date range, which we passed to Entity View to set the calendar so articles match the dashboard counts in each category. This ensures alignment between the article counts displayed in the dashboard and the users' clicking through to view them on Entity View.

Listing 6.29: Django Accounts Middleware

```
def process_request(self, request):
if request.user.is_authenticated:
    now = timezone.now()
    user = request.user
```

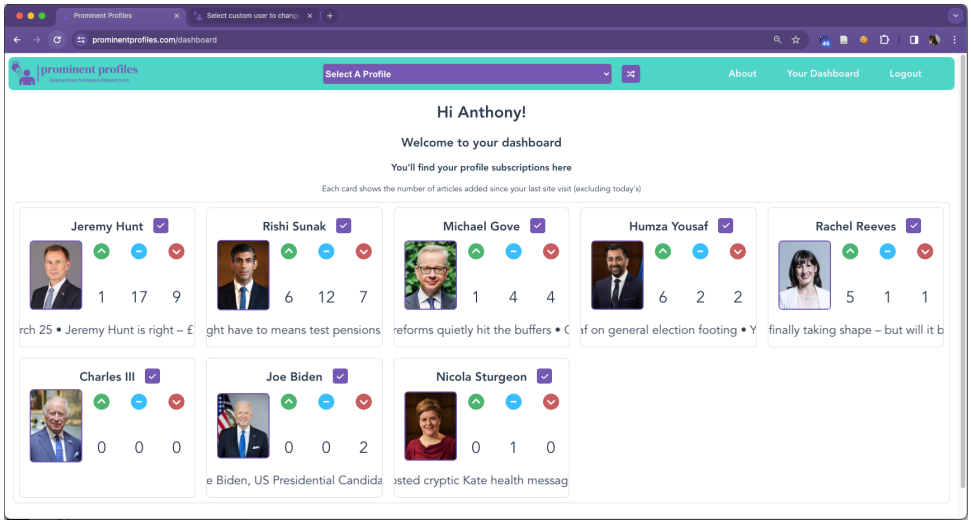


Figure 6.14: Dashboard View

```
5
6       # Check if the last visit excluding today is either not set or was before
            today
7       if not user.last_visit_excluding_today or user.last_visit_excluding_today.
            date() < now.date():
8
9           # Update last_visit_excluding_today to the previous value of last_visit
10          # only if last_visit was set and was on a previous day
11          if user.last_visit and (
12                  not user.last_visit_excluding_today or user.last_visit.date()
                        != now.date()):
13              user.last_visit_excluding_today = user.last_visit
14
15      # Update last_visit to now on every authenticated request - point of
            middleware!
16      user.last_visit = now
17      user.save()
```

```
1     setDateRangeFromURL () {
2       const urlParams = new URLSearchParams(window.location.search)
3       const lastVisit = urlParams.get('last_visit')
4       if (lastVisit) {
5         const lastVisitDate = new Date(lastVisit)
6         const currentDate = new Date()
7         this.dateRange = {
8           start: lastVisitDate,
9           end: currentDate
10        }
11      } else {
12        // If `last_visit` is not provided, set the date range to the last 14 days...
```

Listing 6.30: `SortToggle.vue` using URL Dates

Changes from Wireframe Design

We have made enhancements to improve UX and UI aesthetics. The layout of the cards has been revamped to utilise horizontal space, facilitating a more compact presentation more efficiently. The SubscriptionButton component has been integrated here, allowing users to easily unsubscribe from an entity with minimal development effort due to its reusable nature from Enity View. Font-Awesome icons have replaced text, enhancing the interface's visual appeal. Additionally, article headlines now scroll across the screen like a news ticker, capturing user interest at a glance.

6.3 Django API Views

Our Vue frontend required `profiles_app` and `accounts` API endpoints in Django (Table 6.2) to fetch content and interact with user data. The `profiles_app` endpoints provide publicly accessible content, allowing anonymous users to view entity profiles and their article-by-article overall sentiment data without needing authentication, ideal for services that benefit from open access[1]. Meanwhile, the accounts endpoints are protected by JSON web tokens to manage state and user ID because they necessitate user authentication and handle sensitive actions like user registration, password management, and subscriptions.

[1] Creating an Entity View record is an excpetion as it controls Trending Profiles determiantion.

App	API Name	Description	Vue Usage
accounts	get_sub_list	Gets a list of every entity a user is subscribed to for the dashboard page.	`SubProfilesGrid.vue`
accounts	get_sub_status	Identifies if a user is subscribed to the entity they are viewing in frontend.	`SubscriptionButton.vue`
accounts	get_user_data	Obtains first name for a personal dashboard welcome.	`DashboardPage.vue`
accounts	password_reset	Provides a method to request a password reset token based on an e-mail address.	`ForgotPassword.vue`
accounts	password_reset _validate_token	Verifies a password reset token is valid.	`ResetPassword.vue`
accounts	password_reset _confirm	Provides a method to reset a password with a valid reset token.	`ResetPassword.vue`
accounts	register	Registers a user provided their email and phone is unique (check is in serializer).	`SignUpPage.vue`
accounts	toggle_sub	Processes a users request to (un-)subscribe.	`SubscriptionButton.vue`
accounts	token	Provides an authentication token.	`LoginPage.vue`
accounts	token_refresh	Takes a refresh type JSON web token and returns an access type JSON web token if the refresh token is valid.	`auth.js`, `axios.js`
profiles_app	bing_entities_mini	Provides a minimal view (name, picture) of entity details obtained from Bing, primarily for faster content loading.	`ArticlePage.vue`, `ProfileEntry.vue`, `SubProfileCard.vue`, `TrendingProfileCard.vue`
profiles_app	bing_entities	Provides the full view of entities complementing details obtained from Bing (adds description), for more detailed pages.	`EntityPage.vue`
profiles_app	entities	Fetches and returns a list of all entities marked as visible by admins.	`EntitySelection.vue`
profiles_app	overall_sentiments _exp_article	Gets exponential sentiment analysis results for all entities mentioned in a specific article.	`ArticlePage.vue`
profiles_app	overall_sentiments _exp	Retrieves exponential scaled sentiment scores for a specific entity.	`ArticleEntriesContainer.vue`, `SubProfileCard.vue`, `TrendingProfileCard.vue`
profiles_app	overall_sentiments _lin	Retrieves linear scaled sentiment scores for a specific entity.	Interchangeable w/ above cell contents.
profiles_app	entity_name	Returns the name of an entity from provided ID, or an error message if not found.	`ProfileEntry.vue`, `SubProfileCard.vue`, `TrendingProfileCard.vue`
profiles_app	get_trending_entities	Identifies and returns trending entities based on view counts over gradually increasing time ranges.	`TrendingProfiles.vue`
profiles_app	create_entity_view	Creates a new view to support trending entities functionality.	`EntityPage.vue`

Table 6.2: Django API Endpoints

6.4 Deployment

6.4.1 Docker Containers

Docker containerization (Fig. 6.15) allowed a microservices architecture, with each service running in its container. This separation of concerns makes the system more modular and allows for easier scaling and maintenance of individual components. It also complimented our CI/CD pipeline as the dependencies are installed automatically via the build files, and container isolation avoids conflicts between each component.

Web is our Django app, served via Gunicorn to productionise it by handling any incoming requests and serving our API views for the Vue app. The Vue app is built and copied into the static volume. Nginx is a reverse proxy that serves static content (Vue app) or redirects URLs to the Django back end for API requests or administrator activities. Certbot triggers to renew our certificate so we can continue to serve HTTPs over 443. Celery mounts the media volume to read the article JSON files obtained from Bing API calls.

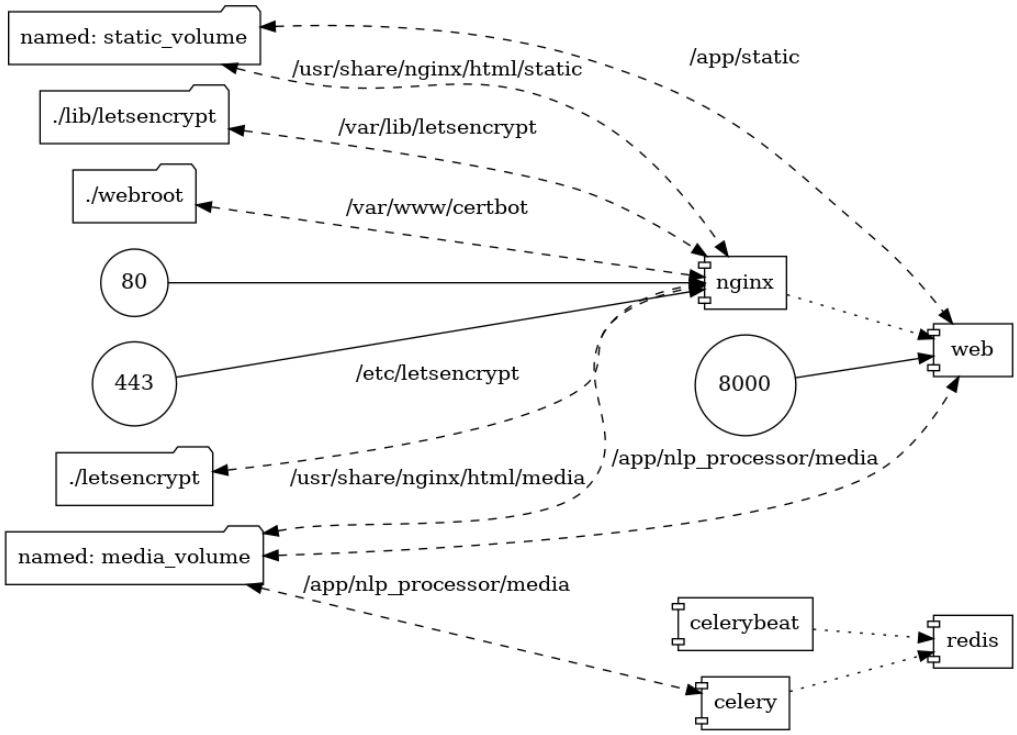


Figure 6.15: Docker Compose Configuration

6.4.2 Automating Regular Jobs

In the initial weeks, article collection, download and pipeline operation were chores our admin had to remember (and occasionally forgot) to take care of. So, a solution to automatically run these essential operations was required. Celery is a task queue, allowing for the execution of asynchronous and scheduled jobs[79]. Celery Beat is a scheduler that integrates with Django, triggering our management commands and providing an admin page to set schedules and monitor activity[80]. Redis acts as a message broker over port 6379, between Django and Celery, storing the messages and job queues that Celery workers use to process background tasks[81].

Fig. 6.16 shows the collection of articles from Bing occurs twice a day, with the pipeline running 30 minutes later. SimilarEntityPairs creation takes place as often to react to new unresolved entities from

[1]The `web` and `celerybeat` containers communicate via `redis` over port 6379, but this isn't shown in Fig. 6.15 as it is not exposed for security reasons. The `certbot` container is omitted for clarity.

NAME	ENABLED	SCHEDULER	INTERVAL SCHEDULE	START DATETIME	LAST RUN DATETIME	ONE-OFF TASK
Update Similar Entity Pairs	✓	every 12 hours	every 12 hours	March 2, 2024, 5 p.m.	April 4, 2024, 5 a.m.	✗
celery.backend_cleanup	✓	0 4 * * * (m/h/dM/MY/d) GMT	-	-	April 4, 2024, 4 a.m.	✗
Scrape Articles + NLP Analysis	✓	every 12 hours	every 12 hours	April 4, 2024, 12:45 p.m.	April 4, 2024,12:45 a.m.	✗
Collect Article Data (Bing API)	✓	every 12 hours	every 12 hours	April 4, 2024, 11:45 a.m.	April 4, 2024, 11:45 p.m.	✗
Visible Entities - Bing Entity LookUp	✓	every day	every day	-	April 3, 2024, 4:35 p.m.	✗
One Off Visible Entities - Bing Entity LookUp	✗	every 10 seconds	every 10 seconds	-	-	✓
One Off Scrape Articles + NLP Analysis	✗	every 10 seconds	every 10 seconds	March 12, 2024, 4 a.m.	-	✓
Test Hello Task	✗	every minute	every minute	-	-	✓

Figure 6.16: Celery Beat Admin Overview

fresh articles. Finally, the fetching of supplementary info (for entities set visible for first time)from Bing Entity API takes place once a day.

6.4.3 Forgot Password

We required an email address to implement a forgot password functionality that went beyond security questions. We used our Namecheap domain and Zoho mail to create info@prominentprofiles.com by configuring MX DNS records.

Initially, Django's built-in password reset view was used. Yet, this meant users would have to visit a Django template page outside of the Vue app, and this looked unprofessional, so we used `django-password-reset` to enable API views allowing integratation of 'forget password' into a journey that appears if the user gets their password wrong and stays within the SPA.

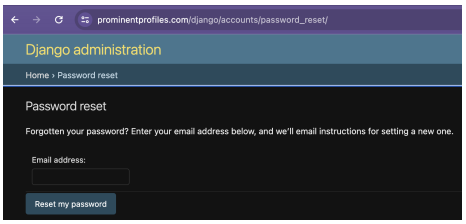


Figure 6.17: Django Pwd Reset

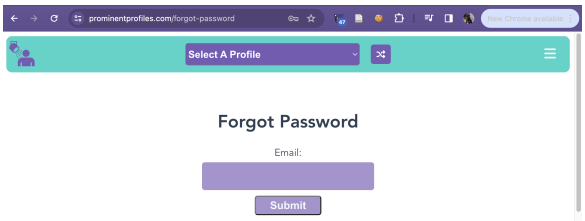


Figure 6.18: SPA Pwd Reset

Additional Email Benefit: The email set-up could also be used to log the successful start and end of jobs to pinpoint if a deployment or memory error had interrupted the pipeline job, making it more convenient for admins to become aware of failed runs (e.g. by setting up email folder for subject contains 'Job').

Listing 6.31: Send Email Start/End of Job

```
1       send_mail(
2           'NLP Job Finished',
3           'The scrape and analyse articles job has finished.',
4           settings.EMAIL_HOST_USER, # Sender
5           [settings.EMAIL_HOST_USER], # Recipients - could add admins emails
6           fail_silently=True,
7       )
```

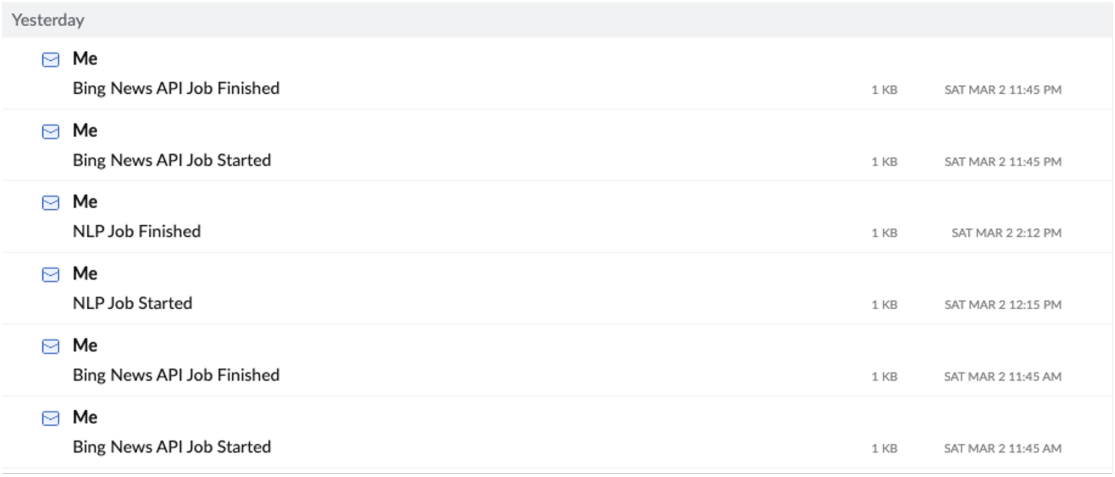


Figure 6.19: Job Email Updates

6.5 Challenges and Unexpected Additions

6.5.1 Entity Resolving

The Merge Review interface was created using custom templates to provide administrators with a semi-automated way to resolve entities. Fuzzy matching is deployed in a daily job, to identify pairs of entities with similar string patterns. Prominent Profiles presents these to administrators for review. Admins can examine the origins of each entity and decide whether to merge or ignore them. Merging updates all database references and logs the action in ry. Ignoring an entity creates a record to prevent it from reappearing as a candidate. This approach replaced on-demand matching to avoid long loading times due to the growing number of entities.

6.5.2 Article Duplication

By January, when the frontend prototype was complete, and a basic Entity View showed article cards, a problem became apparent: very similar articles appeared in the front end.

This presents a few issues:

- 1. As a user, it looks awry to see the same article card two (or more) times with often the same sentiment score, and this could undermine trust in us as a news aggregator.
- 2. NewsSentiment application is expensive, taking approx. 30 to 120 seconds per article, duplication wastes computation and needlessly extends the processing time.

Initially, we defined URL as a unique field when designing the Article model, but the issue runs deeper. News Agencies Reuters and Associated Press have been found to supply remarkably high proportions of routine reports to outlets[82]. Moreover, Trinity Mirror (now Reach plc) bought the local paper publisher Local World in 2015, and so it logically follows that news is shared between Reach's national newspapers (e.g. Daily Mirror, Daily Star) and its 240 regional papers[83, 84].

The Northern Scot and Forres Gazette seem similar in appearance; we find they are both operated by Highland News & Media Ltd.

Metadata Comparisons

The initial solution proposed was string equality comparisons between article headlines in the database. This approach was flawed. Firstly, for some straightforward headlines, 'Jeremy Hunt cuts NI by 2%', there is a risk multiple stories may use the same headline but write very different text to cover the event.

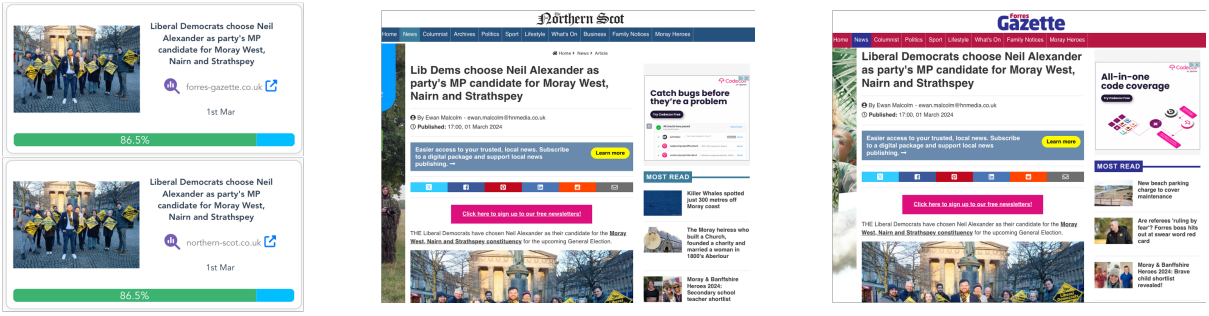


Figure 6.20: Similar Articles in PP from the Northern Scot and Forres Gazette

Secondly, if you look closely at Fig. 6.20, the left headline uses 'Lib Dems', but the right uses 'Liberal Democrats', so using equality alone would fail. These instances arising from editorial decisions are not uncommon, or those where the headline is updated on a single news site (e.g. a quotation corrected, a word is replaced), causing a slight change in the URL and, thus, our analysis of a very similar article at the same publisher.

To investigate this headline likeness, we explored our database of article headlines using Levenshtein distance (Ld), which is the minimum number of single-character edits required to change one string into another. To tackle the straightforward headlines, we explored combining them with the author(s) extracted by Trafilatura. We developed `similar_headline_clean.py`: it compares all articles with each other and if $LD(headline_1, headline_2) < 5$ characters different to another and the $LD(author_1, author_2) < 3$ characters then the 2nd instance of the article was deleted. Also, if the $LD(headline_1, headline_2) < 10$ (e.g. a whole word changed), with $LD(author_1, author_2) < 3$, then the terminal would display both articles headline, authors, and publishers to an administrator and they could manually review whether deletion should take place.

While this solution did successfully identify some duplicates it remained unsatisfactory as authors are not always successfully extracted or provided by news sites, it required infeasible manual intervention (PP now considers 100s of articles a day) and currently only solved issue 1 as this operation was taking place at the admins convenience not at the specific, strategic times we would like to scrape new articles.

Linguistic Comparisons

The project supervisor advised starting by counting occurrences of 'the' . Indeed, the exploration of function words ('prepositions, common adverbs, pronouns') as indicators of style frequencies can be traced back to 1963[85]. Examples of function and stop words overlap. We see counts of them used to construct formulas to quantify particular features of an author's writing style or the development of vector space models in attempts to detect plagiarism[86]. Genre detection has also leveraged such common words[87, 88].

To develop our approach, we explored using various linguistic and statistical methods to determine a composite metric to apply to our articles rather than relying on a single statistic. In the references above, many of these methods are mentioned. However, we took a practical approach, defining a set of ten verified duplicate article pairs and ten unrelated articles and applying all measures available in the LexicalRichness package. We then calculated the % difference in these values for the pool of similar and unrelated and then selected measures based upon those that maximised similar % and minimised unrelated % differences.

From this exploratory work, we retained:

- Word Count
- Terms: Number of unique words.
- Vocd: Vocabulary Diversity.
- Yulek: A measure of text diversity focusing on word frequency distribution.
- Simpsond: A measure of diversity using number of occurrences of each word.

Hashing Comparisons

Alongside small editor changes, duplicates occurring on other news sites will have slightly different article text body extractions. We sometimes see picture/advertisement captions, related news headlines, and text below an article body detected. This ruled out using any secure hash on the article text since the level of variation would result in entirely different hashes.

We required a means to produce the fingerprints of the articles to give us an estimate of the closeness between them. Fuzzy hashing has evolved from use in spam and malware detection to the ability to detect duplication on web servers even where CSS, headers, or metatags differ, and variations in spacing, paragraphs, and diacritics are present[89]. Our use case suitably complements this, and ideally, [90] has brought this capability to Python.

Models

The ArticleStatistics model stores the fuzzy hash and linguistic statistics and has a one-to-one relationship with an Article.

We used a one-off job to create SimilarArticlePairs performing `ppdeep.compare(h1, h2)`, and calculate percentage differences on other measures on all combinations of ArticleStatistics records. The linguistic statistic thresholding was driven by our exploratory work and the observation of candidate SimilarArticlePairs that were, in fact, duplicates, which are easier to spot manually in an admin view with all metadata presented in the Django Admin view.

Listing 6.32: Extract from article_update.py

```
    def calculate_all_percentage_differences(pair):
        stat1 = pair.article1
        stat2 = pair.article2
        pair.words_diff = calculate_percentage_difference(stat1.word_count,
            stat2.word_count)
        # ...
        pair.with_diff = calculate_percentage_difference(stat1.with_count,
            stat2.with_count)
        pair.save()
```

If $compare(article1, article2) > 90$, then without considering other measures, this is regarded as a duplicate pair. Else > 65 would be used with the determined % thresholds.

Initially, as we had analysed the duplicates already, we introduced a SimilarArticlePair check into the OverallSentimentView call using the determined logic. For performance reasons, we introduced a time range of articles published 14 days before the previous one.

Listing 6.33: Extract from profiles app views

```
has_similar_pair = SimilarArticlePair.objects.filter(
    Q(article2_id__in=article_ids_for_similarity_check),
        Q(hash_similarity_score__gte=90) |
        (
        Q(hash_similarity_score__gte=65, words_diff__lt=10, terms_diff__lt=10,
        vocd_diff__lt=5, yulek_diff__lt=10, simpsond_diff__lt=10,
        the_diff__lt=20, and_diff__lt=20, is_diff__lt=20,
        of_diff__lt=20, in_diff__lt=20, to_diff__lt=20,
        it_diff__lt=20, that_diff__lt=20, with_diff__lt=20
        ) & Q(
            article2__article__publication_date__lte=F(
            'article1__article__publication_date') + timedelta(days=14)
        )
```

At this stage, the obvious perception of duplicates in the Vue app was resolved (Issue 1).

Pipeline Update

To stop duplication detection from being an ongoing job performed after the pipeline and thus resolve Issue 2, we incorporated our new methods into the pipeline around6.1.2. An ArticleStatistics record is created for an Article, and then comparisons are made with other article statistics for articles published up to four days before.

Listing 6.34: Pipeline with Similarity Check

```
    article_obj = Article(url, headline, article_text, ner, publication_date,
                                                author, site_name, source_file)

    start_time = time.time()
    article_obj.get_statistics()
    # print(f"Statistics Calculation: {time.time() - start_time} seconds")
    # About 2 seconds

    article_obj.save_to_database()
    too_similar = check_similarity_with_timeout(article_obj)

    if too_similar:
        article_obj.set_db_processed(is_processed=False, similar_rejection=True
            )
        print(f"{article_obj.headline} removed on similarity grounds!")
        article_obj.text_body = None  # Free memory explicitly
```

In the event of hash > 65, a SimilarArticlePair is created - this gives future flexibility to adjust the linguistic statistics criteria.

