UNIVERSITY OF LIVERPOOL

COMP390
2020/21

A Full Stack Application and Bot to Analyse Political Commentary on Reddit

Student Name:	Andrew Georgiou
Student ID:	201199188
Supervisor Name:	Stuart Thomason

DEPARTMENT OF
COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Dedicated to Joseph Bolan

Acknowledgements

I want to thank my family for supporting me during my studies, my girlfriend for keeping my spirits high every day, and also a huge thank you to my university friends and colleagues who have made my time at The University of Liverpool that much more enjoyable, especially Dr Stuart Thomason who has guided and encouraged my work throughout this past year.

I dedicate this dissertation to my late grandfather Joseph Bolan who sadly did not get the chance to see me graduate. He encouraged me to do what I loved, and I owe not only my degree to him but my future as well.

COMP390

2020/21

A Full Stack Application and Bot to Analyse Political Commentary on Reddit

DEPARTMENT OF
COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3BX

Abstract

Political divide has taken a significant hold of social media. Dangerous echo chambers develop in the advent of an ability to detect them. *r/The_Donald* was a 790,000-user right-wing subreddit that became a haven for its followers before it was banned for racism and harassment in June 2020 from the platform Reddit.

This project aimed to develop a machine learning model to classify the political orientation of Reddit's commentary and communities, build a full-stack application that enables interaction with the model and live data. Facilitating users to understand further how the platform works through identifying echo chambers and political safe havens in the wake of the presidential election.

The dissertation discusses the successful process of building the full-stack application, including extraction of meaningful data, overcoming over-fitting in pre-trained language models, creating a backend to host the model in the cloud, and a frontend that enables the visualisation of these components.

Table of Contents

1. Introduction & Background

1.1 Introduction

1.1.1 – Dissertation Overview

This dissertation describes the motivation and background of the project, the aims and objectives of which the project achieved. The initial design of the data, machine learning model, backend, and frontend are discussed, as well as implementing, testing, and evaluating each of these components. The dissertation is concluded with an in-depth reflection of the project and the work behind it.

1.1.2 – Motivation

A study conducted in January 2016 showed 62% of American adults get their news from social media, with 70% of Reddit users stating they receive their news from the social media platform [1]. Reddit allows users to subscribe to communities (subreddits) they wish to see on their timeline and un-subscribe to those they want to see less. This feature can limit users' exposure to a diversity of opinions as they choose to subscribe to like-minded communities. This tendency to favour information that aligns with our beliefs and joining these communities to develop a shared narrative is referred to as an echo chamber [2].

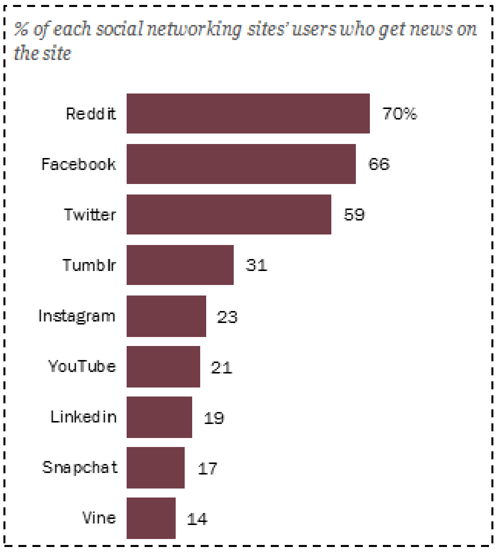


Figure 1: Study of social network users [3]

"Internet users can seek out interactions with like-minded individuals who have similar values, and thus become less likely to trust important decisions to people whose values differ from their own. This voluntary Balkanisation and the loss of shared experiences and values may be harmful to the structure of democratic societies as well as decentralised organisations." [4]

In June 2020, one of the largest subreddits on the platform *r/the_donald* was described as *"a never-ending rally dedicated to the 45th President of the United States, Donald J.*

Trump." [5] with 790,000 subscribed users at its peak was banned following cases of targeted harassment and racism against other communities of differing political views [6].

Following the 2020 United States presidential election, social media has seen tensions between these opposing communities only rise as their overlap of communities becomes further polarised [7]. The desire to identify these communities on Reddit, letting users understand how they may be creating harmful political echo chambers of their own.

1.1.3 – Aims

- Develop a model that can categorise left-wing and right-wing commentary sentiment.
- Build a platform for users to interact with the model.
- Develop a react frontend that allows users to connect with live data.

1.1.4 – Objectives

- Define and categorise left-wing and right-wing subreddits into a list.
- Mine the data using PRAW into a CSV file.
- Train a model to categorise the left-wing and right-wing sentiment.
- Host the model as a RESTful API on a cloud instance.
- Create a microservice that enables interacting with live Reddit data.
- Develop an interactive React frontend that can make calls to the hosted services.

1.1.5 – Ethical Considerations

Following the University of Liverpool Ethical Conduct Guide and its policies are thoroughly regarded in every component of this project.

All the data used is personally scraped to ensure it is trusted and personal to this project. This is done to reduce bias in data which could skew results and opinion of the final implementation.

Consideration into the mining of Reddit data took the concerns of Reddit's API guidelines surrounding not mining personal user information. Instead, only user-generated content is used, and personal information relating to usernames and user identifying features are removed before data is processed and stored.

All data scraped needs to be stored in personally protected file stores that contain password security to ensure data cannot be released publicly. The only data held externally to the local file store remains on a password-protected Google Drive.

Previous projects relating to political prediction allow for user political prediction, which lets Redditors enter a username and predict the political alignment of other users [8]. However, specific user prediction enabled people to target different users based on their political alignment, so this project does not feature specific user/Redditor prediction.

1.2 Background and Related Work

1.2.1 – Background Introduction to Natural Language Processing

A lot of language in this dissertation relates to the field of Natural Language Processing, this background introduction into NLP helps to briefly explain what the field is and what developments in the field have enabled for this project.

Natural Language Processing is a sub-field of linguistics, and artificial intelligence focused on the understanding and analysis of human language. For example, one application is text classification, such as spam detection or sentiment analysis. The aim is to classify text into different classes by using a model for machine learning that is trained in that specific task.

For the task of NLP, text must be converted into some float or integer values to be fed into the machine learning model as input. This data is passed through a series of parameters trained on the data by tweaking their weights until they give the desired output. The complexity of these models has increased exponentially over time. Cutting-edge pre-trained models such as GPT-3 can exceed 175 billion parameters [9].

Performance of these pre-trained models on the *General Language Understand Evaluation* (GLUE) benchmark exceeds a 10-point improvement over previous non-transformer models [10]. These models are trained on large corpora of text to understand the human language similarly to how humans can understand words by the context in which they appear.

These models can then be fine-tuned, which uses additional data to train the model's weights at the lower level to classify it for your task while keeping intact the contextual word understanding. This project fine-tunes these pre-trained models on the text classification problem of classifying the political orientation of Reddit comments.

Some of the pre-trained language models used during this project and will often be discussed, BERT (Bi-directional Encoder Representation from Transformers)[11], published by Google in 2018, uses two unique methods of training to perform effectively. Masked Language-Modelling (MLM) masks 15% of an input sequence and asks the model to predict the original word based only on the context provided by the rest of the sequence. This enabled bi-directional learning that was previously considered impossible. Simpler bi-directional models have existed for much longer, such as bi-directional LSTM (Long-Short-Term Memory). However, these models can only predict left-to-right and right-to-left but not bi-directionally from each word in the input sequence, which helps increase BERT's understanding of context.

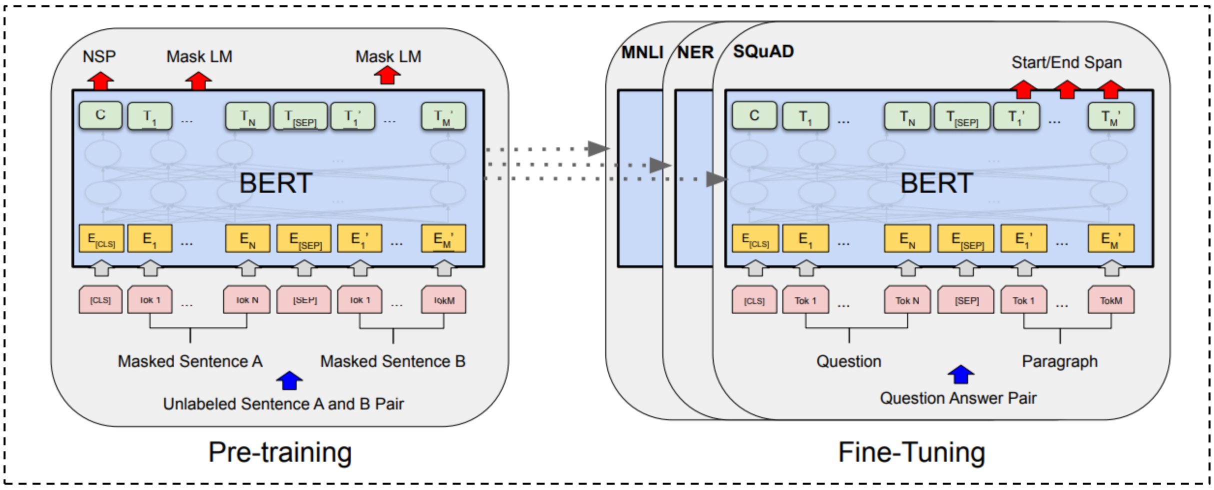


Figure 2: Overall pre-training and fine-tuning procedures for BERT [11]

The other unique method is Next-Sentence-Prediction (NSP), the model receives pairs of sentences as input and attempts to predict the order in which the sentences appear within the original document. This uses special tokens discussed in *Section 2.2*, defining the beginning and end of an input sequence.

Bert consists of two different versions, the base model, which consists of 110 million parameters and the large, which has 340 million. The other pre-trained language models are RoBERTa and DistilBERT, and these are both descendants of BERT. RoBERTa is a model developed by Facebook in 2019 [12]. Consisting of a similar number of parameters RoBERTa improved the performance of BERT by re-training the entire model on a larger corpus of data. Most importantly, it removed NSP from its pre-training task and introduced dynamic masking that enabled the masked words to change mid-training at random. This gave it a 20% improvement in performance over BERT in the GLUE benchmark.

DistilBERT offers slightly less performance than both BERT and RoBERTa, but in return, it offers greater efficiency through shorter training and prediction time. This is possible through distillation, which uses the BERT model and approximates the output between layers using a smaller network. DistilBERT, therefore, has half the number of parameters as BERT at around 60 million. [13]

	BERT	**DistilBERT**	**RoBERTa**
Model Parameters	Base: 110 million Large: 335 million	Base: 66 million Large: N/A	Base: 125 million Large: 355 million
GLUE Performance	80.5%	77.0%	88.5%
Training *(words per corpus)*	*BooksCorpus: 800M* *Wikipedia: 2500M*	*BooksCorpus: 800M* *Wikipedia: 2500M*	*BooksCorpus: 800M* *Wikipedia: 2500M* *CommonCrawl News: 6.5 Billion* *CommonCrawl stories: 7 Billion*
Features	*MLM, NSP*	*MLM, NSP*	*MLM*

Figure 3: Comparison of pre-trained language models, BERT, DistilBERT and RoBERTa

1.2.2 – Related Work

Little work has been done to implement a methodology for identifying subreddits or commentary on the platform for political bias since *A.Soliman et al.* attempted to classify subreddits by utilising a word frequency list for left and right-leaning language [14]. A similar word frequency list was generated through this project, although used as a tool for visualisation whilst more complex methods were developed for classifying the communities.

The use of similar language models to classify political inclination through Twitter achieved 87.7% accuracy, finding that the terminologies used by either source had a significant difference [15]. One study into the same problem on Reddit, which identified users as liberal or conservative, attempted to classify their comment history. They found 70% accuracy in a smaller Support Vector Machine model, finding that additionally utilising controversiality and sentiment of the posts can reduce the ability for the model to generalise [16].

Political prediction tools that aim to predict users' political leaning on the platform Reddit exist, such as https://reddit-lean.com/, which uses a logistic regression trained on 20,000 users' comment history from the subreddit *r/politicalcompassmemes* to attempt multi-classification of 4 political classes.

Users of the tool have used it to predict other users' political orientation, having attempted to label users by their political ideology and encouraged users to target others based on their political alignment. This informed the ethical consideration of this project not to include any user prediction or targeted prediction that may tag specific users with a political orientation.

2. Design

2.1 Software Design Breakdown & Decisions

This section will cover a brief breakdown of the project into its main categories and a detailed explanation of the different software design decisions made throughout this project, with reasoning for each decision.

2.1.1 – Breakdown

This project is broken down into three separate parts, which will be seen as a common theme of separating these sections throughout this dissertation.

Mining and Training: This section of the project evolves around the extraction of data from Reddit, the processing of data, and the training of different artificially intelligent models used in the project's backend section.

The Backend: This part of the project is the server that serves data to the frontend application. It includes serving the model's prediction data in response to user input and live-processing Reddit data for visualisation tools.

The Front-End: The area of the project the user interacts with, hosted at *https://www.reddit-political-analysis.com/*. This is the visual and interactive elements of the project, including building visualisation tools that utilise the backend for data relating to user input.

2.1.2 – Extraction and Training Programming Language & IDE

This section of the project required Python, as it allows access to the library called Pandas, which is needed for handling and processing the amount of data we would be scraping. Pandas is an "open-source data analysis and manipulation tool" [17] which is helpful for handling and storing data. Python also has multiple wrapper libraries for Reddit's different APIs, PRAW and PSAW.

The IDE used for this project is called Spyder, an "open-source scientific environment written in Python, for Python." [18] enabling a few useful features, such as a built-in plot view which is practical for visualising the mined data using the library matplotlib. It also allows us to use a variable explorer to analyse our data at each stage of processing.

2.1.3 – Comment Extraction API

When mining data, there exist two commonly used Reddit APIs, The Official Reddit API (*PRAW*)[19] and PushShift API (*PSAW*)[20]. Both these API complete the same task but differ in their methods. Reddit's API is considered the standard comment extraction tool, although PushShift is an ElasticSearch API, a high-speed extraction method for data.

Unlike Reddit's official API, PushShift does not extract comments from the website live. It instead crawls Reddit daily and updates the ElasticSearch database to enable indexing data faster than Reddit's API. Unfortunately, this is ineffective for this project's extraction method. The model needed to be trained on accurate, up-to-date data to maintain political relevance as political opinion is constantly changing.

It is also insufficient as PushShift's aggregate functionality has been removed, meaning it requires more requests per submission to extract comment data than using Reddit's Official API.

2.1.4 – Subreddit's For Extraction

Left and Right-wing subreddits were initially planned to be chosen subjectively by the author. However, upon early training of the model, it generalised poorly on the subjective data with around 50% testing accuracy in BERT, DistilBERT, and RoBERTa models with more significant rates of overfitting in early epochs.

Inspection of the data showed that some of the subreddits were entirely sarcastic in their creation and intended to be mocking. In addition, small subreddits suffered from brigading where politically opposing users would mass post and upvote their differing opinion to "take over" a given subreddit [21].

A method designed to handle this was creating software that would extract frequently posting users from each subreddit and build a frequency map of other commonly visited subreddits. This method aimed to help in discovering overlapping and popular subreddit communities, which can help identify more objective data. This is *discussed further in section 3.3.2*.

