# Decoding Democracy: Exploring AI-Driven Insights for Participatory Democracy

University of
St Andrews

Bruno Ceccolini
210001111

School of Computer Science
University of St Andrews
March 22, 2024

# Abstract

Participatory democracy has been strongly influenced by the vast amounts of data generated from public consultations in recent years. "Decoding Democracy" is a project that investigates the application of natural language processing and machine learning techniques to analyze public consultation responses within the context of democratic participation. By leveraging citizen responses data from three Scottish consultations [14] [15] [21], the project employs various techniques such as sentiment analysis and topic modelling to gauge public sentiment on commonly discussed issues. Additionally, the project explores the potential of semantic similarity for identifying similar opinions. These insights contribute to the field of participatory democracy by offering a data-driven method to understand the complexities of public opinion, potentially improving future policy making and consultation processes.

## Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is NN,NNN* words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

## Acknowledgements

Completing this project would not have been possible without the help of my supervisor, Dr Rosa Filgueira, who guided me and provided me with invaluable advice and resources throughout the year. I would like to thank her for making the process more enjoyable and insightful, as well as using her expertise to give me a clear direction for where this project could be taken.

I would also like to thank my father, Andrea Ceccolini, and mother, Leena Ceccolini. Their support has been a driving factor throughout my studies, and I will be forever grateful for the sacrifices they have made to give me the opportunity to attend this outsanding University.

# Contents


**1 Introduction** **7**
- 1.1 Problem Statement . . . 7
- 1.2 Selected Consultations . . . 7
  - 1.2.1 Draft Referendum Bill . . . 8
  - 1.2.2 First Electoral Reform (2018) . . . 9
  - 1.2.3 Second Electoral Reform (2023) . . . 10
- 1.3 Objectives . . . 11
  - 1.3.1 Primary Objectives . . . 11
  - 1.3.2 Secondary Objectives . . . 12
- 1.4 Contributions . . . 12
- 1.5 Report Structure . . . 12
- 1.6 Terminology . . . 14

**2 Context Survey** **15**
- 2.1 Public Participation in Policy Decisions . . . 15
- 2.2 Artificial Intelligence Applications . . . 15
- 2.3 Large Language Models . . . 16
- 2.4 Transfer Learning . . . 17
- 2.5 Previous Work On Decoding Democracy . . . 17

**3 Requirements Specification** **19**
- 3.1 Consultation Requirements . . . 19
- 3.2 Database Framework Requirements . . . 19
- 3.3 Deep Learning Analysis Requirements . . . 20
- 3.4 Web Tool Development . . . 21

**4 Software Engineering Process** **22**
- 4.1 Supervisor Meetings . . . 22
- 4.2 Work Approach . . . 22

**5 Ethics** **23**

**6 Design** **24**
- 6.1 Project Pipeline . . . 24
  - 6.1.1 Data Collection . . . 25
  - 6.1.2 Exploratory Data Analysis . . . 26
  - 6.1.3 Database Creation . . . 26
  - 6.1.4 Data preprocessing and Text representation . . . 26
  - 6.1.5 Model Selection . . . 26
  - 6.1.6 Deep Learning Analysis . . . 26
  - 6.1.7 Web Dashboard Development . . . 26
  - 6.1.8 Reporting Results . . . 27
- 6.2 Database Design . . . 27
- 6.3 System Architecture . . . 30

6.3.1 Jupyter Notebook and Google Colab . . . 30
6.3.2 Data Flow . . . 30
6.3.3 Client . . . 31
6.3.4 Server . . . 32
6.4 User Interface Design . . . 34
6.4.1 Usability and Responsiveness . . . 34

**7 Implementation** **35**
7.1 Text Preprocessing . . . 35
7.2 Model Selection for Sentiment Analysis . . . 37
7.3 Topic Modelling . . . 40
7.3.1 Clustering Techniques . . . 40
7.3.2 Agglomerative clustering . . . 41
7.3.3 Latent Dirichlet allocation . . . 42
7.3.4 Combining methods for optimal topic modelling . . . 45
7.4 Semantic Response Similarity . . . 46
7.4.1 Bi-Encoders vs Cross-Encoders . . . 47
7.5 Web Dashboard Frontend Implementation . . . 50
7.5.1 Codebase Structure . . . 51
7.5.2 User Interface . . . 51
7.5.3 Server Communication . . . 52
7.6 Backend Implementation . . . 52
7.6.1 Structure . . . 53
7.6.2 Serving AI Analysis Results . . . 53
7.7 Testing . . . 54

**8 Evaluation and Critical Appraisal** **55**
8.1 Government Analysis Comparison . . . 55
8.1.1 Draft Referendum Bill Analysis . . . 55
8.1.2 Electoral Reforms Analysis . . . 59
8.2 Database Ontology And SQLite . . . 62
8.3 Sentiment Analysis . . . 63
8.4 Topic Modelling . . . 63
8.5 Semantic Response Similarity . . . 65
8.6 Web Dashboard . . . 66
8.7 Requirements Validation . . . 67
8.7.1 Database Requirements . . . 68
8.7.2 Deep Learning Analysis Requirements . . . 69
8.7.3 Web Tool Requirements . . . 71

**9 Conclusions** **73**
9.1 Key achievements . . . 73
9.2 Drawbacks and Future Work . . . 74
9.3 Reflection . . . 74

**A Testing Summary** **76**

**B User Manual** **79**
B.1 Responsiveness of the website - using a mobile phone . . . . . 88

**C Other Appendices** **93**
C.1 Python packages . . . . . 93
C.2 Ethical Approval Document . . . . . 94

# 1 Introduction

## 1.1 Problem Statement

Public consultations are vital to Scotland's democratic process, as they provide citizens with a platform to express their opinions on policies and issues that directly affect their lives. However, the traditional methods of manually analysing large volumes of consultation responses can be time-consuming and subjective, and often will not capture the full spectrum of topics or public sentiment within citizen feedback.

This project addresses these challenges by using various natural language processing techniques to analyse public consultation responses in Scotland. By employing sentiment analysis, we aim to evaluate the emotions expressed by citizens on various aspects of policy and voting proposals. This can offer policymakers a data-driven understanding of the public mood and highlight areas of consensus or disagreement. Furthermore, through topic modelling, we seek to uncover common themes and concerns within discussions. We will delve into the semantic context and relationships between ideas expressed by citizens, in order to offer a comprehensive overview of the key issues that resonate with the Scottish public.

The combination of natural language processing techniques promises a more in-depth understanding of public consultations in Scotland, which could enable policymakers to uncover valuable insights on the issues that truly matter to the citizens, without having to tediously read through the thousands of responses manually.

## 1.2 Selected Consultations

A consultation typically involves the government seeking input from the public on proposed policies or decisions. The government will usually ask a combination of multiple choice questions and open-ended questions. Additionally, there will often be some prerequisite materials, such as background information or policy documents, to help participants understand the context and make informed contributions. The aim is to engage citizens and organisations in the decision-making process, ensuring that diverse perspectives are considered before finalizing policies or actions.

The Decoding Democracy project is concerned with three consultations, chosen by Dr Tarik Olcay and Dr Rosa Filgueira. This section will cover a brief background of each consultation, to give some context into what issues the questions and responses are addressing.

| Consultation | Multiple Choice Questions | Open-ended Questions | Individuals Responses | Organisation Responses |
|---|---|---|---|---|
| Draft Referendum Bill | 0 | 5 | 7,157 | 41 |
| 2018 Electoral Reform | 22 | 23 | 844 | 67 |
| 2023 Electoral Reform | 30 | 23 | 486 | 31 |

Table 1: Question and response details for about each consultation

Firstly, Table 1 outlines the features of each consultation. The first contains only open-ended questions, whilst the two others contain many multiple choice questions too.

### 1.2.1 Draft Referendum Bill

The following summary of the first consultation was taken from the Scottish Government's page [13]:

> On 20 October 2016, the Scottish Government launched a consultation on a Draft Referendum Bill. The consultation paper set out the Scottish Government's proposals for legislation for a possible referendum on independence for Scotland. The consultation ran for three months, and closed on 11 January 2017.
>
> The consultation paper proposed a number of changes to the procedures followed in the 2014 Scottish independence referendum. Some of these related to legislative changes implemented after 2014, and some were intended to address specific issues raised following the 2014 referendum. The consultation contained five open questions which all took the form: 'What are your views on...?' These addressed:
>
> 1. Arrangements for managing the referendum
> 2. Technical changes to polling and count arrangements (since the 2014 referendum)
> 3. Changes to rules on permitted participants
> 4. Campaign rules and rules on spending
> 5. Changes to the rules on permitted participants' expenses and transactions between qualifying and non-qualifying persons

As an example, the first question, *"What are your views on the proposed arrangements for managing the referendum?"*, received many responses like *Why more uncertainty? We don't know what's going on after brexit. We have just hit a bus. The first minister is trying to throw us under it."*

### 1.2.2 First Electoral Reform (2018)

The next consultation was about reforming Scotland's electoral process. The following summary is again taken from the Government's analysis page [16]:

> In late 2017, the Scottish Government carried out a public consultation to explore options for reforming and modernising electoral processes in Scotland. The consultation contained 25 questions. It discussed and sought views on:
>
> 1. How often elections should be held (Questions 1 and 2)
> 2. Who runs elections and how they are run (Questions 3 to 16)
> 3. Who can register and vote in elections in Scotland (Questions 17 to 22)
> 4. Ways of improving the accessibility of voting and elected office (Questions 23 to 25)
>
> The consultation ran from 19 December 2017 to 29 March 2018. The consultation contained a mix of closed (tick-box) and open questions. Thus, both quantitative and qualitative analysis was undertaken.

Many of the questions in this consultation took the form: "Do you agree that ...?", and gave multiple choice options of "Yes" and "No". Another type of multiple-choice question was "Do you think the term length for the Scottish Parliament and local government should be ...?", and gave the following options:

1. 5 years
2. 4 years
3. Other length (please specify)

The majority of people opted for five years, which actually led to the term length being changed from four years following the consultation.

With most of the questions giving multiple choice options, the open-ended questions usually followed on from a multiple choice one, and asked something like *"Do you have any other comments on this issue?"*. For instance, the third question followed on by asking *"Do you have any other comments or suggestions on term lengths?"*, and a few example responses were:

- "Elections should occur in years where there are no other elections scheduled so that each candidate can have a fair chance of Campaigning on the issues they will stand up for/against if elected to Holyrood/Council."

- "Longer term may reduce voter fatigue and overlap with other elections. Should we leave the UK and EU (thus removing the need for elections at those levels) it may make sense to reconsider this."
- "We are living in an era of rapid economic, political and social change. It is important that the electorate have regular opportunities to vote for SP & LG representatives. Politicians should not enjoy a greater degree of job security than the average elector/employee! Three years would concentrate the minds of politicians."
- "5 years seems fair."
- "No."

As seen by the variety of responses, some elaborate more than others, and many responders skip these open-ended questions entirely.

#### 1.2.3 Second Electoral Reform (2023)

The third and final consultation was concerned with increasing voter turnout and encouraging people to stand for election in Scotland [22]. This consultation followed up on the previous consultation in 2017/2018, which had led to significant changes relating to Scottish Parliament and Scottish Local Government elections (together referred to as 'devolved Scottish elections'). Devolved elections had moved from a four to a five-year term, and significant changes had been made in relation to roles of the Electoral Commission, the Electoral Management Board for Scotland (the "EMB"), and Boundaries Scotland. Also, in 2021, legislation was passed to ensure that year's Scottish Parliament election could be safely held in the context of the Covid-19 pandemic.

Looking at this new consultation, some of the key areas they were looking for input on include:

1. Whether 16 and 17 year olds should be allowed to stand for election
2. How to increase voter registration and participation among under-represented groups, including young people and foreign nationals
3. How to make voting more accessible
4. How to improve the electoral administration system
5. How to handle scheduling elections in exceptional circumstances
6. The consultation paper also considers questions arising from the UK Elections Act 2022.

This consultation was also composed largely of multiple choice questions. For instance, the first question was *"Do you think that 16- and 17-year-olds should be able to stand for election in:"* and the options were:

1. Neither Scottish Parliament nor Local Government elections
2. Both Scottish Parliament and Local Government elections
3. Scottish Parliament elections only
4. Local Government elections only

The vast majority of people answered the first option "Neither Scottish Parliament nor Local Government elections", suggesting there is a strong consensus that 16 and 17 year olds are too young to vote.

## 1.3 Objectives

The main focus of "Decoding Democracy" is to extract insights from each consultation using deep learning techniques. After a comprehensive analysis is conducted, a secondary objective is to construct a web dashboard to visualise the results, however the analysis itself was the most important part.

The objectives of the project were as follows:

### 1.3.1 Primary Objectives

1. Creating a new database framework for storing the consultations and responses
    - 1.1 Develop a robust Entity-Relationship (ER) model or an ontology to capture crucial information about consultations and public responses.
    - 1.2 Establish a structured foundation to efficiently manage and analyse diverse public input.
    - 1.3 Store responses from three distinct consultations for comprehensive insight.
2. Apply deep transfer learning analysis and topic modelling to the consultation responses.
    - 2.1 Master AI methodologies like sentiment analysis and topic modelling to extract valuable insights from citizens' input.
    - 2.2 Build upon existing research using trained deep learning models for enhanced comprehension.
    - 2.3 Explore alternative deep learning models for potential improvement.
    - 2.4 Compare the analyses obtained with the deep learning results with the reports published by the Scottish Government in which they have analyzed those consultations.

### 1.3.2 Secondary Objectives

3 Craft a User-Friendly Web Tool

3.1 Design an intuitive web dashboard serving as the front-end for visualising the consultations, their responses and the AI analysis, automatically delivering insights from selected consultations.

## 1.4 Contributions

The final product is a comprehensive web dashboard allowing users to select any of the three consultations outlined in 1.2, and analyse all the responses to each question, by using the results obtained from the natural language processing techniques used in the initial stages of the project.

1. Primary Objective 1.1 was met and the database design is detailed in Section 6.2.
2. Primary Objective 1.2 was met and is outlined in Section 6.2.
3. Primary Objective 1.3 was met and is discussed in Chapter 6.
4. Secondary Objective 2.1 was met and is discussed in sections 7.2 and 7.3.
5. Secondary Objective 2.2 was met and is discussed in Chapter 7.
6. Secondary Objective 2.3 was met and is assessed in Section 7.2.
7. Secondary Objective 2.4 was met and is evaluated in Section 8.1.
8. Tertiary Objective 3.1 was met and is discussed in Chapter 7.

## 1.5 Report Structure

This report outlines the software development processes for the Decoding Democracy project, and is structured as follows:

- **Chapter 2** explores the background of the project in detail, by looking at similar research involving public consultations and the AI techniques used to analyse them.
- **Chapter 3** outlines the requirements for the Decoding Democracy project.
- **Chapter 4** discusses the software engineering approach that was taken.
- **Chapter 5** covers the ethical considerations that were made throughout the project.
- **Chapter 6** discusses the design and structure of the various components, including the database, the jupyter notebook, and the web dashboard.

- **Chapter 7** delves into the implementation of these components, by going into more detail on certain features and interesting methodologies that were used.
- **Chapter 8** evaluates the effectiveness of the web tool that was made, with respect to the requirements that are outlined in Section 3 and the government analysis that was conducted on the consultations we are concerned with (see Section 1.2).
- **Chapter 9** summarises the work done and discusses possible future work.

## 1.6 Terminology

The following table outlines some of the key technical terminology that will be used throughout the report.

| **Term** | **Definition** |
|---|---|
| Natural Language Processing (NLP) | Machine Learning techniques that enable computers to interpret and comprehend human language. |
| Sentiment Analysis | The use of natural language processing techniques to categorise responses as either positive or negative. |
| Topic Modelling | A type of statistical modelling that uses unsupervised Machine Learning to identify clusters or groups of similar words within a body of text. [**topicmodellingdef**] |
| Semantic Similarity | A metric defined over a set of documents or terms, where the idea of distance between items is based on the likeness of their meaning or semantic content as opposed to lexicographical similarity. [63] |
| Transfer Learning | A machine learning technique where a model trained on one task is adapted to solve a different but related task. In our case, pre-trained language models are repurposed to be used on consultation responses. |
| Deep Learning | A subset of machine learning methods based on artificial neural networks with representation learning. The adjective "deep" refers to the use of multiple layers in the network. Methods used can be either supervised, semi-supervised or unsupervised. [18] |
| Semantic Similarity | A measure of the degree to which two pieces of text are similar in meaning, regarding the underlying concepts and ideas conveyed by the words rather than simple string matching. |
| Text Embeddings | A numerical representation of text data, where words, phrases, or documents are mapped to high-dimensional vectors of real numbers. The goal is to capture the semantic and syntactic meaning of the text in a dense, fixed-length vector format that can be easily processed by machine learning algorithms. |

Table 2: Definitions for the technical terms used throughout the project

# 2 Context Survey

Public consultations usually involve the handling of huge amounts of data submitted by citizens and organisations - not an easy process to interpret with a manual analysis. This section will explore increasing use of natural language processing tools to automate analysis for large volumes of text. In particular, it will explore research involving language models and public consultations. Large language models will also be discussed, with it becoming a trending topic in recent years.

## 2.1 Public Participation in Policy Decisions

The paper "Government and Public Participation in the United Kingdom" by Barnes, et al., [2] explores the history and evolution of public participation in policy decisions in the UK. Public participation in policy-making has a long history in the UK, with one of the earliest examples being the establishment of Royal Commissions in the 1830s. However, it was not until the latter half of the 20th century that public participation became more institutionalized and formalized in the UK.

The paper highlights several key developments that facilitated greater public participation in policy decisions, including the the Wheare Report in 1949, which recommended the facilitation of public participation in decision-making processes related to public services and local government. It also highlights the Growth of Environmental Activism in the 1970s and 1980s which led to increased public participation in environmental policy decisions, and he Modernizing Government Agenda in 1997, which placed emphasis on public participation as a means of improving the transparency and accountability of government decision-making processes.

History has shown that public consultations can improve decision quality whilst providing a means of informing the public of how these decisions will affect their lives. Another paper titled "Putting More Public in Policy Analysis" [60] highlights how important public consultations are, as citizens can provide valuable guidance to policy makers about the direction of public policy through their experiences and values.

## 2.2 Artificial Intelligence Applications

Applying AI techniques like natural language processing (NLP) and topic modelling to analyse public consultation data has significant potential for getting the most out of participatory democracy processes. A relevant study by Weng et al. [62] proposes an "AI Augmented Approach" to identify and visualise shared ideas expressed by citizens during large public consultations. Their method involved extracting linguistic features from consultation responses before categorising them into different themes related to urban infrastructures (e.g. public spaces, transport, buildings, community) using a predefined dictionary. The study also used linguistic cues to differentiate between "shared interests"

and "fixed positions". For instance, the phrase "I want" indicated a fixed position while "I believe" suggests a shared interest. Interactive visualisations were used to effectively communicate the aggregated results to policymakers. This approach was used with a public consultation following New Zealand's 2011 earthquake, and helped analyse of over 100,000 public responses, with the key findings being shared interests in areas like green spaces, better public transport and safer buildings.

Although the authors found a strong cause for using an AI-augmented approach to promote collaborative problem-solving aligned with public values, the paper highlights the challenges with handling incomplete sentences and negative sentiments accurately using NLP, as this led to 26% of the data being excluded from the analysis. To address this, the paper suggests guidelines to encourage complete sentences and differentiate between shared interests and fixed positions to facilitate collaboration. The use of multiple choice questions can also prevent off-topic or incomplete responses.

Another paper by Barbieri et al. [1] addresses the lack of standardised evaluation protocols with NLP processes for large textual data. It presents a suite of seven different Twitter-specific classification tasks, from sentiment analysis to irony detection. The paper discusses the training of RoBERTa [7], a state-of-the-art language model, which presents a useful option in the deep learning part of this project. In the paper, they used several variants of this model:

1. The existing pre-trained RoBERTa-base model (without any additional pretraining)
2. Training RoBERTa from scratch on 60M tweets
3. Starting with the pre-trained RoBERTa-base and continuing pre-training on 60M tweets

Their findings suggest that continuing pre-training an existing language model like RoBERTa on a large Twitter corpus (strategy 3) leads to better performance across most of the tweet classification tasks. Therefore, using the twitter-roberta-base-sentiment model [7] could be the best option when it comes to analysing the large corpus of responses with the Scottish consultations. Overall, this paper acknowledges the importance of a unified evaluation framework for NLP tasks, which aligns with our objective of creating a structured database and analysis pipeline for public consultation responses.

## 2.3 Large Language Models

Following on from the TweetEval paper, we will discuss the relevance of large language models (LLMs) like GPT-3, which have revolutionized the field of natural language processing, and how they differ from language models like RoBERTa. Both are transformer-based language models and are trained on vast amounts of textual data from the internet, allowing them to capture detailed

semantic and contextual information about human language. However, they have different use cases and underlying architectures.

RoBERTa (Robustly Optimized BERT Approach) is an encoder-only model, which means it is designed for tasks that involve processing and understanding input text, such as text classification, sentiment analysis, and named entity recognition. RoBERTa is pre-trained using the masked language modelling objective, where it learns to predict masked-out tokens in a sequence based on the surrounding context.

On the other hand, GPTs (Generative Pre-trained Transformer) like GPT-3 and GPT-4 are decoder-only models, designed for language generation tasks like text completion, summarisation, and open-ended text generation. These models are pre-trained using a different objective called autoregressive language modelling, where they learn to predict the next token in a sequence based on the previous tokens. As well as this GPTs are significantly larger language models, with billions of parameters, while RoBERTa is relatively smaller, with hundreds of millions of parameters.

With the focus of this project being response classification tasks such as sentiment analysis and topic modelling, the RoBERTa model's encoder-based architecture and masked language modelling objective make it a more suitable choice. However, LLMs may also provide a valuable alternative if we want to incorporate text generation capabilities, such as generating summaries or insights from the responses.

## 2.4 Transfer Learning

Fine-tuning language models can be very computationally expensive and usually requires significant computational resources. Additionally, the quality of the fine-tuned model's performance would depend on the availability of high-quality labeled data for the specific task at hand. Due to these challenges, the language models used in this project have not been directly trained on the specific consultation response data used in this project. Instead, their general language understanding capabilities can be leveraged through transfer learning techniques, which involve taking a pre-trained model like RoBERTa and fine-tuning it on a specific task or domain, such as our Scottish consultation responses.