If `true` is returned, the Article is still stored in the database with a similar_rejection flag set to `true` and discarded from further processing (Issue 2). This means we implicitly decided to show the 1st captured publication only on PP.

You may note the decrease from 14 days to 4 days prior in consideration of combinations. This is due to 14 days, causing, in most cases, more time to perform than allowing NewsSentiment to run, making the solution futile. We investigated in a one-off-job and found the average date difference between the current qualifying SimilarArticlePairs was 0.48 days with a standard deviation of 2.7 days so 4 days is beyond adequate.

6.5.3 Responsive UI

For some profiles, we hold over 1K OverallSentiments, creating a noticeable delay when browsing Entity Views as the amount of information to be retrieved and returned by the API call is substantial. We decided to pre-set filters to the last 14 days and adapt the API to allow Vue to perform a quick 0 to 14-day overall sentiment call to populate the view immediately. Then, behind the scenes, the 15 to 180-day range is run, so typically, when the user changes the filter, all the data is ready.

```
mounted () {
        this.fetchData(true)
        this.fetchData(false)
      },

fetchData (quick) {
    const entityId = this.$route.params.id
        // Start from day 0 for quick, and day 15 for the full fetch
        const startDay = quick ? 0 : 15
        // Time period for the API call is based on the 'quick' flag
        const endDay = quick ? 14 : 180
        const apiUrl = `${API_BASE_URL}/profiles_app/overall_sentiments/exp/${entityId
    }/?endDay=${endDay}&startDay=${startDay}`
```

Listing 6.35: 2-Phase API Fetch

6.5.4 Managing Resource Constraints

To address memory limitations in development and production, we optimised our process by creating a fixed number of SentimentAnalyser instances, matching the number of threads instead of one per article; this minimised time and memory-intensive TSC model initialisations. These instances were efficiently distributed across threads using a queue, significantly enhancing our system's performance and memory usage. The explicit closing of database connections was also added in numerous locations to preserve them and stop job failures.

Listing 6.36: Queuing SentimentAnalyser Instances

```
    self.sa_queue = queue.Queue()
    self.max_concurrent_threads = ARTICLE_THREADS
    for _ in range(self.max_concurrent_threads):
        sa = SentimentAnalyser()
        self.sa_queue.put(sa)

    def process_article_wrapper(self, args):
        article, i, total_objects = args
        print(f"Current Progress: {i} of {total_objects}")

        sa = self.sa_queue.get()
        try:
            self.process_article(article, sa)
        finally:
            # Putting the SentimentAnalyser back in the queue, even if
                exception takes place
            self.sa_queue.put(sa)
            # Close connection used in thread due to digital ocean limit (22).
            connection.close()
```

It was also necessary to set and tweak a max article batch size for the FastCoref application to ensure we preserved server memory. A discrete bug that was tricky to identify was the unexpected exhaustion of all of a Celery worker's memory if the batch size was greater than the article texts given—the fix was simple: identifying it was trial and error.

Listing 6.37: FastCoref Batch Size Adjustment

```
batch_size=ARTICLE_BATCH_SIZE
if len(article_texts) < batch_size:
    batch_size = len(article_texts)
model = FCoref(device=F_COREF_DEVICE)
prediction = model.predict(texts=article_texts, max_tokens_in_batch=batch_size)
```

```
[2024-02-26 22:10:08,359: ERROR/MainProcess] Process 'ForkPoolWorker-1' pid:21 exited with 'signal 9 (SIGKILL)'
[2024-02-26 22:10:08,382: ERROR/MainProcess] Task handler raised error: WorkerLostError('Worker exited prematurely: signal 9 (SIGKILL) Job: 1.')
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/site-packages/billiard/pool.py", line 1264, in mark_as_worker_lost
    raise WorkerLostError(
billiard.einfo.ExceptionWithTraceback:
"""
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/site-packages/billiard/pool.py", line 1264, in mark_as_worker_lost
    raise WorkerLostError(
billiard.exceptions.WorkerLostError: Worker exited prematurely: signal 9 (SIGKILL) Job: 1.
"""
[2024-02-26 22:10:10,212: INFO/ForkPoolWorker-3] Scrape and analyse articles job started at 2024-02-26 22:10:10.212386
[2024-02-26 22:10:11,415: WARNING/ForkPoolWorker-3] /app/nlp_processor/media/api_articles/prince+harry/prince+harry_no_dup_articles_loop_26-02-2024-01:15.json
```

Figure 6.21: Celery Memory Exhaustion

6.5.5 Mobile Optimisations

Due to functional test failures on mobile devices we made numerous improvements:

- Defining minimum widths for forms and containers that use % screen width to scale on larger screens to utilise mobile screen space effectively.
- Creating a compact logo that appears when there isn't enough header width.
- Showing a hamburger menu when 'About', 'Your Dashboard', and 'Logout' extend beyond the view.
- Permitting components to scale to a smaller width by adjusting the publisher domain preview length.
- Introducing margins on vertically stack layouts, so differentiating between view and component scrolling is easier.
- Limiting containers' max height to wholly within the mobile frame.
- Moving the magnifying glass away from the external link icon to minimise mistaps.

```
const substringLength = computed(() => {
  if (windowWidth.value <= 500) {
  /* Card containers are vertically stacked at this point but mobile! */
    return 7
  } else if (windowWidth.value <= 1275) {
  /* Card containers are vertically stacked at this point */
    return 20
  } else if ..... continues
```

Listing 6.38: Publisher Domain Adjusted by Device

[1]Similar approaches to (Lst. 6.5.5) used to control logo change and hamburger appearance.

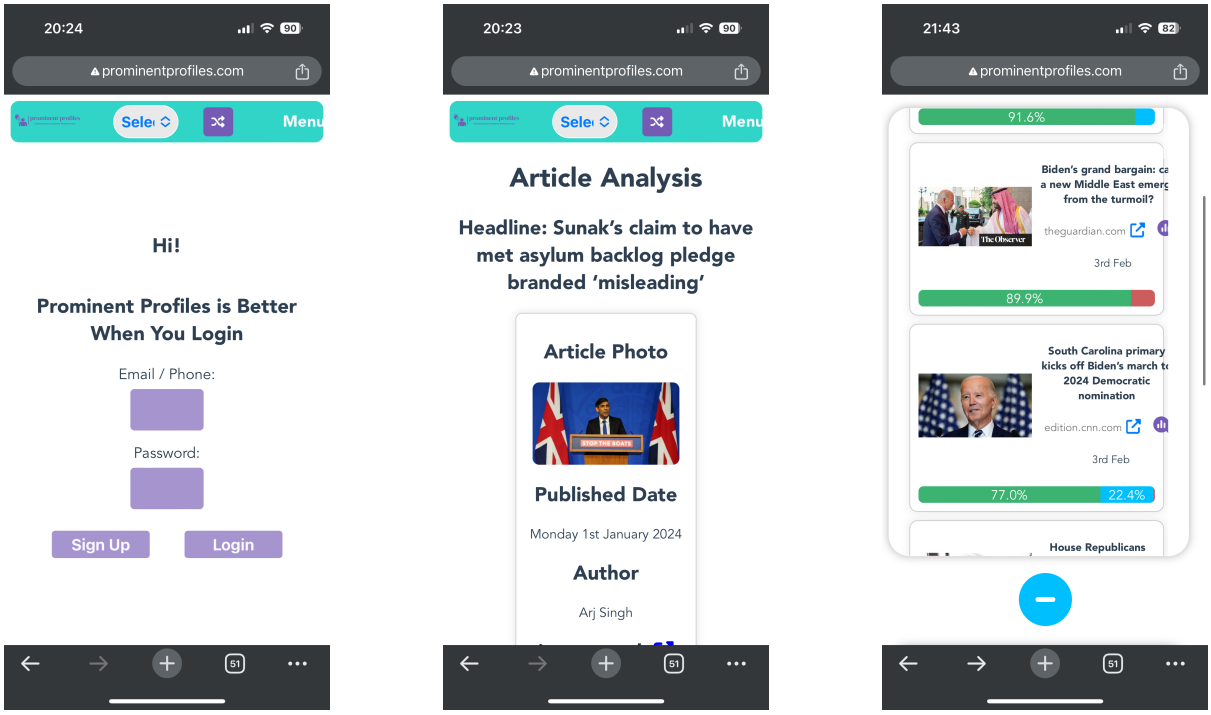


Figure 6.22: Before Mobile Optimisations

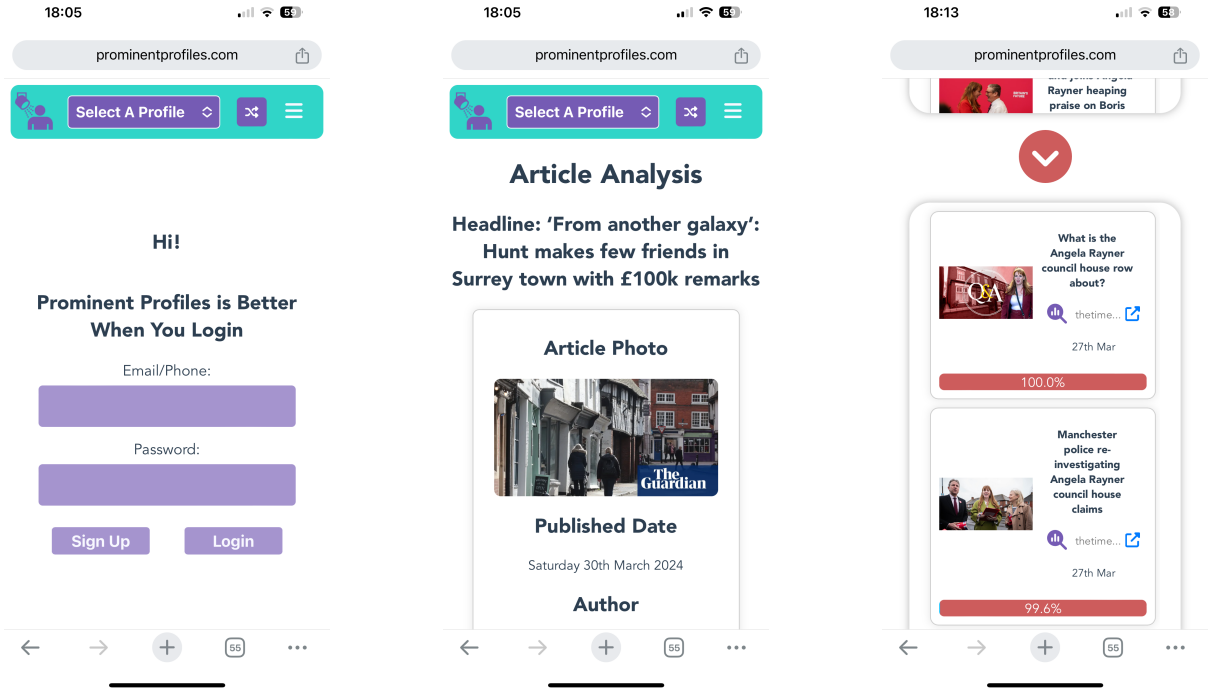


Figure 6.23: After Mobile Optimisations

Chapter 7

Testing

When developing Prominent Profiles, we were keen to identify any errors in our software. Testing is the ideal candidate for this. While some may consider testing to be the measure of success or reassurance; we took the view that for our tests to be successful, some test cases must fail so that we could improve the code and eventually get them to pass[91]. If all our tests passed at the outset, they were not sufficiently challenging or well-defined.

The developer initially relied upon manual, informal testing of the system's components but evolved to leverage automation for checks with fixed outcomes and at risk of being missed by human error due to their repetitive nature. Formalised (functional) testing was also introduced to provide an additional layer of robustness and to cover circumstances where automated testing would have been too complex to implement. As a bonus, the tests outlined below neatly serve as a documented set of cases describing how our APIs and UI should behave under various conditions, which is valuable for development and maintenance.

7.1 Automated Testing

Approach

Automated tests were adopted as they would serve as a shield between a buggy commit and the production environment, protecting end users from a loss of functionality or, in rare but notable cases, the entire app's availability; to do this, they were adopted into the CI/CD pipeline following the build phase and would prevent deployment running in the event of a single test failure.

The approach to automated testing was first to consider each Django app and then consider the models, and views that featured within them (each app has a test.py or testing directory), though sometimes models were imported across to support just as they are in the genuine environments.

For instance, model testing for `profiles_app` began with unit tests by checking individual model attributes, such as field validation and constraints (e.g. unique, not null). Then it expanded into integration tests by emulating the model relationships the pipeline relied on: creating a ProcessedFile record, then an Article 'obtained' from it with some Entities that 'appeared' in it and a 'similar' Article to pair it with. This way, if any changes to its model definitions were pipeline impacting, these tests should catch it.

Meanwhile, view testing asserted the expected HTTP status codes and the nature of the data structures returned. Models were declared in a setup phase to create mock data, so in the action and assertion phases, the APIs had actual data to retrieve, and our tests had expected content to check for. These tests ensure the views serve the correct information, enforce defined permissions and visibility constraints, that URLs were correctly mapped to views and that those views can handle standard and erroneous data gracefully.

However, the most crucial tests were for each NLP pipeline aspect. As research took place across several notebooks, they served as a starting point for writing tests, as (broadly) each cell could become a test

action, and its output would become the assertion. We expanded them with additional cases (e.g. boundary), paying particular attention to utility functions that were simple to write unit tests for but rewarding as they underpin functionality. Finally, we incorporated integration testing to ensure that the end-to-end functionality, from data input to processing, analysis, and storage, worked as intended.

```
Ran 114 tests in 9.126s

OK
Destroying test database for alias 'default'...
```

Figure 7.1: 114 Tests Live In Pipeline Stage

Errors Fixed Examples

1. Clean Up Substring: A bug in the clean_up_substrings method meant it only updated the first coref cluster rather than all provided. Its example purpose is: 'Sunak', 'Rishi Sunak' → 'Rishi Sunak' retained 'Sunak' removed, and since we have examples of unresolved entities in the production environment like 'Sunak' and 'Kate', there is a strong chance this bug caused them.

2. Overall Sentiment API: The API view always returned nothing when specified with a start and end day range to query the database. The developer had mixed up the greater than and less than parameters.

3. Percentage Contribution: This method lacked built-in divide-by-zero protection. It was protected from being called with the potential for this error. Still, if the method were repurposed, simulated by the unit test that caught this error, this protection would not exist, so we added it.

On reflection, we strongly consider using test-driven development from the start of the process in future work, especially if our development involves a team.

```
    # Identify longest names in cluster ID and remove shorter ones

    for entity in clustered_entities:
        cluster_id = entity['Cluster Info']['Cluster ID']

        entity_name = entity['Entity Name']
        current_longest = longest_names.get(cluster_id, "")


    if len(entity_name) > len(current_longest):
            # Remove the shorter entity without adding it to the list of entities to keep
            if current_longest:
                entities_to_keep = [e for e in clustered_entities if
                                    e['Cluster Info']['Cluster ID'] != cluster_id]
            longest_names[cluster_id] = entity_name

            # Add the entity to the list of entities to keep
            entities_to_keep.append(entity)
```

```
    # Group entities by their Cluster ID
    clusters = {}

    for entity in clustered_entities:
        cluster_id = entity['Cluster Info']['Cluster ID']

        if cluster_id not in clusters:
            clusters[cluster_id] = []
        clusters[cluster_id].append(entity)


    cleaned_entities = []
    for cluster_id, entities in clusters.items():
        # Find the entity with the longest name in each cluster
        longest_entity = max(entities, key=lambda e: len(e['Entity Name']))
        entities_to_keep.append(longest_entity)
```

```
-               article__publication_date__gte=start_date,
-               article__publication_date__lte=end_date
+               article__publication_date__lte=start_date,
+               article__publication_date__gte=end_date
```

```
-    total = sum(elements)
-    percentage_contributions = [(element / total) * 100 for element in elements]
+    try:
+        total = sum(elements)
+        percentage_contributions = [(element / total) * 100 for element in elements]
+    except ZeroDivisionError:
+        print('Division by zero error in percentage contribution!')
+        return [0, 0, 0]
```

Figure 7.2: Errors Fixed Examples

7.2 Functional Testing

It could be argued we employed functional testing implicitly throughout as each major feature addition was tested locally in the development environment before deployment to check its expected behaviour and typically again in production. However for the later sprints we formalised these checks as a potential user journey that touched all functional elements of our site.

We created 97 functional tests (12.8.2) focusing on the UI and the user journeys between different views. Additionally, to check the key admin activities, we established 18 further tests. Test failures helped us uncover issues including:

- **Vue Watchers:** The tests led to the incorporation of Vue watchers, which ensure that content reacts to a change in the article or entity referenced in the URL.
- **CSS Adaptations and Hamburger Menu:** CSS adjustments were made to improve UX, especially on mobile devices, and a hamburger menu was added. These changes aim to reduce the likelihood of content overflow on smaller screens.
- **Project Backlog Issues:** Some minor issues have been identified and added to the project backlog these include:
  - Preventing API calls when Vue form validation fails in less frequently used user journeys, such as the 'Forgot Password' process. We can fix these using the same protective measures implemented in the Register journey.
  - Improving error messaging for email validation in the 'Forgot Password' flow to be as informative as in the login and sign-up interfaces.
- **API Endpoint Accessibility for non-visible Entities:** If an admin sets an entity's visibility as false, its API endpoint remains accessible. Users who know the Entity View URL or are subscribed could still view these entities. A simple solution proposed is to apply an additional filter in the API views to address this issue.

The testing outcomes have also provided insights into the effectiveness of the test suite:

- The test pass rate for the Vue app views is now 97%.
- The test pass rate for admin functionality tests is now 88.9%.

These results demonstrate the thoroughness of our functional testing approach.

7.3 Load Testing

The non-functional requirements state that 'the system MUST support access by up to 100 concurrent users without significant degradation in response times'.

Locust is an open source load testing tool and is developer friendly by allowing tests to be defined directly in Python code[92].

We defined 11 different http requests encompassing a mix of Vue routes and API calls to cover the typical experience of users browsing the site and content loading.

Listing 7.1: Extract of Load Test Cases

```
entity_ids = [2, 4, 42, 482, 664, 169, etc...
entity_article_id_pairs = [(169, 22376), (1195, 2211), etc...
start_end_days = [(0, 14), (14, 180)]

    @task
    def overall_sentiments_api(self):
        """Simulates visiting Entity View"
        entity_id = random.choice(self.entity_ids)
        days = random.choice(self.start_end_days)
        self.client.get(f"/django/profiles_app/overall_sentiments/exp/"
                        f"{entity_id}/?endDay={days[1]}&startDay={days[0]}")

    @task
    def visit_article_page(self):
        article_id = random.choice(entity_article_id_pairs)
        self.client.get(f"/article/{article_id[0]}/{article_id[1]}")

    @task
    def visit_home_page(self):
```

```
20          self.client.get("/")
```

Notice the declared arrays of values. These made the tests more realistic by defining arrays of values for the API calls to randomly pick. Otherwise, we ran the risk of the database caching a common query and providing us with unrealistic results.

All tests added one user per second.

- Test 1: Ran for 125 seconds (25 seconds to ramp up) maxing out at 25 users.
- Test 2: Ran for 150 seconds (50 seconds to ramp up) maxing out at 50 users.
- Test 3: Ran for 200 seconds (100 seconds to ramp up) maxing out at 100 users.

Test Number	**OverallSentiment API max days setting**	
	180 day max (default)	**90 day max**
1	309.49 ms	136.7 ms
2	848.94 ms	304.6 ms
3	1878.94 ms	645.26 ms

Table 7.1: Initial Digital Ocean Plan 1 Avg Response Times

After investigating the issue, we recalled that the Digital Ocean database at $15 monthly has a 22 connection limit and only 1GB of RAM. The heavy OverallSentiment query at the longer days amount monopolised these 22 connections, revealing why quicker 90-day queries made improvements.

We experimented with increasing the tier to 47 connection limit ($30 per month) with 2GB of RAM, and it improved the situation.

Test Number	**180 day max (default)**
1	196.9 ms
2	413.92 ms
3	1381.34 ms

Table 7.2: Basic Digital Ocean Plan 2 Database Avg Response Times

Still unsatisfied, we found that the OverallSentiment API was significantly hindering performance. This query fetches all articles associated with an entity and assesses their similarity to other articles within a 14-day window for each API call. This similarity check, embedded in the view before the introduction of ArticleSimilarityPair creation and automatic rejection in our pipeline, was operating under a less efficient system where articles were stored without consideration of duplicity (6.5.2).

It's important to highlight that the most time-consuming requests, which contributed to the increased average response times, were attributed solely to the OverallSentiment view. Despite this backend processing, the impact on the end-user experience is mitigated; the API request for the default 0 to 14-day range is notably faster, allowing these operations to proceed in the background without delaying user interactions.

A different angle is focusing load testing on the 0 to 14-day range, demonstrating the time for Vue pages to load as they are currently filtered. This limitation depicted a best (albeit moderately unrealistic with 0 to 180 load still occurring in reality) case for user-facing performance in its designed state.

Test Number	**14 day max**
1	196.9 ms
2	156.11 ms
3	153.86 ms

Table 7.3: Digital Ocean Plan 1 with 14 day limit (User noticeable Range)

In response to our findings, we implemented a new one-off job, which marks the second Article in all SimilarArticlePairs with a similar_rejection flag set to `true`. This streamlines the logic within the OverallSentiment view as only a straightforward boolean condition is required to filter now. This approach

replaces the previous, resource-intensive method of continuously comparing similarity statistics for each API call—a process that became increasingly burdensome as the volume of articles in our database grew - a direct result of storing every Article retrieved from February 1st onward, in conjunction with an uptick in article URLs due to expanded Bing API search terms and pull frequency.

Caching has also been introduced for all `profiles_app` API views. This is via Redis since we already use it for Celery automation anyway. It reduces the need to query the database, preserving the 22 connections and providing faster responses.

Test Number	**180 day max**
1	43.65 ms
2	91.57 ms
3	124.7 ms

Table 7.4: Digital Ocean Plan 1 with simplified API and redis caching

We are delighted with the improved performance and can confidently say the MUST requirement has now been met more effectively than spending more on Digital Ocean.

A more demanding test of adding two users at a time over 300 seconds, maxing out at 500 users, achieved an average of 98.06 ms. In conclusion, load testing was essential for identifying this area for improvement, and we are very impressed with the difference it has made to the smoothness and responsiveness of our UI.

Chapter 8

Legal, Social, Ethical and Professional Issues

8.1 Web Scraping

Web scraping, the automatic retrieval of data from the internet, is a grey area regarding legality and ethics[93, 94].

Prominent Profiles (PP) follows best ethical practice by checking the robots.txt instructions for blank user agents ('*') to identify if a website can be scraped[95]. If a robots.txt can not be found on a website domain, we take the cautious approach and do not proceed to scrape despite some courts discussing that a lack of one may constitute a licence to use the data anyway[94]. Furthermore, there is no legal obligation to follow any of its instructions[94].

Listing 8.1: Robot.txt Check in PP

```
parsed_url = urlparse(url_to_check)
base_url = parsed_url.scheme + "://" + parsed_url.netloc

rules = RobotFileParser()
try:
    with urllib.request.urlopen(base_url + "/robots.txt", timeout=5) as
        response:
        rules.parse(response.read().decode('utf-8').splitlines())
    return rules.can_fetch("*", url_to_check)
except urllib.error.URLError as e:
    print(f"Error accessing robots.txt: {e}")
except socket.timeout as e:
    print(f"Timeout occurred: {e}")

# Default to False as if robot.txt can't be checked.
return False
```

We do not crawl (navigating to other links on each page); loading one page is much more friendly for a server load than recursively browsing deeper into its site map.

For practical reasons, we do not check the individual websites' Terms of Service (ToS) to see if scraping activity is prohibited because, ideally, they should maintain their robots.txt file appropriately. In the US, court cases have determined a user must make an affirmative confirmation, e.g. tick box, to become a 'user' of the website and thus be bound by ToS - an automated scraper can not do this[94]. Even so, the Ninth Circuit determined breaking the ToS alone as insufficient to become liable under the US Computer Fraud and Abuse Act[94]. Any claim of the 'breach of contract' nature would require proof of the incurring of material damages resulting from the scraping activity[94]. Ultimately, checking and accepting each ToS would severely reduce the diversity of news available on PP.

Regarding copyright, which articles have from inception, we are not financially gaining from their use, so this is viewed more favourably ('fair use'). Plus, copyright law covers the specific form of expressing ideas, while PP creates a (unique) summary of the data extracted[94].