Concrete left, and right-wing subreddits were entered into the software, such as "Conservative" or "liberal" subreddits and from that, it generated this output of subreddits for each term:

Left-Wing Subreddits:	**Right-Wing Subreddits:**
r/Liberal	*r/Conservative*
r/ShitLiberalsSay	*r/LouderWithCrowder*
r/LateStageCapitalism	*r/ Capitalism*
r/Socialism_101	*r/AskThe_Donald*
r/BreadTube	*r/HillaryForPrison*
r/Socialism	*r/Tucker_Carlson*
r/Communism	*r/Anarcho_Capitalism*
r/Communism101	*r/Republicans*
r/Anarchism	*r/Walkaway*
r/Anarchy101	*r/NEWPOLITIC*
r/CompleteAnarchy	*r/SocialJusticeInAction*
r/DemocraticSocialism	*r/Conservatives*
r/ToiletPaperUSA	
r/AccidentallyCommunist	
r/AntifaStoneToss	
r/TheRightCantMeme	
r/AntiFascistsOfReddit	
r/LateStageImperialism	
r/Capitalism_In_Decay	
r/SocialistRA	
r/ABoringDystopia	
r/EnlightenedCentrism	
r/DankLeft	

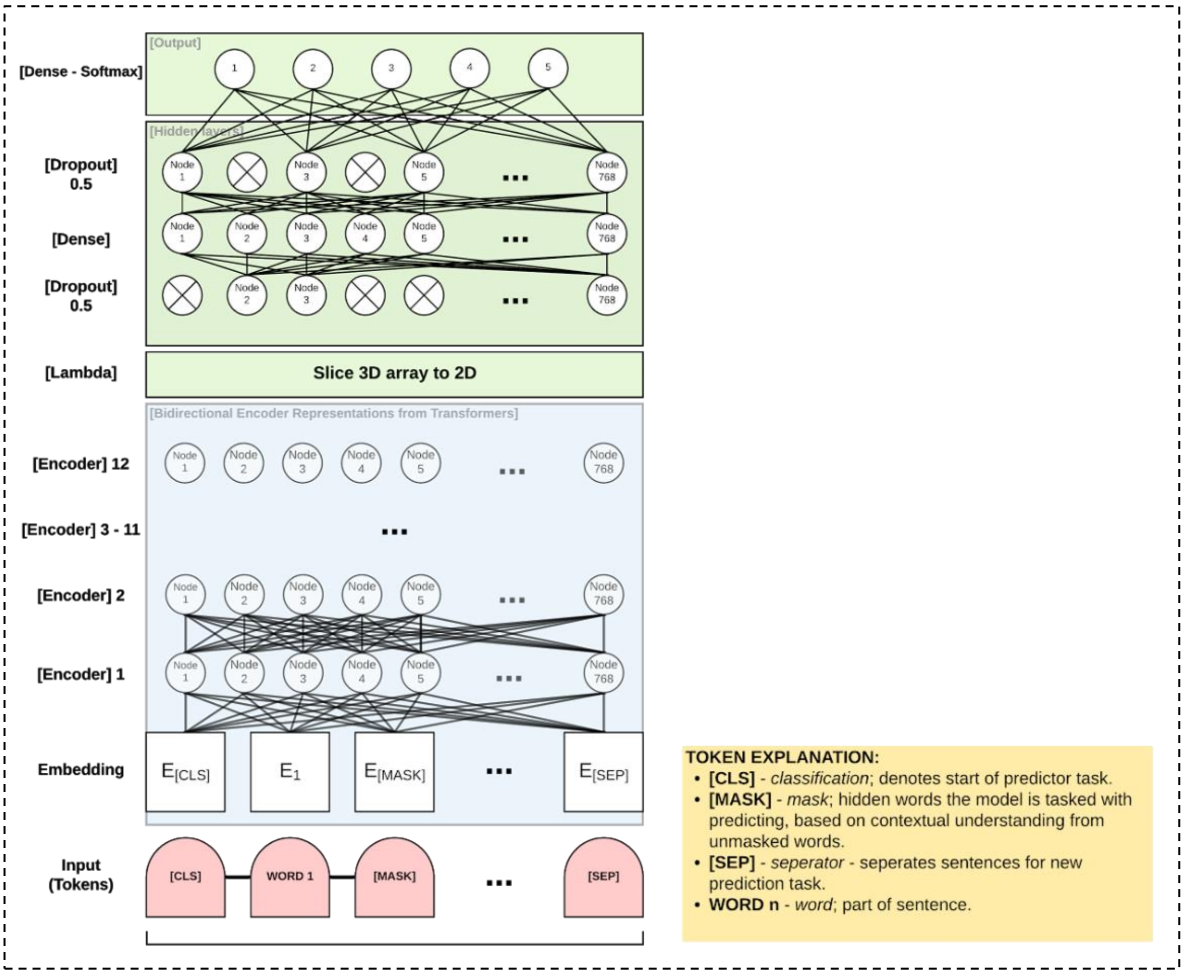
Figure 4: List of Subreddits generated by Common Subreddit Tool

The tool scraped the top 1000 comments from each of the top 100 users on these subreddits and produced this political subreddit frequency list. One thing of note, it was able to identify much more left-leaning subreddits than right-leaning. So, potentially, Reddit is a more left-centric platform or users who frequent right-wing subreddits do not actively participate in many similar political communities.

2.1.5 – Artificially Intelligent Model Design

Deciding between creating an artificially intelligent model from scratch or utilising a pre-trained language model was a decision that had to be made at the beginning of the project.

The decision to choose a pre-trained language model was based on the development timeframe and the financial undertaking of this project. Since the model development cycle was estimated to be two to three months, it was not feasible to create a model from scratch. It was also not feasible to train a similar performing model to the pre-trained models without a significant financial investment for hardware costs.



TOKEN EXPLANATION:

- **[CLS]** - *classification*; denotes start of predictor task.
- **[MASK]** - *mask*; hidden words the model is tasked with predicting, based on contextual understanding from unmasked words.
- **[SEP]** - *seperator* - seperates sentences for new prediction task.
- **WORD n** - *word*; part of sentence.

Figure 5: A representation of a pre-trained model, BERT's architecture.

An early introduction to pre-trained language models, BERT consists of 12 layers with over 125 million parameters. The reason a model like this is fine-tuned instead of trained from scratch is related to cost. It took 64 TPU chips over four days of training at $7000 to pre-train BERT. Due to the batch sizing used in training, it also makes it impossible for a single person to train this model on their GPU, therefore requiring hardware rental costs.

Natural Language Processing has matured as a field, ground-breaking models which have been pre-trained on large corpora of text, such as Wikipedia's corpus which features over 2-billion-word vectors. These models have a much greater understanding of the human language than what is possible with any single developer created model. Not utilising these pre-trained models and fine-tuning them further on a specific task within a particular domain is wasting their developmental purpose.

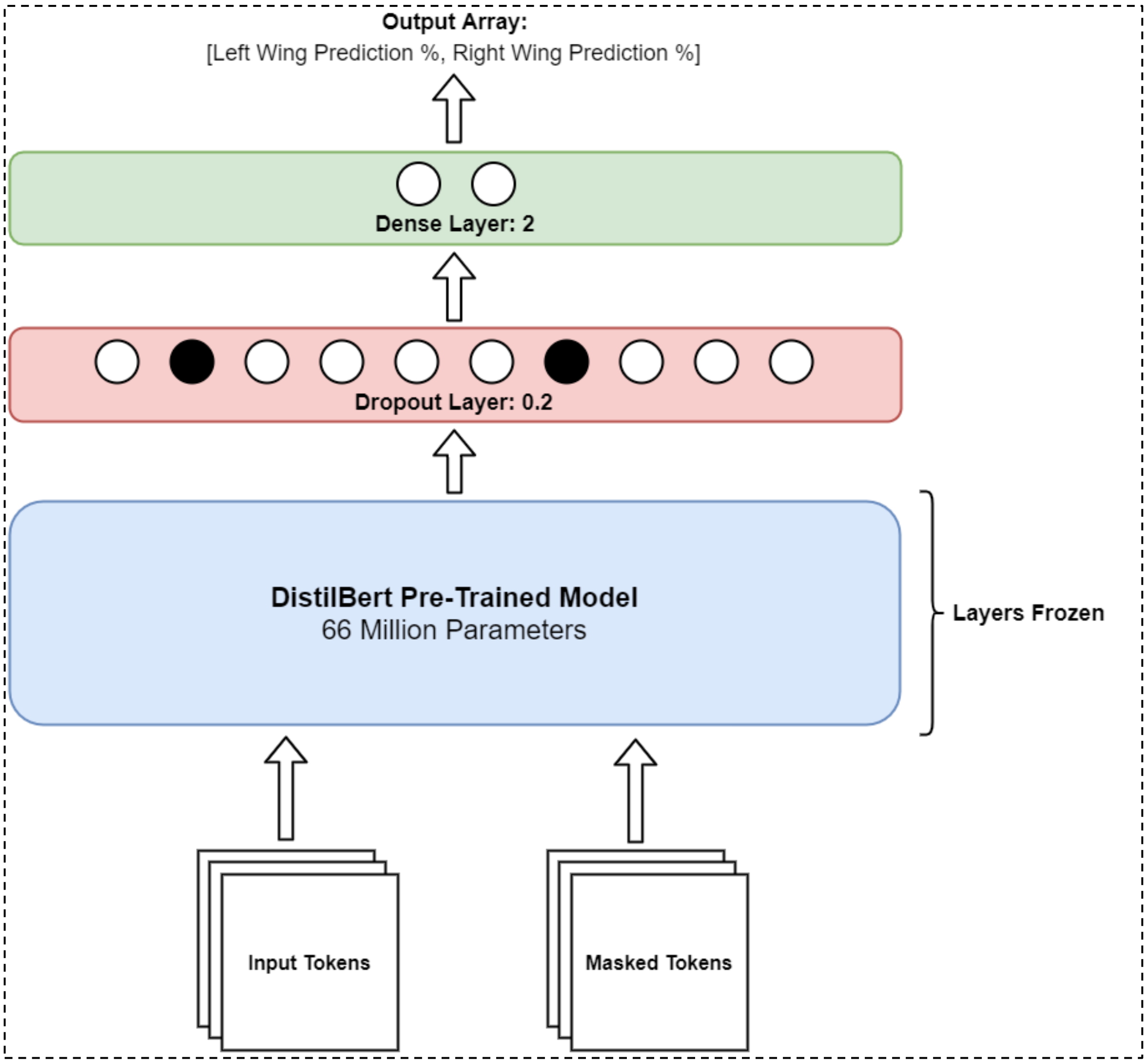


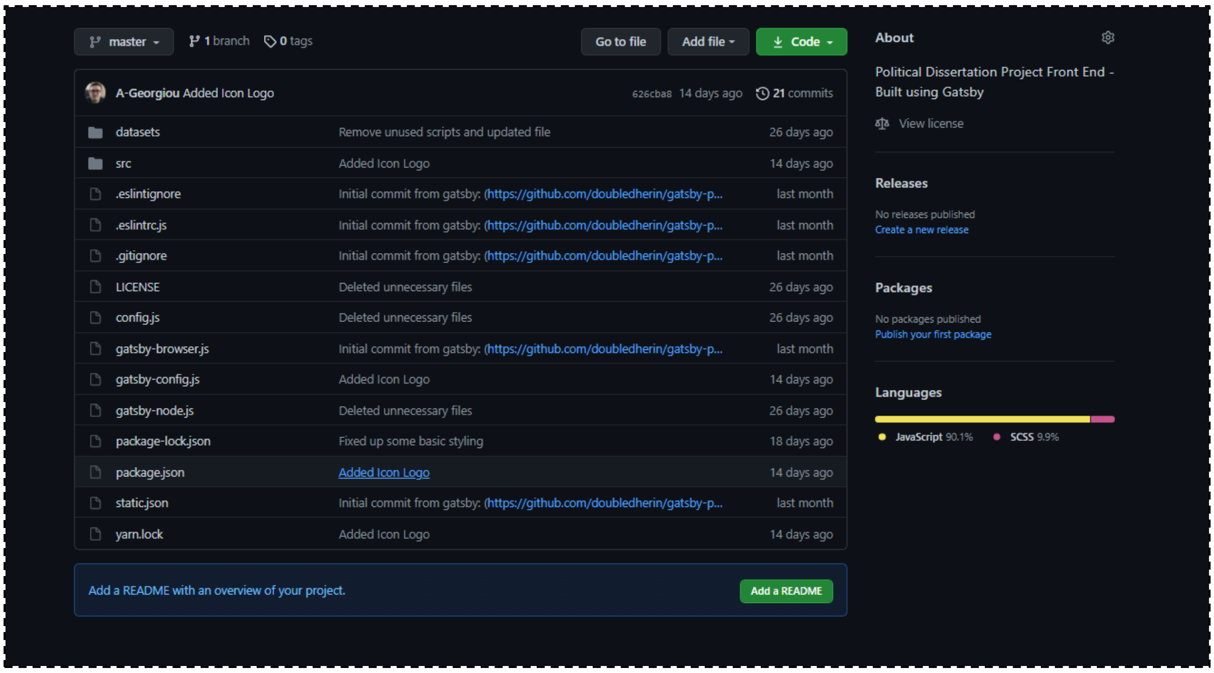
Figure 6: Diagrammatic representation of the DistilBERT TensorFlow Architecture.

Fig. 6 shows a representation of the final model, TensorFlow's HuggingFace model for DistilBERT. It has an input of tokenised text and masked tokens. DistilBERT uses next-word prediction, which will attempt to predict masked words to increase the contextual understanding of a single word/token. The Dropout layer reduces the number of output nodes and helps understand the model's output which is fed into a Dense layer with two nodes as the prediction output.

The DistilBERT model layers are frozen, which stops the large 66 million parameters from being adjusted during training. Without this, the model could overfit the data much easier.

2.1.6 – Hosting the Front-End

The decision to host the React frontend through Netlify aided publishing content by pulling the React repository from GitHub to automatically build and deploy the frontend on each push to the production branch. This meant changes could happen quickly, going from development to live server just by committing and pushing changes to GitHub.



A-Georgiou Added Icon Logo 626cba8 14 days ago 21 commits

datasets	Remove unused scripts and updated file	26 days ago
src	Added Icon Logo	14 days ago
.eslintignore	Initial commit from gatsby: (https://github.com/doubledherin/gatsby-p...	last month
.eslintrc.js	Initial commit from gatsby: (https://github.com/doubledherin/gatsby-p...	last month
.gitignore	Initial commit from gatsby: (https://github.com/doubledherin/gatsby-p...	last month
LICENSE	Deleted unnecessary files	26 days ago
config.js	Deleted unnecessary files	26 days ago
gatsby-browser.js	Initial commit from gatsby: (https://github.com/doubledherin/gatsby-p...	last month
gatsby-config.js	Added Icon Logo	14 days ago
gatsby-node.js	Deleted unnecessary files	26 days ago
package-lock.json	Fixed up some basic styling	18 days ago
package.json	Added Icon Logo	14 days ago
static.json	Initial commit from gatsby: (https://github.com/doubledherin/gatsby-p...	last month
yarn.lock	Added Icon Logo	14 days ago

Add a README with an overview of your project. Add a README

About

Political Dissertation Project Front End - Built using Gatsby

View license

Releases

No releases published
Create a new release

Packages

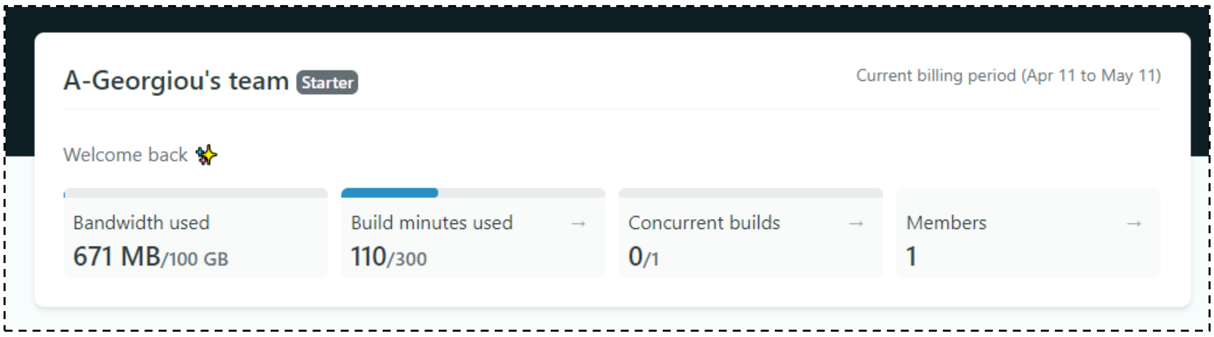
No packages published
Publish your first package

Languages

● JavaScript 90.1% ● SCSS 9.9%

Figure 7: GitHub source code for Gatsby frontend that is replicated to Netlify live server.