The potential applications of language models in this project include performing sentiment analysis and topic modelling more accurately than traditional machine learning models, as well as retrieving semantically similar responses from the dataset, thus enabling users to explore related viewpoints and opinions more effectively.

## 2.5 Previous Work On Decoding Democracy

Before embarking on this project, my supervisor, Dr Rosa Filgueira, had completed some introductory analysis on the first consultation (the consultation

on the draft referendum bill) [14]. This provided me with a useful starting point, guiding the initial exploratory data analysis and deep learning techniques.

# 3 Requirements Specification

For each project objective, software requirements are outlined and prioritised based on their importance. The following list is captures the properties that the final solution should exhibit. Each requirement is categorised as either functional or non-functional criteria, and is prioritised with either high (H), medium (M), and low (L) labels to indicate their relative importance.

## 3.1 Consultation Requirements

### Non-Functional Requirements

1. All three consultations which are outlined in Section 1.2 must be analysed [14] [15] [21], so means each of the following requirements applies for all three consultations.

## 3.2 Database Framework Requirements

### Functional Requirements

1. Database Schema Requirements (H): The system must support the development of a strong Entity-Relationship (ER) model to encapsulate information related to consultations including the questions, responses and topics.

2. Data Storage (H): Capability to import and store responses from at least three distinct consultations.

3. Data Transformation (H): Mechanisms for data cleaning and transformation during the importation process, to ensure consistency in how the responses and questions are stored.

4. Automated Data Population (H): Automation of the population of database tables from CSV files.

5. Identification of Question Types (M): Functionality to distinguish and categorise multiple-choice questions from open-ended questions for analysis.

### Non-functional Requirements

1. Data Integrity and Relations (H): Enforcing referential integrity through foreign keys to ensure consistency of data.

2. Efficient Data Handling (H): Optimisation for efficient querying and data retrieval to support the large number of responses.

## 3.3 Deep Learning Analysis Requirements

### Functional Requirements

1. Model selection (H): Evaluate the performance of several ML models for sentiment analysis and compare the results with human-annotated sentiment scores before selecting the most effective and accurate model for extracting meaning out of the consultation responses.

2. Text preprocessing (H): Responses should be preprocessed in the same way before being passed into any models to ensure consistent and comparable analysis for any response. This means removing stopwords, lemmatising to reduce words to their base form, and removal of punctuation and emojis to prevent models from interpreting similar responses differently.

3. Sentiment Analysis (H): Apply ML techniques to perform sentiment analysis on consultation responses.

4. Dynamic Topic Discovery (H): Utilise Clustering and LDA (Latent Dirichlet Allocation) to dynamically discover topics within open-ended responses.

5. Embedding Generation (H): Generate and store text embeddings for responses.

6. Topic Labelling (H): The system should label clusters and topics with descriptive keywords based on the content of responses within each group.

7. Database Update Mechanism (H): Implement mechanisms to update the database with sentiment values and topic assignments for each response.

8. Visualisation Support (M): Provide support for data visualisation, including word clouds for topics and scatter plots for cluster visualisation.

### Non-functional Requirements

1. System Accessibility (H): The jupyter notebook should be executable on google colab [26], using the free resources available (i.e. the TPU, GPU or CPU runtimes).

2. Optimized Model Processing (M): Choice of a model that can generate accurate sentiment analysis and topic modelling on a large corpus of responses without taking too much time, so that the system is able to expand and manage more responses in the future.

3. Maintainability (M): The system must be maintainable and supported by documentation.

## 3.4 Web Tool Development

**Functional Requirements**

1. Allow users to select one of the three consultations to analyse. (H)
2. Allow users to see all question from a consultation. (H)
3. Allow users to analyse the distribution of responses for multiple choice questions with a bar chart. (H)
4. Allow users to view all the responses for open ended questions, with search functionality matching specific words. (H)
5. Allow users to search for similar responses to a certain input. (M)
6. Allow users to view the sentiment distribution for each question, with positive and negative bars on a bar chart. (H)
7. Allow users to see what the common topics were for each question, and what the sentiment distribution is for each topic. (H)
8. Allow users to chat with an AI assistant implementing Large Language Model capabilities, in order to get insights from the responses. (L)

**Non-Functional Requirements**

1. Intuitive Interface (H): The interface for the web dashboard must be easy to use, simplifying the process of selecting consultations, viewing analysis results, and understanding insights.
2. Accessibility (H): Ensure the web tool is accessible to non-technical users such as policy makers and the general public, incorporating guidelines for usability and accessibility.
3. Low Latency (M): The website must respond to queries quickly, avoiding long waiting times when searching for responses and viewing analyses.
4. Maintainability (M): The system must be designed to be maintainable so that updates can be published quickly and additional functionalities can be implemented as required.
5. Responsiveness (M): The user interface must be responsive across various screen sizes, such as phones, tablets and desktop computers.
6. Environment compatibility (M): The system should be supported across multiple operating systems, including MacOS and Linux.

# 4 Software Engineering Process

Whilst the development approach taken for this project involved frequent communication with my Supervisor, the majority of the time was spent working on the tasks at hand individually.

## 4.1 Supervisor Meetings

Throughout the year, weekly supervisor meetings were help in Dr Rosa Filgueira's office, which mostly consisted of progress updates and discussion of issues and next steps. These were crucial for the project's evolution, as they ensured the project was worked on incrementally and priorities could be shifted each week. Also, the regularity insured that problems were resolved quickly, especially thanks to Dr Filgueira's expertise and previous work with artificial intelligence projects. Apart from the meetings, Dr Filgueira was available via email, and provided me with useful resources throughout the process.

## 4.2 Work Approach

GitHub was used for version control, and contained the Jupyter Notebook, the raw data for the consultations, and the web dashboard code. The repository was designed for reproducibility, so the Readme file explains how one can reproduce the whole project and run the web tool locally. This remote repository was cloned on the school servers, so that it could be backed up in case it was somehow removed from GitHub.

Notion's desktop application [30] was used to keep track of what tasks had to be completed and any notes from meetings, providing a clean interface to categorise various parts of the project. At every meeting, priorities were discussed and noted down for the following week, which left me to work on them and tick them off throughout the rest of the week. Notion was also used to record any issues I had which were to be discussed in the supervisor meetings.

Overall, the development approach used was flexible and resulted in fast progress, thanks to the detailed list of requirements which were worked on incrementally (see Section 3).

# 5 Ethics

Regarding the ethical considerations for this research, we meticulously adhered to principles ensuring the anonymity and confidentiality of all participants involved. Although all the responses are publicly available on the Scottish Government website [14] [15] [21], the names of participants were removed from the excel files before any analysis or exploration of the data. Therefore, no personal information was collected, aligning with ethical research standards. On top of this, the machine learning models used all have a permissive license [10] [11], allowing work with them to be distributed whilst preserving their copyright.

The ethical approval document is included in Appendix C.2.

# 6 Design

Taking the project's objectives and requirements into consideration, this section aims to establish a robust foundation for efficient analysis of public consultation responses. We will cover the decisions made regarding the system's architecture, database design and user interface design. But first, it is crucial to understand the overall structure and stages that will be completed in the "Decoding Democracy" project.

## 6.1 Project Pipeline

This section will provide a detailed overview of each stage, explaining why the specific approaches and techniques were chosen. Figure 1 illustrates the project's pipeline, with each step to be discussed.

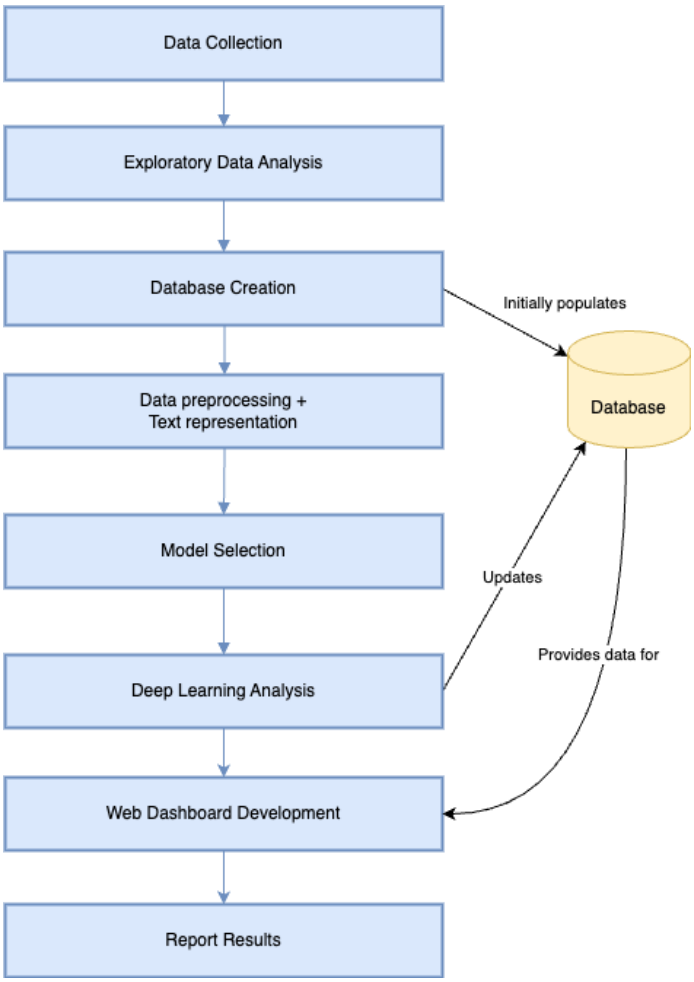



Figure 1: Project pipeline showing how each stage in the project ties together

Each of the components in this Figureare explained in detail in subsections 6.1.1 to 6.1.8.

### 6.1.1 Data Collection

The initial phase involves gathering data on the Scottish consultation' to our study, ensuring the dataset aligns with the ethical principles outlined, such as not collecting any personal information. The datasets containing the responses to the three consultations introduced earlier in Section 1.2 were provided by my supervisor, who had downloaded them from the government website [14] [15] [21].

### 6.1.2 Exploratory Data Analysis

After data collection, an exploratory data analysis (EDA) was conducted to gain insights into the dataset's characteristics. This phase helped to identify trends in the public's responses, such as submissions over time and the distribution of multiple choice responses. It also helped distinguish which questions were multiple choice and which were open-ended so that a boolean `multiple_choice` field could be added to the `Question` entity (see 6.2).

### 6.1.3 Database Creation

The database creation stage involved organising the collected data into a suitable format to be stored in a database, whilst maintaining the relational properties of the data, such as which consultation each response belonged to. The database was designed to facilitate easy manipulation during the analysis phase, such as updating it with information on the sentiment and topics within the responses (see Figure 1). More details about this are given in Section 6.2.

### 6.1.4 Data preprocessing and Text representation

This step involved preprocessing the responses, which is important because it normalises and standardises the responses, making them consistent and comparable (more details given in Section 7.1). The preprocessed responses were then transformed into a numerical format that machine learning algorithms can process. Text embeddings and Term Frequency-Inverse Document Frequency (TF-IDF) were used to capture the semantic meaning of words and the importance of terms within documents (see Section 7.3).

### 6.1.5 Model Selection

Following this, several language models were compared to find the most accurate one in the context of analysing consultation responses. Some of the models that were evaluated include RoBERTa [7] and DistilBERT [20]. More details will be given in Sections 7.2 and 7.3.

### 6.1.6 Deep Learning Analysis

After selecting a model, sentiment analysis was applied on the whole dataset to gauge the emotional tone towards different questions. This process involved quantifying the positive or negative sentiment of responses to enable a deeper understanding of public opinions. On top of this, topic modelling was conducted to discover common themes in responses. As shown in Figure 1, the database was then updated with the results from these processes.

### 6.1.7 Web Dashboard Development

After updating the database with the results of the deep learning analysis, the web dashboard was developed to visually present the consultations,

their questions and citizen answers, along with the analysis result to offer an interactive interface for users to explore.

#### 6.1.8 Reporting Results

Finally, the last step was to combine the findings of the analysis into an insightful report, which will hopefully serve as a valuable resource for policy-makers and members of the public.

### 6.2 Database Design

As mentioned in the Project Pipeline section, a database was constructed to store the data related to each consultation. This section will provide a detailed description of the Entity-Relationship (ER) model (shown in Figure 2) that was developed for the database.

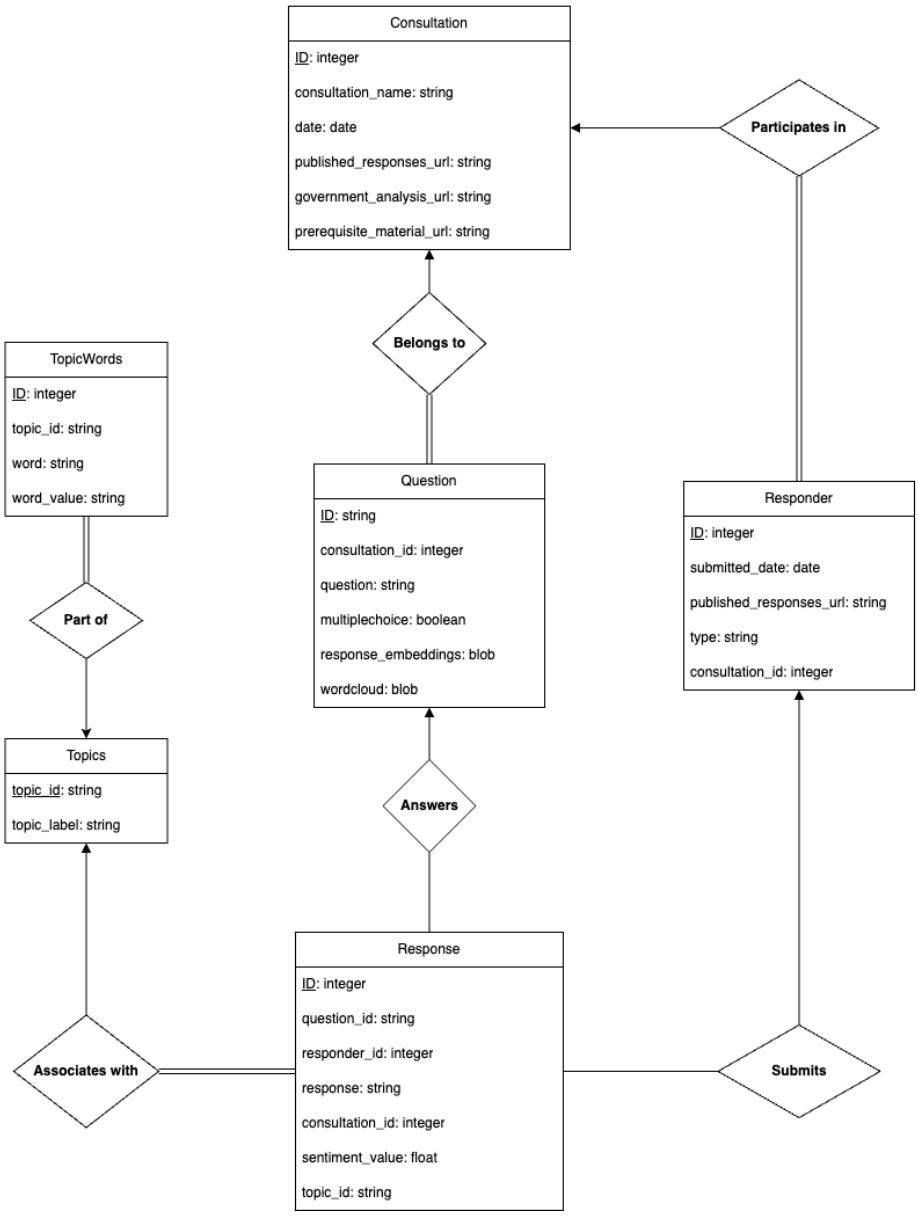



Figure 2: ER Model of the Database to capture consultations, their questions, citizen responses and AI analysis

The Entity-Relationship (ER) model illustrated in Figure 2 is designed to capture and store all the relevant information related to public consultations, including the consultation details, questions, responses, respondents, and the analysis results obtained through the various deep learning techniques. This is

so that during the deep learning processes, information is easy to retrieve and manipulate. On top of storing the details about each consultation, it will store the crucial components of the AI analysis conducted later on. For instance, the results obtained through sentiment analysis and topic modelling will be represented in a format that is easily retrievable from the web dashboard.

The ER model consists of six entities: Consultation, Question, Responder, Response, Topics, and TopicWords, which are related via foreign key constraints, ensuring data integrity and consistency.

The Consultation entity stored information about each consultation, such as its name, date, and links to the published responses, government analysis, and prerequisite materials. This entity is associated with the Question entity through a one-to-many relationship, allowing multiple questions to be associated with a single consultation.

The Question entity stores the actual text of each question, along with information like whether the question is multiple-choice or open-ended. It also includes fields to store wordclouds and response embeddings. Storing response embeddings in this table allows for a more efficient retrieval and organisation than storing them for each response, as we can avoid slower queries to retrieve them for each response individually. The embeddings are stored in a way that preserves the response id for each embedding, so that the response text can be retrieved when performing searches on similar responses.

The Responder entity represents individuals or organisations who have submitted responses to the consultation. It stores information like the submission date, the type of responder (e.g., individual, organisation), and the link to the published responses on the government website. This table will mostly be used for the exploratory data analysis stage, to see statistics on submission timelines etc., because the project is not interested in individual responders, but rather the combined sentiment and concerns of the public.

The Response entity captures the actual text of each response submitted by a responder for a specific question within a consultation. It also includes fields to store the sentiment value and topic id, which are derived from the sentiment analysis and topic modelling processes.

The Topics entity stores the unique topics identified through topic modelling, along with their corresponding descriptions. Each question will have its own set of topics, so the topic ids will be prefixed with the question id they belong to. This design choice allows for more granular and focused topic modelling for each question, ensuring the response topics are contextually relevant to what is being asked. By modelling topics specific to each question, we can avoid confusion with topics that may share common terminology but have different contextual meanings.

Finally, the TopicWords entity is used to associate individual words with their respective topics, allowing for a more detailed understanding of the key terminology making up a certain topic.

This ER model design ensures that all relevant data related to public consultations, questions, responses, and analysis results are captured and stored in a structured manner. The use of foreign key constraints and relationships

between entities helps maintain data integrity and enables efficient querying and retrieval of information for further analysis and visualisation.

## 6.3 System Architecture

The system comprises three main components: the database, the jupyter notebook, and the web dashboard. The interactions between each component will be discussed in this section.

### 6.3.1 Jupyter Notebook and Google Colab

The data exploration and deep learning processes were implemented in a Jupyter Notebook. This approach was preferred over creating separate python files because it provided a place to combine all of the analysis needed for this project, along with markdown explanations of each step so that there was a clear workflow and "recipe" of the research conducted for this project [42].

The Jupyter Notebook is hosted on Google Colab [26] mainly due to the free resources Google provides. Using Google Colab gave us the option of using GPU and TPU runtimes, which were crucial in accelerating the more intensive computations for the deep learning tasks. After performing the necessary analysis in the Jupyter Notebook, the SQLite database was populated with the relevant information such as sentiment scores and topics, and downloaded locally to ensure the frontend could access and display the analysis results when developing the dashboard.

### 6.3.2 Data Flow

To re-iterate the data flow within the "Decoding Democracy" system before outlining how the web dashboard ties in, the pipeline is as follows:

1. The Jupyter Notebook reads the raw consultation data and performs necessary cleaning, exploration and preprocessing steps to ensure a consistent data format for each consultation.

2. The processed consultation data is put into the SQLite database.

3. The processed data is then analyzed using NLP techniques.

4. The analysis results are added to the SQLite database, following the predefined schema in Section 6.2, and the database is hosted on a local server using Flask.

5. The frontend sends requests to the Flask backend to retrieve consultation data and analysis results from the database.

6. The backend queries the SQLite database and responds with the requested data in JSON format.

7. The frontend receives the data and renders it in an intuitive manner for users to explore.

The web dashboard follows a two-tier client-server architecture [52]. Figure 3 shows the overall communication between the different system components.

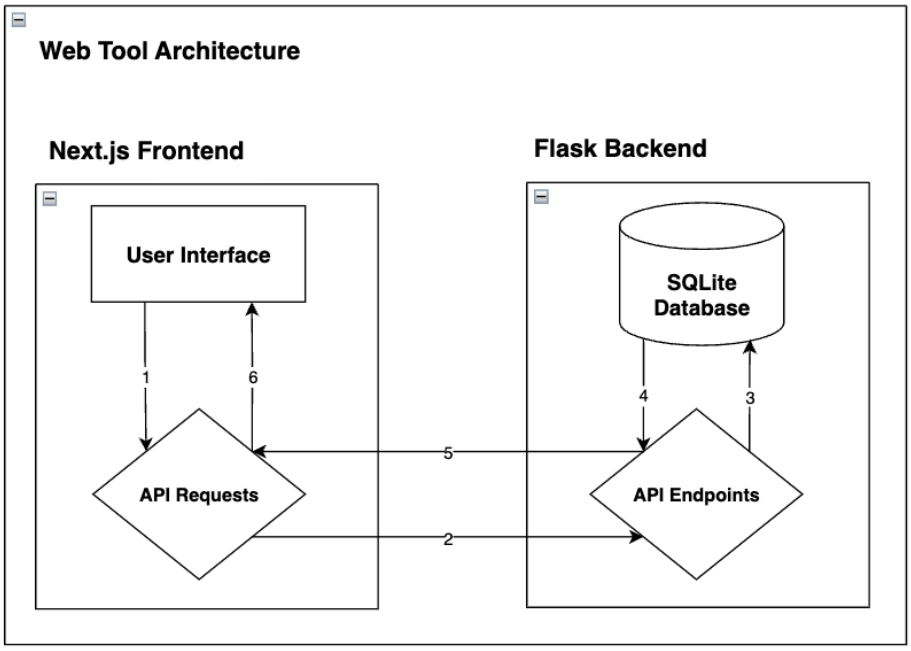



Figure 3: Web dashboard architecture, illustrating how the client and server communicate to retrieve consultation data

### 6.3.3 Client

The frontend of the application is built using Next.js [36], a React-based framework. Next.js provides a robust and efficient way to create user interfaces, ensuring optimal performance and fast loading times by supporting static site generation (SSG). The client communicates with the server using a Representational State Transfer paradigm [34]. REST is an architecture that is commonly used in web applications and was chosen along with Next.js for several reasons. Next.js has a file-based routing system and built in support for API routes, which streamlined the process of adding pages for consultations and their questions. The REST architectural style promotes a clear separation of concerns, with the client and server communicating through well-defined APIs. Also, being language independent, REST was an obvious choice as it enabled communication across the client and server which were written in different langauges (TypeScript and Python). The use of TypeScript was intentional, as it

helped prevent type-related errors at build time and ensured strict use of typing all the way from the database fields to the frontend components [40].