We attempted to find UK-specific papers on scraping legalities but only found other exciting applications, such as real-time web scraping house prices[96]. Like many other academics, they don't address the legalities: in the future, we will likely see greater regulation in this area, as with AI. Finally, no researchers scraping public website data for personal or academic use appear to have caused lawsuits in major Western democratic countries[95].

Our perspective is we are operating a platform similar to aggregators like Google News or Ground News (2.2). Those platforms must read the text to rank the articles or provide left and right-wing summaries while we obtain the profiles and their respective TSC. Moreover, we also display headlines and the main photo in the UI. Then, as no full texts are retained, the user must navigate to the publisher to get the news story, driving traffic to their site, which is likely driven by ad revenues or subscription models, so this is in their best interest.

8.2 Sentiment Analysis

Even though we find PPs TSC core achieves an accuracy of a very respectable (for sentiment analysis) 67% and above on challenging datasets, we must stress that this means 67%+ agreement with the human judgement of a sentiment class and not a 67% accurate identification of it[97]. The difficulty posed by irony, sarcasm, and humour to models can contribute to the fact that it is often challenging for humans to agree on the sentiment of a text[97].

Decisions that impact sentiment determination have been taken in PP's pipeline. For instance, quotations that typically express sentiment more explicitly than the rest of an article's body are included, and these can have a strong contribution to the overall sentiment score. However, we would highlight that the writer chose to include these quotations, and thus, they should determine the sentiment we associate with their coverage.

Due to the possibility of model inaccuracy and our adopted logic, there will be occasions where the sentiment analysis by popular opinion is 'wrong'. Although users of PP should go beyond the headline, we have seen instances where the scoring appears contradictory. Yet, upon reading the body, we find it doesn't align with the expectations the headline sets. In theory, there is a chance we misclassify many articles for a profile and make them appear more negative than they otherwise are portrayed in 'reality'.

To address potential legal implications, e.g. 'Deformation of Character', and fairly inform our users, we have created a disclaimer using Termly, a suite of compliance solutions, shown on our About View and linked in our cookie acceptance banner[98]. There are also ambitions for a reporting inaccuracy feature to address users' concerns quickly and build a dataset for improvements. We are happy to share this paper, and the code is publicly accessible via GitHub for social transparency in the processes used.

Finally, one of the features requested by users is the ability to track the sentiment toward specific profiles over time, including an analysis segmented by individual publishers. We approach this request with caution due to system limitations. Consider a hypothetical scenario where 'Meekly Weekly' publishes eight articles about 'Rishi Sunak', with three negative and five positive. If, due to the limitations of the Bing API, our system only captures the three negative and one positive article at the times of query, the sentiment analysis represented on our platform would not accurately reflect the full spectrum of Meekly Weekly's coverage. This discrepancy would be compounded if extended across multiple publishers. On a trial basis, we could explore the adoption of more reliable RSS feeds or APIs with individual publishers, aiming to ensure complete coverage and a more truthful sentiment portrayal.

8.3 Legal Frameworks

The EU's GDPR rules, which apply to any website regardless of where they are based and the UK's PECR, have been considered in developing PP.

8.3.1 General Data Protection Regulation

Children's Data

GDPR Article 8: Conditions applicable to child's consent in relation to information society services states:

> Where point (a) of Article 6(1) applies, in relation to the offer of information society services directly to a child, the processing of the personal data of a child shall be lawful where the child is at least 16 years old. Where the child is below the age of 16 years, such processing shall be lawful only if and to the extent that consent is given or authorised by the holder of parental responsibility over the child. Member States may provide by law for a lower age for those purposes provided that such lower age is not below 13 years[99].

The UK has chosen to set a lower age limit (13) under the Data Protection Act 2018. However, as PP is available internationally (through a .com domain) and anticipates users from various countries, setting the consent age to 16 is a proactive approach that aligns with the default requirement of GDPR. This approach minimises legal risks since it adheres to the stricter standard. Additionally, most features are accessible without creating an account, so the impact of setting a higher age limit for consent is not particularly damaging for our younger users experience. Although, a parental consent journey could be introduced to allow under 16s to create an account like Google's Family Link.

Privacy Policy

Articles 12 - 14 of GDPR set a requirement for PP to include a privacy policy, which is a document that explains how it will process personal data and how it applies data protection principles. We used Termly to provide a comprehensive policy by answering a questionnaire. It is accessible via the About View and the cookie acceptance banner.

8.3.2 Privacy and Electronic Communications Regulations

Cookies

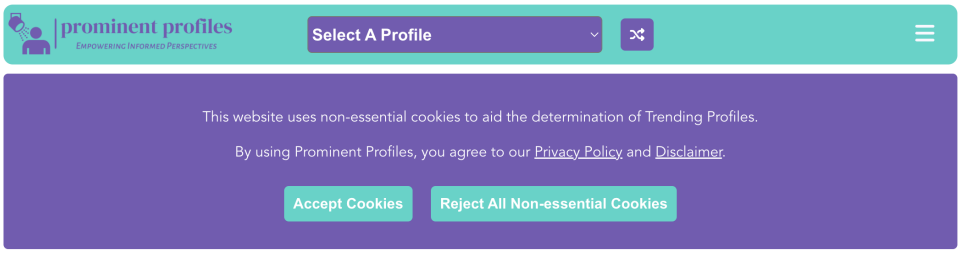


Figure 8.1: Cookie Acceptance Banner

Article 4 of PECR means that when PP uses cookies, it must say what cookies will be set, explain what they will do, and obtain content to do this unless they are essential[100].

Our essential cookies are the CSRF token, access token, refresh token, cookie acceptance status, and session ID. The optional non-essential cookie, viewedProfiles, controls whether a user's profile pages trigger the creation of an EntityView object, which determines the Home View's Trending Profiles feature. To satisfy PECR, a cookie banner is shown below the header for users who have not previously accepted cookies. It also provides links to our Privacy Policy and Disclaimer.

Chapter 9

Evaluation

9.1 NewsSentiment

Due to its use at the project's core, we evaluated NewsSentiment internally to comment on its suitability.

The two most significant labelled TSC collections are:

1. **NewsMTSC** contains 11,361 sentences with 4,227 negative, 4,231 neutral, and 2,903 positive[8].
2. **MAD-TSC** contains 5,110 sentences with 1,839 negative, 2,011 neutral, and 1,260 positive[1].

Both have employed techniques only to retain those with sufficient agreement between annotators, and MAD-TSC is more complex than NewsMTSC (2.1.1).

Two notebooks parsed the JSON datasets into the required left, mention, and right segments for NewsSentiment using the provided character positions. We could then compare the labelled polarity and determined sentiment class to produce performance metrics.


```
{
    "primary_gid": "3685_33_0_6",
    "sentence_normalized": "Merkel knows this, and she could get on well with Martin
         Schulz as the President of the Commission - not least because the SPD
        politician enjoys the confidence of French President Hollande, which could
        help get the stuttering German-French motor back up and running.",
    "targets": [
        {
            "Input.gid": "3685_33_0_6",
            "from": 0,
            "to": 6,
            "mention": "Merkel",
            // [2, 4, 6] = [negative, neutral , positive]
            "polarity": 6.0,
            "result": {
                "sentiment": {
                    "class_id": 2,
                    "class_label": "positive",
                    "class_prob": 0.8332267999649048
                },
                "left_segment": "",
                "mention_segment": "Merkel",
                "right_segment": " knows this, and she could get on well with Martin
                     Schulz as the President of the Commission - not least because
                    the SPD politician enjoys the confidence of French President
                    Hollande, which could help get the stuttering German-French
                    motor back up and running."
            }
        }
    ]
}
```


Listing 9.1: MAD-TSC JSON with NewsSentiment Result Appended

Dataset (Source)	Polarity Counts[1]	Accuracy	Precision	Recall	F1-Score	Confusion Matrix
Train Output (NewsMTSC)	[3316, 3028, 2395]	0.9042	0.9050	0.9042	0.9044	$\begin{bmatrix} 3035 & 180 & 101 \\ 129 & 2743 & 156 \\ 54 & 217 & 2124 \end{bmatrix}$
DevtestRW Output (NewsMTSC)	[429, 455, 262]	0.8403	0.8417	0.8403	0.8396	$\begin{bmatrix} 368 & 41 & 20 \\ 21 & 406 & 28 \\ 19 & 54 & 189 \end{bmatrix}$
Real World sentences[2]						
DevtestMT Output (NewsMTSC)	[482, 748, 246]	0.8470	0.8479	0.8470	0.8454	$\begin{bmatrix} 410 & 67 & 5 \\ 56 & 667 & 21 \\ 18 & 58 & 169 \end{bmatrix}$
Multi-Target mentions sentences						
Train Output (MADTSC)	[1363, 1493, 954]	0.6638	0.6759	0.6638	0.6651	$\begin{bmatrix} 966 & 235 & 162 \\ 219 & 872 & 402 \\ 94 & 169 & 691 \end{bmatrix}$
Test Output (MADTSC)	[364, 401, 235]	0.667	0.6877	0.667	0.6690	$\begin{bmatrix} 264 & 51 & 49 \\ 61 & 228 & 112 \\ 22 & 38 & 175 \end{bmatrix}$
Validation Output (MADTSC)	[112, 117, 71]	0.6667	0.6811	0.6667	0.6666	$\begin{bmatrix} 86 & 15 & 11 \\ 19 & 64 & 34 \\ 10 & 11 & 50 \end{bmatrix}$

Table 9.1: Evaluation Metrics for News Sentiment Classification

Predictably, the training set from the creators of NewsSentiment performs very well, with an F1 of 0.90. There is only a slight drop in performance on their sets containing more realistic and challenging examples, including those with multiple target mentions to an excellent 0.84. However, all performance metrics fall when considering MAD-TSC, achieving a (reasonable) 0.67 across all datasets, highlighting its increased complexity. Most NewsMTSC values across the confusion matrix diagonal are high, while MAD-TSC has more significant confusion. Generally, misclassification tends to occur more between neutral and negative classes, indicating that the model struggles most to differentiate between these two sentiments.

In this evaluation, we investigated improvement through thresholding. Taking both datasets into account (Tables 9.2 - 9.3) thresholding at 0.7 or 0.6 could be an appropriate trade off between retaining sentences and performance as this increases the F1s of NewsMTSC and MAD-TSC by approx 0.04 while retaining 67% to 80% of candidate sentences after model application. Due to the minimum mentions rule (20% of article sentences), the greater the threshold the more articles will be eliminated from consideration.

[1] [Negative, Neutral, Positive]

[2] NewsMTSC say the 'set was created with the objective to resemble real-world distribution as to sentiment and other factors mentioned in the paper.'

Threshold	News MTSC Train		Devtest RW		Devtest MT	
Value	F1-Score	Total Results	F1-Score	Total Results	F1-Score	Total Results
0.1	0.90	8739	0.84	1146	0.85	1471
0.2	0.90	8739	0.84	1146	0.85	1471
0.3	0.90	8739	0.84	1146	0.85	1471
0.4	0.91	8723	0.84	1141	0.85	1464
0.5	0.92	8482	0.85	1110	0.86	1411
0.6	0.94	7942	0.88	1014	0.89	1278
0.7	0.96	7251	0.90	931	0.92	1154
0.8	0.98	6307	0.93	818	0.93	1011
0.9	0.99	4565	0.97	594	0.96	729

Table 9.2: F1-Scores and Total Results by Threshold for News MTSC Datasets

Threshold	MAD_TSC Train		MAD_TSC Test		MAD_TSC Validation	
Value	F1-Score	Total Results	F1-Score	Total Results	F1-Score	Total Results
0.1	0.67	3810	0.67	1000	0.67	300
0.2	0.67	3810	0.67	1000	0.67	300
0.3	0.67	3810	0.67	1000	0.67	300
0.4	0.67	3797	0.67	997	0.67	300
0.5	0.68	3606	0.68	934	0.69	283
0.6	0.72	3143	0.71	800	0.70	256
0.7	0.75	2632	0.75	667	0.76	214
0.8	0.80	2050	0.81	510	0.80	177
0.9	0.85	1220	0.89	314	0.86	113

Table 9.3: F1-Scores and Total Results by Threshold for MAD_TSC Datasets

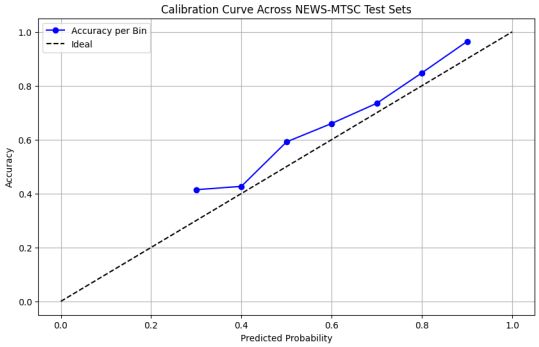


Figure 9.1: Calibration Test using News-MTSC

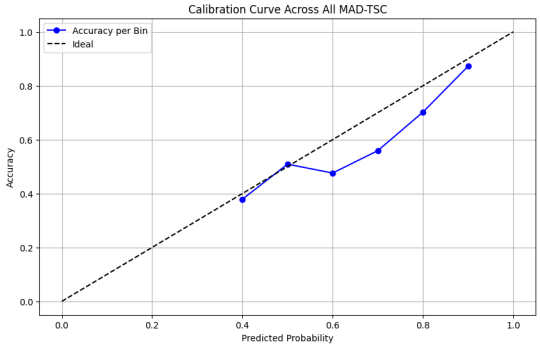


Figure 9.2: Calibration Test using MAD-TSC

Our scoring approach, detailed in 6.1.9, incorporates the probability of each sentiment classification, assigning weighted significance to sentences based on this metric. The use of exponential scaling further amplifies the impact of these probabilities so that higher-confidence predictions disproportionately influence the overall sentiment assessment. Since we rely heavily upon confidence in our overall score determination, we created calibration curves that demonstrate the agreement between predicted probabilities and observed frequencies. We combine the sets of News-MTSC (excluding training) and MAD-TSC to produce two plots (Fig. 9.1 - 9.2) that show little deviation from the perfect diagonal calibration, indicating the model is well calibrated. Hence, it is suitable to rely on its confidence metric in our pipeline.

In retrospect, there's a chance that only low confidence scores contribute to the 100% sentiment bar presented to users. Yet, our threshold tables indicate that instances with confidence scores below 0.5 occur in less than 10% of cases across all datasets; therefore, it's very improbable that an overall sentiment will be determined solely by low confidence results. Even so, a refinement involving tracking the average confidence across an article's BoundMentions and deleting articles that don't meet it could be added in the future.

To the best of our knowledge, there is no labelled TSC dataset of entire articles (rather than sentence samples) that we could use to test the end-to-end NLP pipeline.

9.2 Article Duplication Detection

To investigate the performance of duplicate article detection an extract of SimilarArticlePairs meeting the filter requirements was downloaded from the server.

The spreadsheet reveals that the system has rejected **2635** articles based on their fuzzy hash comparison outcome > 90 and **77** via the 'backup' linguistic method[1].

[1] Of Articles added to PP between 25th December 2023 and 30th March 2024.

During observations, we observed that the system marks Financial Times records as duplicates despite their headlines being entirely different. This is caused by Trafilatura extracting the paywall, which remains broadly similar between articles. None of these articles have an overall sentiment object associated with them, so it does not affect our end-user data. Nevertheless, we should set up an exclusion rule for FT and increase the minimum characters retention threshold beyond 250.

Another situation where headlines differ substantially is live feed style news streams based on hash scores; our analysis suggests that the content was similar at the extraction time. However, there's a challenge: if a user accesses the live feed later, the content may have changed from what was initially analysed. This is because the content on these pages will likely change throughout the day, and our system does not perform reanalysis. We've detected 271 instances of such similarity in live feeds. Therefore, an exclusion for live feeds in general could be helpful.

With FT and live feed styles excluded:

- For 158 duplicates via hash comparisons in manual verification, we find 100% success[2] including five further paywall instances concerning the Australian Daily Telegraph.
- For 20 duplicates via linguistic statistics, we find 100% success[2] and notable cases with low hash outcomes. Upon investigation, this is because the order of the text has been rearranged, quotes slightly changed, or abbreviations like 'DUP' expanded to 'Democratic Unionist Party', which contribute to avoiding capture by fuzzy hashing alone.

We acknowledge that this work doesn't address false negatives; they are deemed less concerning than false positives. False negatives might lead to the creation of duplicate ArticleCards, which, while not ideal, is a relatively minor issue. In contrast, false positives could result in missed opportunities for adding relevant coverage to a profile, which is a more significant problem. Additionally, perfect determination is not required. Our users have indicated that social media feeds often contain repetitive content; the occasional missed duplicate is not detrimental. In some circumstances, not identifying a duplicate is beneficial, particularly when the cost of extending the other ArticleStatistics search beyond four days outweighs the benefits of analysing it (See 6.5.2).

[2]Success within the articles flagged as duplicates. Not to be confused with the undetermined overall detection of duplicates.

Art. 1 ID	Art. 2 ID	Art. 2 Rej.	Art. 1 Headline	Art. 2 Headline	Art. 1 Site	Art. 2 Site	Hash Sim. Score	Marking	Live Latest Feed
27311	27319	TRUE	Sunak did not want to talk about reducing legal migration, says Jenrick	Sunak did not want to talk about reducing legal migration, says Jenrick	Evening Standard	Yahoo News	96	PP_DEEP	FALSE
27292	27297	TRUE	Waterborne diseases have soared under Tories, Labour says amid sewage anger	Waterborne diseases have soared under Tories, Labour says amid sewage anger	Yahoo News	The Independent	99	PP_DEEP	FALSE
27291	27296	TRUE	The quiet 'Blair seats' that line Keir Starmer's path to a majority	The quiet 'Blair seats' that hold the key to a Labour majority	The Times	The Times	100	PP_DEEP	FALSE
27280	27282	TRUE	The pros and cons of Hereford's green-lighted bypass plan	What are the pros and cons of Hereford's green-lighted bypass plan?	Hereford Times	Yahoo News	90	PP_DEEP	FALSE

Table 9.4: Extract of Duplicates Spreadsheet

Art. 1 ID	Art. 2 ID	Art. 2 Rej.	Art. 1 Headline	Art. 2 Headline	Art. 1 Site	Art. 2 Site	Art. 1 Sent. Count	Hash Sim. Score	Marking	Live Latest Feed	Comment
20481	21004	TRUE	Northern Ireland 'open for business', Stormont leaders tell US investors at gala	Northern Ireland 'open for business', Michelle O'Neill and Emma Little Pengelly tell US investors at major Washington event	Inverness Courier	The Irish News	1	77	LINGUISTIC	FALSE	Slightly reworded e.g. 'DUP' changed to 'Democratic Unionist Party'
22466	26204	TRUE	Sunak maintains hope of spring Rwanda flights after seeing off Lords' challenges	Government on course to overturn 10 Lords' amendments to Rwanda Bill	Kent Online	Salisbury Journal	1	69	LINGUISTIC	FALSE	Low hash yet content is very similar
7525	10806	TRUE	Biden hosts Jordan's King Abdullah and UK holds by-elections	What comes next for embattled Commons Speaker Lindsay Hoyle?	Financial Times	Financial Times	0	93	PP_DEEP	FALSE	Paywall
7525	9313	TRUE	Biden hosts Jordan's King Abdullah and UK holds by-elections	English private schools focus on recruiting overseas as Labour VAT hit looms	Financial Times	Financial Times	0	93	PP_DEEP	FALSE	Paywall
10094	10320	TRUE	Politics latest: Motion of no confidence in Speaker after Gaza debate descends into chaos	Politics latest: Big day for Keir Starmer as MPs set to vote on Gaza ceasefire	Sky News	Sky News	1	100	PP_DEEP	TRUE	Live feed with changing news clips but trafilatura may extract same content. Hassle for users seeing feed many weeks later

Table 9.5: Interesting Examples from Duplicates Spreadsheet

9.3 Acceptance Testing

To establish if Prominent Profiles meets the defined requirements and expectations we performed acceptance testing. The system is evaluated against each of the MoSCoW specifications and marked 'pass', 'fail' or 'partial'. In the case of failure or partial achievement we provide commentary.

Functional

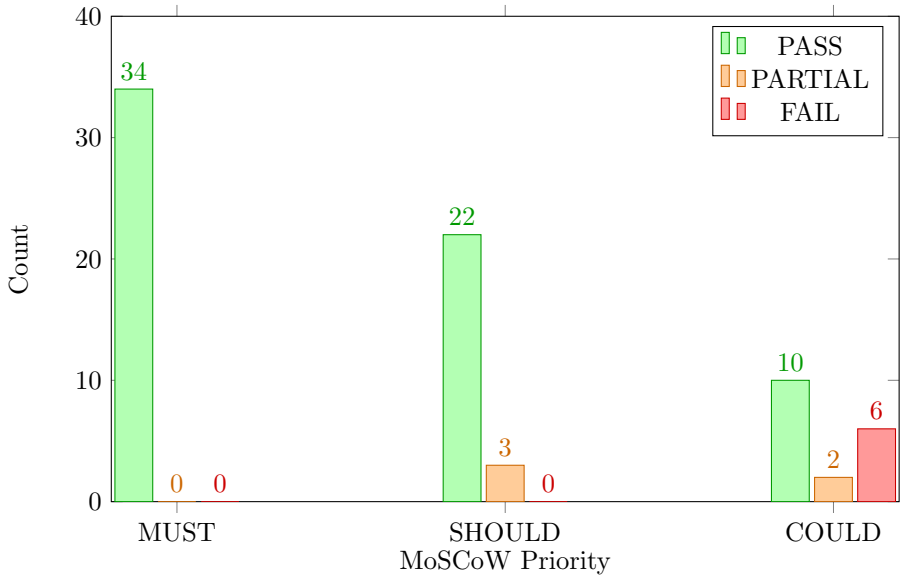


Figure 9.3: Functional Requirements Acceptance Testing

Out of the 77 functional requirements defined, 66 have been fulfilled. Efforts have resulted in progress towards completion for 5 requirements, while 6 are yet to be met; but, these reside within the 'COULD' category, so they are less critical. All 'MUST' requirements are satisfied so we can consider Prominent Profiles a success in delivering its core functionality. Additionally, many important supplementary features have been achieved, highlighting our effective project management.

Ref.	Requirement	FAIL	PARTIAL	Comment
1.19	The system SHOULD exclude news articles from sources known to be incompatible with text extraction libraries (e.g., msn.com) to increase efficiency and reduce processing of unextractable content.		X	We exclude MSN, but FT and Aus Daily Telegraph from duplication evaluation seem to be candidates, too.
1.25	The system COULD apply custom tokenization rules to improve the accuracy of sentence boundary detection, facilitating more reliable sentiment analysis.		X	There is only one custom rule at present.
1.27	The system COULD use entities marked as resolved in the entity NER to coref consolidation phase to update names with 'labelled' data rather than relying solely on spaCy and substrings of the coref clusters.	X		This is something we would like to do in the short term. Although if a new entity not seen before (e.g., George Galloway appearing into the news suddenly) comes along it must be given a fair chance to establish itself as a new entity.

Ref.	Requirement	FAIL	PARTIAL	Comment
2.7	The system SHOULD implement mechanisms to update entity information periodically to ensure data relevance and accuracy.		X	This process is automated for new entities set to visible. An admin can force a reset by deleting all BingEntity records via the admin portal. Then the daily Bing entity app visible job will repopulate with latest info.
2.10	The system SHOULD implement custom filters in the admin interface to segment data for more efficient review and management, such as filtering articles or entities based on sentiment scores or the presence of bound mentions.		X	There is custom filtering for likely duplicates in SimilarArticlePair but there could be more widespread adoption across other models.
2.14	The system COULD implement asynchronous data loading or pagination in admin views with many inline instances (e.g., bound mentions) to improve page responsiveness.	X		The admin views can be a little slow due to the advantageous use of inline mentions - especially so for entities with many OverallSentiment relations.
2.16	The system COULD provide additional reporting tools, widgets or dashboards in the admin interface for enhanced data visualization and analysis.	X		There are no analytics dashboards for purposes like tracking traffic or analysing the flow of articles by day - low priority.
2.17	The system COULD integrate with third-party services or APIs to enrich entity information, such as social media profiles or external databases.		X	We integrate with Bing API but could go further such as linking the Profiles X account so a user can go and read their tweets.
3.17	The system COULD allow password changes after login as an alternative to following the reset path.	X		Password change is via reset pathway only.
3.20	The system COULD allow users to favourite articles for later reference.	X		Not prioritised as a user can simply bookmark the webpage as Article ID / Entity ID is in the URL itself. However, if we created a native iOS/Android app, this would be beneficial. Instagram, TikTok, etc., offer liked and favourites categories, for instance.
4.12	The system COULD provide social media share buttons/icons to share article analyses they wanted and help drive traffic to the app.	X		Many mobile browsers feature a share button in the browser itself, but this is a nice future feature, especially for desktop users.