Netlify is free for up to 100GB bandwidth and 300 build minutes per month. At one point in time, when the project was posted on Reddit, it reached the top post on /r/MachineLearning, /r/LanguageTechnology, and /r/Python all in one day. Hitting fifty unique users simultaneously, the frontend managed to sustain and process each user's request, and bandwidth usage did not exceed 200MB at the time.


A-Georgiou's team Starter

Current billing period (Apr 11 to May 11)

Welcome back

Bandwidth used
671 MB/100 GB

Build minutes used
110/300

Concurrent builds
0/1

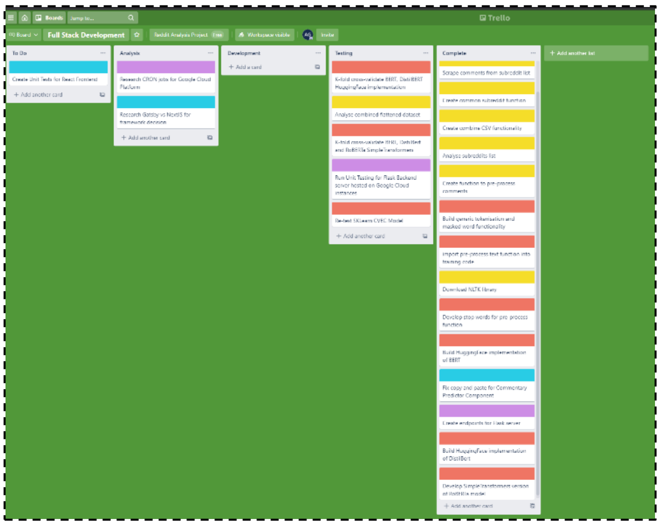
Members
1

Figure 8: Netlify single-month usage for a free account.

2.1.7 – Development Process

The Kanban development method was used for this project. Although often used by agile development teams, this approach suited the modular application of this project. Kanban is the process of using a categorised board and modularly separating each component of the development process. Each component can then be assigned a priority and a deadline. The deadline, in this case, would relate to the Gantt Chart in *Appendix D*. When each component would progress in development it would move along to the next category of the board. The board used for this project has the categories: To-Do, Analysis, Development, Testing and Complete.

Furthermore, this method was paired with The Rapid Application Development (RAD) methodology. Each stage of this project was rapidly prototyped and improved to reduce each stages development time. This was especially useful in managing time since RAD enabled changes to be made in direct relation to how the machine learning model performed in testing, allowing the model to be quickly refined and prototyped before testing again.



To Do
- Create Unit Tests for React Frontend

Analysis
- Research CRON jobs for Google Cloud Platform
- Research Gatsby vs NextJS for framework decision

Development

Testing
- K-fold cross-validate BERT, DistilBERT HuggingFace implementation
- Analyse combined flattened dataset
- K-fold cross-validate BERT, DistilBert and RoBERTa SimpleTransformers
- Run Unit Testing for Flask Backend server hosted on Google Cloud instances
- Re-test SKLearn CVEC Model

Complete
- Scrape comments from subreddit list
- Create common subreddit function
- Create combine CSV functionality
- Analyse subreddits list
- Create function to pre-process comments
- Build generic tokenisation and masked word functionality
- Import pre-process text function into training code
- Download NLTK library
- Develop stop words for pre-process function
- Build HuggingFace implementation of BERT
- Fix copy and paste for Commentary Predictor Component
- Create endpoints for Flask server
- Build HuggingFace implementation of DistilBert
- Develop SimpleTransformers version of RoBERTa model

Figure 9: Kanban board with colour coded components

2.2 Mining and Training Data Design

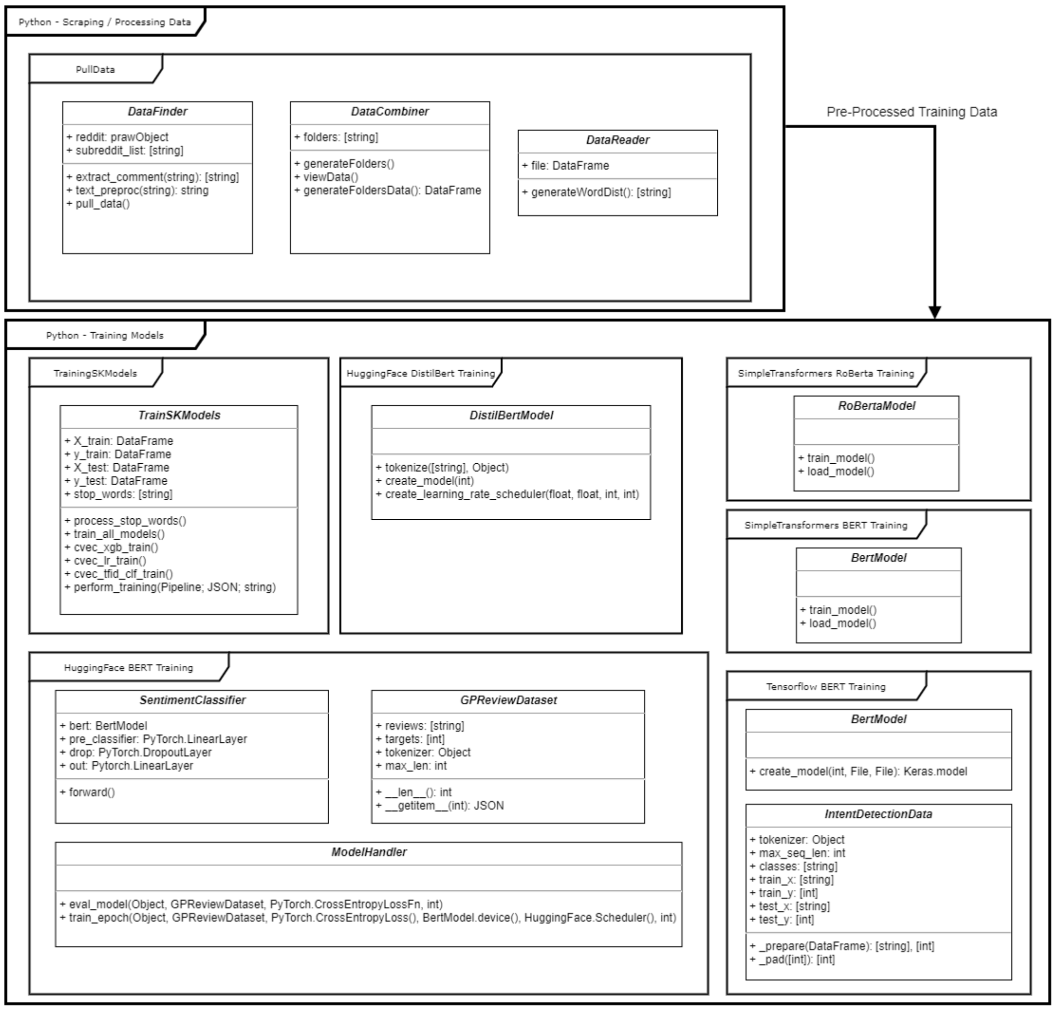


Figure 10: Python Data Scraping and Training Model UML Class Diagram

Scraping and Processing data, extracting and processing the comments from Reddit was done using the *PullData* file, which had the following classes:

DataFinder:

This class enabled extracting comments from Reddit using the prawObject *Reddit*. When initialising the class, it requires a list of strings called *subreddit_list*, which contains all the subreddits used in comment extraction.

The *pull_data()* function will take each comment and pass it into *extract_comment()*. This function would extract each set of comments from a given subreddit and feed the data into *text_preproc()*to pre-process the data by removing unnecessary symbols and punctuation. The overall output for each subreddit is converted into a CSV file.

DataCombiner:

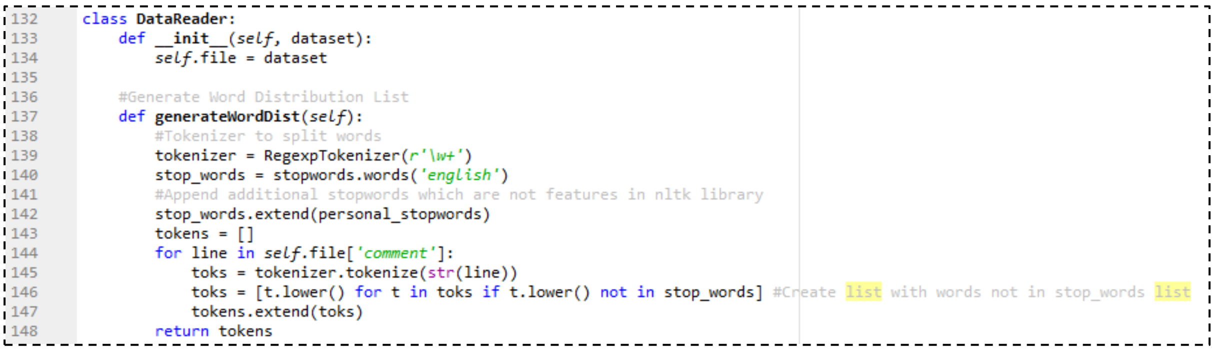
This class is designed to combine the CSV files and set the labels for each class, left and right-wing sentiment datasets are separated into different folders:

generate_folders function combines each folder's CSV files into a single merged dataset.

view_data function flattens the dataset. It takes the merged CSV file and randomly removes indexes of the larger class until both classes are equal.

DataReader:

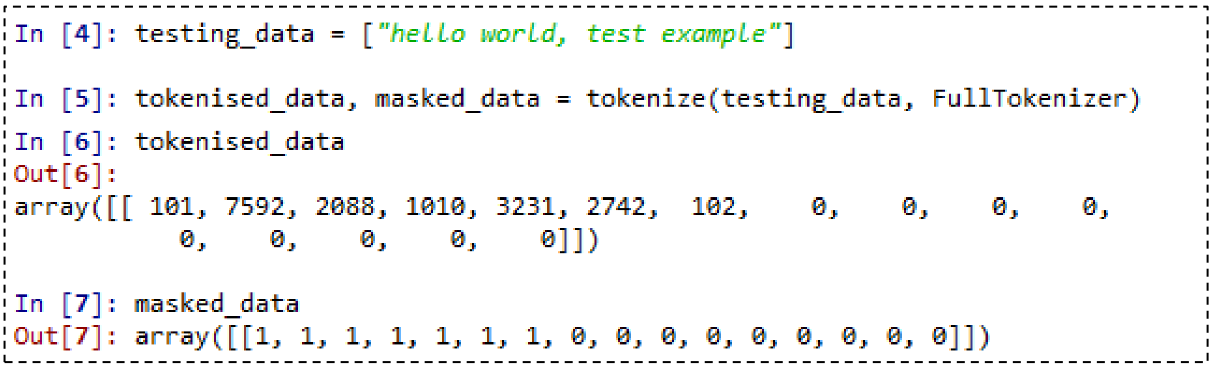
This class generates a word frequency list which is used in the frontend data visualisation. It takes the final flattened, merged dataset, tokenises the dataset and removes and words that are filtered by *stop_words*, and generates a count of each token.



```python
class DataReader:
    def __init__(self, dataset):
        self.file = dataset

    #Generate Word Distribution List
    def generateWordDist(self):
        #Tokenizer to split words
        tokenizer = RegexpTokenizer(r'\w+')
        stop_words = stopwords.words('english')
        #Append additional stopwords which are not features in nltk library
        stop_words.extend(personal_stopwords)
        tokens = []
        for line in self.file['comment']:
            toks = tokenizer.tokenize(str(line))
            toks = [t.lower() for t in toks if t.lower() not in stop_words] #Create list with words not in stop_words list
            tokens.extend(toks)
        return tokens
```