### 6.3.4 Server

The server is implemented using Flask [23], a lightweight and flexible Python web framework. The Flask application handles incoming requests from the client and interacts with the SQLite database [53]. The backend is mostly responsible for retrieving data from the database, but also performs some data processing such as similarity searching which is discussed later in Section 7.4. Being a commonly used framework, it has extensive documentation which made it easy to debug issues encountered during the development process.

The SQLite database schema follows the Entity-Relationship (ER) model specified in Section 6.2. SQLite was chosen as it is a relational database management system, providing a straightforward and efficient way of storing and retrieving data in Python, which is the language used for the initial data processing and deep learning tasks. The lightweight nature of SQLite made it suitable for this project as advanced database features were not required, and it was also chosen to complement the Python backend, as Flask apps integrate seamlessly with SQLite.

Overall, this architecture separates concerns by conducting the NLP processes before developing the web dashboard, so that the results can be seen instantly without users having to wait for analysis tasks to complete. Since all the deep learning techniques used are deterministic, meaning they will produce the same outputs for the same inputs, we can analyse all the consultation data once and store the results.

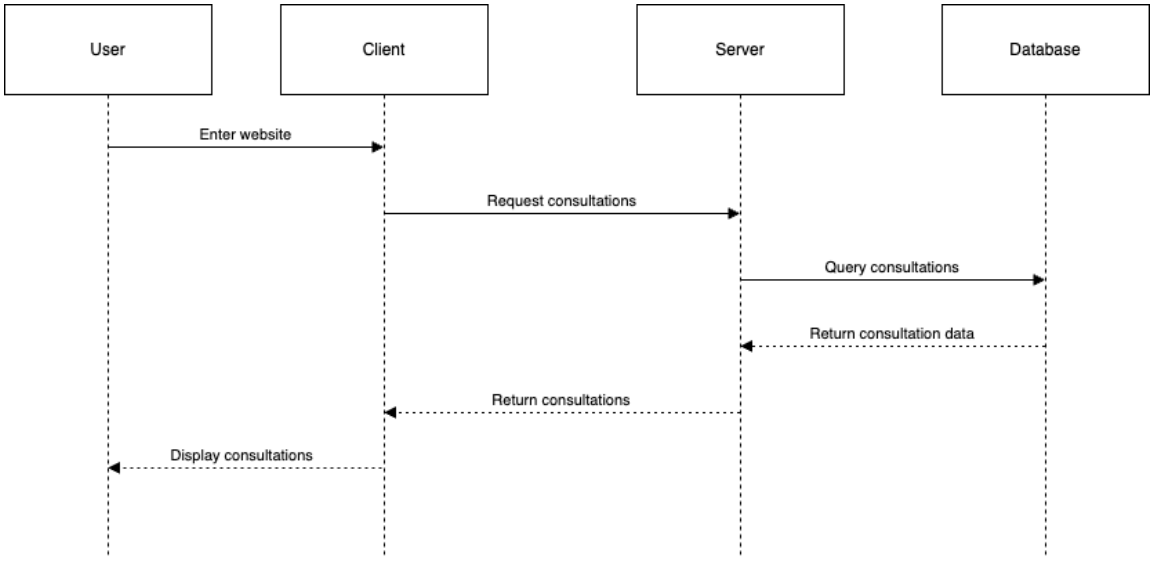



Figure 4: Sequence diagram for retrieving consultation data from the database

Figure 4 illustrates an example query from the client to the server in order to retrieve data on consultations. This sequence of events is similar for any query made from the client. However, when making more complex queries such as querying for similar responses, the sequence is slightly different, as shown in Figure 5 below.

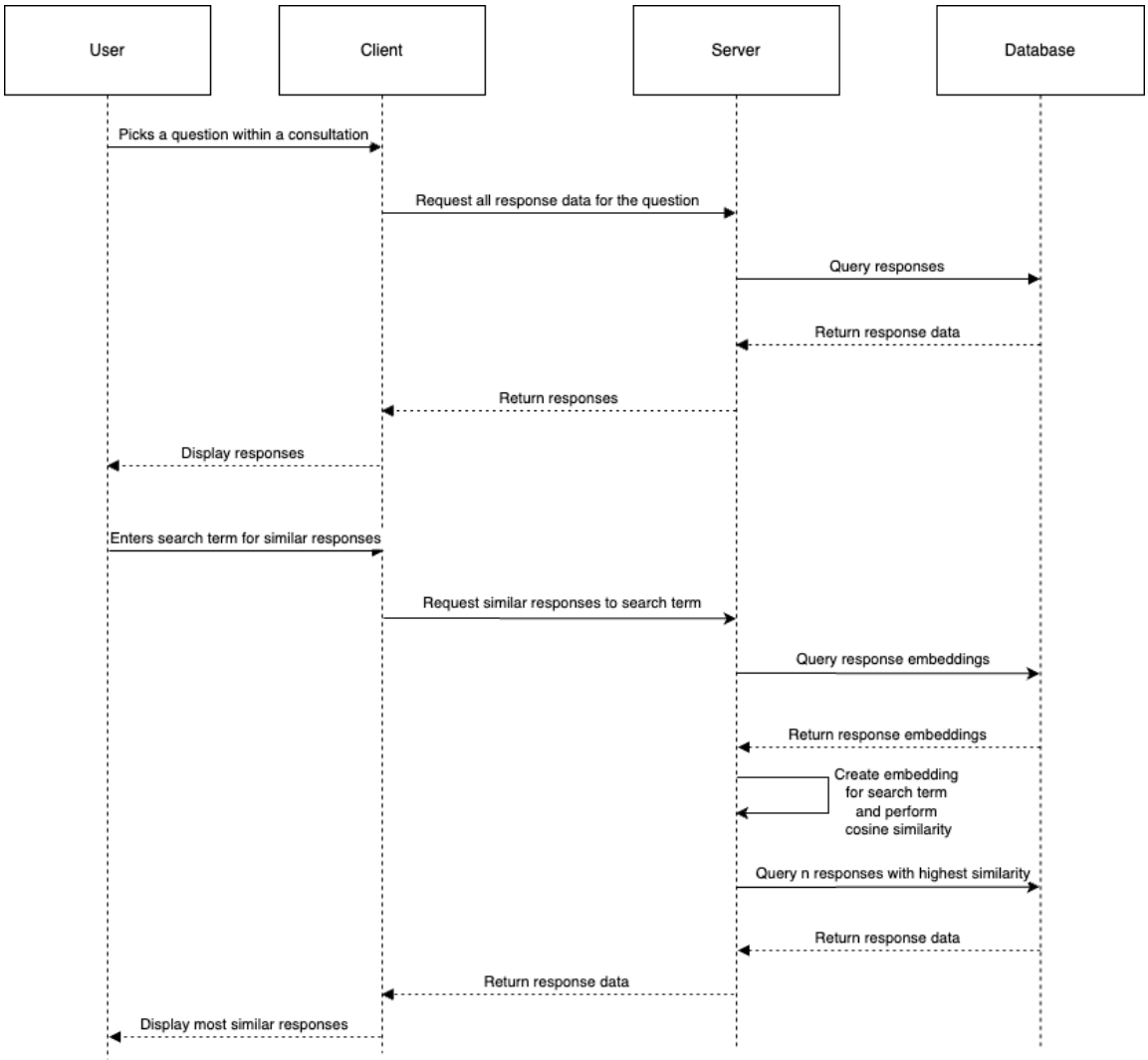



Figure 5: Sequence diagram for searching for similar responses given a user-defined a search term

When the user of the web dashboard is looking at a certain question within a consultation, they will be able to enter a search term and find the most

similar responses for that question. The sequence diagram in Figure 5 shows how a user can select a question for analysis, before finding similar responses. After entering a search term, which will usually be an example response to the given question, the server retrieves the response embeddings (see Section 7.3) from the database and performs cosine similarity (discussed in Section 7.4) between every response to the question and the user's search term. This will return the ids of the responses which were most similar, allowing the server to fetch the actual text for each response and return the N most similar responses to the client.

## 6.4 User Interface Design

The requirements outlined in Section 3 were used to structure the user interface so that users could easily analyse every consultation in an intuitive manner. It was important to create navigable pages for each consultation that displayed every question along with their responses. Also, it was vital to display the analysis results in an informative manner that was easy to understand. The web dashboard composes of 3 main pages or routes, which are described below.

**The home page** displays links for the three consultations, as well as a description of what Decoding Democracy is about. **The questions page** displays a searchable list of questions for the selected consultation. Each question can be clicked into. (See Figure 30 in appendix B). **The open-ended responses page** displays a list of all the responses to the selected question. This page also contains different tabs to visualise various types of analysis on the responses. The first tab allows users to search for exact matches within the responses, to filter responses that have exact words or phrases. The second tab allows users to search for similar responses, and the third tab allows users to select and view the different topics that were found in the corpus of responses, as well as the overall and per-topic sentiment distribution. (See Figure 35 in appendix B). Finally, **the multiple choice response page** displays the distribution of responses with a bar chart. (See Figure 31).

### 6.4.1 Usability and Responsiveness

The user interface was designed with usability in mind, in order to create clear and intuitive pages for users to navigate. This meant keeping a simple, consistent colour theme and making the website responsive to different screen sizes, such as mobiles and laptops. The responsiveness of the website is shown in Figures 42 to 46 in appendix B.

# 7 Implementation

This section goes through the details of how the implementation was carried out, focusing on specific choices for deep learning models, text processing functions, database decisions, and user interface features.

To carry out and document the various tasks in this project, a Jupyter Notebook was created, as discussed in Chapter 6. This notebook contains all the steps from extracting the consultation data from the excel files, to performing sentiment analysis and topic modelling. Initially, it performs an exploratory data analysis to find out the characteristics of each consultation (see Table 1 in Section 1.2). The various python libraries used in the notebook can be found in appendix C, along with their descriptions.

## 7.1 Text Preprocessing

Before evaluating the performance of any models, the open-ended responses were preprocessed to ensure they were standardised and comparable. Figure 6 illustrates a breakdown of what the `pre_process` function does on a given input text.

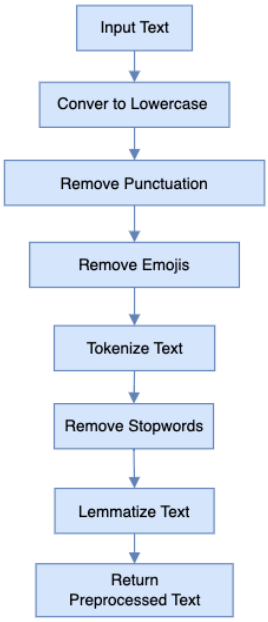



Figure 6: Steps for preprocessing input text (i.e. a response)

The steps are as follows:

1. Convert the input text to lowercase.

2. Remove all punctuation characters from the text.
3. Remove any emojis from the text.
4. Tokenize the text (split response into individual words).
5. Remove "stopwords" from the tokenized text.
6. Lemmatize the remaining words (reduce them to their root form).
7. Return the preprocessed words.

Each step of the preprocessing function is important for improving the efficiency and accuracy of the NLP processing tasks.

Firstly, the conversion to lowercase and removal of punctuation is common practice as it helps to clean the text data, as well as reducing noise and standardising text by ensuring that words with the same semantic meaning but different representations (e.g., "Hello" and "hello") are treated as a single feature. Any emojis are also removed as the models we use cannot interpret their meaning.

Secondly, tokenization is the process of dividing text into smaller units called tokens, which can be words, phrases, subwords, or characters. In our case, the tokens are words. Tokenization helps the model being used to process text more efficiently by enabling it to handle variations in language and handle words with multiple meanings or forms [31].

After this, the text was cleaned of any word that was present in a predefined set of words called "stopwords". Stopwords refer to commonly occurring words in a language (e.g., "the", "a", "and"), which generally do not contribute to the meaning of the text. Retaining these words can negatively impact the topic modelling process by introducing noise and potentially skewing the identified topics. Removing them also improves the efficiency of computations as the overall size of the text corpus is largely reduced. In this context, a custom set of stopwords is used to ensure that domain-specific words are not removed, as this could lead to a loss of valuable information. For example, negations like "not" and "didn't" would usually be removed, but we make sure not to remove them because these words can significantly impact the sentiment polarity of text. For example, the phrase "I am not happy" carries a negative sentiment, whilst the removal of the word "not" would lead to positive sentiment ("I am happy"). Therefore, we keep negation words to ensure the model does not interpret negative responses as positive.

Finally, lemmatization is used to convert words to their root form. This is important because words with the same root can have different forms depending on the tense or plurality (e.g. "go", "going", "went", "gone"). Lemmatization, like the other techniques described, helps to reduce the dimensionality of the feature space by mapping different forms to a single representation, which will ultimately improve the output of the NLP tasks.
For example, the sentence:

> *"Why more uncertainty? We don't know what's going on after brexit. We have just hit a bus! The first minister is trying to throw us under it."*

would be transformed into the following after preprocessing:

> *"uncertainty dont know go after brexit hit bus first minister try throw us under"*

## 7.2 Model Selection for Sentiment Analysis

The sentiment analysis component involved a comparison of multiple models, including Vader (Valence Aware Dictionary and sEntiment Reasoner) [12], TextBlob [57], RoBERTa [7], and DistilBERT [65], to find the best suited for analysing the vast amount of responses for each consultation.

TextBlob and VADER are rule-based sentiment analysis models that rely on predefined rules to determine the sentiment of a given text, making them less computationally expensive and faster but potentially less accurate in understanding context and complex language structures. On the other hand, DistilBERT and RoBERTa are deep learning-based language models that have learnt patterns and representations from vast amounts of text data, so they should achieve higher accuracy in sentiment analysis tasks but are much more computationally intensive, so we will evaluate which model is most practical to use for the large corpus of consultation responses.

From researching, it was expected that Vader would perform better than TextBlob, especially with negative polarity detection [59], as it is trained on content that typically appear on social media, such as informal sentences, emojis and punctuation. However, it was important to choose different types of models to compare which one is best for the task at hand.

The methodology for comparing models started with choosing a random set of 100 responses from the first consultation [14]. After this, I labelled each response manually with the sentiment value that I thought was suitable, to create a "human values" column. This enabled me to compare the models' results against values that I thought were correct. Each step of the process is detailed below.

1. Choose a random set of 100 responses from the draft referendum bill consultation.

2. Assign a human sentiment value to each response, ranging from very negative (-1) to very positive (1). Values were one of {-1, -0.5, 0, 0.5, 1}.

| A | B | C | D | E | F |
|---|---|---|---|---|---|
| answer | human | vader | textblob | roberta | distilbert |
| I have confidence in | 0.5 | 0.586 | 0.45 | 0.639 | -0.011 |
| Why more uncertain | -1 | -0.401 | 0.375 | -0.823 | -0.572 |
| This appears to be s | 0.5 | 0.586 | 0.43 | 0.877 | -0.274 |
| Proposals seem fair | 0.5 | 0.318 | 0.425 | 0.285 | 0.076 |
| Words can't explain | 1 | 0.077 | 0.2 | 0.485 | -0.062 |

Figure 7: Comparison of each model vs human sentiment values

3. Run each model on every response, and store results in a csv file with each column being a different model. A snapshot of what the csv file looks like is shown in Figure 7, with the numbers representing human-assigned and model-calculated sentiment scores.

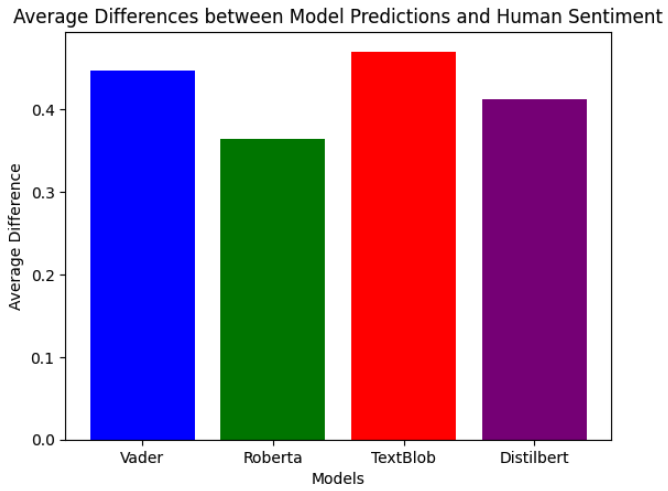



Figure 8: Average differences between model sentiment values and human sentiment values

4. Calculate the average difference for each model's values against the human values (results shown in Figure 8).

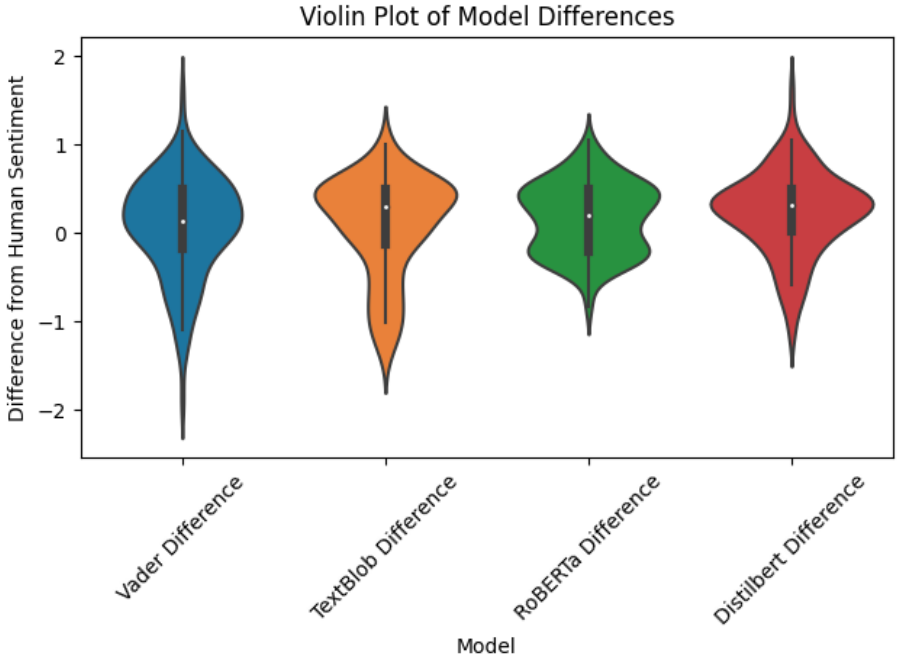



Figure 9: Sentiment Analysis Model Comparison

5. Create violin plots (Figure 9) to show the distribution of values between each model, and see which ones are most similar to the human values.

From the bar chart in Figure 8 and violin plots in Figure 9, it can be concluded that RoBERTa produced the most accurate sentiment values. This is because the green violin plot is wider around zero values, suggesting most of the difference in values between human and RoBERTa sentiment was close to 0. However, DistilBERT also shows close similarity to human values, and since RoBERTa had a much slower execution, we opted for the more efficient DistilBERT model.

The DistilBERT model was proposed in the paper titled "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter" [20]. It is a "student" transformer model trained by extracting essential knowledge from its "teacher", the BERT base model. It has 40% less parameters than Google's BERT [27], and runs 60% faster while preserving over 95% of BERT's performances as measured on the GLUE language understanding benchmark [61]. This makes it an ideal choice for our project as we need to process around 100,000 responses.

Following the choice of the pre-trained DistilBERT model from HuggingFace [65], it was used to process and update sentiment values for every individual preprocessed response from the database.

## 7.3 Topic Modelling

Arguably the most important part of "Decoding Democracy" is finding the key topics people are discussing in their responses. Therefore, it was crucial to implement a strong topic modelling process which accurately found underlying themes. This type of deep learning analysis is more intensive than assigning sentiment scores, so it required a different approach to compare models. Rather than running different models and comparing them, research was conducted prior to choosing our method. On top of analysing research papers in the field, I consulted my supervisor Dr Rosa Filgueira who has some expertise in the area. In the end, a hybrid approach of using Clustering techniques along with Latent Dirichlet allocation (LDA) was chosen. These are two different approaches to discovering patterns within textual data, and this section covers an overview of how each method works and the benefits of using them over alternative methods.

### 7.3.1 Clustering Techniques

Clustering is an unsupervised machine learning technique used to group data points based on similarity. In this context of this project, the data points are responses. Clustering is able to identify natural groupings without any prior knowledge of the categories, so it is especially useful when analysing responses to open-ended questions, because people discuss a range of topics, making it hard to predict how many topics or clusters there will be in advance.

As described in the paper "A Survey In Clustering Techniques" by Rai et al. [41], the steps in a clustering algorithm are as follows:

1. First, text preprocessing is performed (see the steps outlined in Section 7.1) which involves steps such as tokenisation (breaking text into words or phrases), stopword removal (eliminating common words) and lemmatisation (reducing words to their base form).

2. Secondly, feature extraction if performed to convert text into a numerical form for models to understand, such as text embeddings (see Table 2).

3. With the text embeddings, weights are assigned to each word based on their frequency, through a process called TF-IDF (Term Frequency-Inverse Document Frequency).

4. Next, a similarity metric (cosine similarity in our case - see Section 7.4.1) is used to calculate the similarity between text documents.

5. After this, the clustering algorithm is chosen based on the data characteristics. In our case, hierarchical clustering is used over alternatives like K-means or DBSCAN for several reasons explained in Section 7.3.2.