Table 9.6: Partial/Fail Functional Requirements

Non-Functional

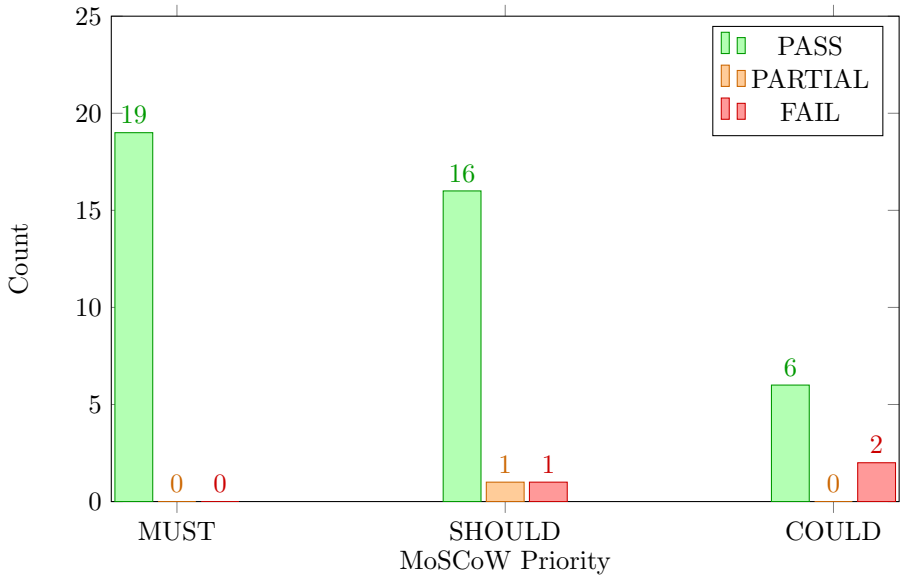


Figure 9.4: Non-Functional Requirements Acceptance Testing

4 out of our 45 non-functional requirements were not accomplished. The most important area to improve is accessibility, and we have already started making changes, including allowing keyboard navigation beyond the headers and into the components themselves. Other remaining NFRs would be desirable if the platform grew. Overall, though, the attainment of 41 non-functional requirements, especially all those in the MUST category, is an excellent outcome for the project.

Ref.	Non Functional Requirement	FAIL	PARTIAL	Comment
2.8	The system SHOULD be accessible to users with disabilities, following WCAG for accessibility.		X	We have made changes (April) to improve keyboard accessibility with tab and enter for most items. Still, we could go further, such as providing more labeling so screen readers can provide an overview of features. WCAG 2.1 should guide future work.
3.6	The system SHOULD implement rate limiting on API endpoints to prevent abuse and ensure service availability.	X		For the amount of traffic we are receiving currently, this is not a concern, but if the project gains more attention, put protections in place.
3.8	The system COULD have 2-factor authentication using a user's email or mobile phone.	X		We have collected user emails and UK phone numbers with validation in preparation for implementing such a system. While Prominent Profiles doesn't hold sensitive info like payment cards, we note that users concerned about security are beginning to expect 2FA.

Ref.	Non Functional Requirement	FAIL	PARTIAL	Comment
5.8	The system COULD offer training materials or sessions for administrators to ensure they are well-equipped to manage the system effectively.	X		As we currently have a sole administrator, this is not needed. This report serves as an overview, though, and we describe the test cases in code comments and the appendix.

Table 9.7: Partial/Fail Non-Functional Requirements

9.4 User Study

9.4.1 Questionare Design

On a near-final iteration, users were invited to participate in a study to evaluate Prominent Profile's (PP) effectiveness at meeting its aims and identify areas for improvement. The questionnaire began by prompting users to visit the live app and watch a short introductory video. We then asked 23 questions, including Likert scale, binary, multiple-choice, open-ended, and demographic questions, to cover user satisfaction, usability, feature effectiveness, demographic information, and open feedback for improvements. This variety was designed to gather easy-to-analyse quantitative and qualitative data for greater depth and context. Anonymity was emphasised to seek more honest feedback.

9.4.2 Analysis

15 responses were collected over a 4 day period[1].

Quality Assurance

To ensure data quality, we checked each response and removed one due to contradictory answers such as providing a 1 for overall experience but a 5 for 'recommend to others', giving false answers such as 'Yes' to registering when internally we could see they had not made an account, and lazy responses don't know; it was completed without care, and (we suspect) a genuine app visit.

Usability and Interface Design

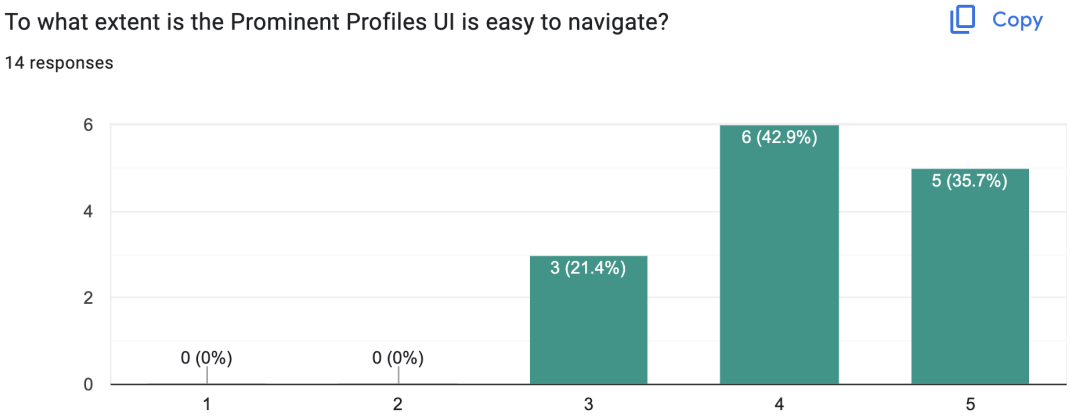


Figure 9.5: UI Ease of Use

Most users find the Prominent Profiles UI easy to navigate, with 11/14 rating it 4+/5 and stating that it is easy to understand and straight to the point. Those who responded with a score of 3 would like to see greater clarity around the purpose of each icon via text or a guided tutorial. Some users didn't feel it was clear that filters were applied by default, so some profiles didn't show articles until reset was

[1]Full Results in Appendix: 12.9.

clicked. In response, we added a message when no articles are present in a category. While many users find the app clear, clean and minimalist, one wants to see more professional graphic design, and another stated it should be possible to swipe on Trending Profiles animated headlines. We tend to agree with this, and think a native iOS/Android app could enable an even better UI/UX for mobile users.

Features and Functionality

All but one respondents were satisfied with the variety and coverage of news topics, and 11/14 rated the sentiment analysis as 4+/5 on a scale from not at all to very accurate. We asked users to pick their most favourite features from a tick box selection and then their least favourite features from an identical list, so we now subtract the least desirable from the most desirable to provide an overview.

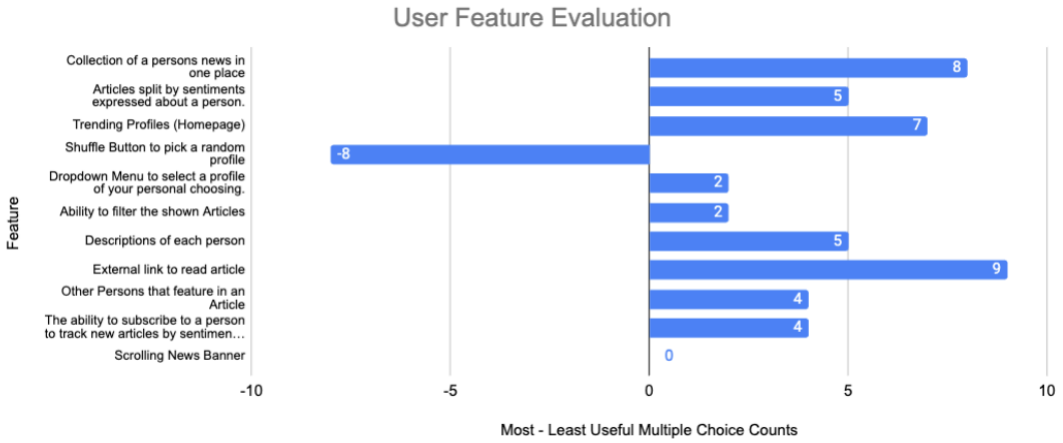


Figure 9.6: Feature Enthusiasm

There appears to be a significant lack of enthusiasm for the shuffle button. We originally intended it as a discovery feature, but its removal will now be considered, along with the scrolling news banner, which is not clickable and takes patience to read.

The drop-down menu didn't receive much praise either, indicating that more major features overshadowed it or that a better approach is needed; one user suggests a searchable menu of options, especially as the number of profiles grows. Filtering enthusiasm is low, suggesting our approach of setting default filters (max sentiment, last 14 days) is appropriate, given that only advanced users seem to engage with the toggles.

Users are very enthusiastic about having news organised in a person-by-person format, where they can easily use an external link to read it and find Trending Profiles suitable homepage content. Findings also suggest the split by sentiments expressed and descriptions from Wikipedia compliment the format well. Only 2 users made an account, which may explain why enthusiasm for subscriptions is more middling. The same can be said for the other profiles present, which relies on users discovering an article with them in their brief interactions.

Content Quality and Bias Awareness

No respondent identified another app that delivers news similarly, and many users commented on our app's uniqueness as a reason for liking it. Users enjoy that it can also make people aware of biases in the content they are reading. Although some would like to know more about the NLP processes to trust scores, one suspects, our developer could have built-in politically motivated bias (we haven't!). Regarding legitimacy, our inclusion of cookie consent, privacy policy, HTTPs, and about view make (13/14) users trust PP more, highlighting the importance of achieving these regulatory expectations. Finally, one respondent liked our goals but didn't trust the media. Instead, they preferred analysing an MP's voting record, so they felt our execution was poor. We would respond by saying tools analysing voting records already exist, e.g. theyworkforyou.com.

Users Overall Opinions and Aims Evaluation

We translated our defined aims (1) into 4 statements and asked users to express their agreement with them. In Fig. 9.7 we find our respondents overwhelmingly feel the project has achieved the outcomes we strived for.

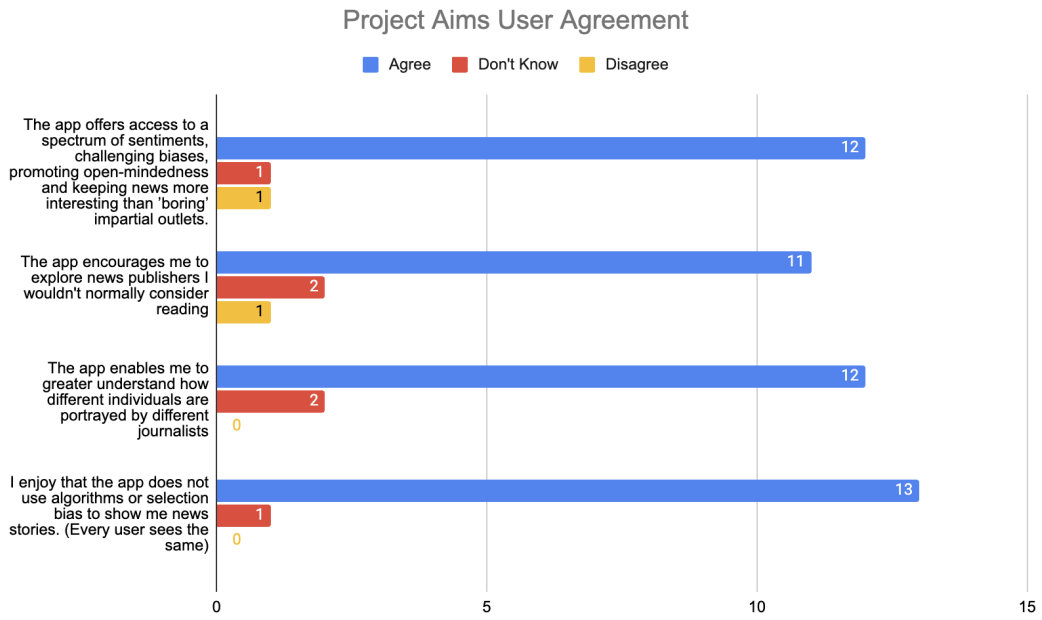
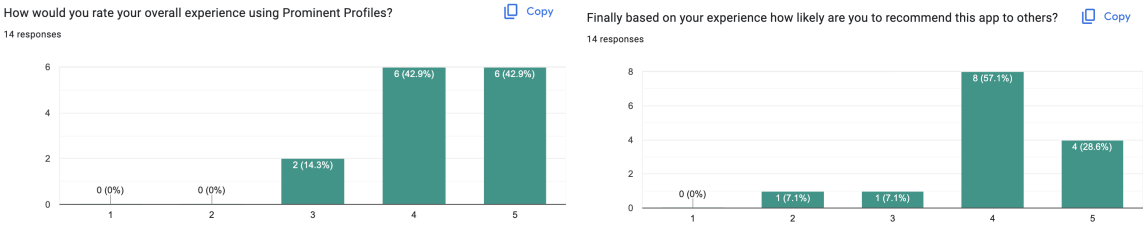


Figure 9.7: Project Aims User Agreement

To further test this, we asked users a hypothetical question about whether they would visit PP if they saw many negative news stories about an individual: 61% said yes, while the rest specified maybe. The lack of 'no' responses is excellent.

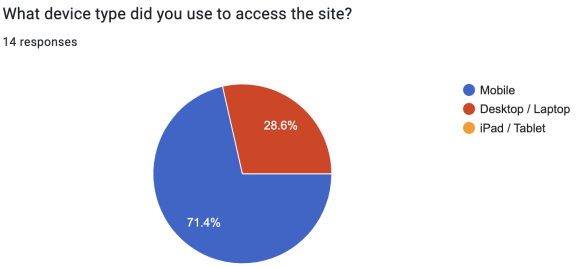


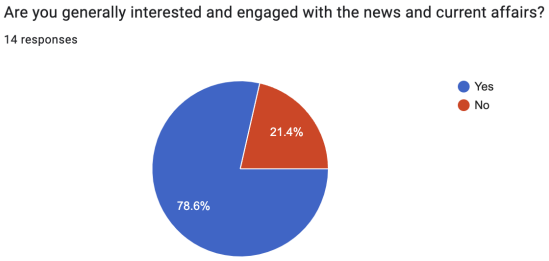
The success of our web app's UI, content variety and quality, along with its unique goals and presentation, translate well into users' overall rating of PP. Moreover, they strongly recommend it to others, and diving into those with less enthusiasm raises that their friends aren't interested in politics or that the purpose could be 'niche'. We accept these findings but believe enabling profiles like 'Taylor Swift', 'Kate Middleton' etc. could open up the appeal. Indeed, sites like the Daily Mail let users customise their content between news and showbiz.

User Devices and Demographics

Approximately 70% of users accessed the app via a mobile device, highlighting the value of our mobile optimisations. We think the UX is better on desktop, as it is easier to see the contrast in perspectives, so it's encouraging that users still found the app successful via mobile.

Notably, only one respondent was over 25, so we appreciate comments around the UI may not be shared by those much older and less technically able. Also, the younger generation may be more willing to embrace and trust AI/NLP than older generations. Additionally, $\frac{3}{4}$ of respondents considered themselves generally engaged with current affairs, so it must be considered they may be more enthusiastic about tools like PP. Consequently, these demographics motivate a broader study capturing the missing groups.





Future Features

Most respondents from the pre-defined improvements list would like the ability to favourite an article. We could highlight better to users that bookmarking the article since the analysed data for an article is static (i.e there is no meaningful update PP can provide like with subscriptions) will serve the same purpose. But, we can certainly understand how in a native app environment a likes and favourites feature like TikTok could be beneficial so users can easily find analysis again.

A few respondents desired login via an existing social media account presumably for the convenience and password reduction, so we could aim to introduce this feature. Equal numbers of respondents express interest in the provision of sentiment by publisher over time but we hold ethical concerns, discussed earlier in (8.2). Finally, a long-answer response requested a browser plugin to perform real time analysis across the internet. We like this concept but think a more realistic technical implementation would be a plugin that sends the URL to the server for inclusion in the next article obtain run and then informs the user if analysis was successful with an email.

9.4.3 Overview

Our user study demonstrates that Prominent Profiles meets its intended goals, with a solid positive reception to its UI, content curation, and unique features, as reflected in users' willingness to recommend the app to others. This positive feedback, coupled with constructive suggestions for future enhancements, underscores the app's potential to broaden its appeal and further refine its user experience.

Chapter 10

Conclusion

Prominent Profiles pursued the delivery of a novel news aggregation platform, enabling users to explore news stories via key individuals, particularly politicians, rather than traditional means. By utilising state-of-the-art target-dependent sentiment classification (TSC) models and a suite of Natural Language Processing (NLP) techniques, it has analysed over 25,000 articles to provide targeted insights into how individuals are portrayed across various media outlets.

Our work addresses the challenges posed by existing news discovery and consumption trends, often subject to social media algorithms and editorial biases. By presenting articles segmented by sentiment, Prominent Profiles empowers users to access a spectrum of viewpoints, promoting critical thinking and open-mindedness. This approach provides users with a tool to more easily navigate the complexity of media representations with sentiment scores, enabling transparent, informed consumption and encouraging awareness of bias in media representations.

Internally, to provision Prominent Profiles, our development had to identify suitable methods and packages to obtain articles, detect entities, tokenize sentences, resolve coreferences, identify mentions for analysis, and overcome challenges in entity resolution, article duplication, and job automation. Defining a logical relational database structure to support operations and delivering a Vue frontend with a clean design and navigation that encourages exploration and offers a comprehensive collection of profiles and articles was also essential. The benefits of the agile methodology, complemented by creating a CI/CD pipeline, have been leveraged to achieve all of this.

Deploying a holistic testing strategy involving automated, functional, load and acceptance types during the project has enabled the creation of a system that reliably meets its crucial and most advantageous optional system requirements.

The evaluation of the TSC core on challenging datasets, article duplication detection and a user study has illustrated the platform's effectiveness in meeting its aims. In particular, users have praised our app for being unique, providing a to-the-point interface, collating a person's linked articles together, offering segmentation by sentiment and addressing concerns about algorithms and one-sided perspectives.

Finally, all aspects of the evaluation give rise to the potential for future work both within the frontend to bring new features that enhance user engagement by increasing functionality and in the backend by motivating the creation of a TSC article dataset, a more extensive evaluation of article deduplication and efforts to deliver fair representations of publishers sentiments by profile over time.

Chapter 11

Further Work

Greater Input Article Variety

Bing API was chosen for various reasons, including its ability to gather a spread of relevant news quickly and on demand (3.1.1). However, the **extensive** use of RSS feeds and specific API agreements with outlets could enhance the availability of news and create the opportunity to fairly evaluate sentiments towards a profile by the publisher over time. API agreements are preferable because they avoid the legal grey areas surrounding scraping (8.1). Moreover, a new browser extension could allow users to submit article URLs as candidates for analysis.

Labelled TSC Article Dataset Creation

To enable evaluation of Prominent Profiles' end-to-end performance, work to create a UK-focused collection of articles with annotations and collective agreement would be helpful. Rather than a sentence-level dataset, this would produce an output with an article, linked entities, and their sentences with annotations of representations overall. This work need not be expensive. MAD-TSC used volunteers for this purpose as they were motivated by research highlighting little benefit to paying people to do this[1, 101].

Improving NewsSentiment Performance

Currently, the process concludes before the next batch of articles is sourced from Bing, but a larger article intake would demand performance enhancements. Thanks to NewsSentiment's compatibility, a server equipped with CUDA GPU support could boost processing speed. Refactoring code to use batch processing — a feature added to NewsSentiment in late December 2023 - and increasing thread count on a more capable server could further optimise performance, ensuring our web app keeps pace with the latest headlines.

Entity Recognition Utilising Existing Entity Database

The NER pipeline stage could utilise entities confirmed by admins to provide better matches and reduce unresolved entities. Care would need to be taken to allow new entities to be created when there is no match and to ensure that the matching to existing entities is not too aggressive, especially in the case of common names.

Sentence Tokenization Optimisation

As of 03/04/2024, there were 1.5K BoundError (Fig. 12.9) versus 418K BoundMention entries, so this was not a priority to address. However, further custom tokenization rules could be deployed using existing defined methods, or punctuation restoration and sentence boundary detection models, which help predict if a word is the end of a sentence. These rules could be implemented when the scraping extract is imperfect to improve segments sent for analysis by NewsSentiment.

User reporting

Users should be able to report duplicate articles and ones with 'incorrect' sentiment values (8.2). We could take both types of reports forward to manually review and delete where appropriate, improve backend code, and adjust the scoring system.

References

[1] E. Dufraisse, A. Popescu, J. Tourille, A. Brun, and J. Deshayes, "MAD-TSC: A Multilingual Aligned News Dataset for Target-dependent Sentiment Classification," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 8286–8305. [Online]. Available: https://aclanthology.org/2023.acl-long.461

[2] "Interval Tree," Mar. 2014, section: Advanced Data Structure. [Online]. Available: https://www.geeksforgeeks.org/interval-tree/

[3] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651022000146

[4] W. Gad, A. Alokla, W. Nazih, A.-B. M.Salem, and M. Aref, "DLBT: Deep Learning-Based Transformer to Generate Pseudo-Code from Source Code," *Cmc -Tech Science Press-*, vol. 70, pp. 3117–3123, Oct. 2021.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019, arXiv:1810.04805 [cs]. [Online]. Available: http://arxiv.org/abs/1810.04805

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," Dec. 2014, arXiv:1412.3555 [cs]. [Online]. Available: http://arxiv.org/abs/1412.3555

[7] "ISO 5807:1985," 2019. [Online]. Available: https://www.iso.org/standard/11955.html

[8] F. Hamborg and K. Donnay, "NewsMTSC: A Dataset for (Multi-)Target-dependent Sentiment Classification in Political News Articles," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 1663–1675. [Online]. Available: https://aclanthology.org/2021.eacl-main.142

[9] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao, "Target-dependent Twitter Sentiment Classification," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, D. Lin, Y. Matsumoto, and R. Mihalcea, Eds. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 151–160. [Online]. Available: https://aclanthology.org/P11-1016

[10] D. Team, "t-Test, Chi-Square, ANOVA, Regression, Correlation..." 2024, graz, Austria. [Online]. Available: https://datatab.net/tutorial/cohens-kappa

[11] "News consumption in the UK: 2023," The Office of Communications (OfCom), Tech. Rep., 2023. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0024/264651/news-consumption-2023.pdf

[12] N. Newman, R. Fletcher, K. Eddy, C. T. Robertson, and R. K. Nielsen, "Digital News Report 2023," Reuters Institute for the Study of Journalism, University of Oxford, Tech. Rep.,

Jun. 2023. [Online]. Available: https://reutersinstitute.politics.ox.ac.uk/digital-news-report/2023/dnr-executive-summary

[13] N. Newman and R. Fletcher, "Bias, Bullshit and Lies: Audience Perspectives on Low Trust in the Media," Reuters Institute for the Study of Journalism, University of Oxford, Tech. Rep., 2017. [Online]. Available: https://reutersinstitute.politics.ox.ac.uk/our-research/bias-bullshit-and-lies-audience-perspectives-low-trust-media

[14] Neil T Gavin, "Media definitely do matter: Brexit, immigration, climate change and beyond," *Sage Journals*, vol. 20, no. 4, pp. 827–845, Nov. 2018. [Online]. Available: https://journals.sagepub.com/doi/epub/10.1177/1369148118799260

[15] Cammaerts et al., "Journalistic Representations of Jeremy Corbyn in the British Press: From Watchdog to Attackdog," Academic, Jul. 2016. [Online]. Available: https://orca.cardiff.ac.uk/id/eprint/126231/1/Cobyn-Report.pdf

[16] PWC Research, "BBC News & Current Affairs Review," The Office of Communications (OfCom), Tech. Rep., Oct. 2019. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0024/174075/bbc-news-review-pwc-summary-report.pdf

[17] M. V. Mäntylä, D. Graziotin, and M. Kuutila, "The evolution of sentiment analysis—A review of research topics, venues, and top cited papers," *Computer Science Review*, vol. 27, pp. 16–32, Feb. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013717300606

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019, arXiv:1907.11692 [cs]. [Online]. Available: http://arxiv.org/abs/1907.11692

[19] "Hugging Face." [Online]. Available: https://huggingface.co/about

[20] "spaCy 101: Everything you need to know · spaCy Usage Documentation." [Online]. Available: https://spacy.io/usage/spacy-101