Figure 11: DataReader Class Python Code

Multiple models were trained using different libraries to compare performance, splitting each large model into a separate file aided modularity when transferring the model to the backend. The models all shared the same functionality, just through different libraries. These functions are as follows:



```
In [4]: testing_data = ["hello world, test example"]

In [5]: tokenised_data, masked_data = tokenize(testing_data, FullTokenizer)

In [6]: tokenised_data
Out[6]:
array([[ 101, 7592, 2088, 1010, 3231, 2742,  102,    0,    0,    0,    0,
           0,    0,    0,    0,    0]])

In [7]: masked_data
Out[7]: array([[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Figure 12: Tokenisation Example

Tokenisation:

This function separates each word in each comment and assigns them to an ID defined in the vocabulary file used to train the model. It also appends unique character tokens to denote the beginning and end of a comment. Some models have a process called next-word-prediction in which the tokeniser will also mask certain words which the model will have to attempt to predict.

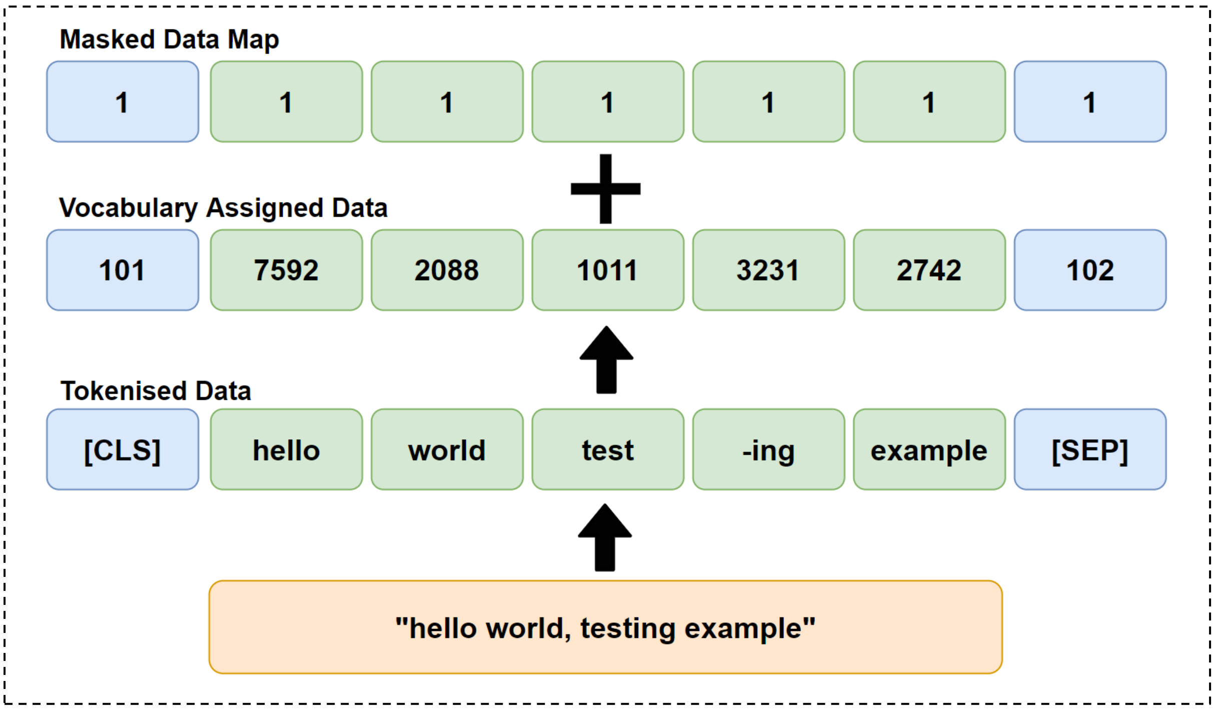


Figure 13: Tokenisation Sketch Visualisation

Train Model:

This function defines the actual training of the model. The parameters for each model were initially experimented across different batch sizes and learning rates but settled on a range of parameters between each model.

Batch Size: 16-32

Epochs: 6-8.

Learning Rate: The learning rate scheduler ran between 5e-5 to 1e-7 with four warmup epoch count.

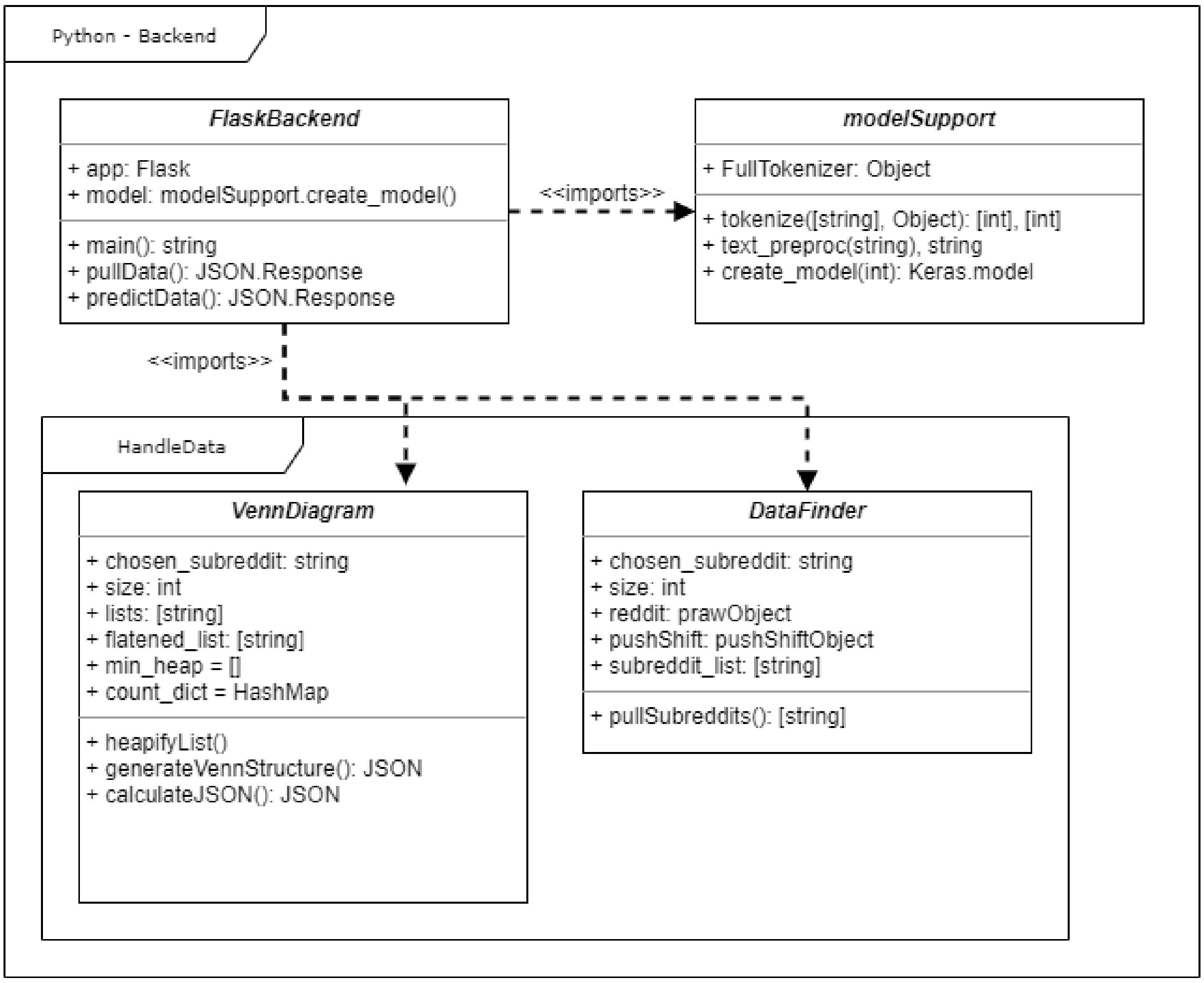
Early Stopping No. Epochs: 2

Max Sequence Length (number of words): 256

Load Model:

The load model function handles loading the model from a saved file and tokenising input data to predict political orientation. The backend server uses this function.

2.3 Backend Class Design

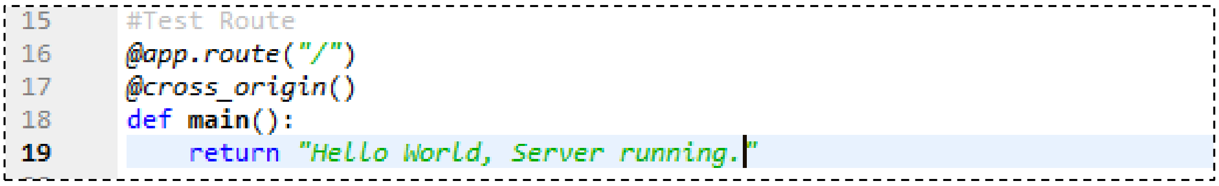

Python - Backend
FlaskBackend
+ app: Flask
+ model: modelSupport.create_model()
+ main(): string
+ pullData(): JSON.Response
+ predictData(): JSON.Response
<<imports>>
modelSupport
+ FullTokenizer: Object
+ tokenize([string], Object): [int], [int]
+ text_preproc(string), string
+ create_model(int): Keras.model
<<imports>>
HandleData
VennDiagram
+ chosen_subreddit: string
+ size: int
+ lists: [string]
+ flatened_list: [string]
+ min_heap = []
+ count_dict = HashMap
+ heapifyList()
+ generateVennStructure(): JSON
+ calculateJSON(): JSON
DataFinder
+ chosen_subreddit: string
+ size: int
+ reddit: prawObject
+ pushShift: pushShiftObject
+ subreddit_list: [string]
+ pullSubreddits(): [string]
Figure 14: Python Flask Backend UML Class Diagram

The backend is separated into classes, one for each endpoint. The classes and each of the endpoints are as follows:

FlaskBackend:

Endpoint: '/'

Description: This is the main body of the Flask server. It holds each endpoint and imports the required classes to process the user data from each request, the functions described in *Fig. 13* for this class are each an endpoint.



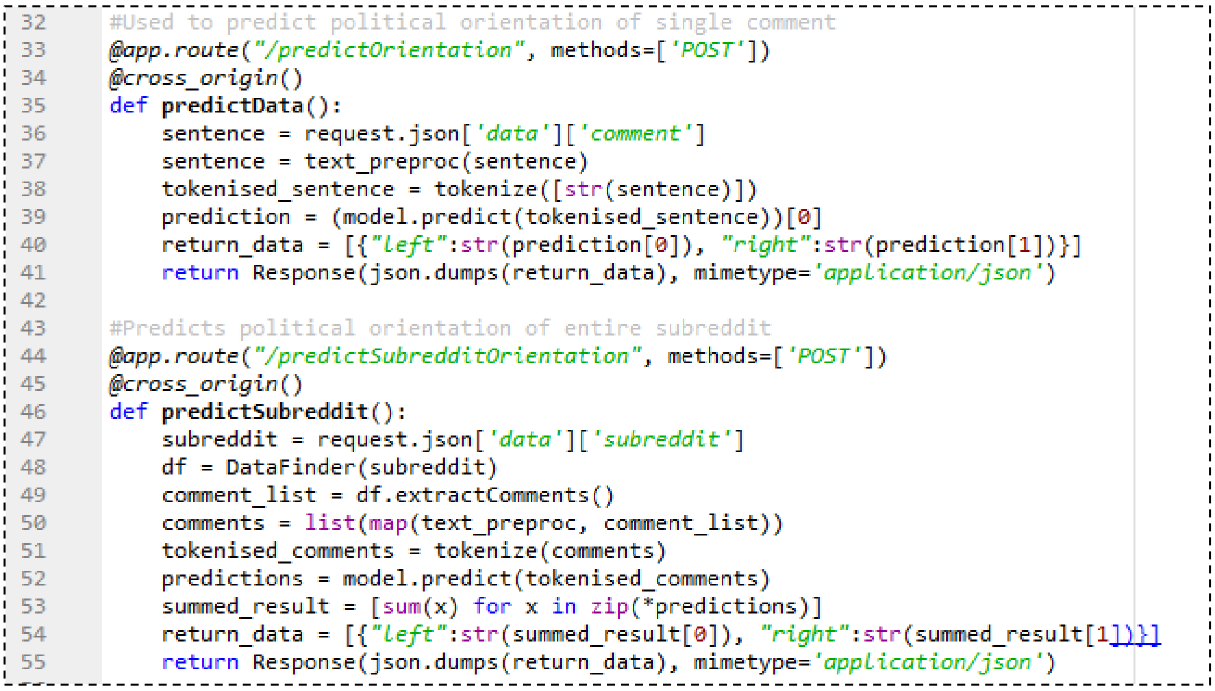
```python
#Test Route
@app.route("/")
@cross_origin()
def main():
    return "Hello World, Server running."
```

Figure 15: Test Endpoint from FlaskBackend class

modelSupport:

Endpoint's: *'/predictOrientation'*, *'/predictSubredditOrientation'*

Description: Handles pre-processing text, tokenising data, and importing Tensorflow model layers to load the models weights.



```python
#Used to predict political orientation of single comment
@app.route("/predictOrientation", methods=['POST'])
@cross_origin()
def predictData():
    sentence = request.json['data']['comment']
    sentence = text_preproc(sentence)
    tokenised_sentence = tokenize([str(sentence)])
    prediction = (model.predict(tokenised_sentence))[0]
    return_data = [{"left":str(prediction[0]), "right":str(prediction[1])}]
    return Response(json.dumps(return_data), mimetype='application/json')

#Predicts political orientation of entire subreddit
@app.route("/predictSubredditOrientation", methods=['POST'])
@cross_origin()
def predictSubreddit():
    subreddit = request.json['data']['subreddit']
    df = DataFinder(subreddit)
    comment_list = df.extractComments()
    comments = list(map(text_preproc, comment_list))
    tokenised_comments = tokenize(comments)
    predictions = model.predict(tokenised_comments)
    summed_result = [sum(x) for x in zip(*predictions)]
    return_data = [{"left":str(summed_result[0]), "right":str(summed_result[1])}]
    return Response(json.dumps(return_data), mimetype='application/json')
```

Figure 16: Predict Data Endpoints from FlaskBackend class

DataFinder:

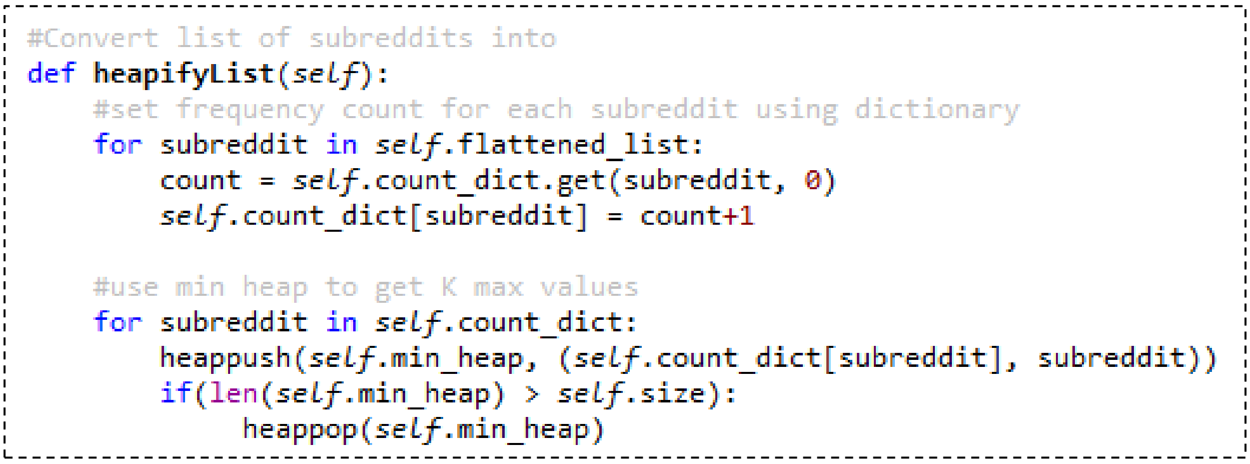
Endpoint: *'/pullSubreddits' 'predictSubredditOrientation'*

Description: Extract data from Reddit using a mixture of Reddit's Official API and The PushShift API. It uses an *extractComments* function to pull comments from a given subreddit. This is used in the */predictSubredditOrientation* endpoint. Its main feature is the *pullSubreddits* function which extracts a list of subreddits commonly used by the top *100* Redditors on a given subreddit.

VennDiagram:

Endpoint: *'/pullSubreddits'*

Description: This class takes a list of subreddits from the *DataFinder* class and calculates a min-heap of the k-most-frequent subreddits. It converts the resultant array into a JSON format.



```python
#Convert list of subreddits into
def heapifyList(self):
    #set frequency count for each subreddit using dictionary
    for subreddit in self.flattened_list:
        count = self.count_dict.get(subreddit, 0)
        self.count_dict[subreddit] = count+1