6. Finally, the selected algorithm is applied to group similar text documents into clusters, like in Figure 10 below.

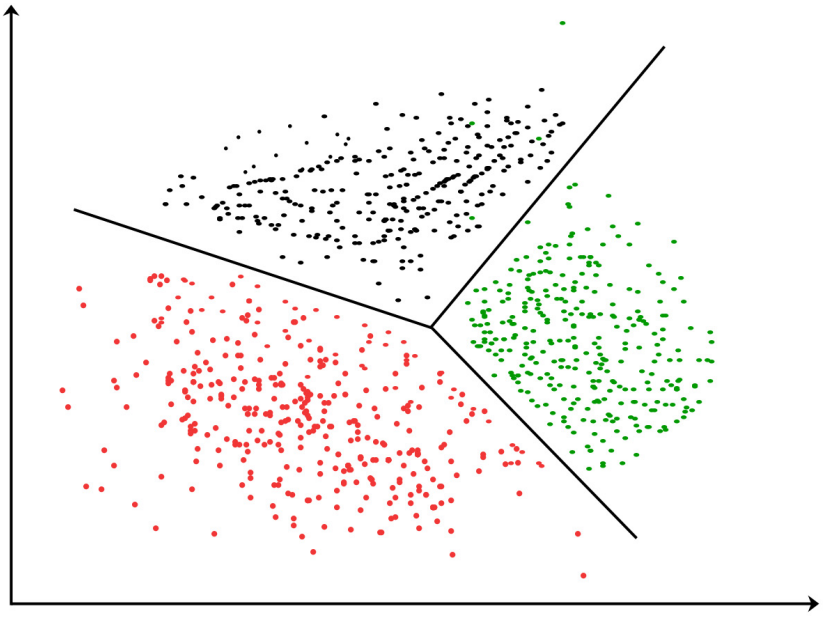


Figure 10: An example of how embeddings in a vector space are clustered

### 7.3.2 Agglomerative clustering

Agglomerative clustering is a hierarchical clustering algorithm [29]. It follows a bottom-up approach, meaning each data point (in this case, each response text) starts as its own separate cluster, and the algorithm iteratively merges the closest pairs of clusters until a stopping criterion is met or all data points are merged into a single cluster. The metric used for the merge strategy can be configured with different criteria, but with our agglomerative clustering approach we use the euclidean distance, which is simply the length of the shortest line connecting two points. A distance threshold is defined to set a limit above which clusters will not be merged.

The algorithm works as follows:

1. Initialize clusters (initially each data point is a separate cluster).
2. Calculate the pairwise distances between all pairs of clusters. This uses the euclidean distance calculated on the vector representations from TF-IDF in our case.
3. Merge the two closest clusters based on the calculated distances
4. Recalculate the distances between the newly formed cluster and all remaining clusters.
5. Repeat steps 3 and 4 until a stopping criterion is met, which in our case is a threshold distance value.

As already mentioned, agglomerative clustering has the advantage of not needing a specified number of clusters in advance. It also has the benefit of being deterministic, meaning that the same input data will always produce the same result, as this will produce consistent results with our dataset. Another advantage is that it can handle arbitrary shapes of clusters, unlike algorithms like k-means that assume spherical clusters. The main downside of agglomerative clustering is that it is computationally expensive, as it needs to calculate the pairwise distances between all clusters at each iteration. However, since a key objective in our project is to perform accurate and reliable topic modelling, it was decided to sacrifice efficiency for better results.

The resulting clusters that are calculated for each set of responses are analyzed using LDA to identify the dominant topics within each cluster, in order to provide a more interpretable understanding of the themes.

### 7.3.3 Latent Dirichlet allocation

LDA is a probabilistic topic modelling technique that assumes each document is a mixture of latent topics, and each topic is a distribution over words. A paper on "Exploring Topic Coherence over Many Models and Many Topics" [54] helped to compare LDA with alternatives like Singular Value Decomposition (SVD) and Non-Negative Matrix factorization (NMF). Stevens et al. found that *NMF learns more incoherent topics than LDA and SVD, and that for applications in which a human end-user will interact with learned topics, the flexibility of LDA and the coherence advantages of LDA warrant strong consideration.*

The paper "Latent Dirichlet Allocation" by Blei et al. [4], which introduced LDA, describes the process as follows.

1. The same steps (1-3) as in Section 7.3.1 on clustering are performed first, with text data being preprocessed and transformed into a bag-of-words representation (describing the frequency of words in a document).

2. After these steps, the selected model (LDAMulticore [25]) is initialised with the number of topics to look for. (This number is taken from the number of clusters found from the clustering techniques which are performed beforehand - see Section 7.3.4).

3. The model initialises topic-word and document-topic distributions for the given documents (responses in our case) with random values. More specifically, each response will be assigned a distribution of topics, as shown by the example in Figure 11.

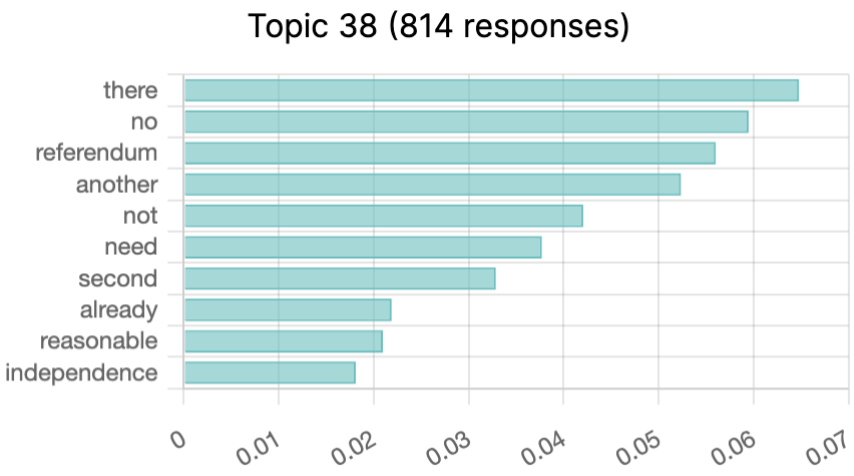



Figure 11: Topic-word distribution for a topic in question 1 of the draft referendum bill consultion

4. An iterative algorithm updates these distributions in an effort to maximise the log likelihood of the data. More specifically, there are two steps that are repeated:
   (a) E-step (Expectation Step): Estimates the posterior distribution of latent variables (the topic assignments) given the observed data. This step uses variational inference or Gibbs sampling.
   (b) M-step (Maximization Step): Updates the model parameters (the topic-word and document-word distributions) based on the estimated latent variables to try and maximize the likelihood of the data given the current estimates.
5. The E-step and M-steps are repeated iteratively until convergence is reached.
6. Finally, after convergence each topic is represented by a distribution over words, and each document is represented by a distribution over topics. These distributions can be used to interpret the themes and assign topic labels.

The two key steps that make LDA so powerful are the E-step and M-Step. The E-step tries to find the best explanation for the data by updating the probabilities of the hidden variables, whilst the M-step updates the model parameters (like topic-word distributions) based on the estimated hidden variables to maximize the likelihood of the data given the current estimates of the hidden variables, improving the model's ability to represent the data accurately.

A paper on "Topic Modelling Using Latent Dirichlet allocation" [9] discusses the strengths of Latent Dirichlet allocation (LDA) for topic modelling. Being an unsupervised machine learning algorithm, it doesn't need labelled data to identify topics and it is also able to handle a large number of topics and a wide range of document collections, which is suitable in the context of large public consultations. As well as this, LDA provides interpretable results by assigning each word in a document to a particular topic, so that users can understand the underlying themes present in the text data.

The specific python LDA model that was chosen was called `LDAMulticore` [25], as it uses all CPU cores to parallelize and speed up execution.

### 7.3.4 Combining methods for optimal topic modelling

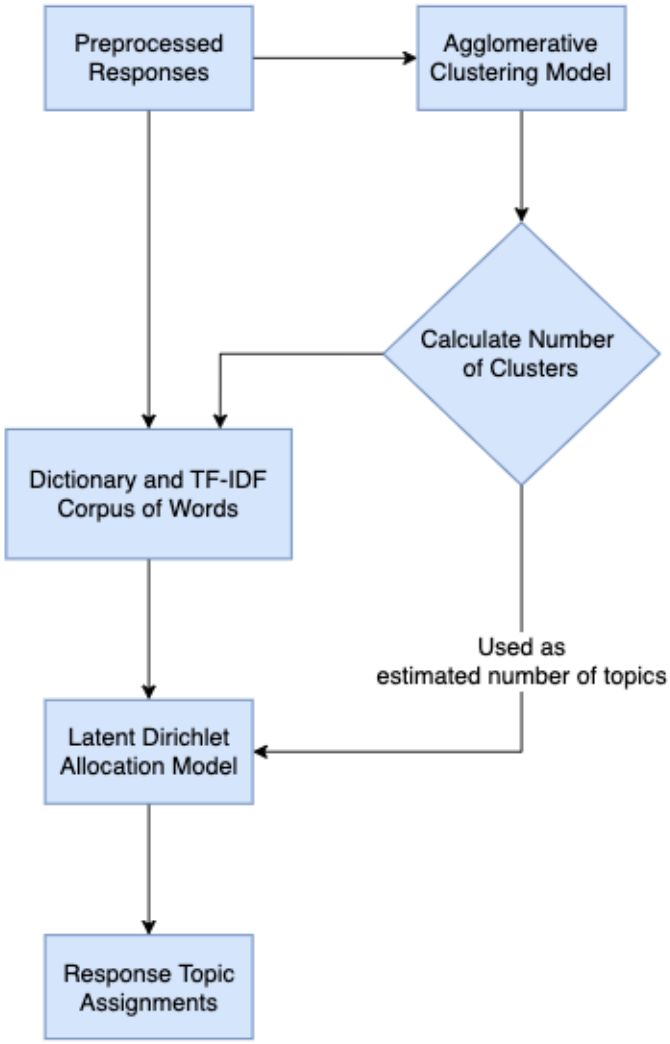



Figure 12: How Agglomerative Clustering is combined with LDA to find topics in the responses

After explaining the applications of both Clustering and LDA, we can see why these two methods are used together in our project, and Figure 12 is a visual representation of how they are combined. Clustering can be used to identify the optimal number of topics for LDA, as suggested in the paper "Finding an Appropriate Dimensionality for Latent Dirichlet Allocation" by Cao et al. (2009) [41]. Since we are finding topics for around 50 different groups

of responses (see Table 1 in Section 1.2), it was impractical and inefficient to tune the LDA model for each different question.

Using clustering as a precursor to LDA topic modelling is a more practical and data-driven approach compared to manually tuning the LDA model because we automatically identify natural groupings within the data based on similarity measures, reducing the computational overhead required for exhaustive parameter tuning of the LDA model. Essentially, clustering ensures that the LDA model focuses on coherent subsets of responses, as opposed to attempting to find a single "one-size-fits-all" number of topics across all responses. This hybrid approach combines the strengths of both techniques, providing a more efficient and scalable solution for analysing diverse public consultation data.

## 7.4 Semantic Response Similarity

In contrast to traditional search engines that will only compare documents based on lexical matches, semantic searching is a method that seeks to improve search accuracy by understanding the content for each query, and even finding synonyms and acronyms [49]. After calculating the text embeddings in the previous step, it is possible to implement this more complete way of searching responses by comparing the embeddings with each other. The idea behind semantic search is to take all the embeddings in the corpus of responses, and form a vector space from them. At search time, the query is embedded into the same vector space and the closest embeddings from the corpus are found. These entries should have a high semantic overlap with the query.

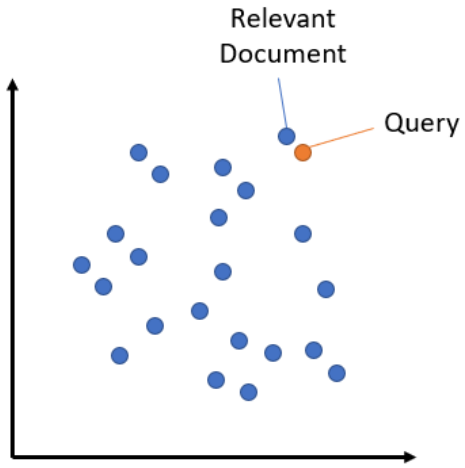



Figure 13: Embeddings vector space for semantic searching

As seen in Figure 13, an encoder (discussed in Section 7.4.1) will map the text embeddings into a vector representation. We are concerned with symmetric semantic search, which is when the search term and the similar entries in the

corpus have a similar length and the same amount of content.

### 7.4.1 Bi-Encoders vs Cross-Encoders

An encoder is a type of neural network architecture used for generating vector representations of input data, so that the semantic information (e.g., similarity) of the data can be explored. Two types of encoders were considered for our use case, shown in Figure 14 [67].

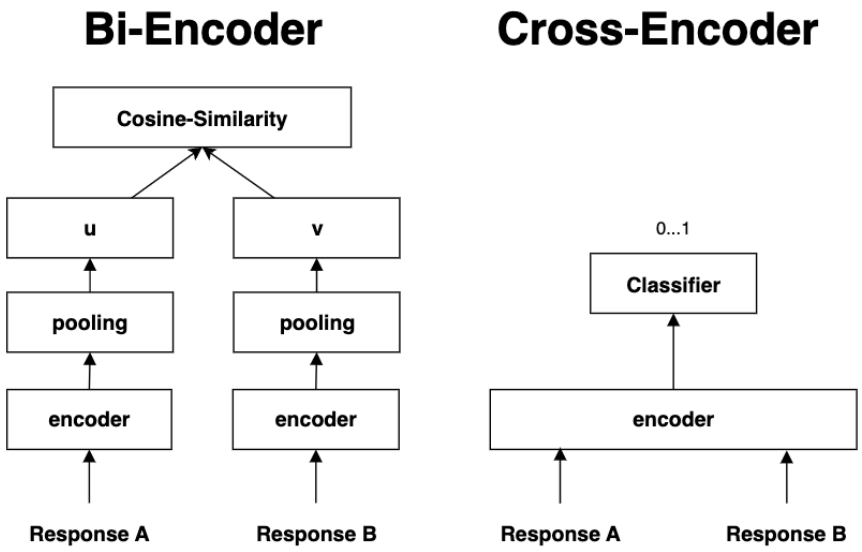



Figure 14: The concept of Bi-Encoders and Cross-Encoders for text similarity

In our case of finding the semantic similarity of responses, a bi-encoder would produce text embeddings for each input, Response A and Response B, independently, before resulting in response embeddings `u` and `v`. These embeddings are then compared using cosine similarity [17], which measures how close the vectors of the embeddings are in a multi dimensional space, as shown in Figure 15. If the cosine similarity is 1, the vectors point in the same direction in the multi-dimensional space suggesting the two responses being compared have the exact same semantic meaning.

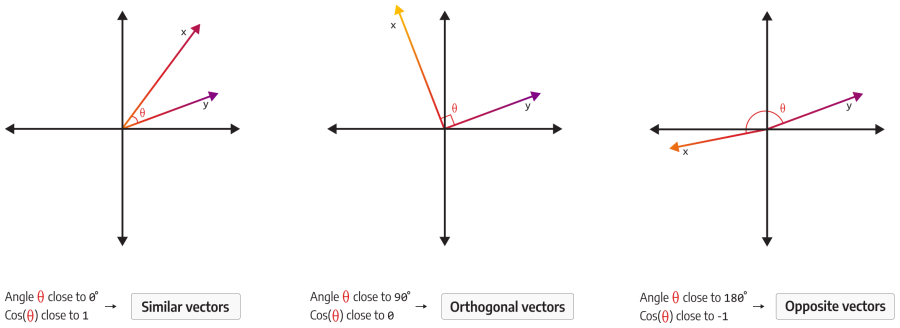



Figure 15: Visualisation of how Cosine Similarity works for different vector pairs

On the other hand, Figure 14 shows how a cross-encoder takes in both responses simultaneously and produces an output value between 0 and 1 indicating the similarity of the response pair. As discussed in the paper "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" [47], cross-encoders are less practical when applying semantic search over a large corpus of data because they do not produce text embeddings to efficiently index or compare data points from.

For this reason, we adopted the bi-encoder paradigm, where each response is independently mapped into a vector space, leading to efficient storage of embeddings for subsequent queries. While cross-encoders may achieve better accuracy for single queries, bi-encoders are more effective over a large dataset like ours, and still manage to achieve a good accuracy. The cross-encoder method was tested, but for certain questions with many responses, the query took too long and caused a timeout error, whilst the bi-encoder method consistently returned a response within just a few seconds.

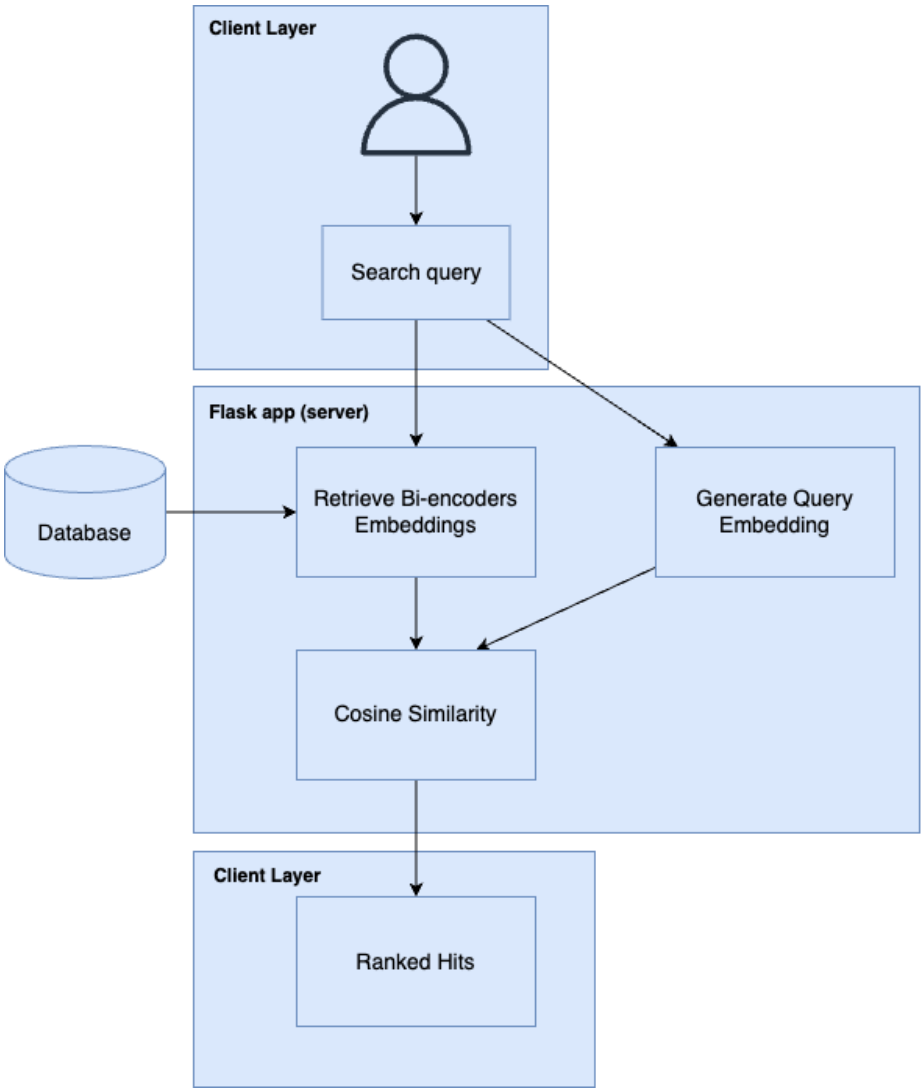



Figure 16: The process of retrieving similar responses using bi-encoders and cosine similarity

Figure 16 illustrates the process of performing a semantic similarity search from the web dashboard. After the user enters a search query, the server produces an embedding for the query whilst retrieving all the other relevant embeddings. Then, it performs cosine similarity to find the closest vectors to the newly generated embedding, before returning them in order of similarity.

## 7.5 Web Dashboard Frontend Implementation

Although the main focus of "Decoding Democracy" was to extract value from the responses using deep learning techniques, the web dashboard was still a primary objective and therefore was designed to include several valuable features, as described in Chapter 6. The frontend of the web dashboard was developed using Next.js [36], which was chosen amongst the many available options mainly due to previous experience with using it. It was a safe choice that would make the process of building a dynamic and interactive user interface a lot easier than if I were to have used plain JavaScript and HTML, as it offers features like built-in API routing and file-based client side navigation. It also comes build in with Tailwind CSS [55], which made the styling process easier and enabled the development of a responsive design across different screen sizes. TypeScript was used throughout the frontend codebase to ensure type safety and catch potential errors during development.

The client and server interact through well-defined API endpoints, using a RESTful architecture, and several third-party libraries were installed to enhance the user experience. Chart.js [8] was used to create charts for displaying analysis results, such as sentiment distributions and topic prevalence.

The frontend implements several components to enable users to explore the analysis conducted on the consultations. Screenshots of the user interface displaying all of the features are shown in appendix B.

Table 3 lists the key packages used in the frontend implementation along with their purposes.

| Package | Description |
|---|---|
| chart.js | JavaScript charting library for designers and developers |
| next | React framework for server-rendered applications |
| react | JavaScript library for building user interfaces |
| react-chartjs-2 | React wrapper for Chart.js 2 |
| react-dom | Entry point to the DOM and server renderers for React |
| react-icons | Icon library for React applications |
| tailwindcss | Utility-first CSS framework for rapidly building custom designs |
| typescript | Typed superset of JavaScript that compiles to plain JavaScript |

Table 3: List of third party packages used for the frontend

### 7.5.1 Codebase Structure

Using Next.js allowed us to intuitively structure the routes and folders in the web application. With its file-based routing system, Next.js automatically handles the routing based on the folder structure. In our case, a clear and logical structure for accessing consultation data was established. By organising the folders as `consultation/[id]/[questionid]`, where `[id]` represents the unique identifier of a consultation and `[questionid]` represents the specific question within that consultation, we can easily define dynamic routes. Other files were created to contain reusable components such as Buttons and Inputs, to minimise code repetition, and these are contained within a `components` folder.