[21] F. Hamborg, "NewsSentiment: Easy-to-use, high-quality target-dependent sentiment classification for English news articles." [Online]. Available: https://github.com/fhamborg/NewsMTSC

[22] "dslim/bert-base-NER · Hugging Face," Apr. 2023. [Online]. Available: https://huggingface.co/dslim/bert-base-NER

[23] S. Otmazgin, A. Cattan, and Y. Goldberg, "F-coref: Fast, Accurate and Easy to Use Coreference Resolution," in *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing: System Demonstrations*. Taipei, Taiwan: Association for Computational Linguistics, Nov. 2022, pp. 48–56. [Online]. Available: https://aclanthology.org/2022.aacl-demo.6

[24] ——, "LingMess: Linguistically Informed Multi Expert Scorers for Coreference Resolution," in *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 2752–2760. [Online]. Available: https://aclanthology.org/2023.eacl-main.202

[25] Namrata Godbole, Manja Srinivasaiah, Steven Skiena, "Large-Scale Sentiment Analysis for News and Blogs," in *The International AAAI Conference on Web and Social Media*, Mar. 2007. [Online]. Available: https://www.icwsm.org/papers/paper26.html

[26] A. Balahur, R. Steinberger, M. Kabadjov, V. Zavarella, E. van der Goot, M. Halkia, B. Pouliquen, and J. Belyaeva, "Sentiment Analysis in the News," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA), May 2010. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2010/pdf/909_Paper.pdf

[27] European Commission – Joint Research Centre (JRC), Ispra, Italy, R. Steinberger, S. Hegele, H. Tanev, and L. Della Rocca, "Large-scale news entity sentiment analysis," in *RANLP 2017 - Recent Advances in Natural Language Processing Meet Deep Learning*. Incoma Ltd. Shoumen, Bulgaria, Nov. 2017, pp. 707–715. [Online]. Available: http://www.acl-bg.org/proceedings/2017/RANLP%202017/pdf/RANLP091.pdf

[28] J. Wei, "The Stanford Sentiment Treebank (SST): Studying sentiment analysis using NLP," Oct. 2020. [Online]. Available: https://towardsdatascience.com/the-stanford-sentiment-treebank-sst-studying-sentiment-analysis-using-nlp-e1a4cad03065

[29] F. Hamborg, K. Donnay, and B. Gipp, "Towards Target-dependent Sentiment Classification in News Articles," 2021, vol. 12646, pp. 156–166, arXiv:2105.09660 [cs]. [Online]. Available: http://arxiv.org/abs/2105.09660

[30] M. Cinelli, G. De Francisci Morales, A. Galeazzi, W. Quattrociocchi, and M. Starnini, "The echo chamber effect on social media," *Proceedings of the National Academy of Sciences*, vol. 118, no. 9, p. e2023301118, Mar. 2021, publisher: Proceedings of the National Academy of Sciences. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.2023301118

[31] Anna Bredava, "How to do news monitoring: the complete guide," Oct. 2022, section: Public Relations. [Online]. Available: https://awario.com/blog/how-to-do-media-monitoring-guide/

[32] Connell, Adam, "Awario Review 2023: How Good Is This Social Listening Tool?" Jun. 2023. [Online]. Available: https://bloggingwizard.com/awario-review/

[33] "Ground News - Mission." [Online]. Available: https://ground.news/mission

[34] D. Ma, "Use of RSS feeds to push online content to users," *Decision Support Systems*, vol. 54, no. 1, pp. 740–749, Dec. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923612002370

[35] S. Alekhya, "Search for news with the Bing News Search API - Bing Search Services," Apr. 2022. [Online]. Available: https://learn.microsoft.com/en-us/bing/search-apis/bing-news-search/how-to/search-for-news

[36] A. Barbaresi, "Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, H. Ji, J. C. Park, and R. Xia, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 122–131. [Online]. Available: https://aclanthology.org/2021.acl-demo.15

[37] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei, and J. Launay, "The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only," Jun. 2023, arXiv:2306.01116 [cs]. [Online]. Available: http://arxiv.org/abs/2306.01116

[38] J. Piskorski, N. Stefanovitch, G. Da San Martino, and P. Nakov, "SemEval-2023 Task 3: Detecting the Category, the Framing, and the Persuasion Techniques in Online News in a Multi-lingual Setup," in *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, A. K. Ojha, A. S. Doğruöz, G. Da San Martino, H. Tayyar Madabushi, R. Kumar, and E. Sartori, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 2343–2361. [Online]. Available: https://aclanthology.org/2023.semeval-1.317

[39] X. Schmitt, S. Kubler, J. Robert, M. Papadakis, and Y. LeTraon, "A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, Oct. 2019, pp. 338–343. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8931850?casa_token=fJo3SlJBpJwAAAAA:NDC2pdp-uPMriGiymUNM-QHTGL0UTKVuYfpzLAXdYoMOPlksW2gY8dlaaIydfA4ROrf2mbIqgw

[40] L. Karttunen, "Discourse Referents," in *International Conference on Computational Linguistics COLING 1969: Preprint No. 70*, Sånga Säby, Sweden, Sep. 1969. [Online]. Available: https://aclanthology.org/C69-7001

[41] "neuralcoref · spaCy Universe." [Online]. Available: https://spacy.io/universe/project/neuralcoref

[42] K. Clark and C. D. Manning, "Deep Reinforcement Learning for Mention-Ranking Coreference Models," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016, pp. 2256–2262. [Online]. Available: http://aclweb.org/anthology/D16-1245

[43] S. Otmazgin, "fastcoref Python package." [Online]. Available: https://github.com/shon-otmazgin/fastcoref

[44] G. Grefenstette and P. Tapanainen, "What is a word, What is a sentence? Problems of Tokenization," Apr. 1994.

[45] S. Loria, "textblob Documentation," p. 13, Mar. 2024.

[46] M. De Berg, O. Cheong, M. Van Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. [Online]. Available: http://link.springer.com/10.1007/978-3-540-77974-2

[47] C. L. H. Tretyakov, Konstantin, "intervaltree: Editable interval tree data structure for Python 2 and 3." [Online]. Available: https://github.com/chaimleib/intervaltree

[48] M. Patel, "Understanding The Importance Of Tokenization In Machine Learning." [Online]. Available: https://www.c-sharpcorner.com/article/understanding-the-importance-of-tokenization-in-machine-learning/

[49] S. J. Mielke, Z. Alyafeai, E. Salesky, C. Raffel, M. Dey, M. Gallé, A. Raja, C. Si, W. Y. Lee, B. Sagot, and S. Tan, "Between words and characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP," Dec. 2021, arXiv:2112.10508 [cs]. [Online]. Available: http://arxiv.org/abs/2112.10508

[50] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://aclanthology.org/P16-1162

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Aug. 2023, arXiv:1706.03762 [cs]. [Online]. Available: http://arxiv.org/abs/1706.03762

[52] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," Jul. 2016, arXiv:1607.06450 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1607.06450

[53] D. Nozza, F. Bianchi, and D. Hovy, "What the [MASK]? Making Sense of Language-Specific BERT Models," Mar. 2020, arXiv:2003.02912 [cs]. [Online]. Available: http://arxiv.org/abs/2003.02912

[54] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[55] "WordPiece tokenization - Hugging Face NLP Course." [Online]. Available: https://huggingface.co/learn/nlp-course/en/chapter6/6

[56] M. Hosseinia, E. Dragut, and A. Mukherjee, "Stance Prediction for Contemporary Issues: Data and Experiments," in *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, L.-W. Ku and C.-T. Li, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 32–40. [Online]. Available: https://aclanthology.org/2020.socialnlp-1.5

[57] "Django overview." [Online]. Available: https://www.djangoproject.com/start/overview/

[58] "Security in Django." [Online]. Available: https://docs.djangoproject.com/en/5.0/topics/security/

[59] K. Das, "Introduction to Flask — Python for you and me 0.5.beta1 documentation." [Online]. Available: https://pymbook.readthedocs.io/en/latest/flask.html

[60] F. Fuior, "Introduction in Python Web Frameworks for Development," *Revista Română de Informatică și Automatică*, vol. 31, pp. 97–108, Sep. 2021.

[61] "Comparison of FastAPI with Django and Flask," Nov. 2023, section: Geeks Premier League. [Online]. Available: https://www.geeksforgeeks.org/comparison-of-fastapi-with-django-and-flask/

[62] M. Wanyoike, "History of front-end frameworks," Oct. 2018. [Online]. Available: https://blog.logrocket.com/history-of-frontend-frameworks/

[63] H. Dhaduk, "Best Frontend Frameworks for Web Development," Nov. 2022. [Online]. Available: https://www.simform.com/blog/best-frontend-frameworks/

[64] tkrotoff, "Front-end frameworks popularity (React, Vue, Angular and Svelte)." [Online]. Available: https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190

[65] "React – A JavaScript library for building user interfaces." [Online]. Available: https://legacy.reactjs.org/

[66] "Angular - What is Angular?" [Online]. Available: https://angular.io/guide/what-is-angular

[67] "Vue.js." [Online]. Available: https://vuejs.org/

[68] P. Achimugu, A. Selamat, R. Ibrahim, and M. N. Mahrin, "A systematic literature review of software requirements prioritization research," *Information and Software Technology*, vol. 56, no. 6, pp. 568–585, Jun. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914000354

[69] A. Y. Aleryani, "Comparative Study between Data Flow Diagram and Use Case Diagram," vol. 6, no. 3, 2016. [Online]. Available: https://testwp.adpi.co.id/wp-content/uploads/2022/04/DFDresearch.pdf

[70] E. Lumba and L. Hakim, "Digital Library Application Design," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, vol. 6, no. 2, pp. 193–197, 2023. [Online]. Available: https://tinyurl.com/3c8f4yhk

[71] "Chapter 6. Data-Flow Diagrams," University of Cape Town. [Online]. Available: https://www.cs.uct.ac.za/mit_notes/software/pdfs/Chp06.pdf

[72] N. R. Tague, *The Quality Toolbox, Second Edition*. Quality Press, Jan. 2005, google-Books-ID: G3c6S0mzLQgC.

[73] W. Vincent, "Django Best Practices: Projects vs. Apps," Oct. 2023. [Online]. Available: https://learndjango.com/tutorials/django-best-practices-projects-vs-apps

[74] H. Yang, "Implementation Principles of Components," in *Vue. JS Framework: Design and Implementation*, H. Yang, Ed. Singapore: Springer Nature, 2023, pp. 343–371. [Online]. Available: https://doi.org/10.1007/978-981-99-4947-2_12

[75] J. Smith, "Should I Use A Carousel?" [Online]. Available: https://shouldiuseacarousel.com/

[76] "Accessibility expert warns: stop using carousels," Jul. 2013. [Online]. Available: https://www.creativebloq.com/accessibility-expert-warns-stop-using-carousels-7133778

[77] S. Feinberg and M. Murphy, "Applying cognitive load theory to the design of Web-based instruction," in *18th Annual Conference on Computer Documentation. ipcc sigdoc 2000. Technology and Teamwork. Proceedings. IEEE Professional Communication Society International Professional Communication Conference an*, Sep. 2000, pp. 353–360. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/887293

[78] E. Kukkonen, "The influence of persuasive software features to the addictiveness of YouTube," Ph.D. dissertation, University of Oulu, Dec. 2023. [Online]. Available: https://oulurepo.oulu.fi/handle/10024/46842

[79] "Celery - Distributed Task Queue — Celery 5.3.6 documentation." [Online]. Available: https://docs.celeryq.dev/en/stable/

[80] "django-celery-beat - Database-backed Periodic Tasks — django_celery_beat 2.5.0 documentation." [Online]. Available: https://django-celery-beat.readthedocs.io/en/latest/

[81] "Redis Documentation." [Online]. Available: https://redis.io/docs/

[82] B. Franklin, *Pulling Newspapers Apart: Analysing Print Journalism*. Routledge, Mar. 2008, google-Books-ID: mz7R9OocNVgC.

[83] "Trinity Mirror buys newspaper rival Local World," *BBC News*, Oct. 2015. [Online]. Available: https://www.bbc.com/news/business-34660516

[84] "Reach PLC - Our Brands." [Online]. Available: https://www.reachplc.com/about-us/our-brands

[85] F. J. Damerau, "The Use of Function Word Frequencies as Indicators of Style," *Computers and the Humanities*, vol. 9, no. 6, pp. 271–280, 1975, publisher: Springer. [Online]. Available: https://www.jstor.org/stable/30204237

[86] S. Eissen, B. Stein, and M. Kulig, "Plagiarism Detection Without Reference Collections," Jan. 2006, pp. 359–366.

[87] E. Stamatatos, N. Fakotakis, and G. Kokkinakis, "Text Genre Detection Using Common Word Frequencies," in *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, 2000. [Online]. Available: https://aclanthology.org/C00-2117

[88] M. Zechner, M. Muhr, R. Kern, and M. Granitzer, "External and Intrinsic Plagiarism Detection Using Vector Space Models," 2009, pp. 47–55.

[89] C. G. Figuerola, R. G. Díaz, J. L. Alonso Berrocal, and A. F. Zazo Rodríguez, "Web Document Duplicate Detection Using Fuzzy Hashing," in *Trends in Practical Applications of Agents and Multiagent Systems*, J. M. Corchado, J. B. Pérez, K. Hallenborg, P. Golinska, and R. Corchuelo, Eds. Berlin, Heidelberg: Springer, 2011, pp. 117–125.

[90] M. Ulikowski, "ppdeep: Pure-Python library for computing fuzzy hashes (ssdeep)." [Online]. Available: https://github.com/elceef/ppdeep

[91] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*. John Wiley & Sons, Sep. 2011, google-Books-ID: GjyEFPkMCwcC.

[92] "Locust Documentation — Locust 0.1.dev129 documentation." [Online]. Available: https://docs.locust.io/en/stable/

[93] M. Khder, "Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application," *International Journal of Advances in Soft Computing and its Applications*, vol. 13, pp. 145–168, Dec. 2021.

[94] V. Krotov, L. Johnson, and L. Silva, "Tutorial: Legality and Ethics of Web Scraping," *Faculty & Staff Research and Creative Activity*, Dec. 2020. [Online]. Available: https://digitalcommons.murraystate.edu/faculty/86

[95] A. Luscombe, K. Dick, and K. Walby, "Algorithmic thinking in the public interest: navigating technical, legal, and ethical hurdles to web scraping in the social sciences," *Quality & Quantity*, vol. 56, no. 3, pp. 1023–1044, Jun. 2022. [Online]. Available: https://doi.org/10.1007/s11135-021-01164-0

[96] J.-C. Bricongne, B. Meunier, and S. Pouget, "Web-scraping housing prices in real-time: The Covid-19 crisis in the UK," *Journal of Housing Economics*, vol. 59, p. 101906, Mar. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S105113772200078X

[97] H. Kennedy, "Perspectives on Sentiment Analysis," *Journal of Broadcasting & Electronic Media*, vol. 56, no. 4, pp. 435–450, Oct. 2012, publisher: Routledge _eprint: https://doi.org/10.1080/08838151.2012.732141. [Online]. Available: https://doi.org/10.1080/08838151.2012.732141

[98] "Termly." [Online]. Available: https://app.termly.io/dashboard/website/247f92f7-3a9b-4fdc-acbd-3c7ebcf8c9ec

[99] "Article 8 GDPR." [Online]. Available: https://gdprhub.eu/index.php?title=Article_8_GDPR

[100] "What are the rules on cookies and similar technologies?" May 2023, publisher: ICO. [Online]. Available: https://ico.org.uk/for-organisations/direct-marketing-and-privacy-and-electronic-communications/guide-to-pecr/guidance-on-the-use-of-cookies-and-similar-technologies/what-are-the-rules-on-cookies-and-similar-technologies/

[101] A. Mao, E. Kamar, Y. Chen, E. Horvitz, M. Schwamb, C. Lintott, and A. Smith, "Volunteering Versus Work for Pay: Incentives and Tradeoffs in Crowdsourcing," *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 1, no. 1, pp. 94–102, Nov. 2013. [Online]. Available: https://ojs.aaai.org/index.php/HCOMP/article/view/13075

Chapter 12

Appendix

12.1 Credentials

Email: csoffice@contacts.bham.ac.uk

Password: OldJoe1908

Account has restricted admin access (view-only) at www.prominentprofiles.com/admin

12.2 Demonstrations

PPT_DEMO.pptx
Overview Video (aimed at survey participants)

12.3 GitHub

My GitHub repository hosts the code, and GitHub Actions handles the CI/CD pipeline with their runners (free for public repos).

12.4 Running Code

12.4.1 Localhost

The most straightforward way to run the code locally is without docker:

1. Create a Python virtual machine environment: `python3 -m venv env`
2. Clone the git repo: `git clone https://github.com/meeky37/FY-Project.git`
3. Change directory to the requirements file: `cd Django_App/prominent_profiles/` and then navigate to `requirements.txt`
4. Install required Python packages: `pip install -r requirements.txt`
5. Install a specific package without dependencies: `pip install newssentiment==1.1.25 --no-deps` (Note: Installing without dependencies to prevent unnecessary downgrade of torch, which on M1 Mac prevents 'mps' GPU use)
6. Make and apply migrations:
   `python manage.py makemigrations`
   `python manage.py migrate`
7. Create a superuser for the Django admin: `python manage.py createsuperuser`

8. Run the Django development server with custom settings: `python manage.py runserver 8000 --settings=prominent_profiles.test_settings`
9. To run Bing API commands, source a key from Azure and add it to `config.py`.
10. Collect article data: `python manage.py collect_article_data`
11. Trigger the extract and NLP processes: `python manage.py scrape_articles_concurrent`
12. Populate BingEntity for visible entities: `python manage.py visible_entity_bing` (Entities must be set to visible = TRUE via the admin page or dbshell)
13. Explore and try out management jobs in `nlp_processor` and `profiles_app` by reading the docstrings and applying them to your analyzed data.
14. Navigate to the Vue.js project directory from the git root: `cd Vue_js/prominent_profiles`
15. Install required front-end libraries: `npm install`
16. Set the API base URL: Set `API_BASE_URL = 'http://localhost:8000'` in the project configuration.
17. Serve the Vue.js application: `npm run serve`
18. Access the Django admin and Vue app: Typically, `localhost:8000/admin` will serve the Django admin interface, and `localhost:8080` will serve the Vue.js application, although ports may vary.

12.4.2 Deployment

1. Create a GitHub repository (others may work but we use GitHub Actions so this is the most compatible route with YAML files as is).
2. Set up Ubuntu web server on Digital Ocean using the most recent Docker and Ubuntu configuration provided with 8GB memory if you plan to scrape and run the pipeline (Fig. 12.2) - you can resize down to 2GB once you have some data.
3. Generate SSH keys.
4. Store the private key as a secret in GitHub.
5. Add a database (base price) when prompted by Digital Ocean.
6. Obtain a domain name (e.g., Namecheap).
7. Set up a Zoho mail account using this domain.
8. Configure the DNS records using documentation provided by your domain and mail provider.
9. Create a GitHub Container Registry Personal Access Token (GHCR PAT).
10. Set the following in GitHub settings → Security → Secrets and Variables → Actions:

- BING_API_KEY
- CERTBOT_EMAIL
- DB_HOST
- DB_NAME
- DB_PASSWORD
- DB_PORT
- DB_USER
- DJANGO_DEBUG
- DROPLET_HOST
- DROPLET_PUBLIC_KEY
- DROPLET_USER

- EMAIL_PASSWORD
- GHCR_PAT
- GHCR_USERNAME
- RUNNING_IN_DOCKER
- SECRET_KEY
- SSH_KNOWN_HOSTS
- SSH_PRIVATE_KEY

10. (Clone locally if you wish) and make a commit.
11. The build, test and deploy CI/CD pipeline should trigger automatically.
12. Visit your domain name to check Vue.
13. Visit your domain name /admin to check Django.

DNS records

Type	Hostname	Value	TTL (seconds)	
TXT	zmail._domainkey.prominentprofiles.c...	returns v=DKIM1; k=rsa; p=MIGfMA0GCSqGSI...	3600	More
TXT	prominentprofiles.com	returns v=spf1 include:zoho.eu ~all	3600	More
MX	prominentprofiles.com	mail handled by mx3.zoho.eu.	50 14400	More
MX	prominentprofiles.com	mail handled by mx2.zoho.eu.	20 14400	More
MX	prominentprofiles.com	mail handled by mx.zoho.eu.	10 14400	More
TXT	prominentprofiles.com	returns zoho-verification=zb55685600.zmveri...	3600	More
A	www.prominentprofiles.com	directs to 157.245.46.42	3600	More
A	prominentprofiles.com	directs to 157.245.46.42	3600	More
NS	prominentprofiles.com	directs to ns1.digitalocean.com.	1800	More
NS	prominentprofiles.com	directs to ns2.digitalocean.com.	1800	More
NS	prominentprofiles.com	directs to ns3.digitalocean.com.	1800	More

Figure 12.1: Digital Ocean DNS Records

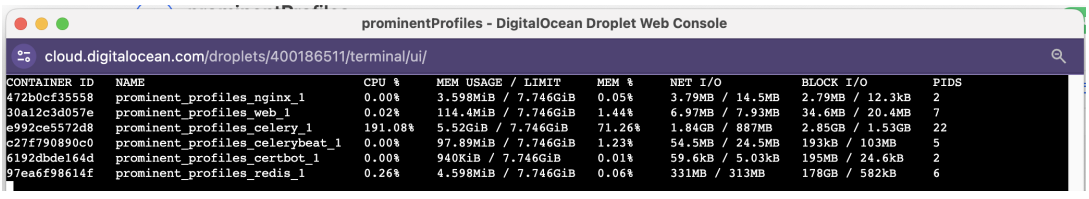


Figure 12.2: Memory Use during Scrape and NLP Pipeline operation

12.5 System Requirements

12.5.1 Functional

1. **News Article Collection and Processing**

1.1. The system MUST retrieve URLs for news articles from the Bing Search API and store them.

1.2. The system MUST track the storage location of API responses containing article URLs.

1.3. The system MUST extract text from the retrieved news article URLs using Trafilatura.

1.4. The system MUST respect scraping robots.txt scraping rules.

1.5. The system MUST not scrape the same URL if the article has already been processed.

1.6. The system MUST perform NER on the article text to identify entities of type PERSON, using spaCy or equivalent.

1.7. The system MUST perform coreference resolution on extracted text to identify further NE mentions, e.g., pronouns.

1.8. The system MUST tokenize sentences to obtain the broader context for mentions of NE, using TextBlob or equivalent.

1.9. The system MUST map NER of type PERSON to their respective coreference clusters, using FastCoref (or LingMess if resources allow).

1.10. The system MUST threshold mentions of entities to 20% of an article's sentences so a representative score can be determined.

1.11. The system MUST identify the overlaps between sentences and mentions of NEs, using IntervalTree or equivalent.

1.12. The system MUST apply sentiment analysis to sentences containing entities using the NewsSentiment package.

1.13. The system MUST determine an overall set of sentiment scores for a given article and entity significantly within it.

1.14. The system MUST store the processed data, including entities, sentiment scores, and article metadata, in a database for future retrieval.

1.15. The system MUST perform entity and coreference consolidation, enhancing entity resolution accuracy with other articles.

1.16. The system SHOULD track whether a set of articles obtained via an API call has been analysed to prevent needless re-analysis.

1.17. The system SHOULD discard articles with the same headline and author of an article already processed.

1.18. The system SHOULD consolidate entities linked to multiple coreference clusters (and vice-versa) to accurately associate named entities with their mentions within articles and enhance entity resolution accuracy with other articles.

1.19. The system SHOULD exclude news articles from sources known to be incompatible with text extraction libraries (e.g., msn.com) to increase efficiency and reduce processing of unextractable content.

1.20. The system SHOULD implement threading for parallel processing of articles through the NLP pipeline to reduce processing time and continuous load on the server.

1.21. The system SHOULD have a single command to scrape, apply NLP processing, analyse sentiments, and store in the database that can be run on demand by an admin.

1.22. The system COULD have an automated command to perform the above.

1.23. The system COULD automatically retrieve URLs for news articles from the Bing Search API.

1.24. The system COULD implement duplicate detection and avoidance mechanisms for articles to prevent expensive reprocessing of (near-)identical content.

1.25. The system COULD apply custom tokenization rules to improve the accuracy of sentence boundary detection, facilitating more reliable sentiment analysis.

1.26. The system COULD store error logs for failed sentiment analysis attempts to aid debugging and improve processing accuracy over time.

1.27. The system COULD use entities marked as resolved in the entity NER to coref consolidation phase to update names with 'labelled' data rather than relying solely on spaCy and substrings of the coref clusters.

1.28. The system WON'T HAVE the ability to display a summary of news outlets' representations of a particular entity.

1.29. The system WON'T HAVE the capability to obtain the TRUE publisher of content republished on news websites like Yahoo News.