    #use min heap to get K max values
    for subreddit in self.count_dict:
        heappush(self.min_heap, (self.count_dict[subreddit], subreddit))
        if(len(self.min_heap) > self.size):
            heappop(self.min_heap)
```

Figure 17: K-nearest-Neighbour algorithm implementation for most-frequent subreddits

2.4 Front End Class Design

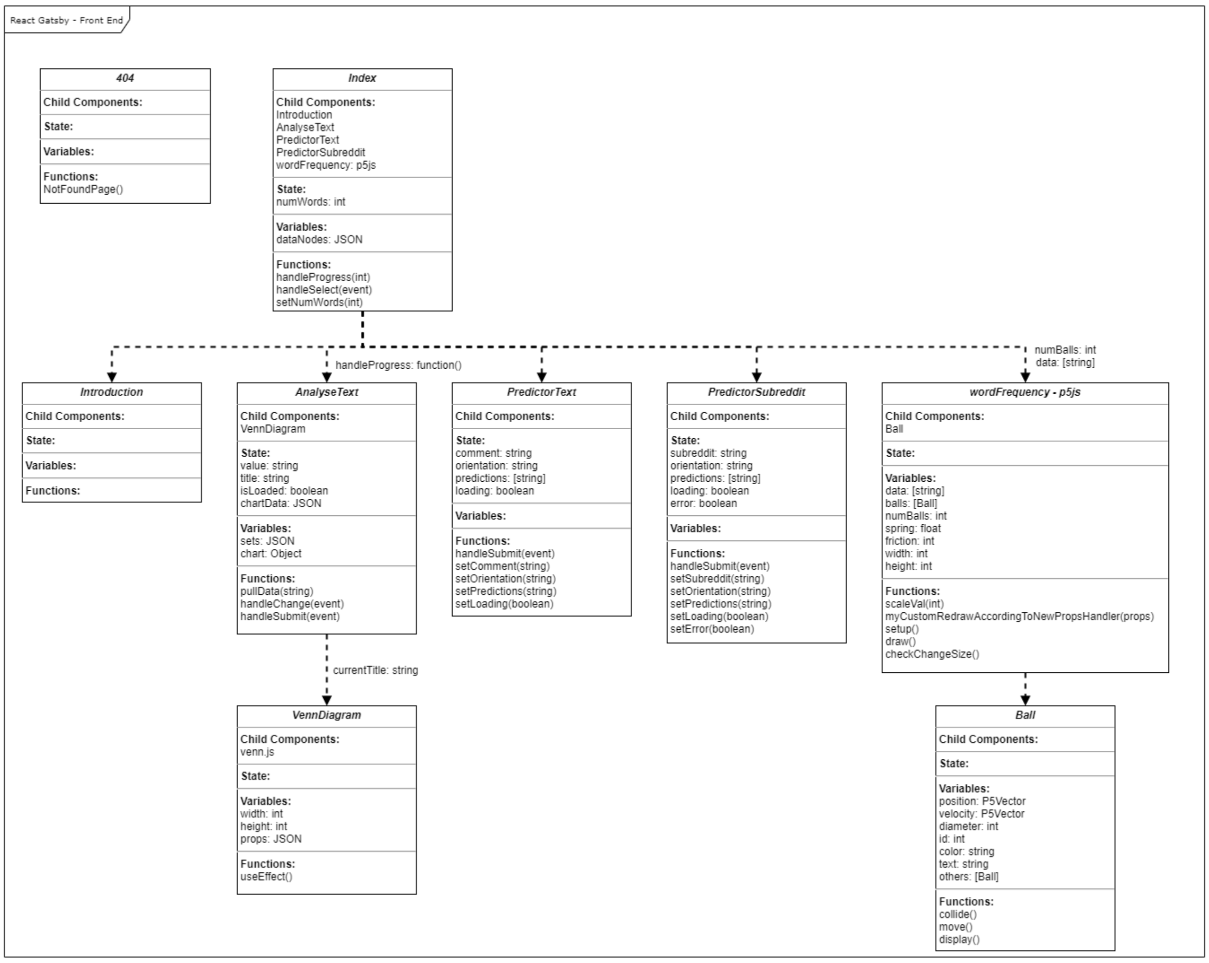


Figure 18: React Components Class Diagram

2.4.1 – Section Breakdown

The Front-End section of the project is a single page design, which is split up into five visual components which belong to the main class *Index*,

The Introduction (Introduction): a clean introduction page with a slanted div styling to introduce users to the project.

The Venn-Diagram (AnalyseText): visually appealing, bright Venn diagram tool with interactive visual elements that enable the user to search for intersecting subreddits.

Word Frequency List (WordFrequency): sketch playground that enables a unique and user-friendly method for viewing the generated word frequency list.

Text Predictor (PredictorText): A clean form for user input used to predict commentary orientation which is saved into a table of elements.

Subreddit Political Predictor (PredictorSubreddit): A similar clean form to the *text predictor* that enables prediction of a chosen subreddit in which the data is stored in an HTML table.

2.4.2 – Component Design and Implementation

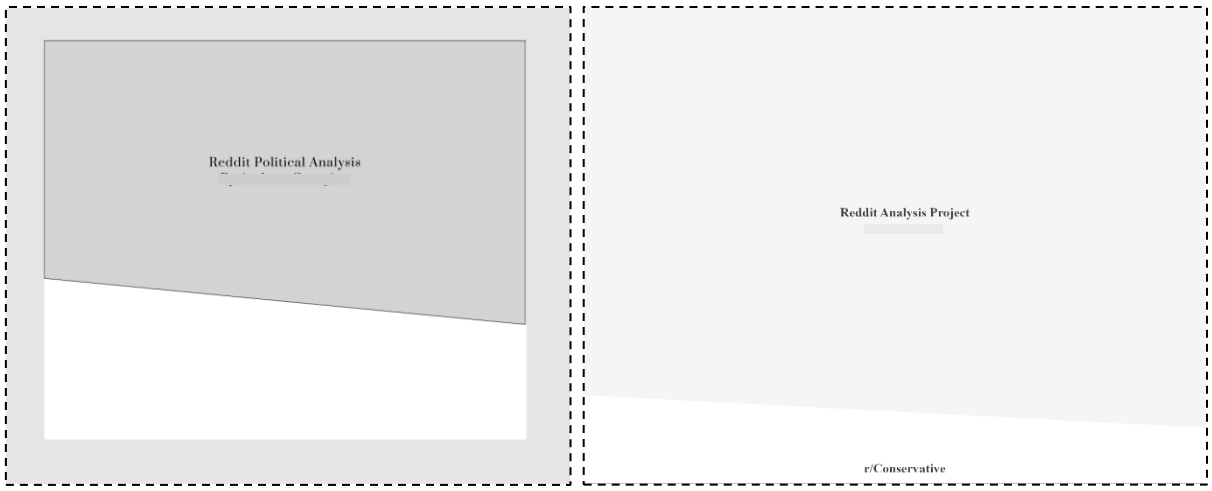


Figure 19: Introduction Design in Figma / Implemented Introduction Design

The Introduction

The initial design for the introduction intended to be professional for this academic project, the light grey background intended to let the component stand out.

The slanted div is a common design feature to show the content below that suggests to the user to scroll down. Since this is a single page design, it is essential to signify this to the user.

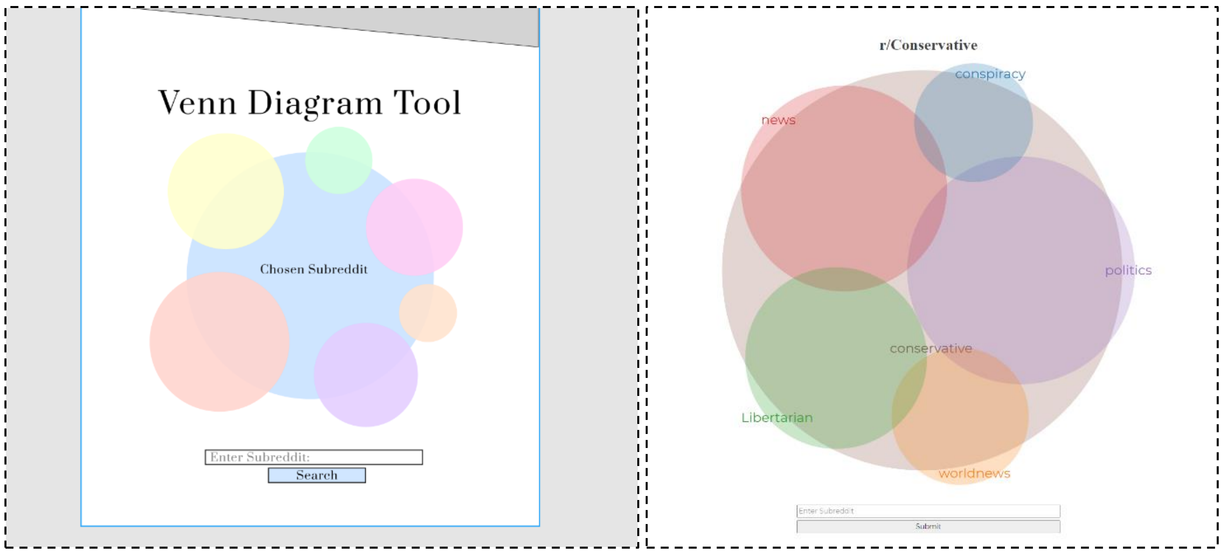


Figure 20: Venn Diagram Figma Design / Implemented Venn Diagram Tool

Venn Diagram Tool:

This tool was intended to be a very user-friendly, colourful tool for viewing intersecting subreddits. The Figma design intended to use bright colours, although in implementing the tool, there were limitations with the d3 library that enabling outward-facing circle groups would cause issues with rendering. Overall, the implementation still bears some semblance to the original design.

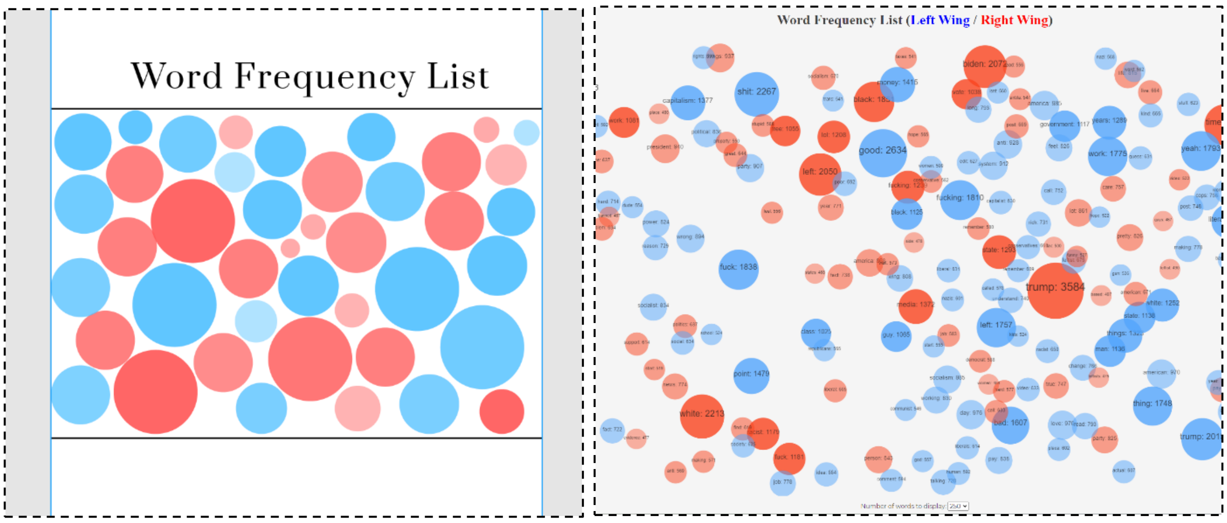


Figure 21: Word Frequency List Design / Implemented Word Frequency List

Word Frequency List:

Initially designed to have an appealing method of visualising words with each word being held in a word bubble, the bubbles size dependant on the frequency of that word. Filling the screen with no bubbles overlapping as per the original design was processor-heavy. Instead, it created bubbles randomly positioned and used collision detection to keep them separate works equally well with less user lag.

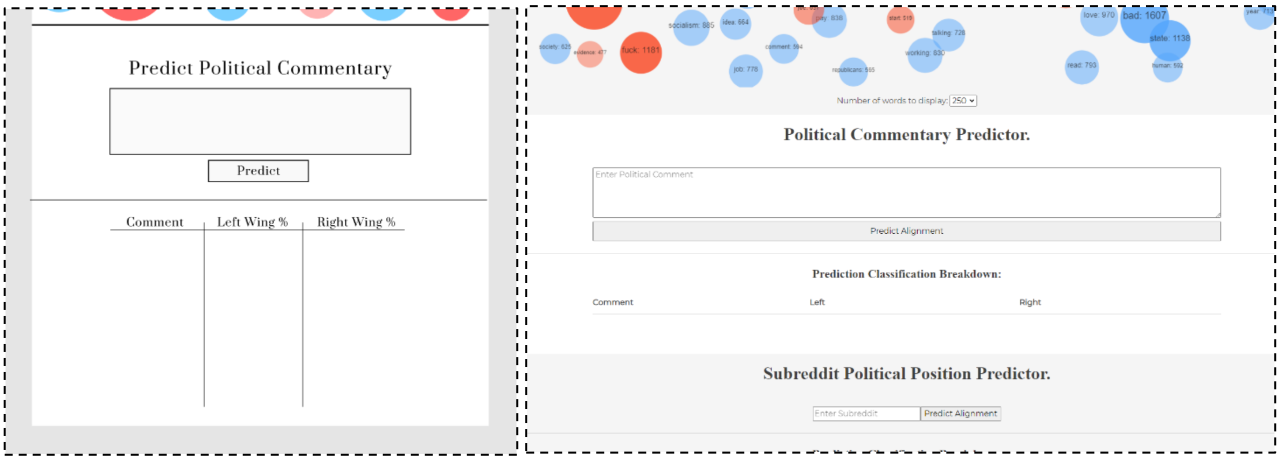


Figure 22: Political Predictor's Design / Actual Implementation

The political commentary and subreddit position predictor share the same design, a simple form with a table element that expands as data is entered. This design allows the content to be compact and easily scalable for mobile devices.

3. Implementation

This section will discuss implementing each component of this project and discuss some of the issues faced during the implementation. Each section is broken down into a problem description that describes the outline of the task before implementation, and the sections following highlight stages of the development process.

3.1 Mining and Pre-processing data

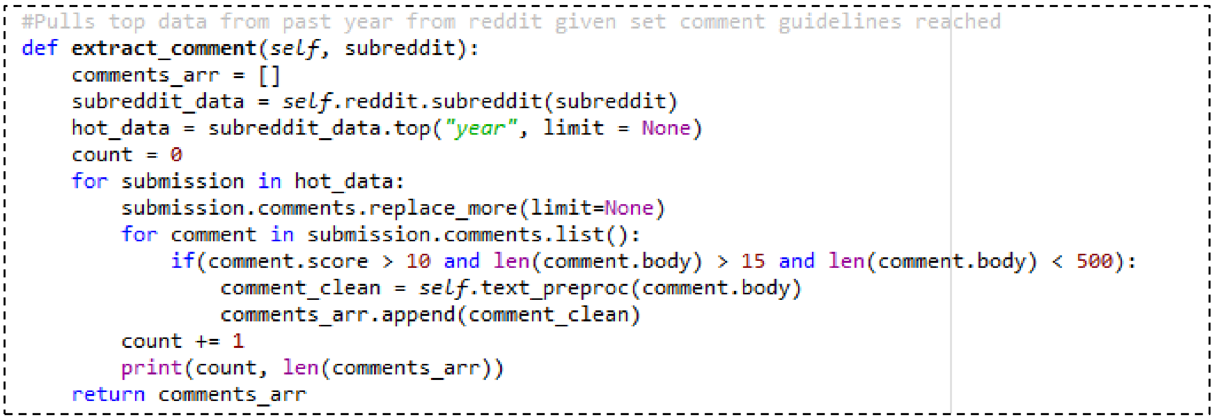
3.1.1 – Chapter Description

This chapter covers the first part of implementing this project: scraping, processing, and labelling the data from Reddit. This was done through the programming language Python. It discusses the issues surrounding extracting valuable non-subjective data for training a machine learning model.

3.1.2 – Extracting Data

Reddit's Official API, PRAW, allows for the extraction of up to 100 items per request, although requests greater than 100 items are split up into multiple requests, each with a 2-second delay as per their API guidelines [22]. This was the first issue faced with using Reddit's API to request a million comments for processing and extraction, taking over five and a half hours. In reality, this many comments can take a few days due to code execution times.

Filters were used to help extract more useful data to save extraction time, and one such filter is to only extract comments from the 'top′ posts within the past year. This helps as these posts represent the subreddit more than new or unpopular posts that may be contrary to community opinion.

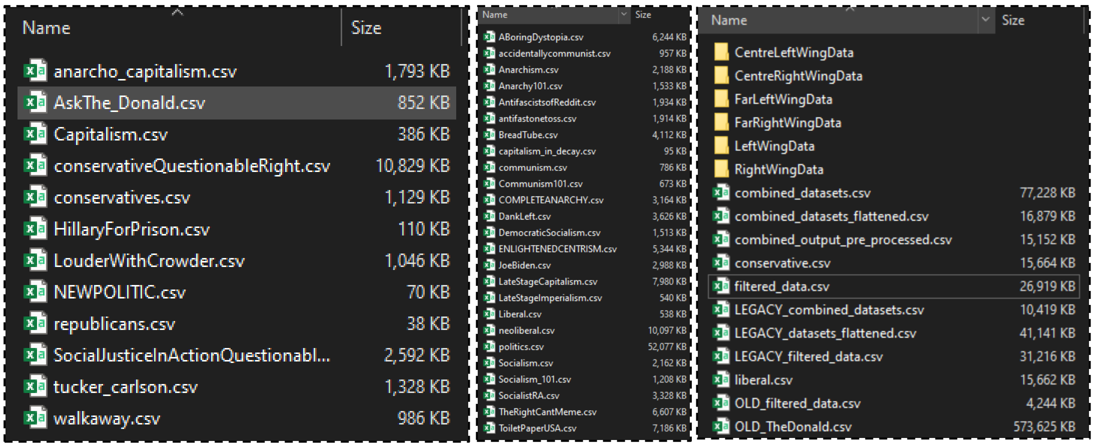