### 7.5.2 User Interface

Next.js provides a `layout.tsx` file which defines the layout for the current route (i.e., the containing folder), so we added a header component that persists on each page and contains a dropdown to select any consultation, making it easy for the user to change consultations, no matter what page they are currently looking at.

The topic modelling functionality discussed in 7.3 was implemented as part of the question analysis page. This page contains a tab to visualise the topics and sentiment for a particular questions, as shown in appendix B. The topics are represented by bar charts containing the topic-word distribution which

displays the 10 most prevalent words within the topic. Initially, the three most common topics are displayed to the user, but they can select and filter responses by topic.

The sentiment analysis functionality discussed in 7.2 is shown under the same tab. The sentiment distribution is visualised in a bar chart displaying the percentage of positive and negative responses. An example is shown in appendix B.

### 7.5.3 Server Communication

The frontend communicates with the backend using a set of well-defined API endpoints. The API was designed to be modular and easy to read by having reusable functions like `getRequest` to abstract away the complexities of making HTTP requests. This function is used by various other functions to fetch specific data from the backend, such as retrieving responses for a question or searching for responses based on a search term. The API endpoints follow a consistent naming convention, making it easy to understand their purpose. For example, `/api/responses/:questionid` retrieves responses for a specific question. The frontend also handles query parameters, such as pagination offsets and limits, to control the amount of data returned by the backend and prevent slow responses.

## 7.6 Backend Implementation

The backend was built using Flask, which was again chosen mainly for familiarity reasons. It also provides extensive documentation and integration with various Python libraries. The backend bridges the frontend and the SQLite database, handling client requests by retrieving and processing data. Table 4 outlines the main packages used in the backend implementation along with their descriptions. The table outlining all packages used for the deep learning analysis can be found in appendix C.

| Package | Description |
|---|---|
| numpy | Numerical computing library for Python |
| Flask | Lightweight WSGI web application framework in Python |
| matplotlib | Comprehensive library for creating static, animated, and interactive visualizations in Python |
| nltk | Natural Language Toolkit library for Python |
| pandas | Data manipulation and analysis library for Python |
| scikit-learn | Simple and efficient tools for predictive data analysis in Python |
| demoji | Library for dealing with emojis in Python |
| sentence-transformers | Library for state-of-the-art sentence embeddings in Python |

Table 4: List of third party packages used for the backend

### 7.6.1 Structure

The backend provides RESTful API routes that the frontend uses to access consultation data and analysis results. These endpoints are written in several files to separate concerns in terms of the type of data that is being retrieved. For instance, endpoints retrieving responses data are written in `responses.py`, whilst endpoints relating to topic retrieval are written in `topics.py`. This improved the manageability of the code as it ensured shorter files which were easier to modify and understand. There is also a `db.py` file that handles the connection with the SQLite database.

In order to optimise queries, caching was used to cache the endpoints for a given set of parameters [5]. This means that when the client makes the same request twice (such as requesting the responses for the same question), the second time will be much quicker as the server saves the previous result, thereby skipping unnecessary work.

### 7.6.2 Serving AI Analysis Results

As discussed in sections 7.4, the backend performs data processing for several tasks such as semantic searching, to output the results of the NLP tasks performed in the Jupyter Notebook in a structured format that can be easily

visualised by the user. For example, when the user performs a semantic similarity search, the backend finds the most similar responses and returns them in JSON format along with their similarity scores, so that the frontend can easily output a table of information. To facilitate semantic similarity search, the Sentence Transformers library [49] was utilized to load the transformer model that generated the semantic embeddings for consultation responses. As the embeddings for all responses are already stored in the database (see Section 6.2), it saves time as the backend only has to generate embeddings for the user input. The implementation leverages efficient data structures, such as dictionaries, to store and retrieve the embeddings whilst preserving the information relating to each response, so that the response text can be retrieved after processing the corresponding embedding.

## 7.7 Testing

This section will cover the testing that was conducted to ensure reliability and correctness. In the Jupyter Notebook, the functions were naturally tested by running them in subsequent cells with different parameters. With this incremental approach, each cell was tested before moving onto the next functions. For example, sentiment analysis functions were completed and tested before moving onto the topic modelling functions.

Unit tests were written to test the backend functionality, as this contained the important data processing and API endpoints that retrieved and sent data back to the frontend. The frontend was not as rigorously tested due to time constraints and it only being a secondary objective for our project. Nevertheless, during development of the user interface, the functionality was constantly being validated by trying out all the features and fixing any bugs that occurred.

The backend routes and API endpoints were thoroughly tested using the unittest framework in Python [24]. Various test cases were developed to cover every scenario and edge cases for each API endpoint. These are outlined in Table 11 of appendix A. Figure 27 in appendix A shows all of the tests passing, and a code coverage report was conducted which shows almost every line of code was covered in the backend (see Figure 28 in appendix A).

In particular, the unit tests tested the individual functions in isolation to verify their correctness, by passing in different input parameters and comparing the expected outputs with the actual results.

By testing the web dashboard and the integrated AI analysis components, we validated the end-to-end functionality of the system. This testing approach ensured that the insights derived from the Jupyter Notebook were correctly translated and presented to the users through the web interface.

# 8 Evaluation and Critical Appraisal

In this chapter, the "Decoding Democracy" project will be evaluated against several criteria to assess its effectiveness in analysing public consultation responses and providing actionable insights. The evaluation will focus on comparing the analysis obtained from the project's AI techniques with the official reports published by the Scottish Government. Additionally, the performance of key components such as the ontology, sentiment analysis, topic modelling, and semantic response similarity will be critically appraise and compared against similar work. Finally, the project requirements outlined in Section 3 will be revisited to validate if they have been met.

## 8.1 Government Analysis Comparison

To evaluate the usability and practicality of the system developed, it was important to compare the analysis features from the web dashboard against the manual analysis performed by the government. This comparison will help to see whether the key takeaways that were found by the Scottish Government could also be found in the web dashboard, and thereby evaluate how useful and effective the dashboard is. With there being a vast number of questions and responses, the comparison will involve selecting a few example questions and extracting insights from both the web dashboard and the government website to see if there are similarities.

### 8.1.1 Draft Referendum Bill Analysis

For the draft referendum bill consultation [13], we will focus on the first Question: *"What are your views on the proposed arrangements for managing the referendum?"*. To give some context on what the proposed arrangements were, they are summarised in Table 5.

| **Proposed arrangement** | **Description** |
|---|---|
| Eligibility to vote | The referendum was to follow the same eligibility criteria as Scottish local government and Scottish Parliament elections, meaning any British, Commonwealth or EU citizen residing in Scotland could vote. Additionally, 16 and 17-year-olds were eligible to vote. |
| Absent Voting | Proxies must be registered voters, confirming their identity through the individual electoral registration process. |
| Checking of Postal Votes | The draft bill proposes 100% checking of personal identifiers on postal voting statements, aligning with previous practice and accepted norms at other elections. |
| Polling and Count Staff | It was prohibited for the counting officer to appoint individuals involved in campaigning during the referendum. |
| Verification Statements | Counting officers were mandated to provide copies of verification statements confirming ballot counts to any counting agent upon request. |

Table 5: Proposed arrangements for the draft referendum bill

The government's analysis found several underlying themes in citizen responses.

1. Concerns about security and potential electoral fraud, particularly regarding postal voting, handling of ballot boxes, and neutrality of polling staff. Respondents suggested measures to tighten security.

2. Support for prohibiting the appointment of campaigners as polling staff, though some felt this didn't go far enough to ensure neutrality.

3. Disagreement on voter eligibility, especially regarding Scottish expatriates' right to vote. Some argued for voting rights based on being born in Scotland, while others supported the proposed residence-based franchise. Concerns were raised about holiday home owners registering to vote without being genuine residents.

4. Divided opinions on allowing 16-17-year-olds and EU/Commonwealth citizens to vote.

Overall, while some respondents endorsed the proposed arrangements, many expressed concerns and disagreements, particularly around ensuring the security and fairness of the referendum process. There were calls for stricter measures and independent oversight to build public confidence in the outcome. This is represented in the web dashboard's sentiment distribution chart shown in Figure 17, as there seem to be more negative concerned (or negative) answers than positive.

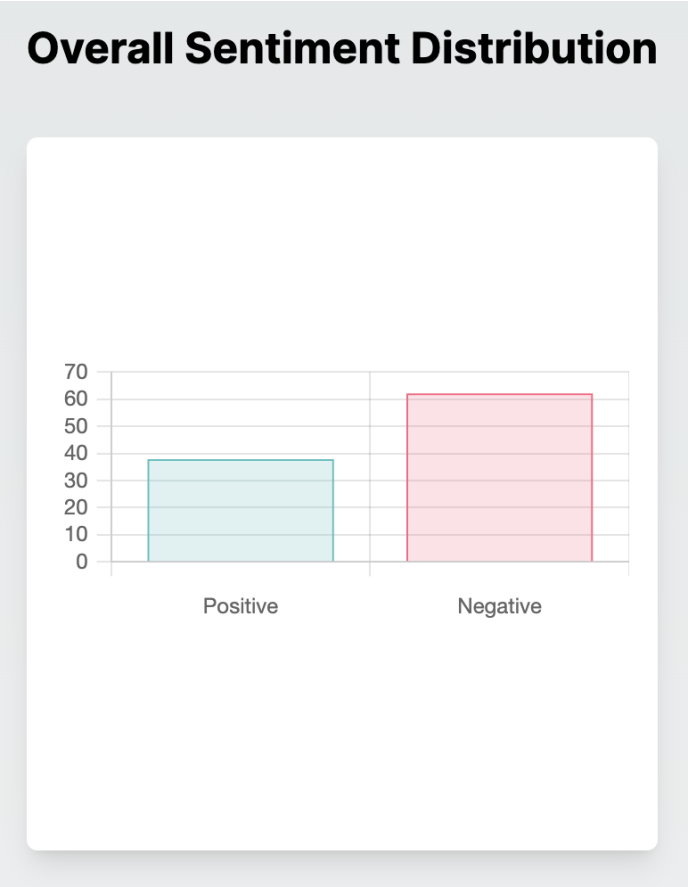



Figure 17: Sentiment distribution for question 1 of consultation 1

Looking at some of the topics found in the responses to question 1, we can see that there were 419 responses that clearly agreed with the proposed arrangements, which is around 6% of the total responses. This is shown in Figure 39 under topic 55.

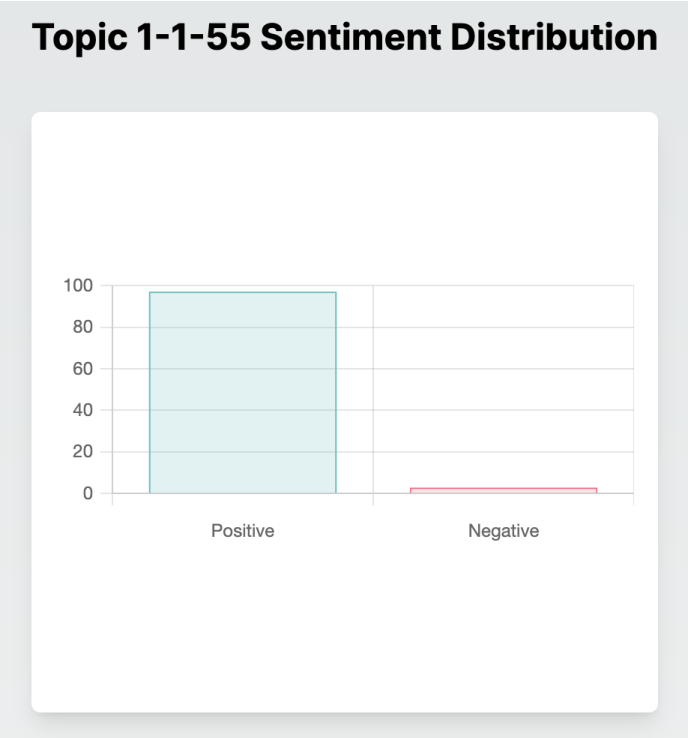



Figure 18: Distribution of words for topic 55 in question 1 of consultation 1

| No. | Response | Sentiment Value | Topic ID |
|---|---|---|---|
| 22 | The existing arrangements work well so the decision to replicate these is sound. | 0.281 | 55 |
| 24 | I agree with the proposed arrangements as detailed in the paper | 0.276 | 55 |
| 44 | I agree | 0.35 | 55 |
| 55 | I am happy with the proposed arrangements for managing the referendum. | 0.948 | 55 |
| 71 | I agree with all these proposals | 0.217 | 55 |
| 100 | I broadly agree with the proposed arrangements | 0.354 | 55 |
| 107 | The proposed arrangements are fitting and well explained | 0.314 | 55 |
| 108 | I agree | 0.35 | 55 |

Figure 19: Responses under topic 55, agreeing to question 1, consultation 1

Figure 19 displays some example responses, which all clearly agree with the question.

Another common topic was topic 4, which is shown in Figure 20.

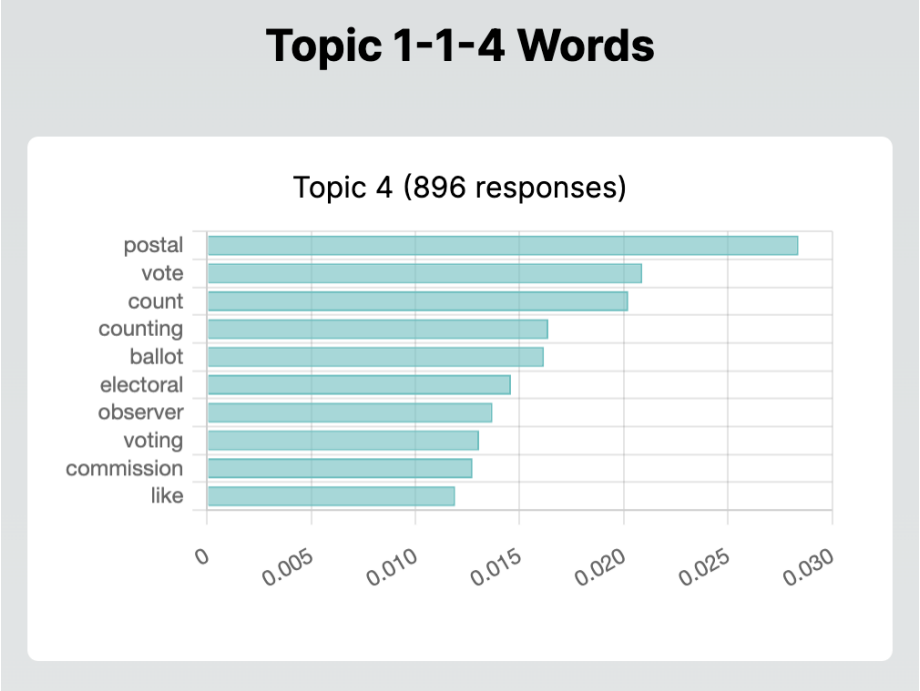



Figure 20: Distribution of words for topic 4 in question 1 of consultation 1

As shown by the bar chart, commonly occurring words included "postal", "vote", "counting" and "ballot", suggesting that responses under this topic were concerned with the security around postal voting and ballot boxes. Answers like "I would like to see better security for ballot boxes" expressed their concerns for potential electoral fraud, which the government analysis also touched upon (see Table 5).

After looking at question 1, we can see that analysing the responses yielded similar conclusions about public opinion to the analysis conducted by the government. This process can be applied to the other open ended questions, by observing the charts of the most common topics and seeing what some of the answers were to gauge the general opinions within that topic.

### 8.1.2 Electoral Reforms Analysis

The next two consultations contained many more multiple choice questions than the first one, providing several benefits over open ended questions in terms of analysing responses. Firstly, they provide a consistent set of responses that are easier to compare and quantify. Multiple choice questions are also clearer and less open to misinterpretation, making the consultation

process more accessible and less demanding for respondents. Nevertheless, a combination of both multiple choice and open ended questions is used in these consultations to provide a comprehensive understanding of respondents' views.

One multiple choice question that was asked in the first electoral reform consultation was *"Do you agree that the franchise should be extended to include everyone legally resident in Scotland?"*, with franchise referring to voting rights in Scottish Parliament and Local Government.

**Table 8.1: Q17 – Do you agree that the franchise should be extended to include everyone legally resident in Scotland?**

| Respondent type | Yes | | No | | Total | |
|---|---|---|---|---|---|---|
| | n | % | n | % | n | % |
| **Organisations** | 24 | 92% | 2 | 8% | 26 | 100% |
| **Individuals** | 568 | 78% | 157 | 22% | 725 | 100% |
| **Total** | **592** | **79%** | **159** | **21%** | **751** | **100%** |

Figure 21: Distribution of responses to question 17 of consultation 2, as shown on the government analysis page

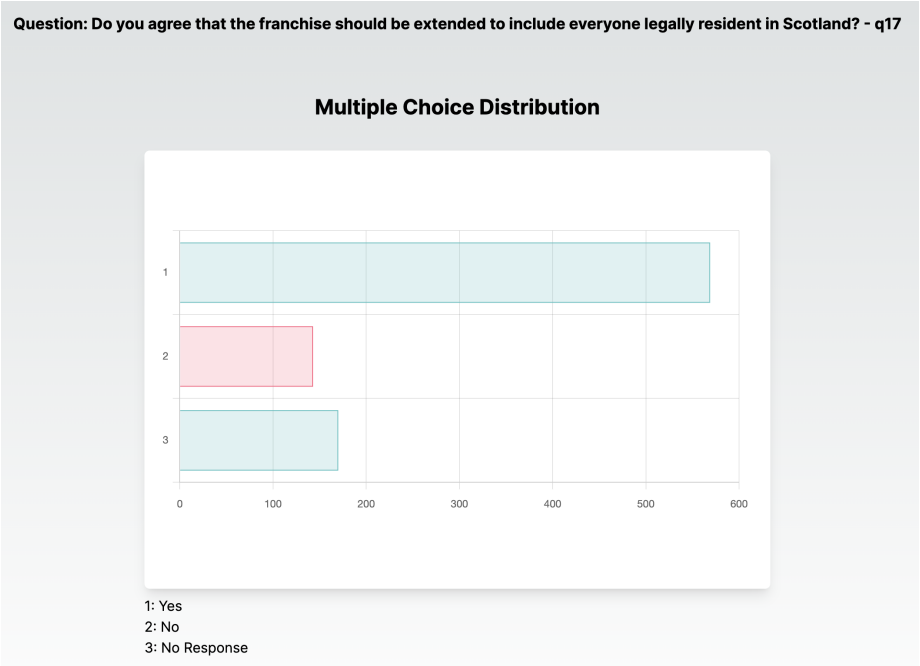



Figure 22: Distribution of responses to question 17 of consultation 2, as shown on the web dashboard

Comparing the government's table (Figure 21) against the web dashboard's bar chart (Figure 22), we can see apart from the government not including a "No Response" field for people that skipped this question, the distribution of "Yes" and "No" responses is equal, displaying close to 600 "Yes" responses compared to 150 "No's". This suggests the public is largely in favour of extending the franchise.


Figure 23: Most common topics for a question

Overall, the web dashboard provides a comprehensive way to analyse responses multiple choice questions as well as open-ended questions.

## 8.2 Database Ontology And SQLite

The database for this project was designed to capture the relevant entities and relationships from the Scottish consultation data. The key entities such as `Consultation`, `Question`, `Responder`, and `Response` are modeled appropriately, along with their respective properties and foreign key relationships. Although the ER model for the database was modified once or twice, the changes were often small and involved adding fields to entities to capture more information. For instance, a `response_embeddings` field was added to the `Question` table after deciding on how to store topic modelling and semantic similarity information in a way that could easily be retrieved by the web dashboard.

Throughout the development process, this database provided a straightforward way to query all the necessary information for each stage of the project. Having said this, the choice of using SQLite to implement the database is perhaps a limiting feature. While SQLite was the natural choice initially for its simplicity, it lacks scalability and more advanced features required for larger-

scale applications. For this reason, it may be sensible to move to another SQL database in the future like PostgreSQL or MariaDB, as these offer benefits regarding remote hosting and performance, which could make the database more robust and scalable.

## 8.3 Sentiment Analysis

As discussed in Section 7.2, the sentiment analysis performance was evaluated by comparing the results obtained from different models (Vader, TextBlob, RoBERTa, and DistilBERT) against manually assigned sentiment values for a sample set of responses. The evaluation revealed that RoBERTa produced the most accurate sentiment values, with the majority of its predictions closely aligned with the human-assigned values. However, considering the computational efficiency and resource requirements, the "student" model, DistilBERT [20] was chosen as the preferred model because it demonstrated comparable performance to RoBERTa while being more lightweight and faster to execute.

The sentiment analysis using DistilBERT was applied to all open-ended responses across the three consultations, and the results were visualised using bar charts, displaying the distribution of negative and positive sentiments for each question.

In summary, the sentiment analysis conducted provides valuable insights into the overall sentiment expressed by respondents for specific questions, and even helps users find outliers by ordering responses to see the most negative or most positive ones. This proved to be an effective way of finding passionate responders with interesting opinions, as the extremely negative responses would usually be much longer than the average response. However, sentiment analysis alone was not enough to inform users of the dashboard what the public was concerned or pleased about, which is why topic modelling was also implemented.

## 8.4 Topic Modelling

The technique used for topic modelling that was discussed in Section 7.3, involving a combination of clustering techniques and Latent Dirichlet Allocation (LDA), was effective in automatically determining the optimal number of topics for each question, overcoming the challenge of manually tuning the LDA model for each set of responses. The agglomerative clustering algorithm was chosen for its ability to handle arbitrary cluster shapes and its deterministic nature, ensuring consistent results across multiple runs.

While this approach provided a practical way to estimate the number of topics, time constraints meant that more advanced techniques to evaluate models such as topic coherence measures were not employed. A paper on "Optimizing Semantic Coherence in Topic Models" by Mimno et al. [33] discussed ways to measure topic coherence from LDA. They propose a formula to calculate the coherence of a topic, namely the UMass measure, by considering the co-document frequencies of word pairs within the topic. Their metric aims to provide a quantitative measure of how well words within a topic relate to each

other, indicating semantic consistency. Another paper titled "Automatic evaluation of topic coherence" [35] evaluated several topic scoring methods and found that PMI (topic co-occurrence) was the best-performing method. The current implementation relies on qualitative assessment and manual inspection of the generated topics and their associated words to gauge the effectiveness. The top words for each topic were visualized using word clouds and bar charts, allowing for a visual evaluation of the topic coherence and relevance to the underlying question. Whilst the generated topic labels were sometimes representative of the main themes within the responses, there were also examples were they did not clear indicate the underlying concerns or opinions.