2. **Admin Interfaces**

2.1. The system MUST allow administrators to manually correct or update entity information as needed to maintain data accuracy.

2.2. The system MUST enable administrators to perform batch updates or deletions of articles, entities, and other data to efficiently manage large volumes of information.

2.3. The system MUST offer advanced search capabilities within the admin interface to allow administrators to quickly locate specific articles, entities, and user accounts based on various attributes (e.g., headline, entity name, email).

2.4. The system MUST allow administrators to manage user accounts, including creating, updating, suspending, or deleting accounts to maintain the user base.

2.5. The system MUST ensure that access to sensitive actions in the admin interface (e.g., data deletion, user account management) is restricted to authorised personnel through appropriate permissions and role-based access controls.

2.6. The system SHOULD provide a management interface that automatically highlights similar entities for review and merging to improve entity resolution.

2.7. The system SHOULD implement mechanisms to update entity information periodically to ensure data relevance and accuracy.

2.8. The system SHOULD provide custom actions in the admin interface for everyday data management tasks, such as marking entities as visible in the app, and merging entities to ensure consistency and accuracy of data.

2.9. The system SHOULD offer advanced search capabilities within the admin interface to allow administrators to quickly locate specific articles, entities, and user accounts based on various attributes (e.g., headline, entity name, email).

2.10. The system SHOULD implement custom filters in the admin interface to segment data for more efficient review and management, such as filtering articles or entities based on sentiment scores or the presence of bound mentions.

2.11. The system SHOULD enable administrators to view and manage user subscriptions, including the ability to add or remove subscriptions on behalf of users.

2.12. The system SHOULD maintain logs of significant administrative actions (e.g., entity merges, data updates) to provide an audit trail for accountability and tracking changes over time.

2.13. The system SHOULD optimise the admin interface for performance, especially when displaying large datasets, to enhance usability and reduce load times.

2.14. The system COULD implement asynchronous data loading or pagination in admin views with many inline instances (e.g., bound mentions) to improve page responsiveness.

2.15. The system COULD allow administrators to semi-automatically review and merge similar entities rather than manually.

2.16. The system COULD provide additional reporting tools, widgets or dashboards in the admin interface for enhanced data visualisation and analysis.

2.17. The system COULD integrate with third-party services or APIs to enrich entity information, such as social media profiles or external databases.

3. **User Account and Subscription Management**

3.1. The system MUST allow users to create accounts using email and password.

3.2. The system MUST display an error message if the login credentials are incorrect.

3.3. The system MUST allow the registered user to logout.

3.4. The system MUST redirect a registered user to the login screen following a logout request.

3.5. The system MUST restrict unauthenticated users from the dashboard view to the login view if they try to access it.

3.6. The system MUST not allow users to sign up with the same email (or phone if implemented) for additional accounts.

3.7. The system MUST allow users to subscribe to entities to appear in their dashboard.

3.8. The system MUST maintain a personalised dashboard for each user, showing subscribed entities and relevant article updates.

3.9. The system SHOULD validate each input box during registration or login for a better UX by alerting to errors before form submission.

3.10. The system SHOULD enable users to receive updates in their dashboard on new articles associated with their subscriptions.

3.11. The system SHOULD redirect unauthenticated users trying to subscribe to an entity to the login view.

3.12. The system SHOULD have access to an email address to enable password resets and allow users to get in contact.

3.13. The system SHOULD provide a basic password reset functionality using built-in Django journeys.

3.14. The system SHOULD authenticate users via JWTs for secure access to personalised features.

3.15. The system SHOULD track user views of entity profiles to determine trending entities for use in a feature.

3.16. The system COULD allow users to log in to accounts using a mobile number.

3.17. The system COULD provide an integrated password reset functionality in the Vue app.

3.18. The system COULD allow password changes after login as an alternative to following the reset path.

3.19. The system COULD allow the user to be logged in from multiple devices.

3.20. The system COULD provide users with customised views of articles of subscribed entities.

3.21. The system COULD allow users to favourite articles for later reference.

3.22. The system WON'T allow users to log in using a so-called 'tech-giant' account, e.g. Google or Facebook.

4. **Data Presentation and User Interaction**

4.1. The system MUST provide an entity summary that displays all articles related to them, categorised by sentiment.

4.2. The system MUST enable users to navigate between different views (Entity, Article Analysis, Home, etc.) seamlessly using the SPA offered by Vue.

4.3. The system MUST not display article text, only its metadata.

4.4. The system MUST display a consistent header across all views for consistency.

4.5. The system MUST provide a way to access a list of all entities set by admins to be visible to all users.

4.6. The system MUST attribute entity images and descriptions sourced from external sources.

4.7. The system SHOULD offer an Article Analysis View detailing the article sentiment analysis results, metadata and other entities mentioned in the article.

4.8. The system SHOULD present a Home View featuring trending entities based on user interaction or alternative criteria.

4.9. The system SHOULD obtain and display additional information, e.g., a description of an entity from Wikipedia.

4.10. The system SHOULD hyperlink entity names and titles for more accessible, intuitive navigation.

4.11. The system COULD allow users to filter and sort news articles by sentiment score or publication date for customised content exploration.

4.12. The system COULD provide social media share buttons/icons to share article analysis they wanted and help drive traffic to the app.

12.5.2 Non-Functional

1. **Performance and Scalability**

   1.1. The system MUST support access by up to 100 concurrent users without significant degradation in response times.

   1.2. The system MUST have scalable memory allocation within the fixed server size.

   1.3. The system MUST utilise its internal APIs efficiently to minimise response times and database load.

   1.4. The system SHOULD implement efficient data processing techniques, such as batch processing coreferences, to reduce article processing time.

   1.5. The system SHOULD temporarily store NewsSentiment results for cases where consolidation didn't create a one-to-one relationship between NER and corefs to save on expensive duplicate model applications to corefs.

   1.6. The system SHOULD utilise external APIs carefully to preserve monthly quotas.

   1.7. The system SHOULD ensure that loading times for all views do not exceed 2 seconds under typical conditions.

   1.8. The system SHOULD ensure high availability and reliability, employing strategies including reverse proxying to effectively manage traffic and server loads.

   1.9. The system COULD automatically process new articles obtained within 1 hour of retrieval.

   1.10. The system COULD implement caching strategies for frequently accessed data (e.g., popular entities and articles) to reduce database load and improve response times.

   1.11. The system WON'T have architectural support for server resource scaling to accommodate increasing data volume and user traffic.

2. **Usability and Accessibility**

   2.1. The system MUST be available on a local machine.

   2.2. The system MUST provide an intuitive and responsive UI for desktop users that encourages exploration and engagement with content.

   2.3. The system MUST have a consistent theme that does not make content difficult to see.

   2.4. The system MUST be compatible with major browsers and adhere to web standards to ensure access to a broad audience.

   2.5. The system SHOULD be available on a public web address with a registered domain name.

   2.6. The system SHOULD implement intuitive navigation and interactive elements (e.g., filtering options, clickable entities) to enhance user engagement.

   2.7. The system SHOULD provide alternative text for images to enhance accessibility.

   2.8. The system SHOULD ensure that navigation is possible through keyboard inputs.

   2.9. The system SHOULD be accessible to users with disabilities, following WCAG for accessibility.

   2.10. The system COULD provide an intuitive and responsive UI optimised for portable device users compatible with various screen sizes to ensure device usability.

3. **Security and Data Protection**

   3.1. The system MUST implement secure password storage using salted hash algorithms.

   3.2. The system MUST implement robust validation and error handling in the backend to prevent security vulnerabilities and ensure data integrity.

   3.3. The system MUST ensure passwords are at least eight characters long and contain at least one letter and number.

   3.4. The system MUST delete all user information when an account is deleted.

   3.5. The system MUST store passwords in an encrypted manner that administrators cannot decrypt.

   3.6. The system SHOULD use HTTPS for all data transactions to ensure data integrity and confidentiality.

3.7. The system SHOULD implement rate limiting on API endpoints to prevent abuse and ensure service availability.

3.8. The system SHOULD use a reverse proxy to prevent direct access to the application server, reducing the attack surface.

3.9. The system COULD have 2-factor authentication using a user's email or mobile phone.

3.10. The system COULD use GitHub security to highlight vulnerabilities in package dependencies.

4. **Compliance and Legal**

4.1. The system MUST comply with data protection regulations (e.g., GDPR) by implementing cookie consent.

4.2. The system MUST comply with data protection regulations (e.g., GDPR) by setting a minimum age for account registration.

4.3. The system MUST comply with data protection regulations (e.g., GDPR) by providing a privacy policy.

4.4. The system COULD implement a mechanism for users to report inaccurate or inappropriate content, ensuring compliance with legal and ethical standards.

5. **Maintainability**

5.1. The system MUST be developed using modular design principles to facilitate maintenance and future updates.

5.2. The system MUST utilise Vue components to promote reusability, minimise maintenance overhead, and provide cleaner code.

5.3. The system code for the pipeline MUST be commented sufficiently to ensure a new developer unfamiliar with the system could understand and maintain it.

5.4. The system MUST include a comprehensive suite of unit tests covering critical functionalities to ensure code maintainability and ease of future updates.

5.5. The system's admin interface SHOULD be designed to allow easy addition or modification of custom actions, filters, and other features to accommodate future requirements without significant refactoring.

5.6. The system SHOULD include comprehensive documentation of its architecture, codebase, and API endpoints.

5.7. The system SHOULD be designed to allow for easy updates and integrating new features without significant restructuring.

5.8. The system's JavaScript code SHOULD conform to the JavaScript Linter (ES-Lint) standards to maintain code quality.

5.9. The system COULD offer training materials or sessions for administrators to ensure they are well-equipped to manage the system effectively.

6. **CI/CD**

6.1. The system SHOULD employ CI/CD pipelines for automated building and deployment to streamline development workflows and ensure reliable software delivery.

6.2. The system COULD utilise containerisation (e.g., Docker) for consistent environments across development, testing, and production stages.

12.6 Design

12.6.1 Lower Level Subprocesses

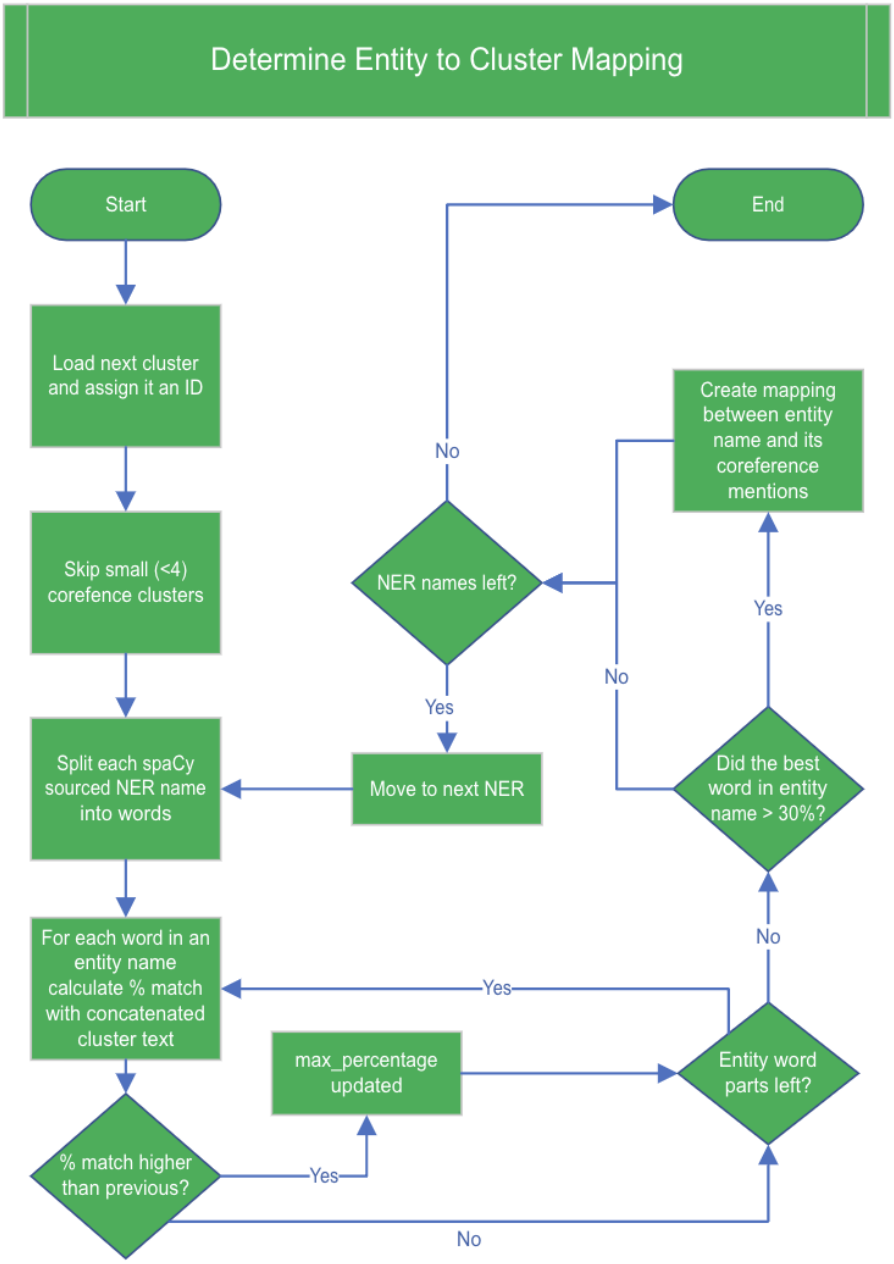


Figure 12.3: Determine Entity to Cluster Mapping Subprocess Flowchart

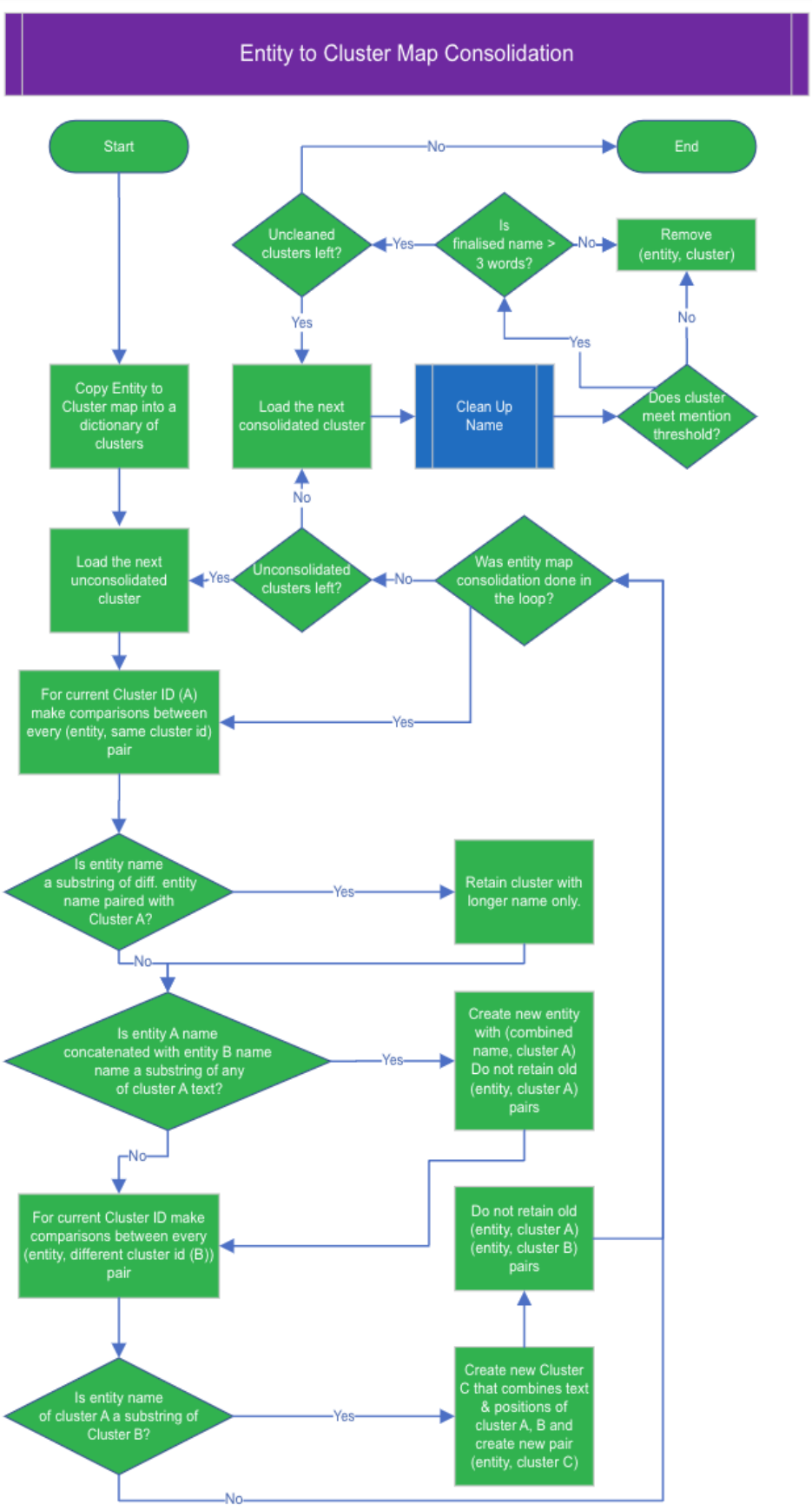


Figure 12.4: Entity to Cluster Map Consolidation Subprocess Flowchart

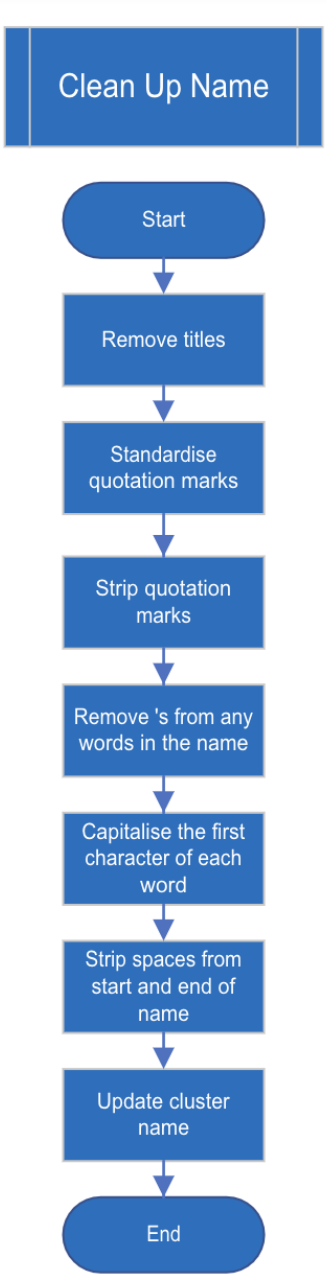


Figure 12.5: Clean Up Name Subprocess Flowchart

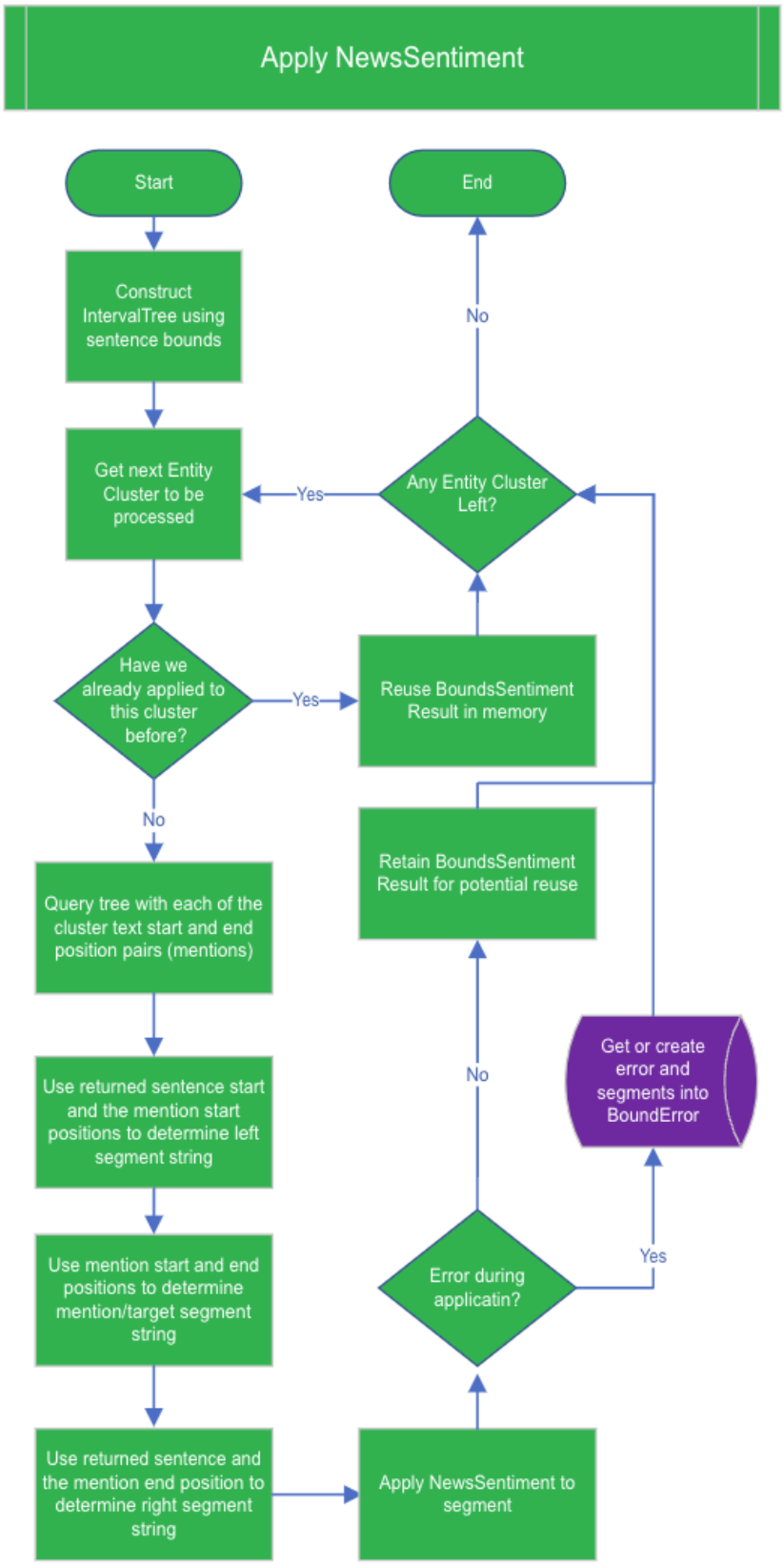


Figure 12.6: Apply News Sentiment Subprocess Flowchart

12.6.2 Notebook Phase Directory Structure

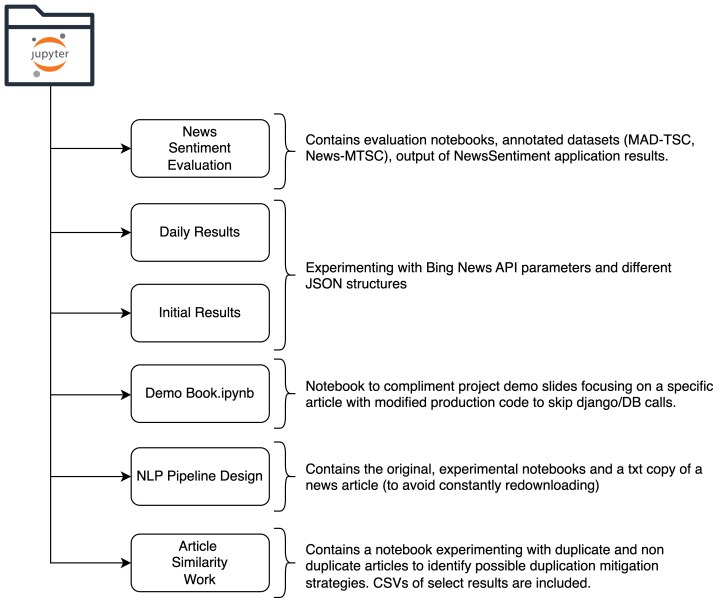


Figure 12.7: Folder Structure of Notebook Phase

12.6.3 Vue Structure

```
/
├── components/
│   ├── ArticleEntriesContainer.vue
│   ├── ArticleEntry.vue
│   ├── ArticleOtherEntities.vue
│   ├── CookieBanner.vue
│   ├── EntitySelection.vue
│   ├── EntitySpotlight.vue
│   ├── HelloWorld.vue
│   ├── PageFooter.vue
│   ├── ProfileEntry.vue
│   ├── SortToggle.vue
│   ├── SubProfileCard.vue
│   ├── SubProfilesGrid.vue
│   ├── SubscriptionButton.vue
│   ├── TrendingProfileCard.vue
│   └── TrendingProfiles.vue
├── interceptors/
│   └── axios.js
├── router/
│   └── index.js
├── shared_methods/
│   ├── auth_methods.js
│   ├── common_requests.js
│   └── validationUtils.js
└── views/
    ├── AboutView.vue
    ├── ArticlePage.vue
    ├── DashboardPage.vue
    ├── EntityPage.vue
    ├── ForgotPassword.vue
    ├── HomePage.vue
    └── HomeView.vue
```

```
├── LoginPage.vue
├── MenuPage.vue
├── PrivacyPolicy.vue
├── ResetPassword.vue
└── SignUpPage.vue
```

12.7 Implementation

12.7.1 Django API Documentation

Visit https://prominentprofiles.com/swagger or https://prominentprofiles.com/redoc depending on your preference for Swagger or Redoc UI.