```python
#Pulls top data from past year from reddit given set comment guidelines reached
def extract_comment(self, subreddit):
    comments_arr = []
    subreddit_data = self.reddit.subreddit(subreddit)
    hot_data = subreddit_data.top("year", limit = None)
    count = 0
    for submission in hot_data:
        submission.comments.replace_more(limit=None)
        for comment in submission.comments.list():
            if(comment.score > 10 and len(comment.body) > 15 and len(comment.body) < 500):
                comment_clean = self.text_preproc(comment.body)
                comments_arr.append(comment_clean)
        count += 1
        print(count, len(comments_arr))
    return comments_arr
```

Figure 23: Python function for extracting "top" comments from a given subreddit.

Higher rated comments within each subreddit represent that community more than unpopular comments. For this reason, filtering out comments below a score threshold can produce fewer volatile data for the model.

Fig. 23 shows the *extract_comment* function, which for each subreddit pulls the top posts from the past year. The maximum limit for submission extraction 1000. For each submission, it loops through the top 1000 comments. Each comment is a tree of further nested comments, accessed by using PRAW's *replace_more* function.



Name	Size
anarcho_capitalism.csv	1,793 KB
AskThe_Donald.csv	852 KB
Capitalism.csv	386 KB
conservativeQuestionableRight.csv	10,829 KB
conservatives.csv	1,129 KB
HillaryForPrison.csv	110 KB
LouderWithCrowder.csv	1,046 KB
NEWPOLITIC.csv	70 KB
republicans.csv	38 KB
SocialJusticeInActionQuestionabl...	2,592 KB
tucker_carlson.csv	1,328 KB
walkaway.csv	986 KB

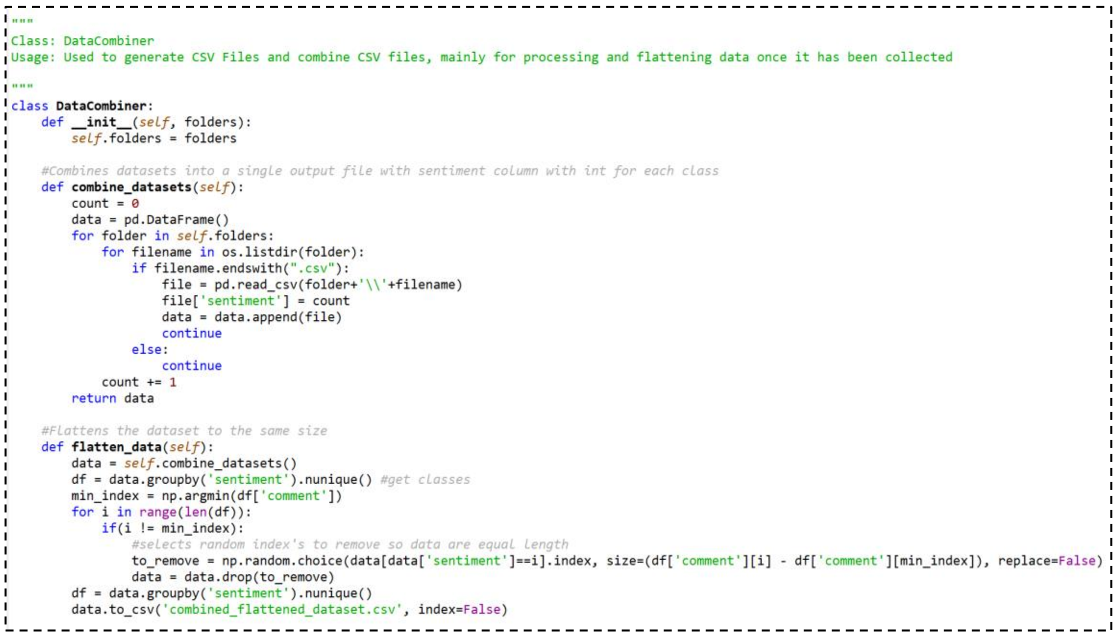
Name	Size
ABoringDystopia.csv	6,244 KB
accidentallycommunist.csv	957 KB
Anarchism.csv	2,188 KB
Anarchy101.csv	1,533 KB
AntifascistsofReddit.csv	1,934 KB
antifastonetoss.csv	1,914 KB
BreadTube.csv	4,112 KB
capitalism_in_decay.csv	95 KB
communism.csv	786 KB
Communism101.csv	673 KB
COMPLETEANARCHY.csv	3,164 KB
DankLeft.csv	3,626 KB
DemocraticSocialism.csv	1,513 KB
ENLIGHTENEDCENTRISM.csv	5,344 KB
JoeBiden.csv	2,988 KB
LateStageCapitalism.csv	7,980 KB
LateStageImperialism.csv	540 KB
Liberal.csv	538 KB
neoliberal.csv	10,097 KB
politics.csv	52,077 KB
Socialism.csv	2,162 KB
Socialism_101.csv	1,208 KB
SocialistRA.csv	3,328 KB
TheRightCantMeme.csv	6,607 KB
ToiletPaperUSA.csv	7,186 KB

Name	Size
CentreLeftWingData	
CentreRightWingData	
FarLeftWingData	
FarRightWingData	
LeftWingData	
RightWingData	
combined_datasets.csv	77,228 KB
combined_datasets_flattened.csv	16,879 KB
combined_output_pre_processed.csv	15,152 KB
conservative.csv	15,664 KB
filtered_data.csv	26,919 KB
LEGACY_combined_datasets.csv	10,419 KB
LEGACY_datasets_flattened.csv	41,141 KB
LEGACY_filtered_data.csv	31,216 KB
liberal.csv	15,662 KB
OLD_filtered_data.csv	4,244 KB
OLD_TheDonald.csv	573,625 KB

Figure 24: Examples of datasets extracted, flattened, filtered, and combined

With Reddit's API timeout of 2 seconds per 100 objects, execution will wait at least 20,000 seconds or five and a half hours per subreddit. However, with converting each submissions generator object into a list to loop all comments and with the processing time of each comment, the actual code execution time for some subreddits exceeded 12 hours.

This execution time was acceptable for mining data as it was not time reliant, but when live data is extracted for the backend, it would not be sensible to have the user wait such a long time between requests. The Reddit API PushShift is used to handle this problem and is *discussed in section 3.3.2.*



```python
"""
Class: DataCombiner
Usage: Used to generate CSV Files and combine CSV files, mainly for processing and flattening data once it has been collected

"""
class DataCombiner:
    def __init__(self, folders):
        self.folders = folders

    #Combines datasets into a single output file with sentiment column with int for each class
    def combine_datasets(self):
        count = 0
        data = pd.DataFrame()
        for folder in self.folders:
            for filename in os.listdir(folder):
                if filename.endswith(".csv"):
                    file = pd.read_csv(folder+'\\'+filename)
                    file['sentiment'] = count
                    data = data.append(file)
                    continue
                else:
                    continue
            count += 1
        return data

    #Flattens the dataset to the same size
    def flatten_data(self):
        data = self.combine_datasets()
        df = data.groupby('sentiment').nunique() #get classes
        min_index = np.argmin(df['comment'])
        for i in range(len(df)):
            if(i != min_index):
                #selects random index's to remove so data are equal length
                to_remove = np.random.choice(data[data['sentiment']==i].index, size=(df['comment'][i] - df['comment'][min_index]), replace=False)
                data = data.drop(to_remove)
        df = data.groupby('sentiment').nunique()
        data.to_csv('combined_flattened_dataset.csv', index=False)
```

Figure 25: Data Combiner Class in Python Programming Language

The above class imports all the CSV files from each folder (*left-wing and right-wing*), combines the files, appends the sentiment class column, and then flattens the dataset randomly to ensure an equal number of left-wing to right-wing comments. Of course, the dataset must be flat, or our model could begin to over-classify a single class.

3.2 Training and Fine-Tuning Models

3.2.1 – Problem Description.

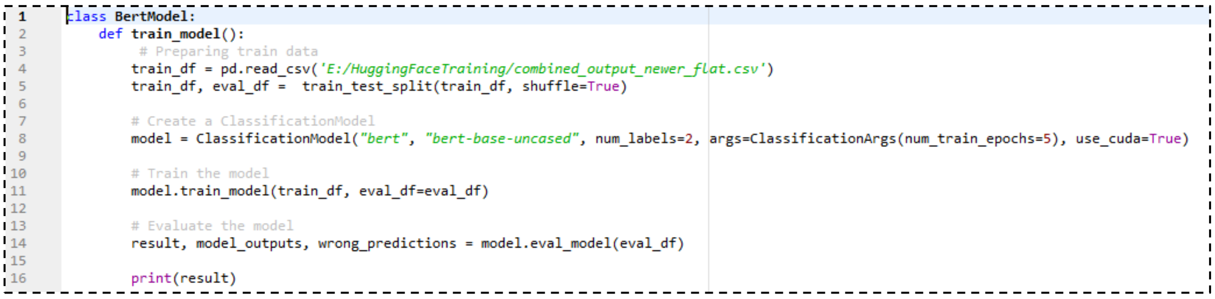
Training the machine learning model used as the commentary classifier was a significant undertaking that took a substantial proportion of time to complete.

Through this chapter, there will be a discussion on identifying some of the roadblocks faced when training models, such as problems with data and pre-trained models overfitting.

3.2.2 – Validating Data Efficacy.

The models were initially fine-tuned using the SimpleTransformers library. This library allows for fast prototyping of pre-trained models on a selection of tasks such as binary classification, question answering, and conversational AI.

SimpleTransformers is built to be a lightweight solution to fine-tuning models in Python, which is helpful in this project for validating the quality of data.



```python
class BertModel:
    def train_model():
         # Preparing train data
        train_df = pd.read_csv('E:/HuggingFaceTraining/combined_output_newer_flat.csv')
        train_df, eval_df =  train_test_split(train_df, shuffle=True)

        # Create a ClassificationModel
        model = ClassificationModel("bert", "bert-base-uncased", num_labels=2, args=ClassificationArgs(num_train_epochs=5), use_cuda=True)

        # Train the model
        model.train_model(train_df, eval_df=eval_df)

        # Evaluate the model
        result, model_outputs, wrong_predictions = model.eval_model(eval_df)

        print(result)
```

Figure 26: Simple example of fine-tuning BERT using SimpleTransformers.

When first fine-tuning the models, the training accuracy would increase exponentially whilst the validation accuracy would stagnate around 50%, a guess for binary classification.

Testing the training parameters is the first measure taken to identify the issue. Testing the model using a trusted dataset such as the spam detection dataset achieved 90%+ accuracy on validation data, therefore showing no issue with the model or the training parameters, which meant an issue must lie with the extracted dataset.

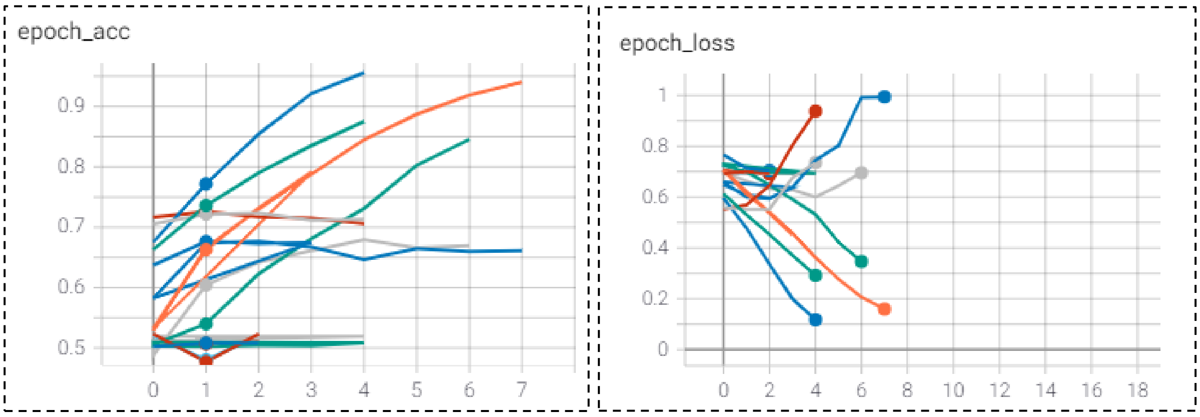
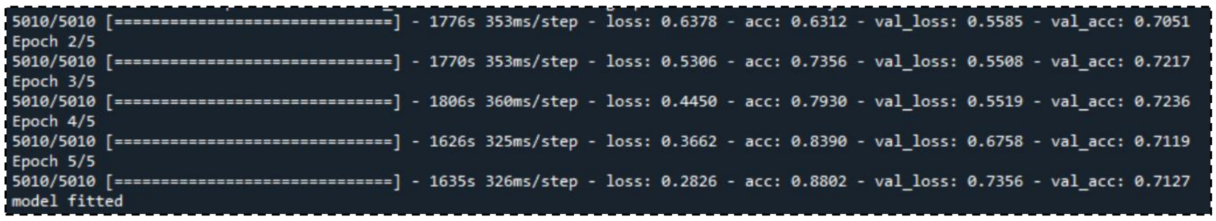


Figure 27: Graph showing initial models, overfitting with rising validation loss.

Examining the dataset and the subreddits used revealed an issue with the subreddits subjectively chosen that were not correctly labelled and required a solution in extracting objective comments, which is where the tool *discussed in section 3.3.2* was developed.

The dataset was re-built using the new extraction method. Results became closer to expectation. However, overfitting remained an issue. The validation accuracy was rising and peaking in the 60-80% range before decreasing as the model would no longer generalise.