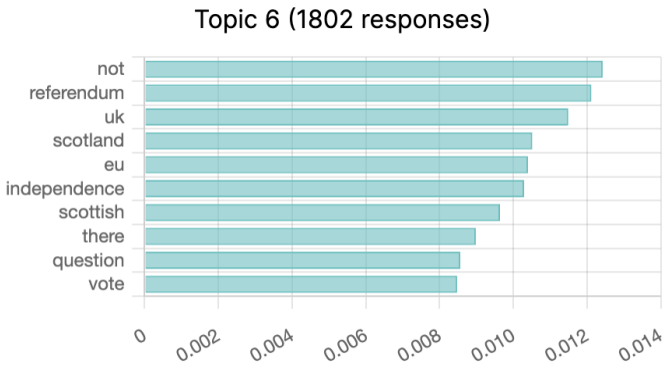



Figure 24: Example of a relatively unclear topic for question 1 of the draft referendum bill consultation

Figure 24 is an example of a discovered topic that is not very coherent. The words in this topic do not really indicate an underlying concern or opinion, making it hard to understand what ties the responses within this topic.

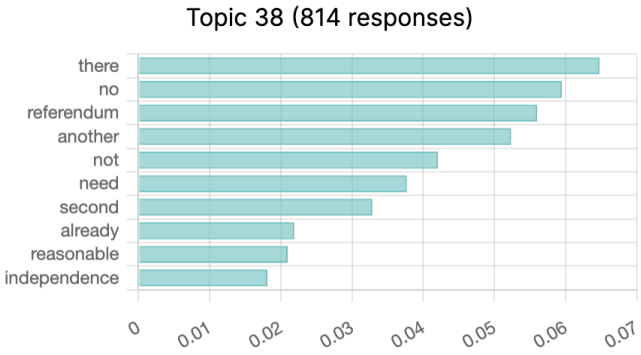



Figure 25: Example of a more coherent topic for question 1 of the draft referendum bill consultation

On the other hand, Figure 25 is an example of a topic that gives a better picture of the responses within it, as the words suggest that the public thinks there is no need for another referendum.

To improve the evaluation of the topic modelling process, future work could involve implementing topic coherence measures. Additionally, exploring other topic modelling techniques, such Hierarchical Dirichlet Process (HDP) [64], could potentially provide better results in terms of the quality of extracted topics.

In conclusion, while the topic modelling evaluation process could be improved, the combination of clustering and LDA proved to be an effective approach for discovering meaningful topics within the consultation responses, offering valuable insights into public opinions and concerns.

## 8.5 Semantic Response Similarity

Response similarity searching enabled users to find responses that are semantically similar to a given input text, by leveraging the power of sentence embeddings and cosine similarity. To implement this feature, the project utilized a Sentence Transformers (SBERT) model [47], specifically the `all-MiniLM-L6-v2` variant which is more lightweight and efficient, to generate embeddings for each response.

Similarly to topic modelling, the effectiveness of the semantic response similarity feature was evaluated through manual testing and qualitative assessment. The system was tested with various search queries, and the retrieved responses were examined for their semantic relevance to the input text. In most cases, the system successfully identified responses that conveyed similar ideas or sentiments, even when the exact words used were different.

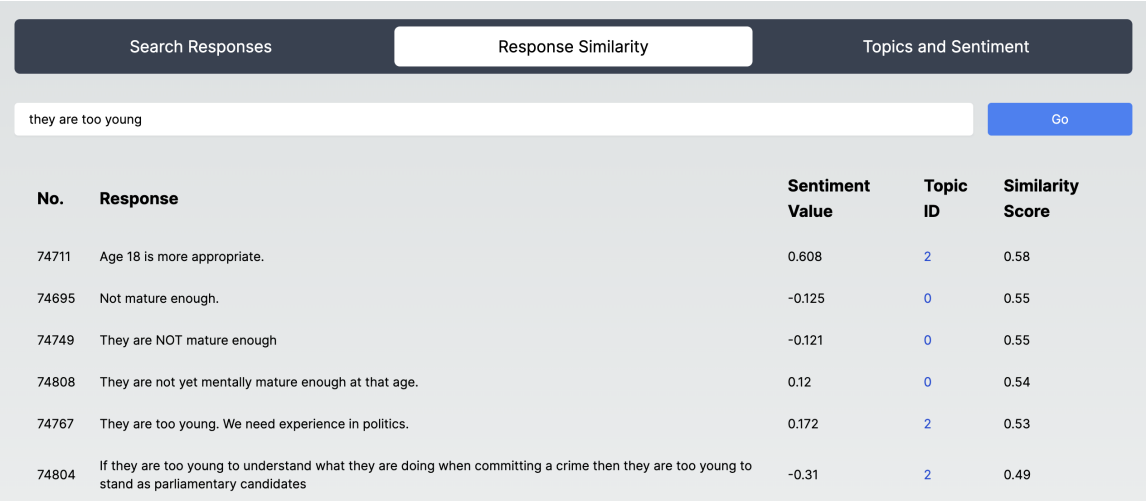



Figure 26: Example of semantic similarity searching with the text "too young". The question was regarding allowing 16/17 year olds to vote.

Figure 26 is an example of well performing semantic similarity search, as it brings up responses that have the same meaning as the phrase entered by the user.

However, the main limitation with this feature was that the current implementation only considers the response text itself for semantic similarity. Incorporating additional context, such as the question associated with the responses, could provide a better understanding of the semantic relatedness. Also, trialling asymmetric semantic search methods [49] on top of the symmetric semantic search that we use now (see Section 7.4) could provide a better metric for response relevance to the question. Asymmetric search involves comparing a short query (a question) with longer paragraphs (responses) answering the query.

Despite these limitations, the similarity search was an effective approach given the time constraints, as it managed to consistently provide good results with low response latency on the client-side.

## 8.6 Web Dashboard

Although the web dashboard was not the main objective of this project, it was implemented in a way that supports several features, and received positive feedback after being reviewed by Dr Tarik Olcay, who primarily researches constitutional change, and my supervisor, Dr Rosa Filgueira. It provides a clear user interface with simple navigation, incorporating the key findings from the deep learning analysis. However, to make it a more comprehensive tool for consultation analysis, more features could be added such as charts for exploratory data analysis, displaying other valuable insights and statistics for each consultation.

With a key requirement being that the web dashboard can be run across multiple environments (see Section 3), it was tested on both MacOS and Linux, and can be found running on my host server bac4.teaching.cs.st-andrews.ac.uk. The Jupyter Notebook, on the other hand, was written using Google Colab [26], which is not dependent on the operating system.

## 8.7 Requirements Validation

The tables below map the implementation of features described in Section 7 to the requirements outlined in Section 3. All requirements have been successfully fulfilled and are supported by appropriate evidence shown in the reference column.

### 8.7.1 Database Requirements

| Num | Description | Reference |
|---|---|---|
| **Functional Requirements** | | |
| 1 | The system must support the development of a strong Entity-Relationship (ER) model to encapsulate information related to consultations including the questions, responses and topics. | Section 6.2 Figure 2 |
| 2 | Capability to import and store responses from at least three distinct consultations. | Section 6.2 Figure 2 |
| 3 | Mechanisms for data cleaning and transformation during the importation process, to ensure consistency in how the responses and questions are stored. | Section 6.3.2 |
| 4 | Automation of the population of database tables from CSV files. | Section 6.3.2 |
| 5 | Functionality to distinguish and categorise multiple-choice questions from open-ended questions for analysis. | Section 6.1 |
| **Non-functional Requirements** | | |
| 6 | Enforcing referential integrity through foreign keys to ensure consistency of data. | Section 6.2 |
| 7 | Optimisation for efficient querying and data retrieval to support the large number of responses. | Section 7.6 |

Table 6: Database Requirements

### 8.7.2 Deep Learning Analysis Requirements

| Num | Description | Reference |
|---|---|---|
| **Functional Requirements** | | |
| 8 | Evaluate the performance of several ML models for sentiment analysis and compare the results with human-annotated sentiment scores before selecting the most effective and accurate model for extracting meaning out of the consultation responses. | Section 7.2 Figure 9 |
| 9 | Responses should be preprocessed in the same way before being passed into any models to ensure consistent and comparable analysis for any response. This means removing stopwords, lemmatizing to reduce words to their base form, and removal of punctuation and emojis to prevent models from interpreting similar responses differently. | Section 7.1 |
| 10 | Apply ML techniques to perform sentiment analysis on consultation responses. | Section 7.2 |
| 11 | Utilise Clustering and LDA (Latent Dirichlet Allocation) to dynamically discover topics within open-ended responses. | Section 7.3 |
| 12 | Generate and store text embeddings for responses. | Section 7.3 |

Table 7: Deep Learning Analysis Requirements

| Num | Description | Reference |
|---|---|---|
| 13 | The system should label clusters and topics with descriptive keywords based on the content of responses within each group. | Section 7.3 |
| 14 | Implement mechanisms to update the database with sentiment values and topic assignments for each response. | Section 6.3.2 |
| 15 | Provide support for data visualisation, including word clouds for topics and scatter plots for cluster visualisation. | Chapter 7 |
| **Non-functional Requirements** | | |
| 16 | The jupyter notebook should be executable on google colab [26], using the free resources available (i.e. the TPU, GPU or CPU runtimes). | Chapter 6 |
| 17 | Choice of a model that can generate accurate sentiment analysis and topic modelling on a large corpus of responses without taking too much time, so that the system is able to expand and manage more responses in the future. | Sections 7.2 and 7.3 |
| 18 | The system must be maintainable and supported by documentation. | Chapter 6 |

Table 8: Deep Learning Analysis Requirements Continued

### 8.7.3 Web Tool Requirements

| Num | Description | Reference |
|---|---|---|
| **Functional Requirements** | | |
| 19 | Allow users to select one of the three consultations to analyse. | Sections 7.5 and 7.6 |
| 20 | Allow users to see all question from a consultation. | Sections 7.5 and 7.6 |
| 21 | Allow users to analyse the distribution of responses for multiple choice questions with a bar chart. | Sections 7.5.2 |
| 22 | Allow users to view all the responses for open ended questions, with search functionality matching specific words. | Sections 7.5, 7.5.2 and 7.6 |
| 23 | Allow users to search for similar responses to a certain input. | Sections 7.5, 7.5.2 and 7.6 |
| 24 | Allow users to view the sentiment distribution for each question, with positive and negative bars on a bar chart. | Section 7.5.2 |
| 25 | Allow users to see what the common topics were for each question, and what the sentiment distribution is for each topic. | Section 7.5.2 |

Table 9: Web Tool Functional Requirements

| Num | Description | Reference |
|---|---|---|
| **Non-functional Requirements** | | |
| 27 | The interface for the web dashboard must be easy to use, simplifying the process of selecting consultations, viewing analysis results, and understanding insights. | Appendix B and Section 7.5.2 |
| 28 | Ensure the web tool is accessible to non-technical users such as policy makers and the general public, incorporating guidelines for usability and accessibility. | Appendix B |
| 32 | The website must respond to queries quickly, avoiding long waiting times when searching for responses and viewing analyses. | Section 7.6 |
| 33 | The system must be designed to be maintainable so that updates can be published quickly and additional functionalities can be implemented as required. | Chapter 6 |
| 34 | The user interface must be responsive and usable across various screen sizes, such as phones and desktop computers. | Section 6.4.1 and appendix B |
| 35 | The system should be supported across multiple operating systems, including MacOS and Linux. | Section 8.6 |

Table 10: Web Tool Non-Functional Requirements

# 9 Conclusions

The main objective of "Decoding Democracy" was to analyse and extract valuable insights from three Scottish consultations. By creating a robust database framework to store consultation data, we were able to apply deep learning techniques to gain insights from the responses, and develop an intuitive web interface for users to explore the results. In evaluating the project, we compared our analysis results with the official reports published by the Scottish Government. The sentiment analysis and topic modelling outcomes aligned well with the government's findings, validating the effectiveness of our approach. As outlined by the requirements validation section (see Section 8.7), the requirements set out at the start in Chapter 3 were mostly met to a high standard, however there were some limitations and areas for improvement which will be discussed.

## 9.1 Key achievements

1. Developed a comprehensive database framework to capture and information about consultations, questions, responses, and topics.

2. Applied natural language processing techniques:

   (a) Sentiment Analysis: Multiple deep learning models were evaluated to accurately gauge public sentiment on various aspects of the consultations, and the chosen DistilBERT model was used to calculate a sentiment score for every response in the database.

   (b) Topic Modelling: A hybrid approach combining clustering techniques and Latent Dirichlet Allocation (LDA) was employed to uncover key themes and concerns within the consultation responses, whilst automatically determined a suitable number of topics for each question.

3. Implemented semantic similarity search to enable users to find responses similar to their queries, and have a more personalised exploration of the data.

4. Developed an intuitive web dashboard to allow users to easily navigate through consultations and view analysis results with some visualisations too. The dashboard was designed to be accessible to non-technical users, such as policymakers and the general public, ensuring that the insights derived from the project could be widely understood.

5. Conducted testing of the backend functionality and the integration of the AI analysis components, utilising the unittest framework and end-to-end testing of the AI analysis features.

6. Evaluated the project's results against the official government analysis, confirming the effectiveness of the developed techniques.

7. Ensured the project is reproducable by providing all necessary resources and a detailed Readme file in the GitHub repository.

## 9.2 Drawbacks and Future Work

While much of the project was successfully implemented, this section will outline some of the limitations. Firstly, the topic modelling process could be further refined by evaluating the techniques against topic coherence measures, as discussed in Chapter 8. Although it would be hard to measure the topic coherence across so many different questions and contexts, this would provide a more data-driven approach to finding the best model for topic discovery. Currently, the topic modelling results can sometimes be unclear for certain topics, so perhaps thinking of a more adaptable approach would be appropriate. For example, finding a way to pre-train models with contextual information regarding the consultations may produce better results. Alternatively, with Large Language Models becoming increasingly competent, their text summarisation capabilities could be used to provide users with a comprehensive summary of the key themes within responses. This could also be extended to the web dashboard, by implementing a "chat" feature which allows user to ask questions about each consultation.

Another feature that has room for improvement is the semantic similarity feature (discussed in Section 7.4), which could be optimised by trialling asymmetric semantic searching to find the responses that answer each question with most relevancy. Currently, the search functionality matches user input based on length, which is somewhat limiting, because if the user wants to find responses that convey an emotion like "anger", the semantic search in our system would not be optimal.

Finally, the current system relies on manually copying the database from the Jupyter Notebook to the backend of the web dashboard. To improve this, a better workflow could be established to automatically update the database when the notebook was ran, as this would enable real-time updates for users of the web tool. Also, whilst adding new consultations would not be too difficult, this process could also be streamlined through a more adaptable python script. Currently, the consultation data is provided through excel files, and the format differs for each consultation. This means the python script to convert the excel files into a consistent format contains some hard-coded variables to extract certain columns etc.

## 9.3 Reflection

In summary, the key achievements demonstrate the successful implementation of the "Decoding Democracy" project, and ultimately show the potential of combining natural language processing with web technologies to extract valuable information from large volumes of textual data. Overall, it was an incredibly rewarding project that provided the opportunity to combine several valuable skills including database design, deep learning, and web development.

The insights gained from this work are exciting and can hopefully be built upon in the future to aid policymakers in understanding public opinion and making more informed decisions.

# A Testing Summary

| Test File | Test Description |
|---|---|
| test_consultations.py | • test_get_all_consultations: Retrieves all consultations and verifies the response status code and data type.<br>• test_get_specific_consultation: Retrieves a specific consultation by ID and verifies the response status code and data. |
| test_questions.py | • ttest_get_questions_for_consultation: Retrieves questions for a specific consultation ID and verifies the response status code and data type.<br>• ttest_get_specific_question: Retrieves a specific question by ID and verifies the response status code and data.<br>• ttest_get_wordcloud_for_question: Retrieves the word cloud for a specific question ID and verifies the response status code and data. |
| test_responders.py | • ttest_get_all_responders: Retrieves all responders and verifies the response status code and data type.<br>• ttest_get_specific_responder: Retrieves a specific responder by ID and verifies the response status code and data. |

Table 11: Backend testing summary

| Test File | Test Description |
|---|---|
| test_responses.py | • ttest_get_responses_for_responder: Retrieves responses for a specific responder ID and verifies the response status code and data type.<br>• ttest_get_responses_for_question: Retrieves responses for a specific question ID with offset and limit parameters and verifies the response status code, data type, and count.<br>• test_search_responses: Searches for responses containing a specific term within a question and verifies the response status code, data type, and count.<br>• test_sentiment_distribution: Retrieves the sentiment distribution for a specific question ID and verifies the response status code and presence of sentiment data.<br>• test_multiple_choice: Retrieves multiple-choice options for a specific question ID and verifies the response status code and data type.<br>• test_similar_responses: Retrieves similar responses for a given question ID and text input with offset and limit parameters and verifies the response status code, data type, and count. |

Table 12: Backend testing summary

| Test File | Test Description |
|---|---|
| test_topics.py | • test_topics: Retrieves topics for a specific question ID and verifies the response status code and data type.<br>• test_common_topics: Retrieves common topics for a specific question ID with a limit parameter and verifies the response status code, data type, and count.<br>• test_top_words: Retrieves top words for a specific topic ID with a limit parameter and verifies the response status code, data type, and count.<br>• test_search_responses: Searches for responses within a specific topic and question ID with offset and limit parameters and verifies the response status code, data type, and count.<br>• test_num_responses: Retrieves the number of responses for a specific topic ID and verifies the response status code and presence of count data.<br>• test_average_sentiment: Retrieves the average sentiment for a specific topic ID and verifies the response status code and presence of sentiment data. |

Table 13: Backend testing summary

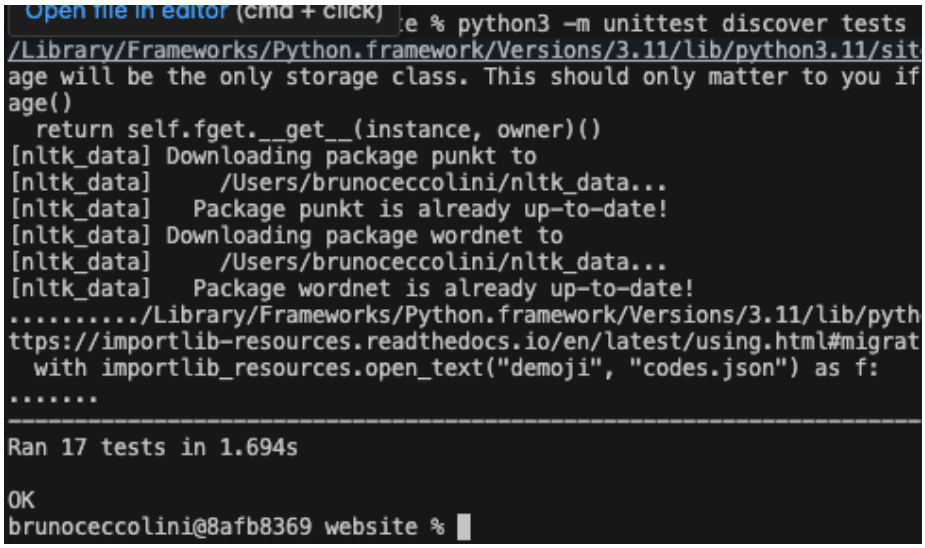



Figure 27: Screenshot showing all tests passing


Figure 28: Backend code coverage report, showing 100% or close to 100% for all files

# B User Manual

The backend and frontend are hosted on the school server at bac4.teaching.cs.st-andrews.ac.uk. Instructions on installing and running the web dashboard can be found in the project GitHub repository.

The screenshots in this section are taken from the web dashboard to display its functionality and the various components of the interface.