12.7.2 Admin Views

Select article to change

ADD ARTICLE +

Search

Action: --------- Go 0 of 100 selected

ID	PROCESSED	SIMILAR REJECTION	HEADLINE	DATE ADDED	PUBLICATION DATE	SITE NAME	AUTHOR	SOURCE FILE
29995	✓	✗	Kate Middleton was first photographed with Prince William 20 years ago — revisit the moment	April 4, 2024, 7:48 p.m.	April 3, 2024, midnight	New York Post	Samantha Ibrahim	ProcessedFile object (1445)
29994	✗	✓	The true story behind Netflix's 'Scoop,' about Prince Andrew's explosive BBC interview	April 4, 2024, 7:48 p.m.	April 3, 2024, midnight	TODAY.com	Scott Stump	ProcessedFile object (1445)
29993	✗	✓	Meghan Markle and Beyonce's Moms Doria Ragland and Tina Knowles Bond in Rare New Photo	April 4, 2024, 7:47 p.m.	April 3, 2024, midnight	Yahoo Entertainment	Ryan Hudgins	ProcessedFile object (1445)
29992	✗	✓	Prince Harry in 'painful place' after writing about Kate Middleton in Spare: expert	April 4, 2024, 7:47 p.m.	April 4, 2024, midnight	Thechronicle	-	ProcessedFile object (1445)
29991	✓	✗	Princess Beatrice, Princess Eugenie 'very upset' after King Charles snub	April 4, 2024, 7:46 p.m.	April 3, 2024, midnight	NZ Herald	NZ Herald	ProcessedFile object (1445)
29990	✓	✗	Why Prince Harry Is Mentioned in Sexual Abuse Lawsuit Against Diddy	April 4, 2024, 7:44 p.m.	April 4, 2024, midnight	PopCulture.com	Michael Hein	ProcessedFile object (1445)

Figure 12.8: Article Admin

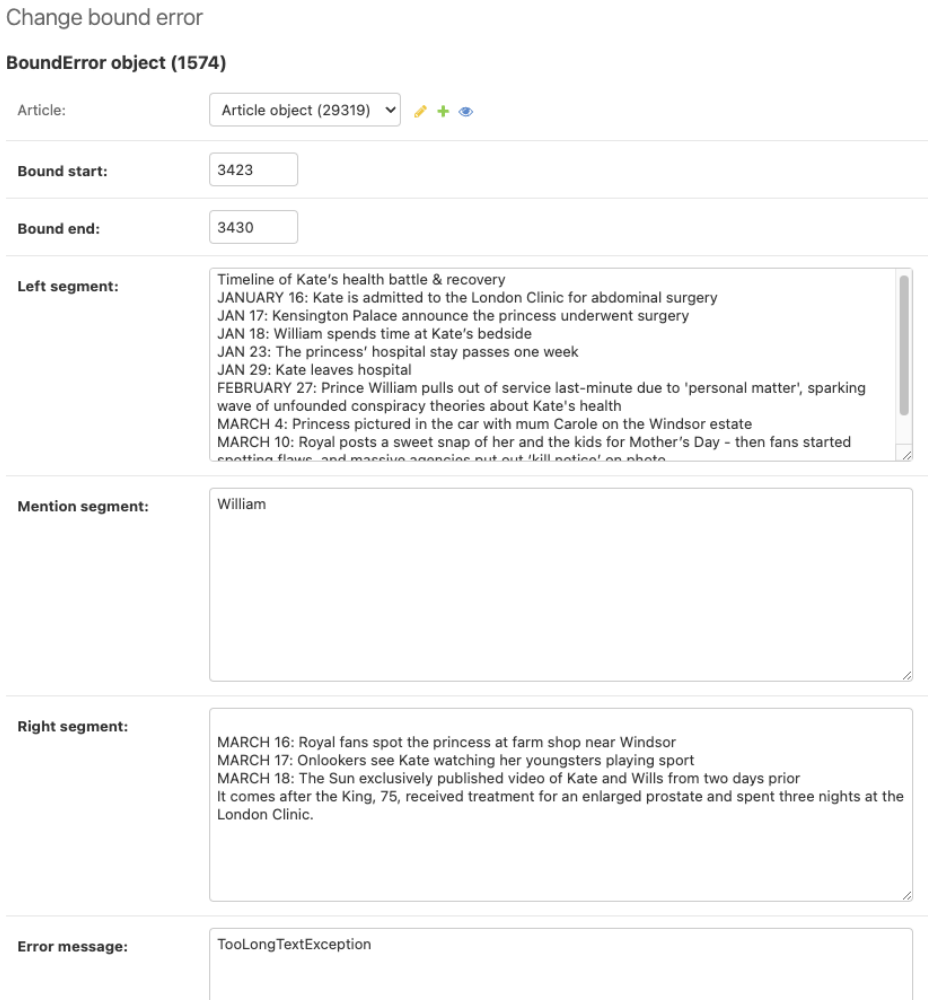


Figure 12.9: BoundError Admin Object

Select bound mention to change

ADD BOUND MENTION +

Action: --------- Go 0 of 100 selected

ARTICLE ID	ENTITY ID	ENTITY NAME	BOUND START	BOUND END	AVG NEUTRAL	AVG POSITIVE	AVG NEGATIVE	BOUND TEXT
30490	2158	Donald Trump	1542	1723	0.00	0.00	0.61	In the 2020 elections, only only 3 per cent of the voters could be categorised as double haters, and Biden's favourability ratings were much better then compared with that of Trump.
30490	2158	Donald Trump	793	937	0.00	0.00	0.92	No wonder, the presidential election this year is witnessing a surge in the number of "double haters''--voters who dislike both Biden and Trump.
30490	2158	Donald Trump	342	454	0.83	0.00	0.00	Ebey said Texas needed at least three options in November: Joe Biden, Donald Trump and 'Literally Anybody Else'.
30490	482	Joe Biden	8368	8410	0.79	0.00	0.00	I think he's probably going to hurt Biden.
30490	482	Joe Biden	8189	8236	0.00	0.00	0.49	Even Trump thinks Kennedy is a threat to Biden.
30490	482	Joe Biden	7489	7667	0.49	0.00	0.00	If he can get on the ballot and splinter Biden's Hispanic coalition from 2020, there is a real chance of Trump winning both states and opening up a clear path to the White House.
30490	482	Joe Biden	7399	7488	0.00	0.00	0.90	Biden's support among this key demographic is slipping and Kennedy seems to be improving.
30490	482	Joe Biden	7101	7256	0.79	0.00	0.00	Kennedy, meanwhile, is improving in polls, and although he has so far qualified to be a candidate only in Utah, his growing support worries the Biden camp.

Figure 12.10: BoundMention Admin

Select Entity to change

Search

Action: --------- Go 0 of 1000 selected

	NAME	TYPE	APP VISIBLE 2	VIEW COUNT	ARTICLE COUNT (NUMERIC) 1
☐	Meghan Markle	-	✓	0	2544
☐	Prince Harry	-	✓	0	2363
☐	Rishi Sunak	-	✓	0	1424
☐	Prince William	-	✓	0	1257
☐	Keir Starmer	-	✓	0	1125
☐	Donald Trump	-	✓	0	903
☐	King Charles III	-	✓	0	893
☐	Kate Middleton	-	✓	0	871
☐	Jeremy Hunt	-	✓	0	419
☐	Tory	-	✗	0	374
☐	Joe Biden	-	✓	0	369
☐	Lee Anderson	-	✓	0	280

Figure 12.11: Entity Admin

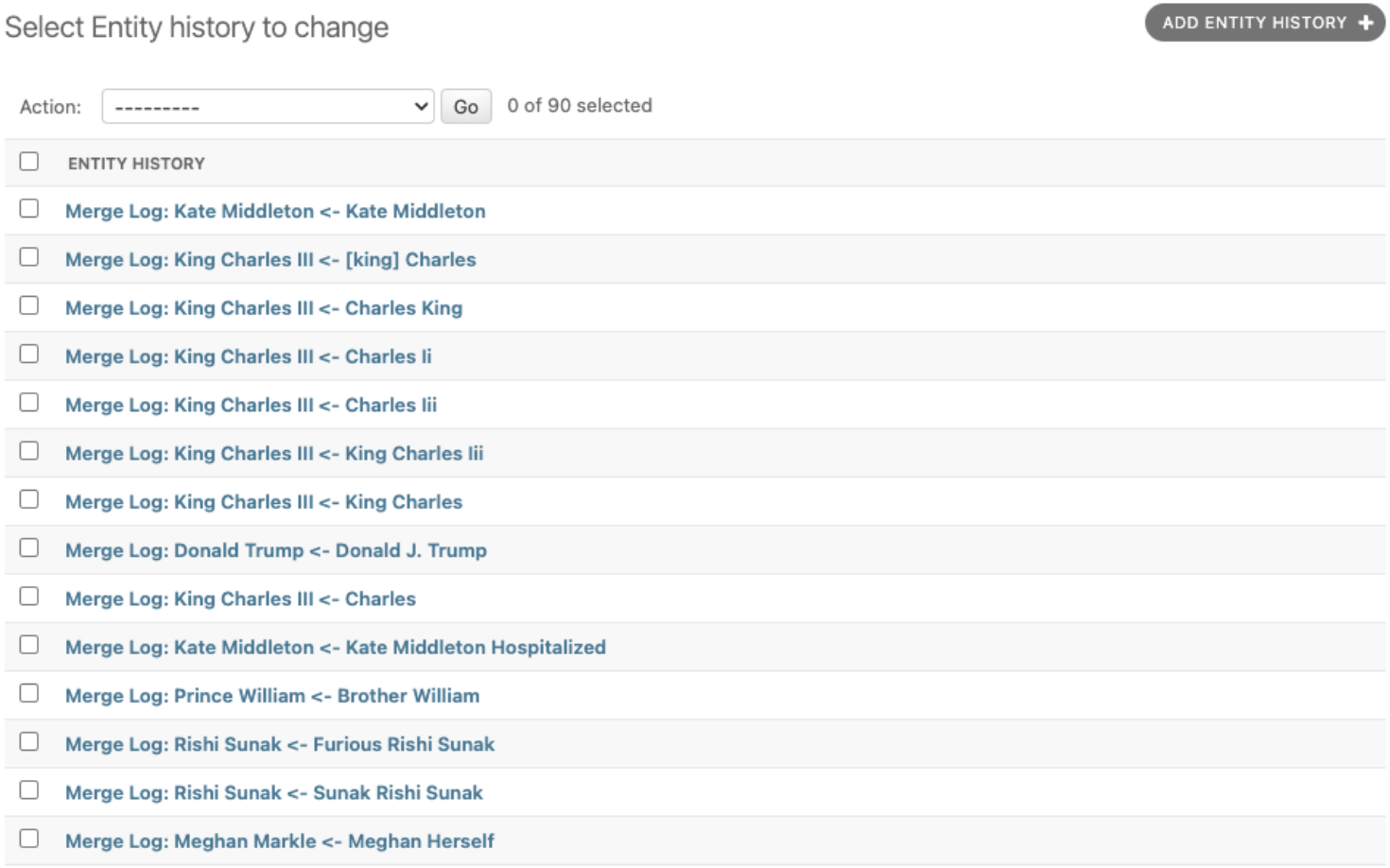


Figure 12.12: EntityHistory (Merge Log)

Select entity view to change

Search

Action: --------- Go 0 of 100 selected

ENTITY	ENTITY NAME	VIEW DT	VIEW TIME
Entity object (21)	Angela Rayner	April 5, 2024	5:13 p.m.
Entity object (33)	David Cameron	April 5, 2024	5:12 p.m.
Entity object (664)	Richard Tice	April 5, 2024	5:12 p.m.
Entity object (1371)	Andy Burnham	April 5, 2024	5:12 p.m.
Entity object (20)	Liz Truss	April 4, 2024	1:31 p.m.
Entity object (190)	Prince Andrew	April 4, 2024	1:31 p.m.
Entity object (42)	Rishi Sunak	April 4, 2024	1:29 p.m.
Entity object (17)	Jeremy Hunt	April 4, 2024	1:29 p.m.
Entity object (8089)	Kate Middleton	April 3, 2024	9:29 p.m.

Figure 12.13: EntityView Admin

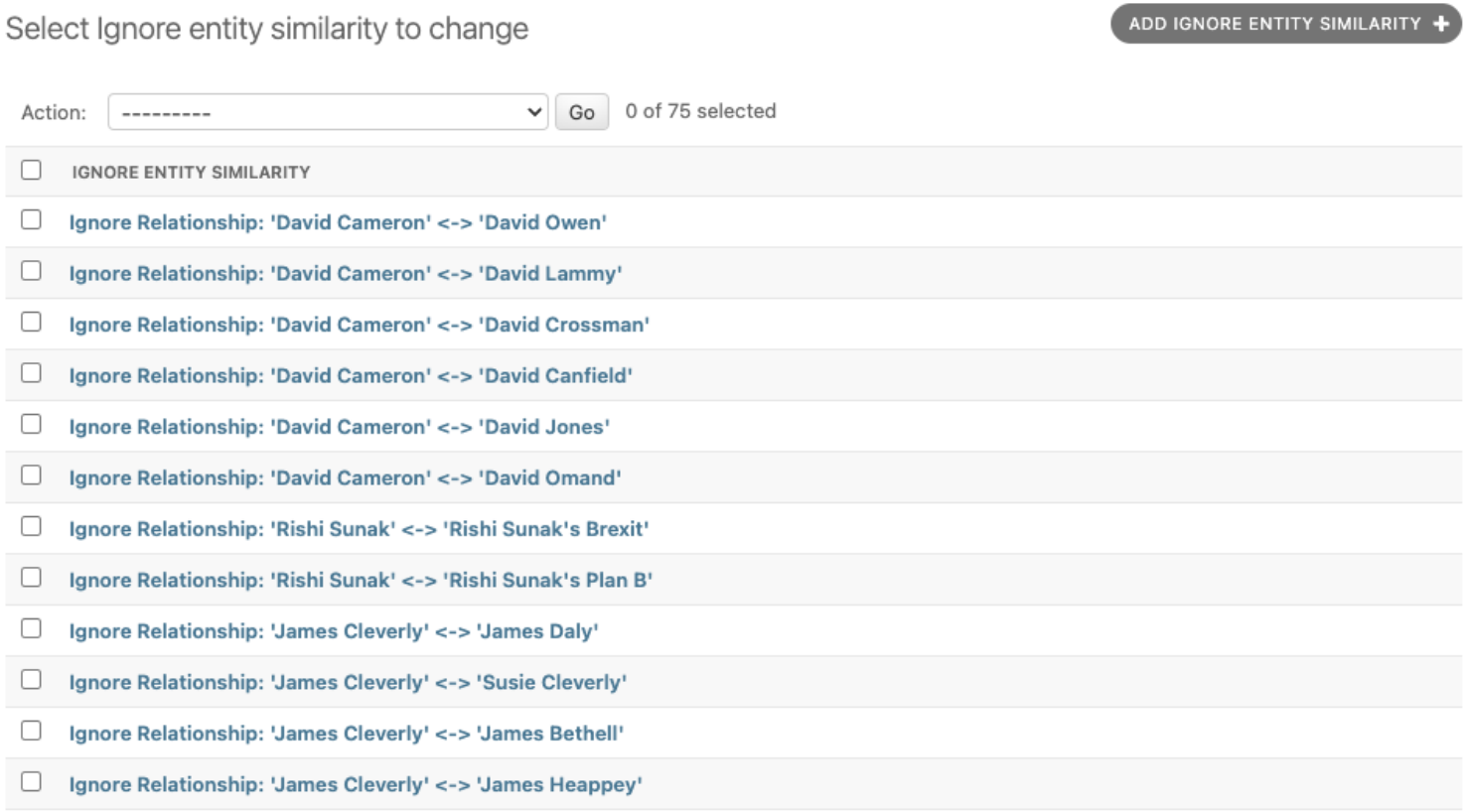


Figure 12.14: IgnoreEntitySimilarity Admin

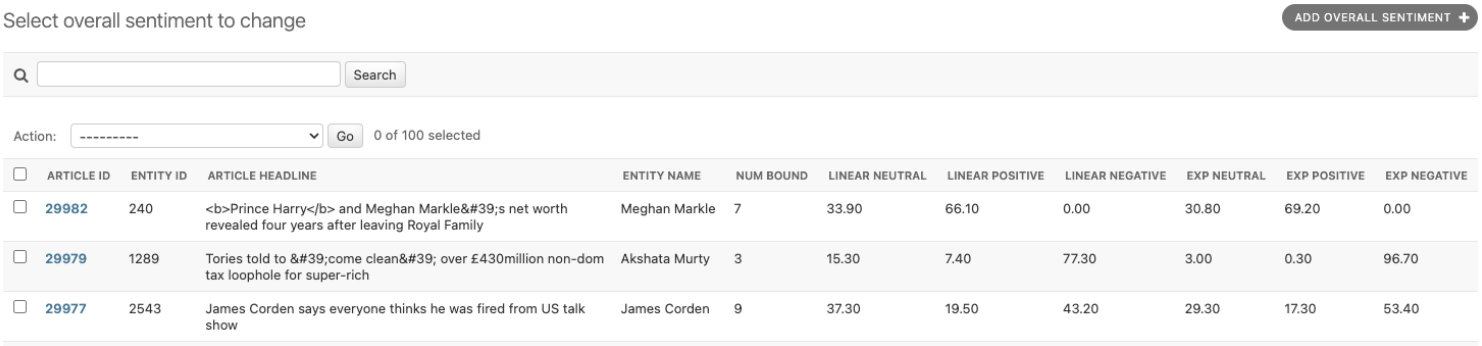


Figure 12.15: OverallSentiment Admin

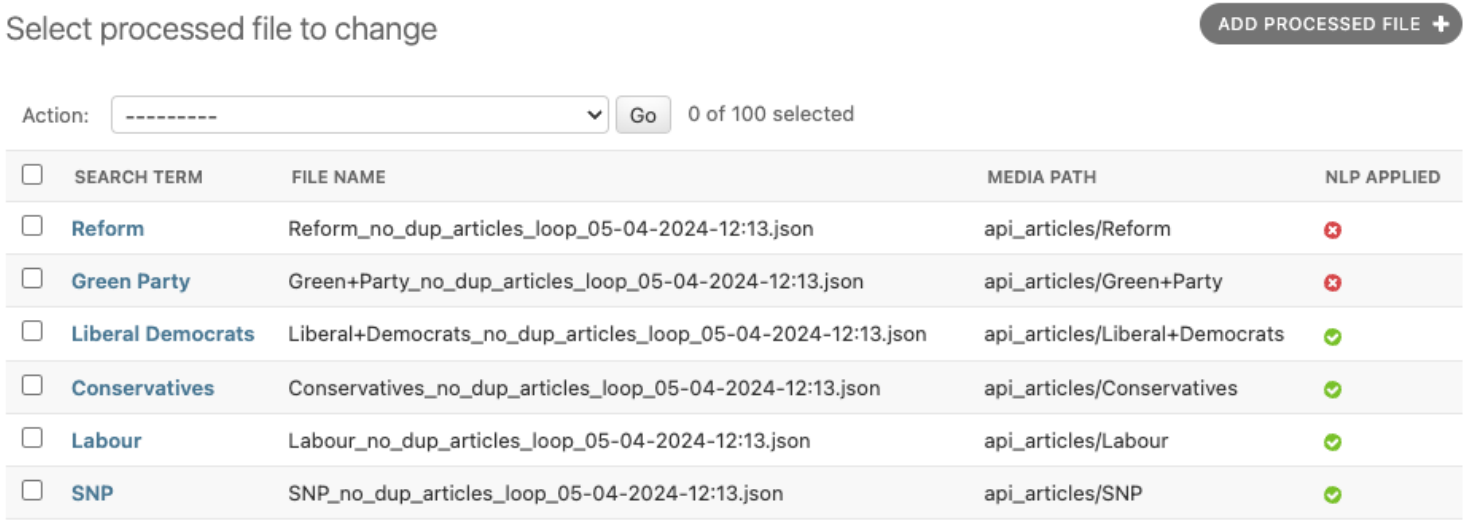


Figure 12.16: ProcessedFile Admin

Select similar article pair to change

Search

Action: --------- Go 0 of 100 selected

ID	ARTICLE 1 HEADLINE	ARTICLE 2 HEADLINE	ARTICLE 1 SITE NAME	ARTICLE 2 SITE NAME	HASH SIMILARITY SCORE	WORDS DIFF	TERMS DIFF	YULEK DIFF	VOCD DIFF	AVG COUNT DIFF	SIMPSOND DIFF	THE DIFF	AND DIFF	IS DIFF	OF DIFF	IN DIFF	TO DIFF	IT DIFF	THAT DIFF	WITH DIFF
33963	Sunak criticised over comment saying border security more important than staying in European court of human rights – UK politics live	Alan Duncan facing Tory disciplinary inquiry over comments accusing senior party figures of being too 'pro-Israel' – UK politics live	The Guardian	The Guardian	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
33962	Are SNP and Scottish Greens really doing worse than Tories?	Are SNP and Scottish Greens really doing worse than Tories?	The Herald	The Herald	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
33961	UK government under growing pressure to stop selling arms to Israel	UK government under growing pressure to stop selling arms to Israel	CNN	Yahoo News	99	2.65	1.79	1.81	1.42	2.27	1.81	0.00	5.56	0.00	0.00	0.00	7.14	0.00	7.69	0.00

Figure 12.17: SimilarArticlePairs Admin

Select similar entity pair to change

ADD SIMILAR ENTITY PAIR +

Action: --------- Go 0 of 100 selected

SIMILAR ENTITY PAIR
Biden Ii - Biden Himself: 66.66666666666667
Biden Ii - Biden: 76.92307692307692
Rankin - Jack Rankin: 70.58823529411764
Kim Adair - Kimai: 71.42857142857143
Kim Adair - Kim Gavin: 66.66666666666667
Joseph Khoury - Joseph Johnson: 66.66666666666667
Joseph Khoury - Joseph May: 69.56521739130434
Joe Bidenspent - Joe Biden's America: 66.66666666666667
Joe Bidenspent - Joseph Biden: 69.23076923076923

Figure 12.18: SimilarEntityPair Admin

Figure 12.19: Subscription Admin Object

12.7.3 Lexical Statistics Duplicates Results

Table 12.1: Measures of Text Diversity Applied to pairs of duplicates and pairs of different articles

Measure	Avg Similar % Diff - Avg Unrelated % Diff	Description
Words	57.5	Word count
Terms	51.6	Unique words
VOC-D	25.6	Vocabulary Diversity
Yule-K	25.1	Measure of text diversity focusing on word frequency distribution
Simpson-D	25.0	Measure of diversity using number of occurrences of each word
Herdan-VM	22.5	Lexical variety measure
RTTR	20.7	Standardised measure of lexical diversity (length less impact)
CTTR	20.7	Corrected TTR, adjusted type-token ratio (length compensated)
MTLD	17.1	Lexical richness via text length and distribution of different words
Yule-I	10.9	Variant of Yule-K
Dugast	8.6	Frequency of single occurring words versus all words
Maas	6.4	Lexical diversity less affected by text length
HDD	3.6	Probability of encountering new words in a text of given size
MATTR	3.4	Moving average lexical richness
TTR	0	Number of unique words / total number of words
Herdan	0	Lexical richness less sensitive to text length
MSTTR	0	Lexical diversity calculated over segments of text