```
5010/5010 [==============================] - 1776s 353ms/step - loss: 0.6378 - acc: 0.6312 - val_loss: 0.5585 - val_acc: 0.7051
Epoch 2/5
5010/5010 [==============================] - 1770s 353ms/step - loss: 0.5306 - acc: 0.7356 - val_loss: 0.5508 - val_acc: 0.7217
Epoch 3/5
5010/5010 [==============================] - 1806s 360ms/step - loss: 0.4450 - acc: 0.7930 - val_loss: 0.5519 - val_acc: 0.7236
Epoch 4/5
5010/5010 [==============================] - 1626s 325ms/step - loss: 0.3662 - acc: 0.8390 - val_loss: 0.6758 - val_acc: 0.7119
Epoch 5/5
5010/5010 [==============================] - 1635s 326ms/step - loss: 0.2826 - acc: 0.8802 - val_loss: 0.7356 - val_acc: 0.7127
model fitted
```

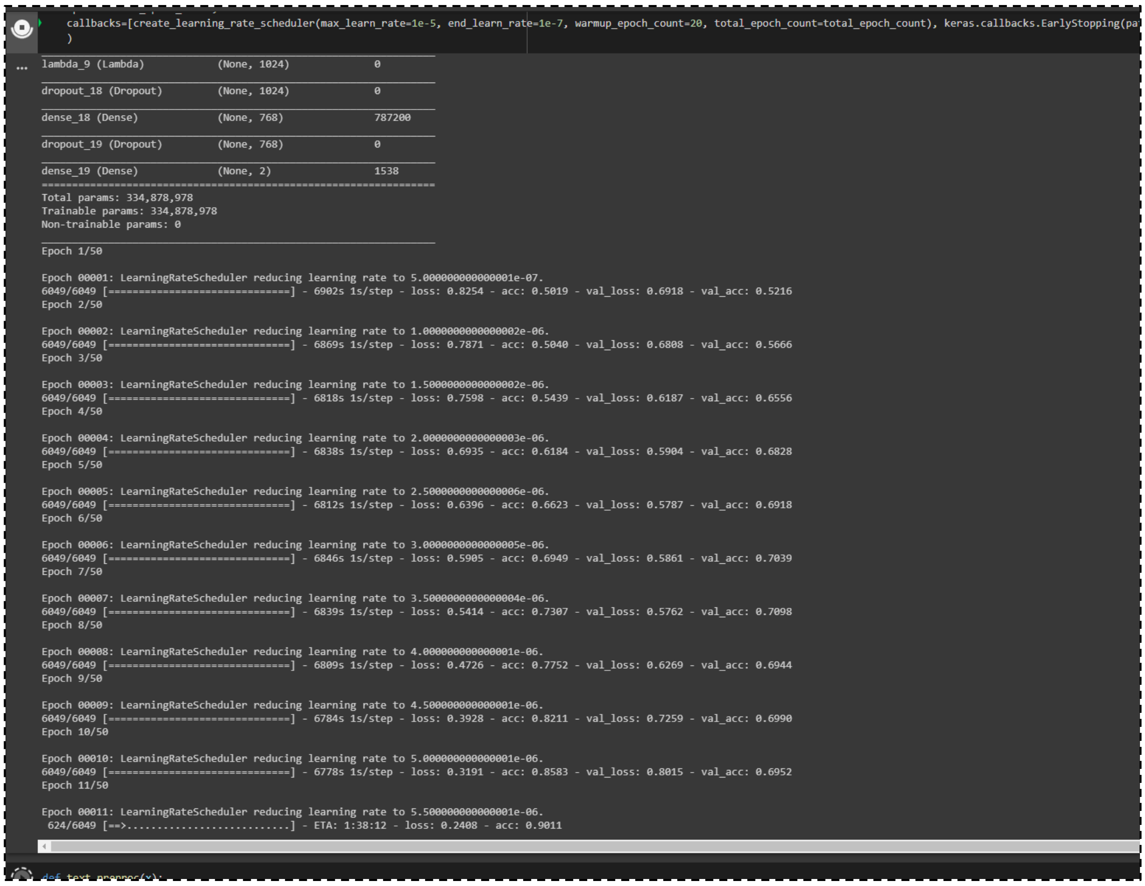
Figure 28: Example accuracy/loss achieved through SimpleTransformers DistilBERT.

This is why SimpleTransformers was not used in the final model. There is a lack of control over the model's output layers since it was not built on PyTorch or Tensorflow. Using a learning rate scheduler was not possible, and GPU utilisation was underperforming using the Tensorflow-GPU library.

3.2.3 – Overcoming Overfitting

The SimpleTransformers library inherits from HuggingFace transformers, a library that allowed more control of the model. TensorFlow layers could be defined and fine-tuned on top of the pre-trained model. In addition, freezing the model layers was possible, decreasing overfitting on such a large parameter model as it reduces the number of trainable nodes.

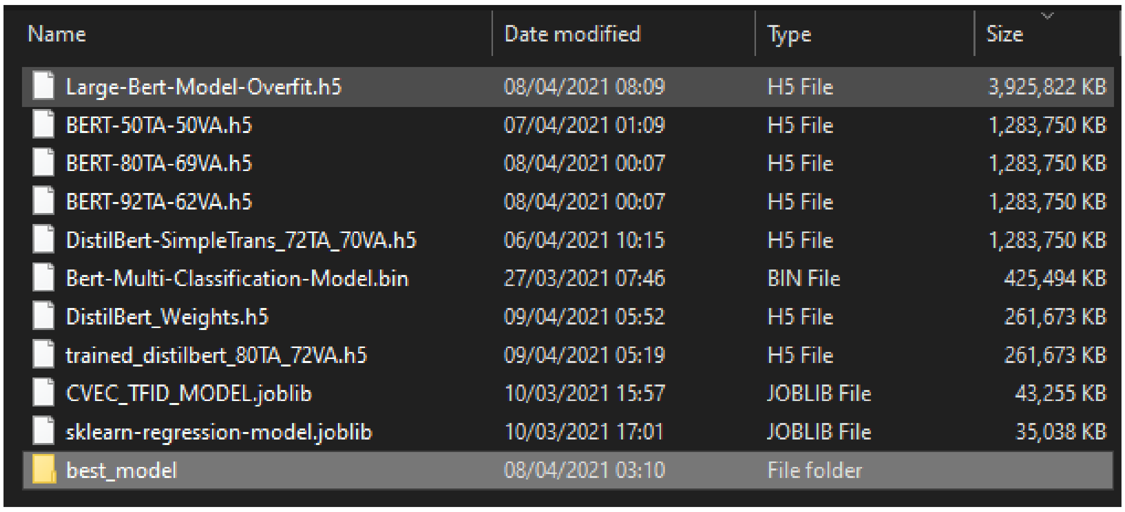
Overfitting was a significant issue with bigger pre-learned models such as BERT or RoBERTa. The optimal model ended up being the much smaller DistilBERT with better performance by utilising the masked word prediction functionality and the smaller number of parameters, limiting the possibility of overfitting.



```
    callbacks=[create_learning_rate_scheduler(max_learn_rate=1e-5, end_learn_rate=1e-7, warmup_epoch_count=20, total_epoch_count=total_epoch_count), keras.callbacks.EarlyStopping(pa
    )
______________________________________________________________
lambda_9 (Lambda)           (None, 1024)             0
______________________________________________________________
dropout_18 (Dropout)        (None, 1024)             0
______________________________________________________________
dense_18 (Dense)            (None, 768)              787200
______________________________________________________________
dropout_19 (Dropout)        (None, 768)              0
______________________________________________________________
dense_19 (Dense)            (None, 2)                1538
==============================================================
Total params: 334,878,978
Trainable params: 334,878,978
Non-trainable params: 0
______________________________________________________________
Epoch 1/50

Epoch 00001: LearningRateScheduler reducing learning rate to 5.000000000000001e-07.
6049/6049 [==============================] - 6902s 1s/step - loss: 0.8254 - acc: 0.5019 - val_loss: 0.6918 - val_acc: 0.5216
Epoch 2/50

Epoch 00002: LearningRateScheduler reducing learning rate to 1.0000000000000002e-06.
6049/6049 [==============================] - 6869s 1s/step - loss: 0.7871 - acc: 0.5040 - val_loss: 0.6808 - val_acc: 0.5666
Epoch 3/50

Epoch 00003: LearningRateScheduler reducing learning rate to 1.5000000000000002e-06.
6049/6049 [==============================] - 6818s 1s/step - loss: 0.7598 - acc: 0.5439 - val_loss: 0.6187 - val_acc: 0.6556
Epoch 4/50

Epoch 00004: LearningRateScheduler reducing learning rate to 2.0000000000000003e-06.
6049/6049 [==============================] - 6838s 1s/step - loss: 0.6935 - acc: 0.6184 - val_loss: 0.5904 - val_acc: 0.6828
Epoch 5/50

Epoch 00005: LearningRateScheduler reducing learning rate to 2.5000000000000006e-06.
6049/6049 [==============================] - 6812s 1s/step - loss: 0.6396 - acc: 0.6623 - val_loss: 0.5787 - val_acc: 0.6918
Epoch 6/50

Epoch 00006: LearningRateScheduler reducing learning rate to 3.0000000000000005e-06.
6049/6049 [==============================] - 6846s 1s/step - loss: 0.5905 - acc: 0.6949 - val_loss: 0.5861 - val_acc: 0.7039
Epoch 7/50

Epoch 00007: LearningRateScheduler reducing learning rate to 3.5000000000000004e-06.
6049/6049 [==============================] - 6839s 1s/step - loss: 0.5414 - acc: 0.7307 - val_loss: 0.5762 - val_acc: 0.7098
Epoch 8/50

Epoch 00008: LearningRateScheduler reducing learning rate to 4.000000000000001e-06.
6049/6049 [==============================] - 6809s 1s/step - loss: 0.4726 - acc: 0.7752 - val_loss: 0.6269 - val_acc: 0.6944
Epoch 9/50

Epoch 00009: LearningRateScheduler reducing learning rate to 4.500000000000001e-06.
6049/6049 [==============================] - 6784s 1s/step - loss: 0.3928 - acc: 0.8211 - val_loss: 0.7259 - val_acc: 0.6990
Epoch 10/50

Epoch 00010: LearningRateScheduler reducing learning rate to 5.000000000000001e-06.
6049/6049 [==============================] - 6778s 1s/step - loss: 0.3191 - acc: 0.8583 - val_loss: 0.8015 - val_acc: 0.6952
Epoch 11/50

Epoch 00011: LearningRateScheduler reducing learning rate to 5.500000000000001e-06.
 624/6049 [==>...........................] - ETA: 1:38:12 - loss: 0.2408 - acc: 0.9011

def text_preproc(x):
```

Figure 29: Example of fine-tuning RoBERTa's 300 Million+ parameters over 18 hours.

Three months were spent fine-tuning different models, mainly due to the time taken for training and experimenting with different data sets. For example, *Fig. 29* shows RoBERTa being fine-tuned in Google Colab on an Nvidia Tesla P100 GPU with a Learning Rate Scheduler applied. Most pre-trained models would take over 24 hours to reach their local or global minimum.



Name	Date modified	Type	Size
Large-Bert-Model-Overfit.h5	08/04/2021 08:09	H5 File	3,925,822 KB
BERT-50TA-50VA.h5	07/04/2021 01:09	H5 File	1,283,750 KB
BERT-80TA-69VA.h5	08/04/2021 00:07	H5 File	1,283,750 KB
BERT-92TA-62VA.h5	08/04/2021 00:07	H5 File	1,283,750 KB
DistilBert-SimpleTrans_72TA_70VA.h5	06/04/2021 10:15	H5 File	1,283,750 KB
Bert-Multi-Classification-Model.bin	27/03/2021 07:46	BIN File	425,494 KB
DistilBert_Weights.h5	09/04/2021 05:52	H5 File	261,673 KB
trained_distilbert_80TA_72VA.h5	09/04/2021 05:19	H5 File	261,673 KB
CVEC_TFID_MODEL.joblib	10/03/2021 15:57	JOBLIB File	43,255 KB
sklearn-regression-model.joblib	10/03/2021 17:01	JOBLIB File	35,038 KB
best_model	08/04/2021 03:10	File folder	

Figure 30: a small portion of best-trained models over 10GB of model data.

3.3 Creating and Hosting Backend

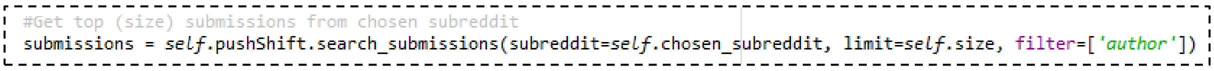
3.3.1 – Problem Definition

The backend was developed using Python's server library, Flask, hosted through Google Cloud; it was intended to be a fast method of hosting the model to enable commentary predictions to be served upon frontend upon request.

In this chapter, there is a discussion on developing the backend to create data science tools and serve the model's predictions. It also covers initial attempts at hosting the backend with the model and the problems faced during this task.

3.3.2 – Building the Venn Diagram Tool

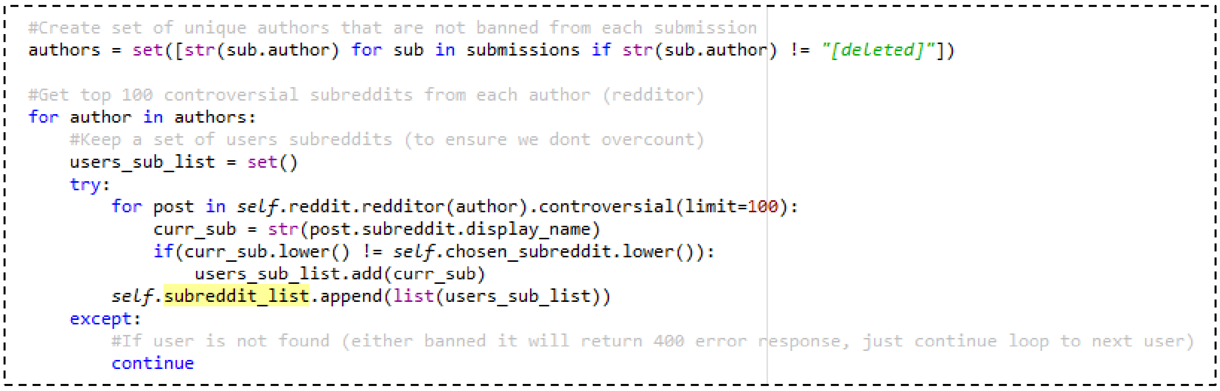
The Venn-Diagram tool was built to take a chosen subreddit and serve a data array of the most frequently used subreddits by users of that chosen subreddit.



```
#Get top (size) submissions from chosen subreddit
submissions = self.pushShift.search_submissions(subreddit=self.chosen_subreddit, limit=self.size, filter=['author'])
```

Figure 31: Function getting 100 Submissions by the author with single PushShift Request.

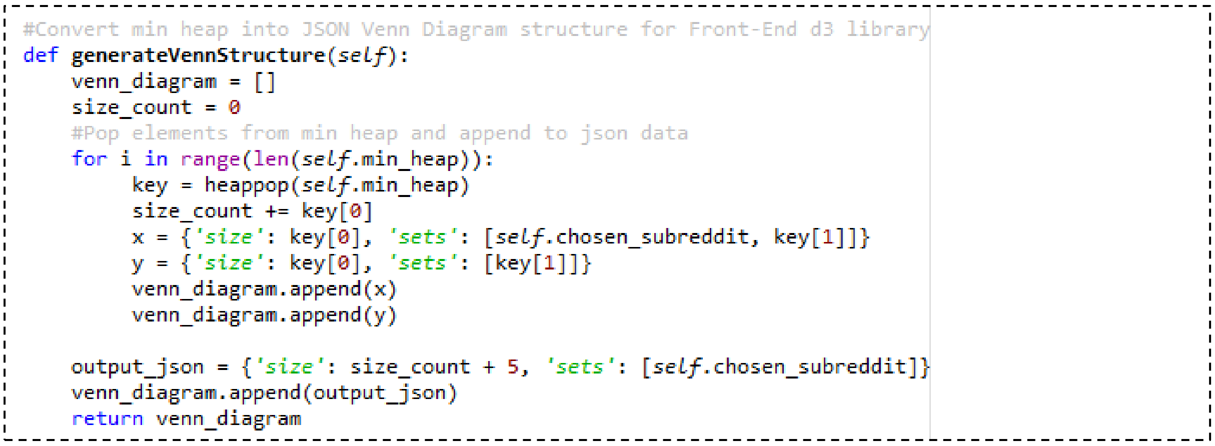
Getting the top 100 users of a subreddit takes 100 separate requests in PRAW, meaning an API delay of 200 seconds per user request. This is not feasible for the end-user experience. This roadblock was overcome by utilising The PushShift API, a Reddit API built using Elasticsearch, which allows for extensive aggregate searches with < 2 second response time.



```
#Create set of unique authors that are not banned from each submission
authors = set([str(sub.author) for sub in submissions if str(sub.author) != "[deleted]"])

#Get top 100 controversial subreddits from each author (redditor)
for author in authors:
    #Keep a set of users subreddits (to ensure we dont overcount)
    users_sub_list = set()
    try:
        for post in self.reddit.redditor(author).controversial(limit=100):
            curr_sub = str(post.subreddit.display_name)
            if(curr_sub.lower() != self.chosen_subreddit.lower()):
                users_sub_list.add(curr_sub)
        self.subreddit_list.append(list(users_sub_list))
    except:
        #If user is not found (either banned it will return 400 error response, just continue loop to next user)
        continue
```

Figure 32: Function getting 100 controversial subreddits per author.

The top 100 controversial submissions were extracted for each author. Controversial submissions, on average, outputted less common subreddits. This can often give more interesting results as more popular subreddits often lack political opinion.