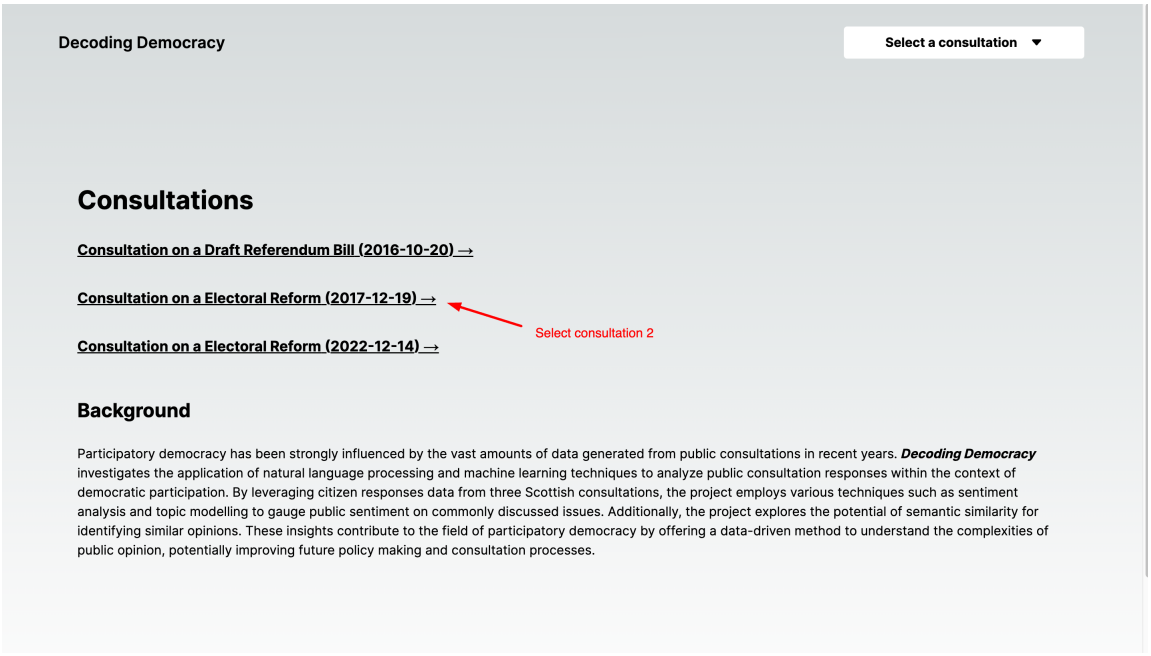



Figure 29: Home page of the web dashboard, displaying consultation links and about the project

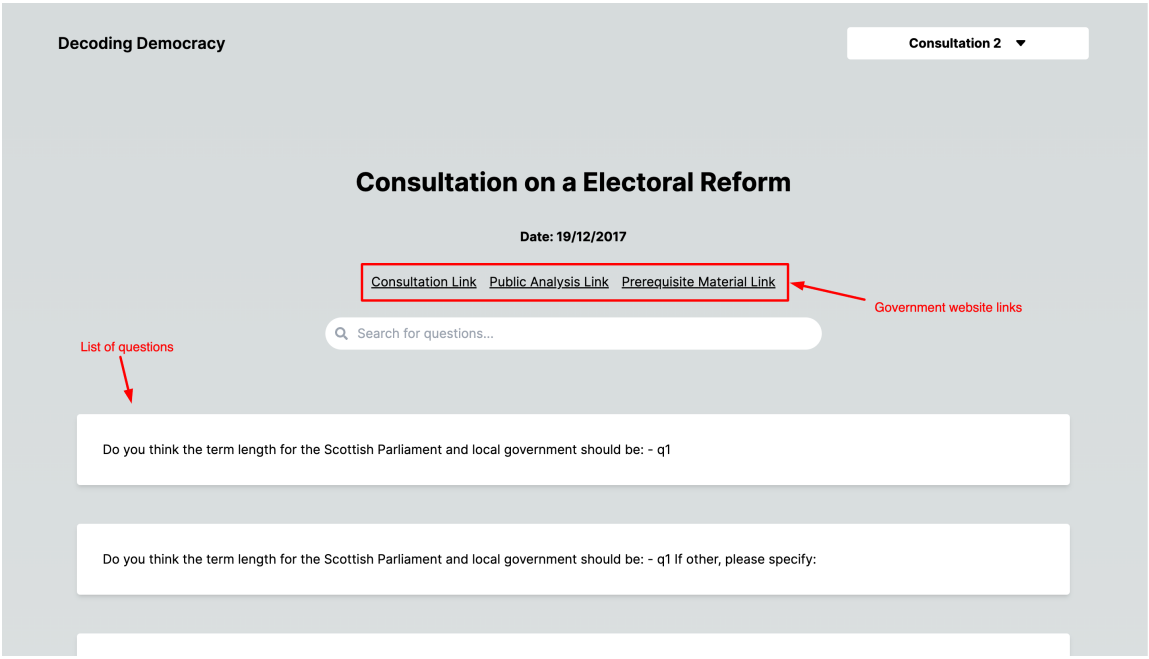
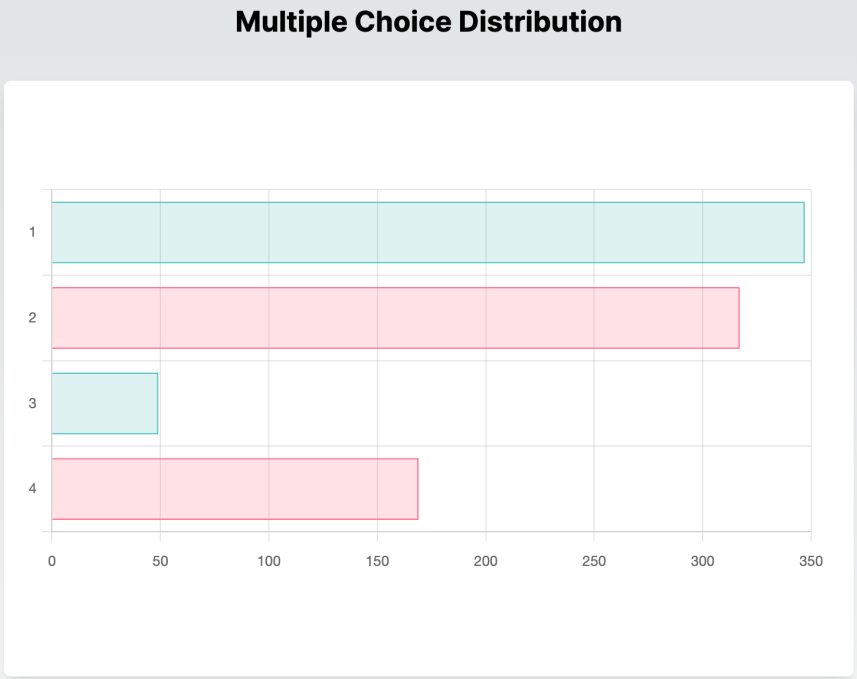



Figure 30: Page displaying a searchable list of questions for consultation 2 and links to the government website pages for the selected consultation

**Question: Do you think the term length for the Scottish Parliament and local government should be: - q1**




1: 5 years
2: 4 years
3: Other length (please specify)
4: No Response

Figure 31: Multiple choice question page, displaying responses for the question 1 in consultation 2

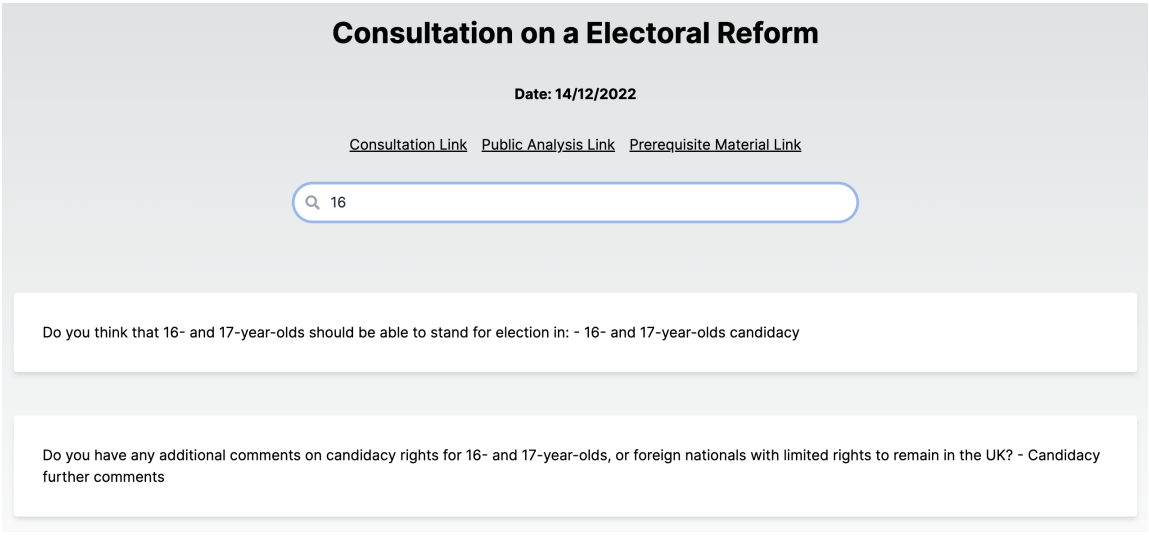



Figure 32: Filtering through questions in a consultation

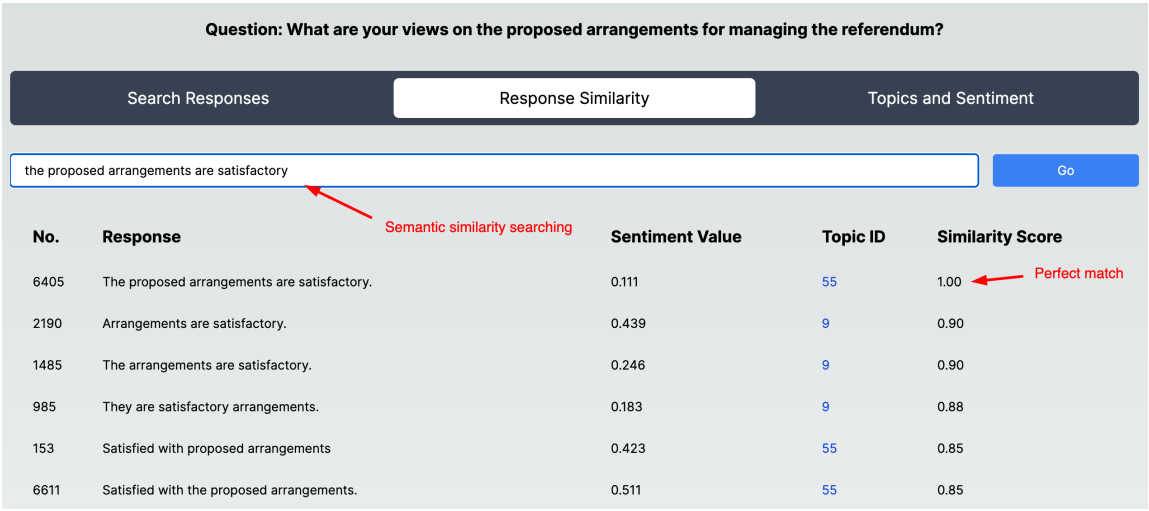



Figure 33: Searching for semantically similar responses within a question, with the similarity score shown

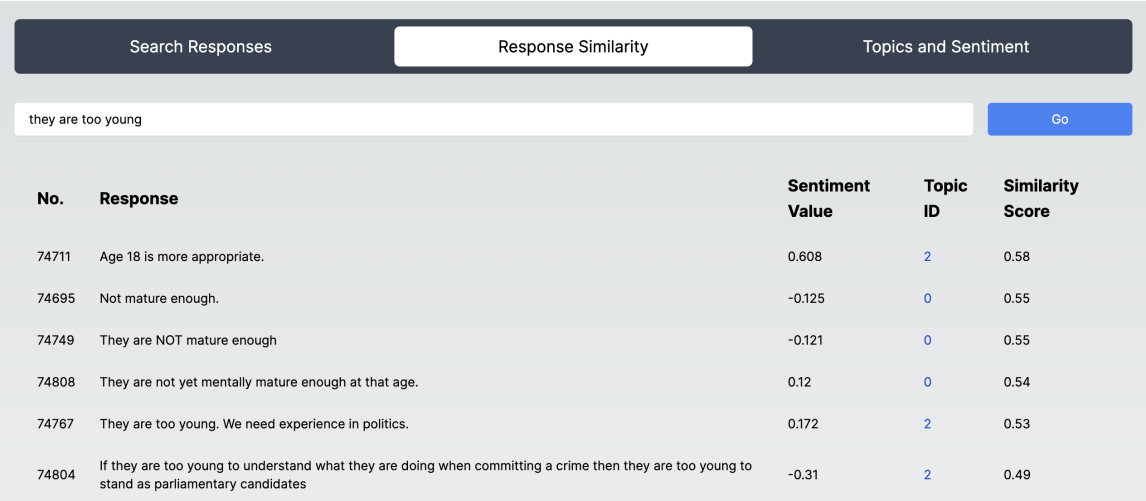



Figure 34: Searching for semantically similar responses within a question, with the similarity score shown

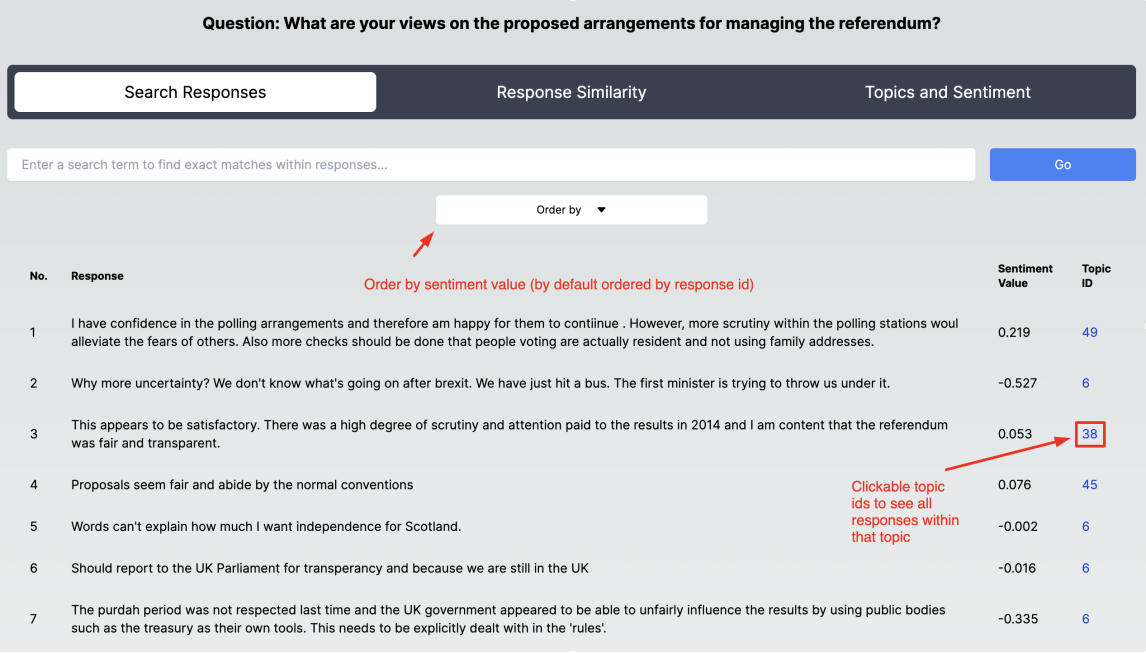



Figure 35: Responses page displaying all responses for a question, along with each response's sentiment and topic id

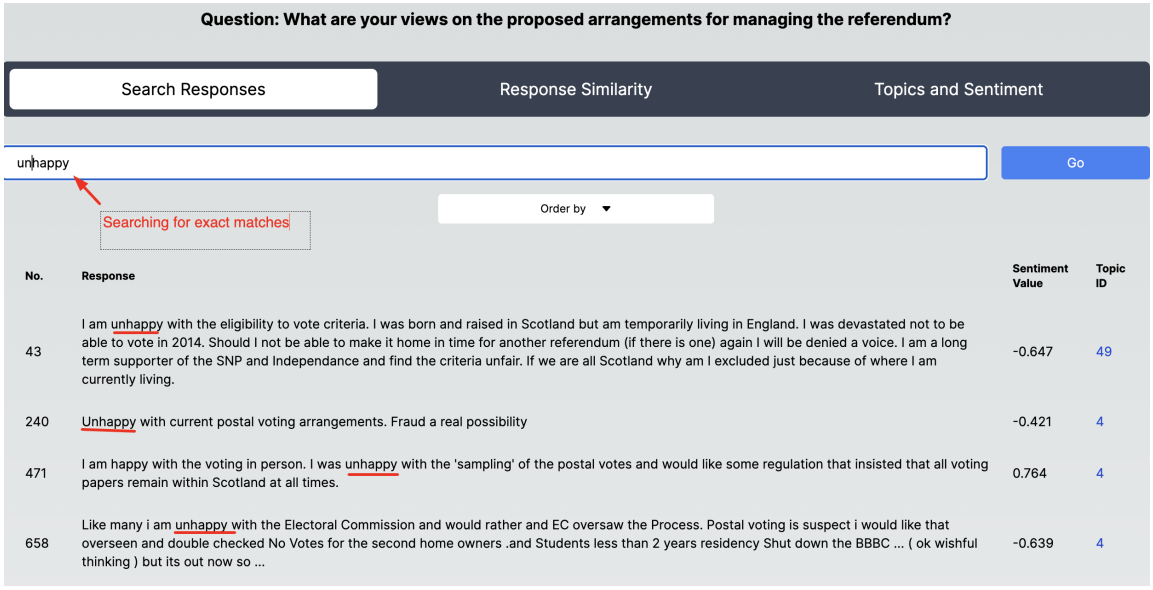



Figure 36: Exact lexical searching of words or phrases within responses

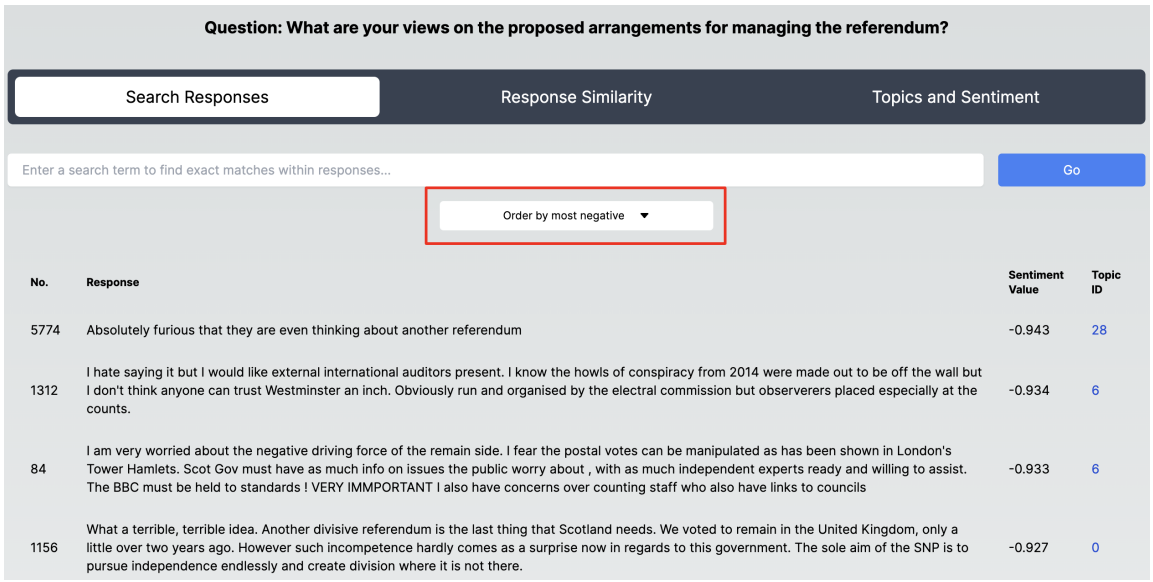



Figure 37: Responses ordered by most negative first

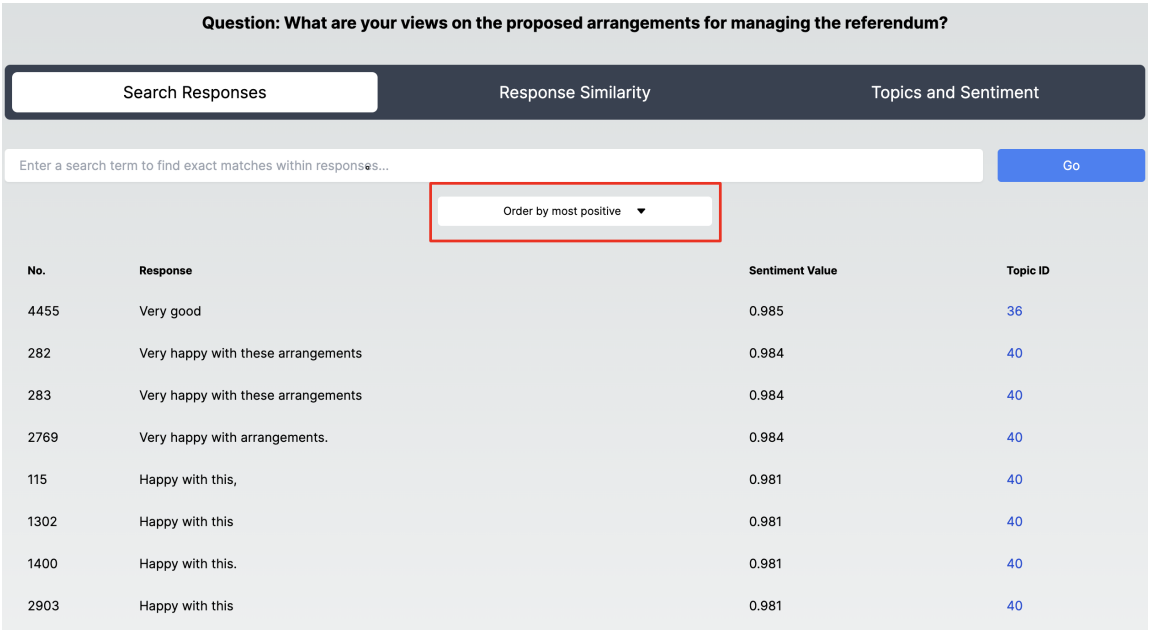



Figure 38: Responses ordered by most positive first

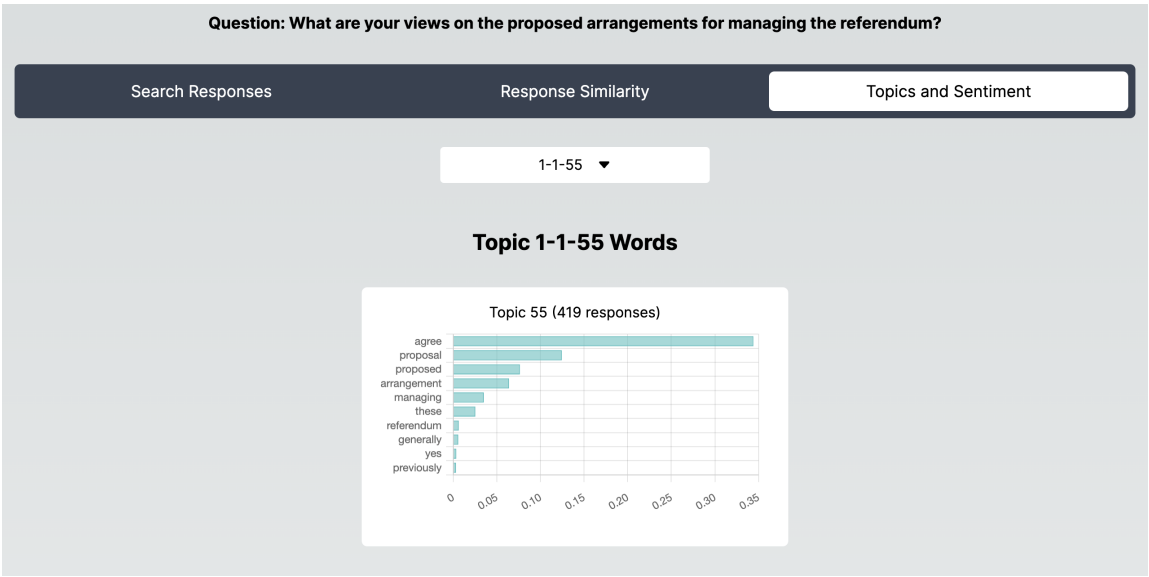



Figure 39: Bar chart showing the word distribution for topic 55

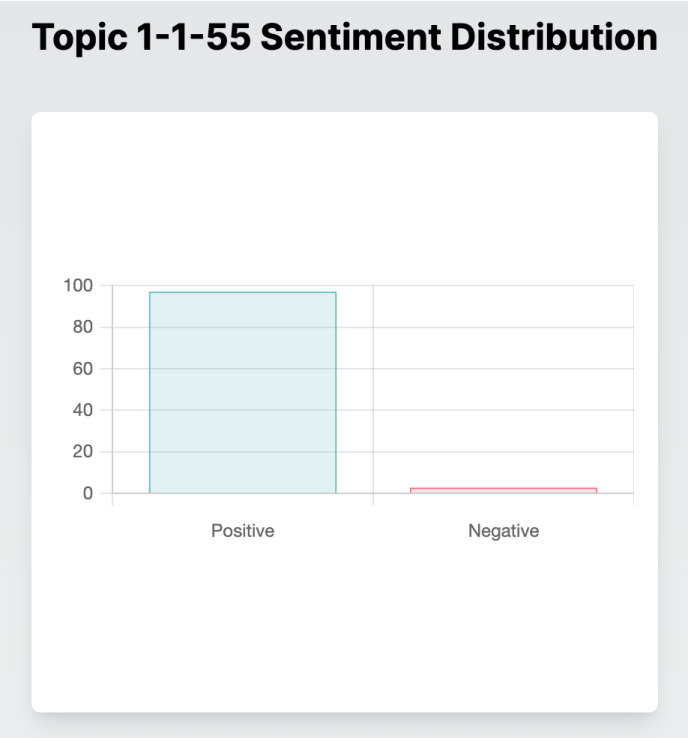



Figure 40: The sentiment distribution for responses under topic 55

| No. | Response | Sentiment Value | Topic ID |
|---|---|---|---|
| 22 | The existing arrangements work well so the decision to replicate these is sound. | 0.281 | 55 |
| 24 | I agree with the proposed arrangements as detailed in the paper | 0.276 | 55 |
| 44 | I agree | 0.35 | 55 |
| 55 | I am happy with the proposed arrangements for managing the referendum. | 0.948 | 55 |
| 71 | I agree with all these proposals | 0.217 | 55 |
| 100 | I broadly agree with the proposed arrangements | 0.354 | 55 |
| 107 | The proposed arrangements are fitting and well explained | 0.314 | 55 |
| 108 | I agree | 0.35 | 55 |
| 130 | I have no issues with the proposals and management of any proposed independence referendum, nor with the voting franchise proposed. | -0.024 | 55 |