12.8 Testing

12.8.1 Automated Tests

- `nlp_processor`
- `profiles_app`
- `accounts`

12.8.2 Functional Tests

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
1	Visit `ProminentProfiles.com`	Home	Home View Shown	Pass	
2	Visit `www.ProminentProfiles.com`	Home	Home View Shown	Pass	
3	Home view loads correctly	Home	Home view containers shown	Pass	
4	Header loads correctly	Home	Header shown at top of page	Fail - Links overflow (Mobile Only)	Introduce hamburger and compact logo
5	Hamburger menu loads on narrow window	Home	Hamburger menu replaces view links when screen width is reduced	Pass	
6	Compact logo loads on narrow window	Home	Logo text removed when screen width is reduced	Pass	
7	Profile Dropdown menu loads correctly	Home	Dropdown menu is present and populated	Pass	
8	Shuffle button loads correctly	Home	Shuffle button is present	Pass	
9	Trending profiles cards load correctly	Home	Name, photo and article headlines shown	Fail - Parent Containers (Mobile Only)	Reduce minimum CSS width to px that will fit typical mobile resolutions
10	Trending profile Cards animated	Home	Articles swipe left to right every 5 seconds	Pass	
11	Dropdown item selected	Any to Entity	User is redirected to chosen Profile	Pass	
12	Shuffle button clicked	Any to Entity	User is redirected to random Profile	Pass	
13	Trending profile card profile name clicked	Home View to Entity	User is redirected to cards Profile	Pass	
14	Trending profile card profile photo clicked	Home View to Entity	User is redirected to cards Profile	Pass	
15	Trending profile card article headline clicked	Home View to Article	User is redirected to Article w/ Card Profile in the Spotlight	Pass	
16	‘About’clicked in header	Any to About	User is redirected to About View	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
17	'Login'clicked in header	Any to Login	User is redirected to Login View	Pass	
18	Footer loads correctly	Any	Prominent Profiles [YEAR] shown at bottom of app	Pass	
19	Unathenticated user clicks Sub Button	Entity to Login	User is redirected to Login View	Pass	
20	Authenticated user clicks Sub Button - not subscribed	Entity	Subscription icon changes from + to a tick	Pass	
21	Authenticated user clicks Sub Button - subscribed already	Entity	Subscription icon changes from a tick to a +	Pass	
22	Profile photo loads correctly	Entity	Profile Photo shown and populated	Pass	
23	Profile description loads correctly	Entity	Profile Description shown and populated	Pass	
24	Profile photo is attributed appropriately	Entity	CREDIT popup below photo link present	Pass	
25	Profile description is attributed appropriately	Entity	Source and date shown in bottom right of description container	Pass	
26	Sort Toggles load correctly	Entity	4 icons and two text buttons shown	Pass	
27	Time sort functions correctly	Entity	Article Entries shown are ordered by published date	Pass	
28	Sentiment sort functions correctly	Entity	Article Entries shown are ordered by the majority sentiment category % value	Pass	
29	Sort direction functions correctly	Entity	Article Entries shown have order reversed when clicked	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
30	Reset functions correctly	Entity	All filters removed, results in the presence of all Article Entries	Pass	
31	Recent functions correctly	Entity	Date range in calendar is set to last 7 days	Pass	
32	Calendar initialised correctly	Entity	Date range in calendar is set to last 14 days	Pass	
33	Calendar functions correctly	Entity	Shows on click, date range selection reflected in Article Entries shown	Pass	
34	Article container loads correctly	Entity	Each container shown, scrollable with no overflow	Pass	
35	Article Entry loads correctly	Entity	Each Article Entry is shown, with no overflow	Partial Pass	There are rare instances where certain headlines and unusual screen widths are not caught by protections and cause slight overflow.
36	Article photo in Article Entry loads correctly	Entity	Article photo is shown	Pass	
37	Article headline in Article Entry loads correctly	Entity	Article headline is shown	Pass	
38	Article publisher in Article Entry loads correctly	Entity	Article publisher is shown	Pass	
39	Article publisher in Article Entry adjusts string length correctly	Entity	When page width is adjusted link string length is reduced or expanded ensuring container doesn't overflow	Pass	
40	Article external link loads correctly	Entity	External link icon shown	Pass	
41	Article published date loads correctly	Entity	Shown in format DDth / nd / rd MMM	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
42	Sentiment Bar loads correctly	Entity	Sentiment Bar shown, with percentages visible only where classification width is sufficient	Pass	
43	Maginifying Glass clicked	Entity to Article	User is redirected to Article w/ Entity View Profile in Spotlight	Pass	
44	Article Photo clicked	Entity	Nothing changes, deliberately not interative to minimise mistaps/clicks when scrolling	Pass	
45	Article Headline clicked	Entity	Nothing changes, deliberately not interative to minimise mistaps/clicks when scrolling	Pass	
46	External Link Icon clicked	Entity to Ext.	User is redirected to external Article URL	Pass	
47	Article Metadata is loaded correctly	Entity/Home to Article	Article photo, author and publisher link shown.	Pass	
48	Profile spotlight is loaded correctly	Entity/Home to Article	Profile shown matches the profile interacted with to reach this page	Fail - Fixed	Previous Profile from a different Article View visit remained. Add Vue watch condition that triggers recreation of donut chart data.
49	Other Profiles Present is loaded correctly	Entity/Home to Article	Where an Article is known to have other overallSentiment records, those entities should be shown here	Fail - Fixed	Previous Other Profile from a different Article View visit remained. Add Vue watch condition that triggers refresh of ProfileEntry components.

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
50	Other Profiles Present for app visible entities is loaded correctly	Article to Entity	When an Entity is visible on the site, their name, photo and a link to their internal entity view is shown	Pass	
51	Other Profiles Present for non app visible entities is loaded correctly	Article to Entity	When an Entity is not visible on the site, their name, and a link to google them is shown	Pass	
52	External Link Icon clicked	Article to Ext.	User is redirected to external Article URL	Pass	
53	Browser back button clicked	Article to Entity	Content quickly reappears as SPA is in use the view should be as the user left it.	Fail - Fixed	Content needlessly refreshed - Update Vue watch condition so that if entity ID remained same on mount of Entity View components no API call is made.
54	Profile Spotlight name is clicked	Article to Entity	User is redirected to spotlights Profile	Pass	
55	Profile Spotlight photo is clicked	Article to Entity	User is redirected to spotlights Profile	Pass	
56	Profile Spotlight entity type hint loaded correctly	Entity/Home to Article	Where avaliable in BingEntity the entity type hint e.g. British politician is shown	Pass	
57	Welcome greeting is loaded correctly	Any to Login	Text to introduce login page shown	Pass	
58	Invalid email format is inputted	Login	Invalid email has been entered message shown below	Pass	
59	Invalid UK mobile number is inputted	Login	Invalid number has been entered message shown below	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
60	Mobile number that does not start +44 inputted	Login	'+44" requirement message shown below	Pass	
61	Password that does not meet requirements inputted	Login	Password criteria is shown	Pass	
62	Login Clicked - Valid credentials	Login to Dashboard	User is redirected to Dashboard View	Pass	
63	Login Clicked - Invalid credentials	Login to Dashboard	Appropriate error message shown	Fail	An error message is shown but it is generic.
64	Login Clicked - Invalid credentials	Login to Dashboard	Forgot Password box shown	Pass	
65	Login with validation messages showing prevents API call	Login to Dashboard	Console log shows no failure due to Vue prevention of request	Fail	Requests still permitted
66	Invalid email format is inputted	Forgot Password	Console log shows no failure due to Vue prevention of request	Fail	Requests still permitted
67	Invalid email format is inputted	Forgot Password	Invalid email has been entered message shown below	Fail	Not implemented yet
68	Submit clicked - Valid email	Forgot Password	Confirmation message shown	Pass	
69	Submit clicked - Invalid email	Forgot Password	Error message shown	Pass	
70	Sign up button clicked	Login to Register	User is redirected to Register View	Pass	
71	Too short first name inputted	Register	First name criteria message shown below	Pass	
72	Too short last name inputted	Register	Last name criteria message shown below	Pass	
73	Invalid email format is inputted	Register	Invalid email has been entered message shown below	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
74	United Kingdom selected	Register	Mobile Number form appears with +44 pre-populated	Pass	
75	Calendar dates load correctly	Register	DOB input prevents dates that would make the user under 18	Pass	
76	Password that does not meet requirements inputted	Register	Password criteria is shown	Pass	
77	Confirm password doesn't match password input	Register	Password must match message shown below	Pass	
78	Too short first name inputted	Register to Login	Prevents API call when Sign Up pressed	Pass	
79	Too short last name inputted	Register to Login	Prevents API call when Sign Up pressed	Pass	
80	Invalid email format is inputted	Register to Login	Prevents API call when Sign Up pressed	Pass	
81	No location is selected	Register to Login	Prevents API call when Sign Up pressed	Pass	
82	No DOB is provided	Register to Login	Prevents API call when Sign Up pressed	Pass	
83	Password that does not meet requirements inputted	Register to Login	Prevents API call when Sign Up pressed	Pass	
84	Privacy policy not accepted	Register to Login	Prevents API call when Sign Up pressed	Pass	
85	All registration fields show no error messages, location, DOB, privacy policy checkbox selected, sign up clicked	Register to Login	Creates user account, redirected to Sign Up page	Pass	
86	Dashboard View loads correctly	Dashboard	Dashboard welcome message and container shown	Pass	
87	Personal welcome greeting loads correctly	Dashboard	The users first name is shown Hi [NAME]!	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
88	Correct subscription cards load in	Dashboard	A card is generated for each profile the user is subscribed to	Pass	
89	SubProfileCard photo is displayed correctly	Dashboard	Correct Profile's photo is shown leftmost in the card, without overflow	Pass	
90	SubProfileCard name is displayed correctly	Dashboard	Correct Profile's name is shown centerd in the card, without overflow	Pass*	Provided BingEntity names are ensured to be under 30 characters, e.g. Catherine, 'Princess of Wales" caused overflow
91	SubProfileCard Sub Btn is displayed correctly	Dashboard	Subscription button with a tick shown (as all entities are in sub dashboard by definition are sub'd to)	Pass	
92	SubProfileCard icons are displayed correctly	Dashboard	Three equally distanced font awesome sentiment icons are shown.	Pass	
93	SubProfileCard photo clicked	Dashboard	User is redirected to Entity View and content is filtered by their last visit	Pass - Automated Test Caught Flaw in API	
94	SubProfileCard title clicked	Dashboard	User is redirected to Entity View and content is filtered by their last visit	Pass - Automated Test Caught Flaw in API	
95	SubProfileCard article counts populate correctly	Dashboard	Counts show sentiment classifications of SubCard Profile in articles since their last visit	Pass	
96	SubProfileCard News Ticker functions correctly	Dashboard	Ticker shows articles since user's last visit scrolling left to right	Pass	
97	Subscription button clicked, visit dashboard again.	Dahsbaord to Dashbaord	SubProfileCard for that profile should no longer appear, user has unsubbed	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
98	Visit `https://www.prominentprofiles.com/admin/`	Admin Home	Page is accessible and loads with anticpated model views visible	Pass	
99	Visit `https://www.prominentprofiles.com/admin/profiles_app/entity/merge_review/app_visible/`	Merge Review Portal	Page is accessible and loads with Django administration banner and home link	Pass	
100	Fuzzy Merge Candidates shown correctly	Merge Review Portal	Each fuzzy merge candidate is shown with a check box, entity name and view link	Pass	
101	Fuzzy Merge candidates view link is clicked	Merge Review Portal	Clicking view on a fuzzy merge candidate takes you to their django admin record view		
102	Merge table title entity is shown correctly	Merge Review Portal	Top of table shows entity name	Pass	
103	Merge table title entity is clicked	Merge Review Portal	Clicking view on a table title entity takes you to their django admin record view	Pass	
104	One candiate selected and merge clicked	Merge Review Portal	Entity merged correctly with table title entity, all model references updated, and EntityHistory record created	Pass	
105	At least 3 candidates selected and merge clicked	Merge Review Portal	As above but for multiple entities.	Pass	
106	One candidate selected and Ignore clicked	Merge Review Portal	IgnoreEntitySimilarity created between table title entity and the selected entity	Pass	
107	At least 3 candidates selected and Ignore clicked	Merge Review Portal	As above but for multiple entities.	Pass	

ID	Test Case	Journey (from / to view)	Expected Output	Pass/Fail	Fix
108	Entities loaded in merge review portal tables correctly	Merge Review Portal	Pairs that exist in the model IgnoreEntitySimilarity are not shown.	Pass	
109	Admin deletes an article record	Article Admin Page	BoundMentions, OverallSentiments associated with article are deleted	Pass	
110	Admin deletes an article record	Article Admin Page	Article if present in dropdown selector is no longer visible	Pass	
111	Article sets an entity record parameter app_visible to TRUE	Article Entity Page	Entity is present in dropdown selector in app header	Pass	
112	Article sets an entity record parameter app_visible to FALSE	Article Entity Page	Entity is not present im dropdown selector in app header	Pass	
113	Article sets an entity record parameter app_visible to FALSE	Article Entity Page	Entity View is not accessible for this entity anymore in app if user knows URL	Fail	Still accessible - fix is simple by adding condition to OverallSentiment view
114	Article sets an entity record parameter app_visible to FALSE	Article Entity Page	Entity View is not accessible for this entity anymore in app if user is subscribed	Fail	Still accessible - fix is simple by adding condition to OverallSentiment view
115	An entity admin view loads in line overallSentiments correctly	Article Entity Page	All an entities overallSentiments are rendered correctly with links to their django admin record views.	Pass	

12.9 Evaluation

12.9.1 SimilarArticlePairs

Extract of SimilarArticlePairs meeting the filter requirements downloaded from the server.

12.9.2 Requirements Acceptance Testing

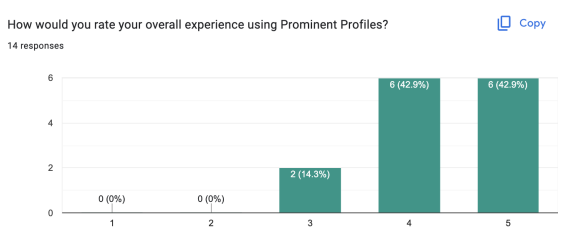
Please see Requirements_Acceptance_Testing.xlsx.

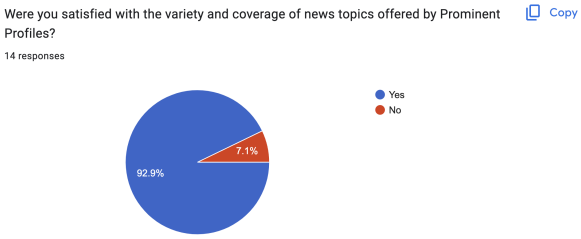
12.9.3 User Study: Form Responses

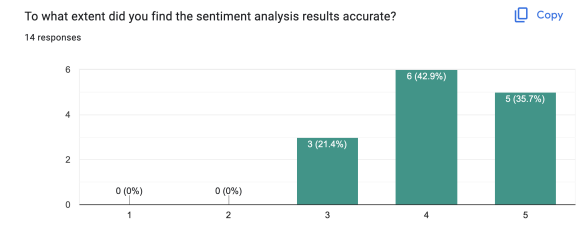
All Responses Link

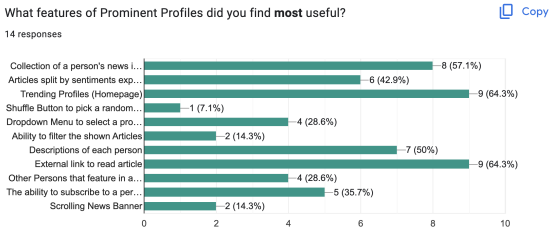
Removed Inconsistent Response

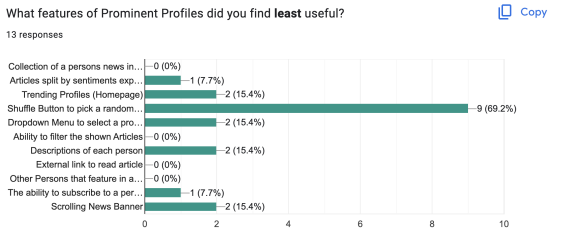
Evaluated Responses CSV

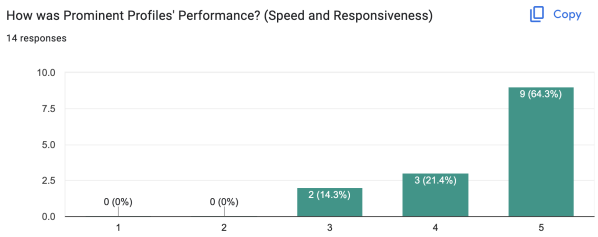






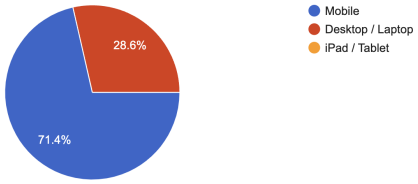






What device type did you use to access the site?

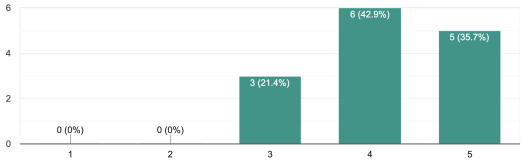
14 responses



To what extent is the Prominent Profiles UI is easy to navigate?

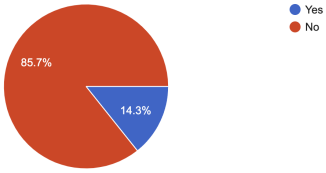
Copy

14 responses



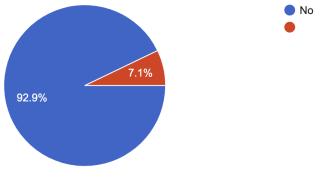
Did you make an account? (be honest - I understand it is time consuming and that's cool - it is just to segment responses!)

14 responses



Have you seen an app that delivers news in the same way as Prominent Profiles? (If so, enter name in box)

14 responses



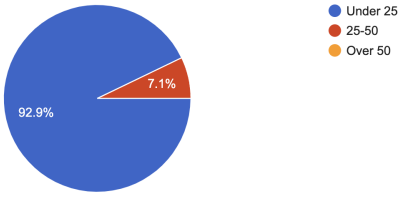
From the following select the most desirable feature you'd like to see added...

Copy

14 responses

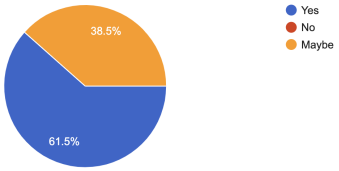
Please select your age bracket

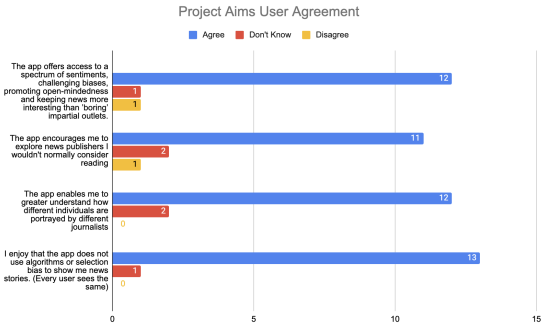
14 responses



If you have been reading particularly negative news stories around a person, would you now visit Prominent Profiles to obtain a variety of coverage?

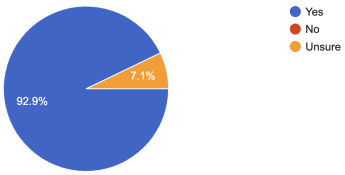
13 responses





The app's inclusion of a Cookie Consent Banner, Privacy Policy, HTTPs, and About Page make it appear more legitimate?

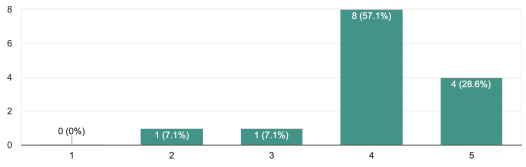
14 responses



Finally based on your experience how likely are you to recommend this app to others?

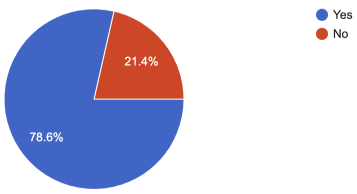
Copy

14 responses



Are you generally interested and engaged with the news and current affairs?

14 responses



Can you briefly describe your overall impression of the app?
The app is easy to navigate, the idea of it is also very helpful for those interested. I like that over a period of time, you'll be able to easily see what you have missed since last logging in too.
Very polished app, unique idea
Super easy to navigate, doesn't require much explanation/induction to start using.
Clean UI and UX, unique enough for it to actually be desired to be used
Fantastic idea with need of work to make the site a bit clearer
Clean, minimalist, easy to understand
Very nice interface, it's clear and to the point and a very good idea
Very clean and professional, relatively easy to navigate
I like the general goal of the app and its formatting, however I find the approach to solve the goal quite unsuccessful
Took a couple of minutes to understand the structure and purpose of the web app but once that became clear the app was really interesting, useful, and intuitive
Great concept, the visuals could be improved to make it look more professional
The website itself has good utility. Would be difficult to understand the website's purpose without the verbal explanation
I recognised most people on the app but I found the UI a little frustrating
Really insightful and easy to navigate.

Table 12.3: Did you encounter any challenges or confusion while navigating Prominent Profiles?

Responses
No
no
None
I don't think it's clear what the difference between the 'clock icon' and 'recent' is. Similarly, I don't think it's obvious you would need to press 'reset' to get older news when on someone's profile. This is especially the case for Paula Vennells who has a blank profile due to no current articles. Maybe a separate 'view older articles'/'later dates' would be better.
While the use of icons is great for allowing any type of user to interact with the app. I think they should also have some text associated to them just to clarify what they do. For instance, the up/down arrow having a label called "sort".
wasn't immediately sure that blue meant neutral sentiment
No serious problems however slight overwhelm of things to interact with all on one page sometimes. (Not major at all)
Again, took a minute to understand the layout and purpose but this was short lived
I thought the red and green arrows were clickable
No I found it straight forward and clear

What do you like about Prominent Profiles?
I like that it gives well-rounded responses on each individual. It's also helpful to see how different articles/perspectives shape these individuals too.
Seems to be unique with lots of useful features to ensure people are getting unbiased news, or are at least aware of the bias of the content they are viewing.
It is very relevant to the news today. More and more emphasis is put on how biased news outlets/articles are, and people can make an informed choice of what they read.
Moves away from traditional perception of left and right-wing ideology and focuses on the overall sentiment.
The goal.
Showcasing the current affairs and political landscape.
I like the idea of making users aware of biases and allowing them to know what to expect from a given article.
Objective coverage of each subject, there's no way of telling the political persuasion of the creator (even though I know what you are *eyes emoji*), very unbiased.
The collection of the information formatted in this way allowing short-form political content and further reading for people who want more.
The ability to view someone's public perception using NLP and unbiased analysis.
It's a new way to look and view prominent individuals in a new format.
Quick and responsive website. Content load times for pages were very good.
I like that there's a good range of people to choose from.
A useful tool to assist with following political people and news with an unbiased view formed from this great approach.

Are there any specific features or functionalities you would like to see in future updates?
A feature showing individual standpoints on big debates.
Maybe the addition of a browser plugin? This could tell you the sentiment of any article you are reading and link to the various sentiments expressed by other outlets.
I think a general summary (i.e., in the form of a pie chart or graph) of how someone's representation in media has changed over time would be effective. Especially, for example, if 'negative breaking news' *Prince Andrew Paula Vennells* had just occurred, you'd be able to see the rise of unfavourable articles over time.
Not necessarily a feature, but as someone who likes to know how things work. Updating the about page to either include more details on how sentiment analysis works or a link to a page describing the methods used. As a common issue when using apps that use ML is that it's considered "magic" and having a place to simplify and break down the "magic" would be nice to have for people who may be sceptical of consuming AI analysed content.
Yes. Voting history in parliament and policies they push. Articles to be separated into a fourth section not based on sentiment but about specific mentions of policy.
No.
A way to see how each news source portrays different political figures over time e.g perhaps over the last 14 days the mirror has been more positive about keith starmer etc.
None I can think of.
The only thing which could be improved would be the aesthetics, perhaps with a graphic designer.
Improved user onboarding when entering the site.

Why did you give this score?
I study Politics and feel that this would be a great resource, it offers a lot on just one platform that is helpful when researching.
Really cool, unique app
The app forces people to be exposed to media they can shelter themselves from, and can stops people living in 'echo chambers'. Good for politics nerds like me to gain well balanced arguments from both sides.
Clean UI and UX, really interesting use of sentiment analysis and actually a good, neutral use of AI
it needs more work, but the idea is brilliant. i would be concerned about any bias as i know ur right wing and i am left wing so i have natural scepticism. this is why i like the idea of having their voting record and policies available
I think it could be a bit niche
I think its a very useful idea to givemore genuine information
maybe more profiles needed, also i have no friends who care enough about politics to use it !
I don't know many people who like politics around me
The app is well made and professional looking with a really unique purpose
in general, the role and function of the app is useful. I value the fact each user is shown the same articles (everyone sees the same thing)
I was really impressed with the structure of the website overall and the features it offered. It is a refreshing approach to a world dominated with algorithms where people may struggle to form opinions. Very much needed - well done *clap hands emoji*