Figure 41: All responses under the topic id 55

## B.1 Responsiveness of the website - using a mobile phone

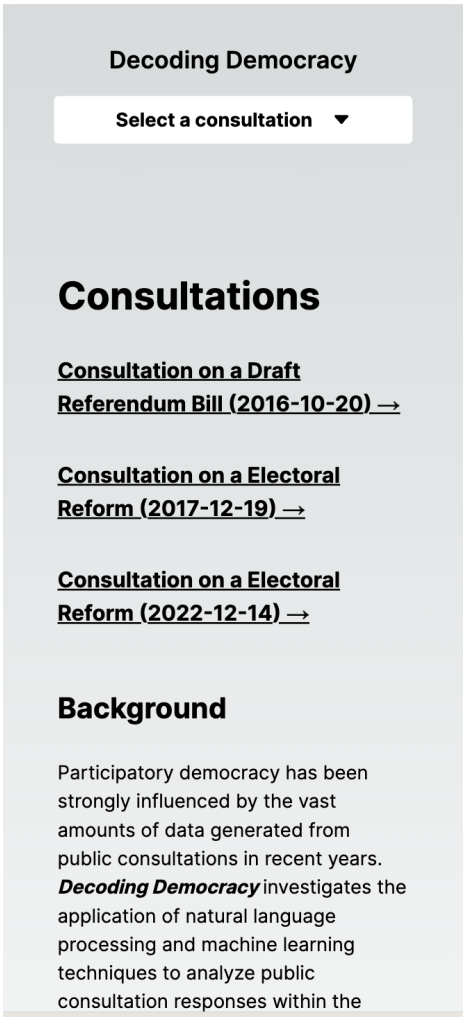



Figure 42: Mobile home page

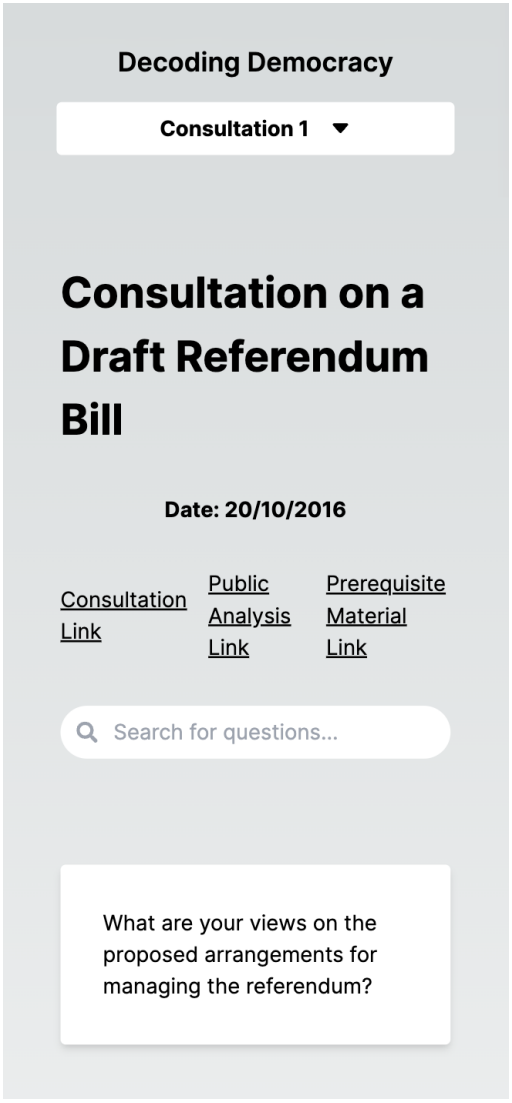



Figure 43: Mobile consultation page

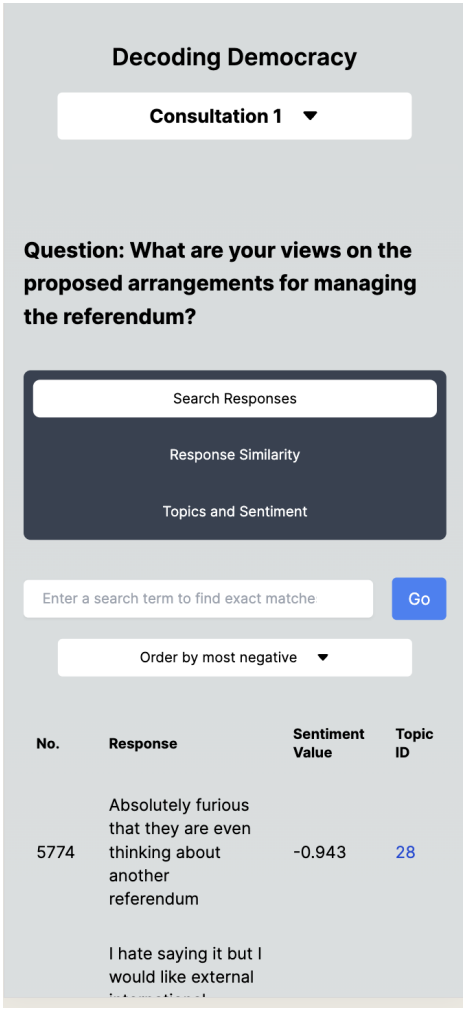



Figure 44: Mobile responses page

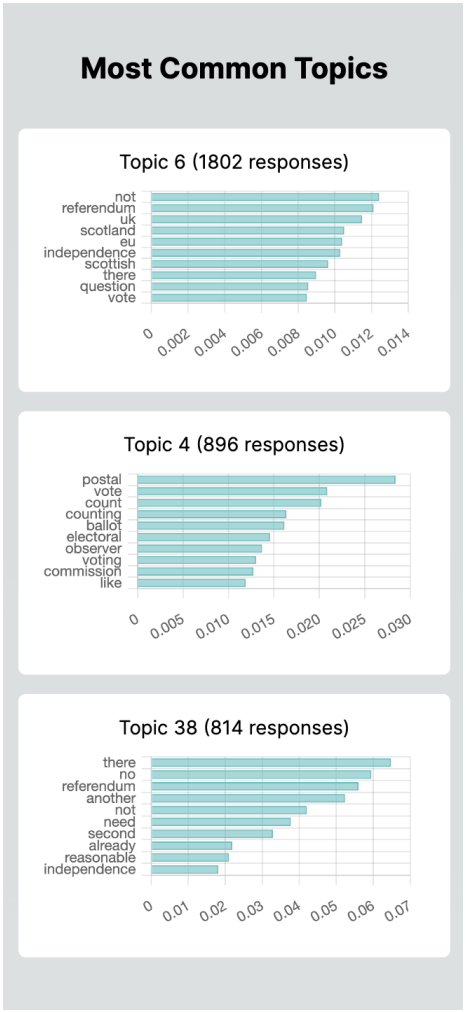



Figure 45: Mobile common topics analysis


Figure 46: Mobile sentiment analysis

# C Other Appendices

## C.1 Python packages

| Package | Description |
|---|---|
| pandas | Data manipulation and analysis library for Python |
| numpy | Numerical computing library for Python |
| nltk | Natural Language Toolkit library for Python |
| demoji | Library for handling emojis in Python |
| vaderSentiment | Rule-based sentiment analysis tool for Python |
| textblob | Library for processing textual data in Python |
| sklearn | Machine learning library for Python |
| gensim | Library for topic modelling and document similarity retrieval |
| transformers | State-of-the-art natural language processing library for Python |
| sentence_transformers | Library for sentence embeddings and semantic textual similarity |
| matplotlib | Data visualization library for Python |
| seaborn | Statistical data visualization library for Python |
| plotly | Interactive data visualization library for Python |
| wordcloud | Library for generating word clouds in Python |
| torch | Deep learning library for Python |
| scipy | Scientific computing library for Python |
| tqdm | Progress bar library for Python |

Table 14: Python packages used throughout the jupyter notebook

## C.2 Ethical Approval Document

University of St Andrews | FOUNDED 1413

School of Computer Science Ethics Committee

08 November 2023

Dear Bruno,

Thank you for submitting your ethical application which was considered by the School Ethics Committee.

The School of Computer Science Ethics Committee, acting on behalf of the University Teaching and Research Ethics Committee (UTREC), has approved this application:

| | | | | | |
|---|---|---|---|---|---|
| **Approval Code:** | CS17313 | **Approved on:** | 08.11.23 | **Approval Expiry:** | 08.11.28 |
| **Project Title:** | Decoding Democracy: Exploring AI-Driven Insights for Participatory Democracy | | | | |
| **Researcher(s):** | Bruno Ceccolini | | | | |
| **Supervisor(s):** | Rosa Filgueira | | | | |

The following supporting documents are also acknowledged and approved:

1. Application Form

Approval is awarded for 5 years, see the approval expiry data above.

If your project has not commenced within 2 years of approval, you must submit a new and updated ethical application to your School Ethics Committee.

If you are unable to complete your research by the approval expiry date you must request an extension to the approval period. You can write to your School Ethics Committee who may grant a discretionary extension of up to 6 months. For longer extensions, or for any other changes, you must submit an ethical amendment application.

You must report any serious adverse events, or significant changes not covered by this approval, related to this study immediately to the School Ethics Committee.

Approval is given on the following conditions:

- that you conduct your research in line with:
  - the details provided in your ethical application
  - the University's Principles of Good Research Conduct
  - the conditions of any funding associated with your work
- that you obtain all applicable additional documents (see the 'additional documents' webpage for guidance) before research commences.

You should retain this approval letter with your study paperwork.

Yours sincerely,

*Natasha Bazeley*

SEC Administrator

School of Computer Science Ethics Committee
Dr Olexandr Konovalov/Convenor, Jack Cole Building, North Haugh, St Andrews, Fife, KY16 9SX
Telephone: 01334 463273 Email: ethics-cs@st-andrews.ac.uk

# References


[1] Francesco Barbieri et al. "TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1644–1650. DOI: 10.18653/v1/2020.findings-emnlp.148. URL: https://aclanthology.org/2020.findings-emnlp.148.

[2] Marian Barnes. "Researching public participation". In: *Local Government Studies* 25.4 (1999), pp. 60–75. DOI: 10.1080/03003939908433967. URL: https://doi.org/10.1080/03003939908433967.

[3] *BERTopic*. https://github.com/MaartenGr/BERTopic.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation". In: 3.null (Mar. 2003), pp. 993–1022. ISSN: 1532-4435.

[5] *Caching*. URL: https://www.geeksforgeeks.org/python-functools-lru_cache/.

[6] Juan Cao et al. "A density-based method for adaptive LDA model selection". In: *Neurocomputing* 72.7 (2009). Advances in Machine Learning and Computational Intelligence, pp. 1775–1781. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2008.06.011. URL: https://www.sciencedirect.com/science/article/pii/S092523120800372X.

[7] CardiffNLP. *Twitter RoBERTa-base for Sentiment Analysis*. https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest. 2021.

[8] *Chart.js*. https://www.chartjs.org/.

[9] Uttam Chauhan and Apurva Shah. "Topic Modeling Using Latent Dirichlet allocation: A Survey". In: *ACM Comput. Surv.* 54.7 (Sept. 2021). ISSN: 0360-0300. DOI: 10.1145/3462478. URL: https://doi.org/10.1145/3462478.

[10] ChooseALicense.com. *Apache License 2.0*. URL: https://choosealicense.com/licenses/apache-2.0/.

[11] ChooseALicense.com. *MIT License*. URL: https://choosealicense.com/licenses/mit/.

[12] *cjhutto/vaderSentiment*. Mar. 2024. URL: https://github.com/cjhutto/vaderSentiment.

[13] *Consultation on a Draft Referendum Bill*. Mar. 2024. URL: https://www.gov.scot/publications/consultation-draft-referendum-bill-analysis-responses/pages/1/.

[14] *Consultation on a Draft Referendum Bill - Scottish Government consultations - Citizen Space*. Mar. 2024. URL: https://consult.gov.scot/elections-and-constitutional-development-division/draft-referendum-bill/.

[15] *Consultation on Electoral Reform - Scottish Government consultations - Citizen Space*. Mar. 2024. URL: https://consult.gov.scot/elections/electoral-reform/.

[16] *Consultation on Electoral Reform summary*. Mar. 2024. URL: https://www.gov.scot/publications/electoral-reform-consultation-analysis/pages/2/.

[17] *Cosine Similarity, Python scikit learn*. URL: https://memgraph.com/blog/cosine-similarity-python-scikit-learn.

[18] *Deep learning*. Mar. 2024. URL: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=1209134089.

[19] *demoji*. https://github.com/bsolomon1124/demoji.

[20] *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. Oct. 2019.

[21] *Electoral reform consultation - Scottish Government consultations - Citizen Space*. Mar. 2024. URL: https://consult.gov.scot/constitution-and-cabinet/electoral-reform/.

[22] *Electoral reform consultation summary*. Mar. 2024. URL: https://www.gov.scot/publications/electoral-reform-consultation-analysis-2/.

[23] *Flask*. https://flask.palletsprojects.com/.

[24] Python Software Foundation. *unittest - Unit testing framework*. URL: https://docs.python.org/3/library/unittest.html.

[25] *Gensim LDAMulticore Model*. URL: https://radimrehurek.com/gensim/models/ldamulticore.html.

[26] Google LLC. *Google Colaboratory*. https://colab.research.google.com/. 2021.

[27] *google-research/bert*. Mar. 2024. URL: https://github.com/google-research/bert.

[28] *Gunicorn*. https://gunicorn.org/.

[29] *Hierarchical Clustering*. URL: https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering.

[30] Notion Labs Inc. *Notion Documentation*. URL: https://www.notion.so/.

[31] Lokaregn. *Preparing Text Data for Transformers: Tokenization, Mapping, and Padding*. https://medium.com/@lokaregns/preparing-text-data-for-transformers-tokenization-mapping-and-padding-9fbfbce28028. Accessed: [Insert access date here]. 2021.

[32] *Matplotlib*. https://matplotlib.org/.

[33] David Mimno et al. “Optimizing Semantic Coherence in Topic Models”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Ed. by Regina Barzilay and Mark Johnson. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 262–272. URL: https://aclanthology.org/D11-1024.

[34] Andy Neumann, Nuno Laranjeiro, and Jorge Bernardino. “An Analysis of Public REST Web Service APIs”. In: *IEEE Transactions on Services Computing* 14 (July 2021), pp. 957–970. DOI: 10.1109/TSC.2018.2847344.

[35] David Newman et al. “Automatic evaluation of topic coherence”. In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. HLT ’10. Los Angeles, California: Association for Computational Linguistics, 2010, pp. 100–108. ISBN: 1932432655.

[36] *Next.js*. URL: https://nextjs.org/.

[37] *NLTK*. https://www.nltk.org/.

[38] *NumPy*. https://numpy.org/.

[39] *Pandas*. https://pandas.pydata.org/.

[40] Jihyeok Park. “JavaScript API misuse detection by using typescript”. In: *Proceedings of the Companion Publication of the 13th International Conference on Modularity*. MODULARITY ’14. Lugano, Switzerland: Association for Computing Machinery, 2014, pp. 11–12. ISBN: 9781450327732. DOI: 10.1145/2584469.2584472. URL: https://doi.org/10.1145/2584469.2584472.

[41] Pradeep Rai and Singh Shubha. “A Survey of Clustering Techniques”. In: *International Journal of Computer Applications* 7 (Oct. 2010). DOI: 10.5120/1326-1808.

[42] Bernadette M. Randles et al. “Using the Jupyter notebook as a tool for open science: an empirical study”. In: *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*. JCDL ’17. Toronto, Ontario, Canada: IEEE Press, 2017, pp. 338–339. ISBN: 9781538638613.

[43] *React*. https://reactjs.org/.

[44] *react-chartjs-2*. https://www.npmjs.com/package/react-chartjs-2.

[45] *react-dom*. https://reactjs.org/docs/react-dom.html.

[46] *react-icons*. https://react-icons.github.io/react-icons/.

[47] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. arXiv: 1908.10084 [cs.CL].

[48] *Routes*. https://routes.readthedocs.io/en/latest/.

[49] *SBERT Sentence Transformers*. URL: https://www.sbert.net/examples/applications/semantic-search/README.html.

[50] *scikit-learn*. https://scikit-learn.org/.

[51] Sorcha Sheridan. *What Is Topic Modeling? A Beginner's Guide*. https://levity.ai/blog/what-is-topic-modeling.

[52] Ian Sommerville. *Software Engineering*. 9th ed. Harlow, England: Addison-Wesley, 2010. ISBN: 978-0-13-703515-1.

[53] *Sqlite3*. URL: https://docs.python.org/3/library/sqlite3.html.

[54] Keith Stevens et al. "Exploring Topic Coherence over Many Models and Many Topics". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Ed. by Jun'ichi Tsujii, James Henderson, and Marius Paşca. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 952–961. URL: https://aclanthology.org/D12-1087.

[55] *tailwindcss*. https://tailwindcss.com/.

[56] Sentence Transformers. *Sentence Transformers - all-MiniLM-L6-v2*. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2. 2021.

[57] *Tutorial: Quickstart — TextBlob 0.18.0.post0 documentation*. Mar. 2024. URL: https://textblob.readthedocs.io/en/dev/quickstart.html.

[58] *typescript*. https://www.typescriptlang.org/.

[59] Analytics Vidhya. *A Comprehensive Guide to Sentiment Analysis with TextBlob and VADER*. https://www.analyticsvidhya.com/blog/2021/10/analysis-with-textblob-and-vader/. 2021.

[60] Lawrence C. Walters, James E. Aydelotte, and Jessica Miller. "Putting More Public in Policy Analysis". In: *Public Administration Review* 60 (2000), pp. 349–359. DOI: 10.1111/0033-3352.00097.

[61] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Ed. by Tal Linzen, Grzegorz Chrupała, and Afra Alishahi. Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. URL: https://aclanthology.org/W18-5446.

[62] Min-Hsien Weng, Shaoqun Wu, and M. Dyer. "AI Augmented Approach to Identify Shared Ideas from Large Format Public Consultation". In: *Sustainability* (2021). DOI: 10.3390/su13169310.

[63] Wikipedia contributors. *Semantic similarity — Wikipedia, The Free Encyclopedia*. [Online; accessed 11-March-2024]. 2024. URL: https://en.wikipedia.org/w/index.php?title=Semantic_similarity&oldid=1209212793.

[64] Matthew J Beal Yee Whye Teh Michael I Jordan and David M Blei. "Hierarchical Dirichlet Processes". In: *Journal of the American Statistical Association* 101.476 (2006), pp. 1566–1581. DOI: 10.1198/016214506000000302. URL: https://doi.org/10.1198/016214506000000302.

[65] Lx Yuan. *DistilBERT Base Multilingual Cased Sentiments Student*. https://huggingface.co/lxyuan/distilbert-base-multilingual-cased-sentiments-student. 2021.

[66] Lx Yuan. *Distilling Zero Shot Multilingual DistilBERT Sentiments Student*. https://github.com/LxYuan0420/nlp/blob/main/notebooks/Distilling_Zero_Shot_multilingual_distilbert_sentiments_student.ipynb. 2021.

[67] Zaynab Zahra, Zihao Li, and Rosa Filgueira. *Laminar: A New Serverless Stream-based Framework with Semantic Code Search and Code Completion*. 2023. arXiv: 2309.00584 [cs.DC].