```
#Convert min heap into JSON Venn Diagram structure for Front-End d3 library
def generateVennStructure(self):
    venn_diagram = []
    size_count = 0
    #Pop elements from min heap and append to json data
    for i in range(len(self.min_heap)):
        key = heappop(self.min_heap)
        size_count += key[0]
        x = {'size': key[0], 'sets': [self.chosen_subreddit, key[1]]}
        y = {'size': key[0], 'sets': [key[1]]}
        venn_diagram.append(x)
        venn_diagram.append(y)

    output_json = {'size': size_count + 5, 'sets': [self.chosen_subreddit]}
    venn_diagram.append(output_json)
    return venn_diagram
```

Figure 33: Function to convert list to JSON output for Front-End.

Initially, this tool was developed to help identify political subreddits objectively for model training datasets. Unfortunately, the version used for the dataset had run-times exceeded 10-15 minutes. This is not feasible for a frontend user who wish to explore the subreddits similarity in a less academically feasible way. The number of authors was therefore reduced from 1000 to 100, and the number of posts reduces equally.

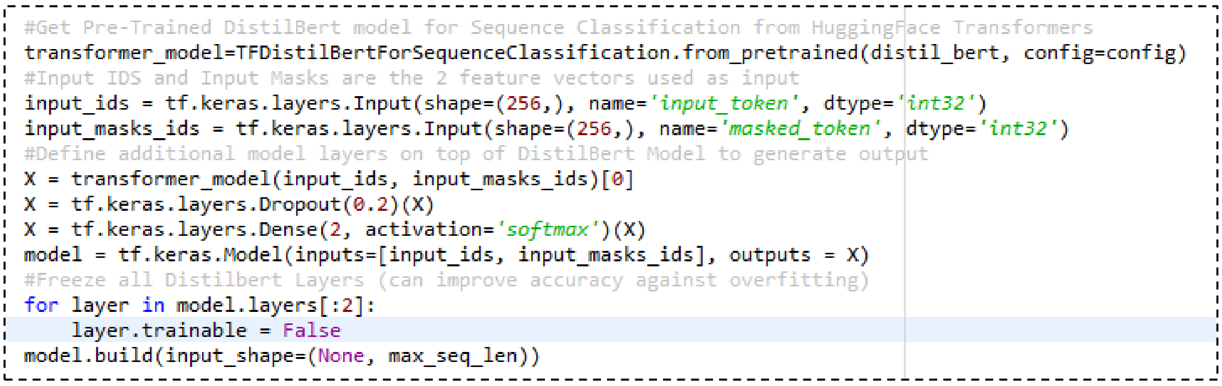
Subreddit	Subreddit's Calculated (k=7)	Frequency	Computation Time
Labour	*UKPolitics*	15	9.176 seconds
	LabourUK	14	
	UnitedKingdom	12	
	GreenAndPleasant	10	
	WorldNews	8	
	Europe	7	
	ABoringDystopia	5	
Conservative	*Politics*	21	11.087 seconds
	Conspiracy	16	
	Libertarian	13	
	Guns	12	
	Political_Revolution	7	
	PoliticalDiscussion	6	
	LouderWithCrowder	3	
Politics	*News*	23	10.897 seconds
	WorldNews	21	
	Technology	10	
	Science	10	
	Europe	9	
	Conservative	8	
	UnitedKingdom	5	

Figure 34: table of example uses of the Venn Diagram tool

3.3.3 – Serving Model Predictions

A significant component of this project is the ability to query the model to predict the political orientation of text. This requires being able to load in the model with all its weights and layers.

Model weights were stored in a 255MB file, although due to the use of custom layers, TensorFlow required the model layers to be built from scratch and the weights to be set into the custom model.



```
#Get Pre-Trained DistilBert model for Sequence Classification from HuggingFace Transformers
transformer_model=TFDistilBertForSequenceClassification.from_pretrained(distil_bert, config=config)
#Input IDS and Input Masks are the 2 feature vectors used as input
input_ids = tf.keras.layers.Input(shape=(256,), name='input_token', dtype='int32')
input_masks_ids = tf.keras.layers.Input(shape=(256,), name='masked_token', dtype='int32')
#Define additional model layers on top of DistilBert Model to generate output
X = transformer_model(input_ids, input_masks_ids)[0]
X = tf.keras.layers.Dropout(0.2)(X)
X = tf.keras.layers.Dense(2, activation='softmax')(X)
model = tf.keras.Model(inputs=[input_ids, input_masks_ids], outputs = X)
#Freeze all Distilbert Layers (can improve accuracy against overfitting)
for layer in model.layers[:2]:
    layer.trainable = False
model.build(input_shape=(None, max_seq_len))
```

Figure 35: Initialise TensorFlow model layers for weights to be loaded onto.

Fig. 35 shows loading the model layers into the backend *modelSupport* class, which will load in and be served tokenised text on the server startup. The output for this model is a two-node Dense layer which means the prediction serving model will generate an array with two weights between 0-1 for the prediction of left-wing and right-wing.

3.3.4 – Issues Hosting Model

Finding where to host the machine learning model was a problem faced before with a previous project where the solution was using Google Cloud. However, with this project looking at cheaper alternatives to the cloud platform was preferred going into development.

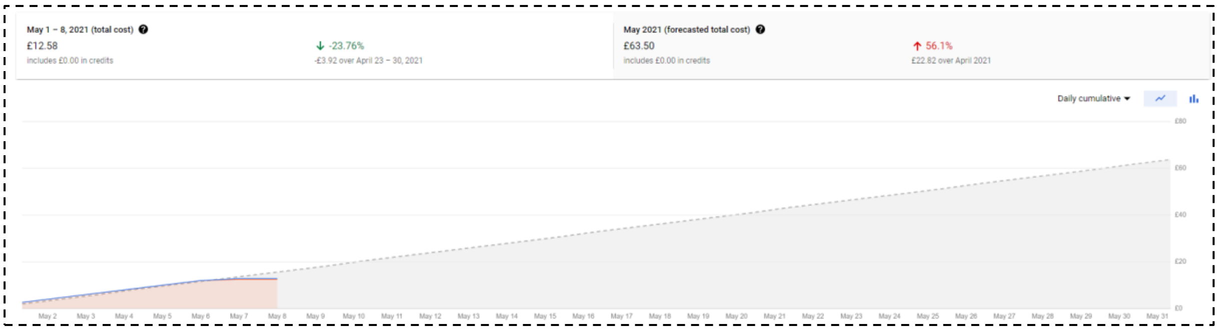


Figure 36: Current Costs and Forecasted Costs According to Google Cloud Platform

Running the model serving backend on 2x E2 instances (2 vCPU, 4Gb Memory), which runs behind a load balancer, costs around £65 per month on Google Cloud. However, they offer a free £300.00 credit to new accounts, which can help for prototyping setting up environments within Google's Compute Engine.

Attempting to use other platforms such as PythonAnywhere to host the model ran into an issue when the platform failed to support TensorFlow 2.0. A recent update has fixed this issue, although the platform cannot scale compute size or GPU size. Larger models require 4GB+ VRAM, which is impossible for PythonAnywhere causing MemoryOverflow crashes.

Setting up the model in Google Cloud's App Engine caused similar issues with scaling, where non-premium users cannot use automatic scaling without enabling premium billing services which meant losing control of the costs incurred for hosting. The solution to this issue was using a fixed service such as Google Cloud's Compute Engine.

3.3.5 – Hosting Backend on GCE

Hosting on Google's Compute Engine (GCE) fixed the issue surrounding hardware requirements. GCE has fixed machines that allow for specifying CPU, GPU, and RAM size on instance creation. However, being fixed means an inability to scale up hardware requirements to meet hosting demands and make it impossible to create or destroy machines without destroying the backend storage data.

Creating a Compute Engine, specifying the hardware requirements, and downloading and launching the backend was a manual process that could meet the performance demands, although it immediately presented some issues.

The hosted frontend could not contact the backend without causing an error due to the lack of security in request data via the external IP of the Compute Engine instance. This error was caused by not having an SSL certificate. Attempts at installing an SSL certificate into the machine through LetsEncrypt failed due to their SSL requirement to have a Fully Qualified Domain Name access point to generate a certificate.

Setting up an FQDN for a Compute Instance was a novel task. The solution being host the instance behind a load balancer which could terminate TLS and act as a Fully Qualified Domain Name through Google's domain name service.

Purchasing the domain name, https://www.RedditPoliticalBackend.com/ allowed generating an SSL certificate via Google Cloud's SSL management service. The DNS records were altered to direct all traffic from the domain to the external static IP of the Load Balancer. This enabled the SSL certificates to be hosted on the Load Balancer itself. Additionally, traffic on the domain is redirected to port 443 for HTTPS, enabling TLS termination at the Load Balancer.

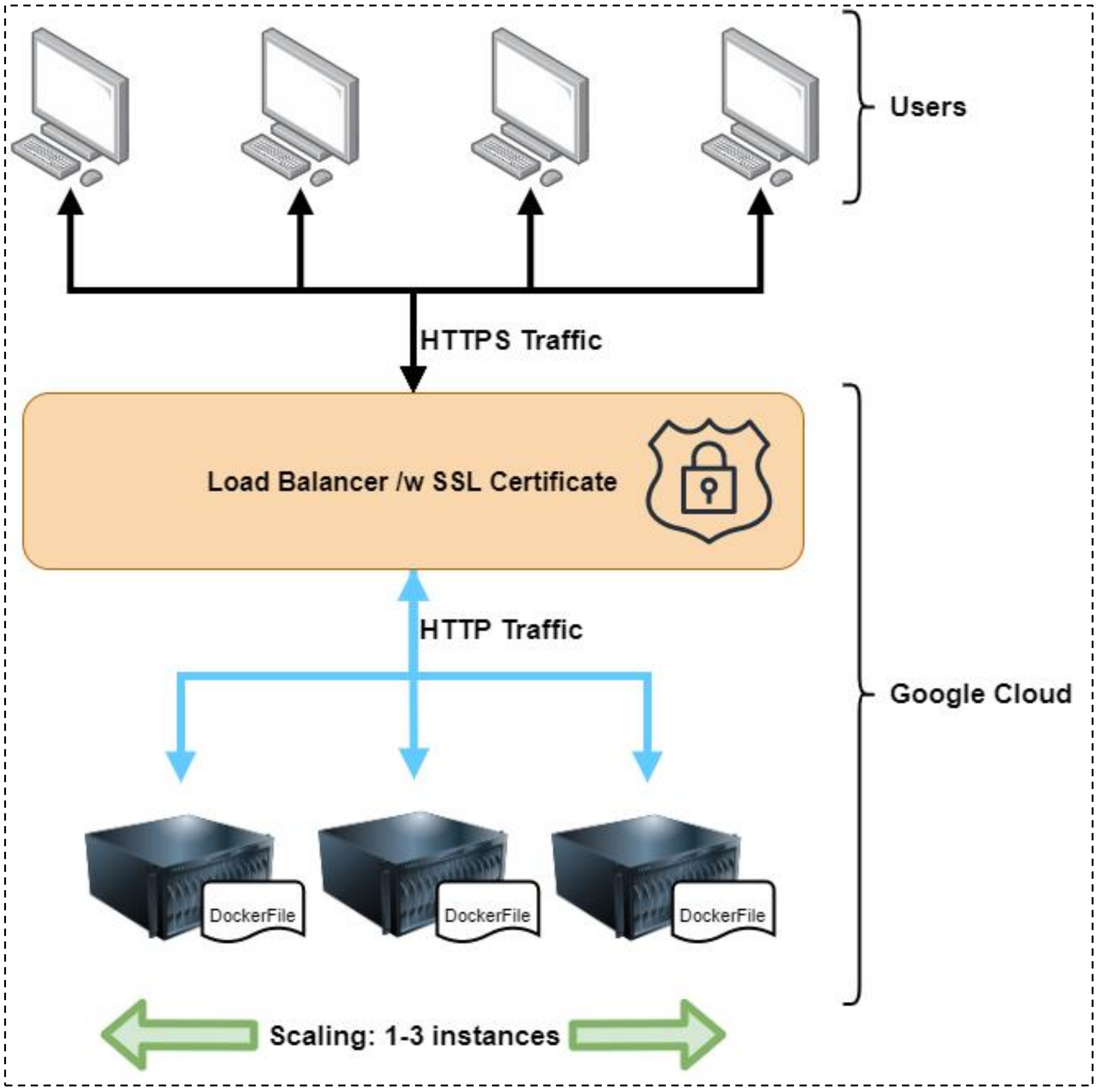


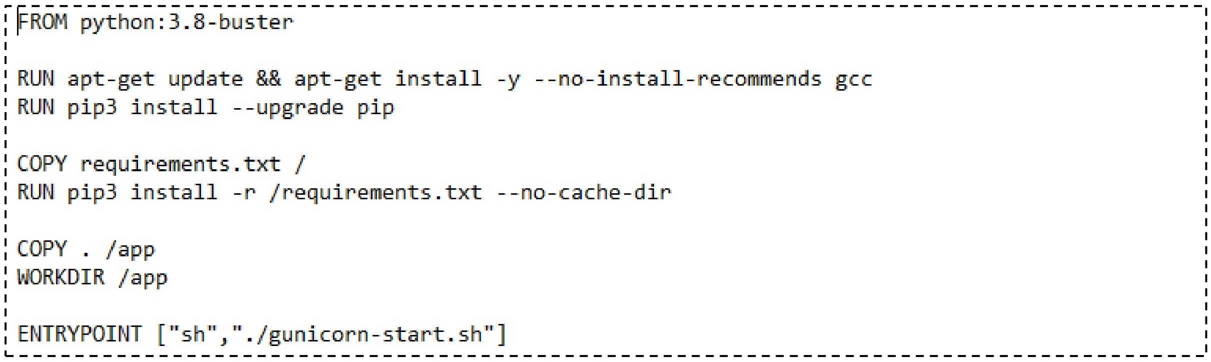
Figure 37: Hosted backend setup with load balancer and instance scaling.

A benefit of the load balancer is being able to run instance groups. This means the load balancer can scale the number of compute instances available to handle an increase in demand for computational power.

The load balancer can run health checks on each instance, and if an instance fails, it can destroy and create a new version of the instance. This caused the next issue, an inability to manually set up the backend and download the model onto each compute instance as they are constantly destroyed and re-created by the scaling group. Automatic instance installation could be managed through the service *Docker*.

3.3.6 – Developing with Docker for Instance Creation

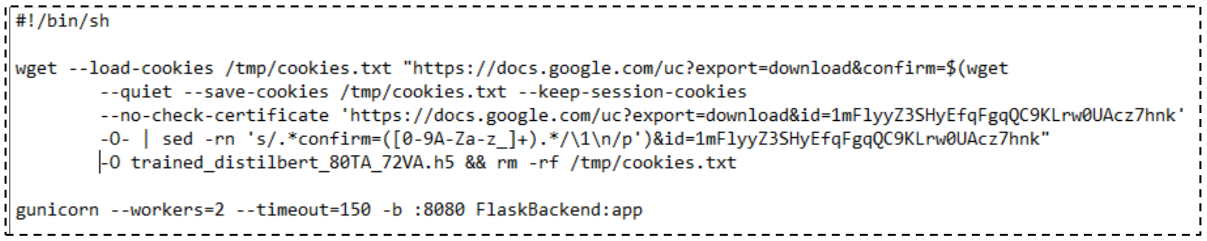
A DockerFile had to be developed to handle the automatic installation of the Flask backend. Docker is a tool for containerising an application and enabling a set of pre-built instructions that run on instance creation. The DockerFile is a set of instructions that tell the instance what commands to run and what order.



```
FROM python:3.8-buster

RUN apt-get update && apt-get install -y --no-install-recommends gcc
RUN pip3 install --upgrade pip

COPY requirements.txt /
RUN pip3 install -r /requirements.txt --no-cache-dir

COPY . /app
WORKDIR /app

ENTRYPOINT ["sh","./gunicorn-start.sh"]
```

Figure 38: DockerFile showing initial instance commands.

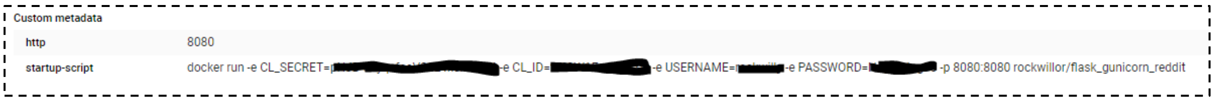
The model was too large to be pushed to GitHub or the Docker repository. Therefore the DockerFile had to pull the model from a hosted Google Drive file. After that, the rest of the code could be pushed into the Docker repository, and by using a *requirements* file for the required libraries, Docker could import all the required libraries in a single command.



```
#!/bin/sh

wget --load-cookies /tmp/cookies.txt "https://docs.google.com/uc?export=download&confirm=$(wget
        --quiet --save-cookies /tmp/cookies.txt --keep-session-cookies
        --no-check-certificate 'https://docs.google.com/uc?export=download&id=1mFlyyZ3SHyEfqFgqQC9KLrw0UAcz7hnk'
        -O- | sed -rn 's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=1mFlyyZ3SHyEfqFgqQC9KLrw0UAcz7hnk"
        -O trained_distilbert_80TA_72VA.h5 && rm -rf /tmp/cookies.txt

gunicorn --workers=2 --timeout=150 -b :8080 FlaskBackend:app
```

Figure 39: gunicorn-start.sh file showing downloading model and running backend.

The large model file required a complex *wget* command to download the model from Google Drive. The Flask backend is executed via *gunicorn*, a Web Server Gateway Interface library that enables running multiple workers on a single instance, allowing for serving multiple requests simultaneously.

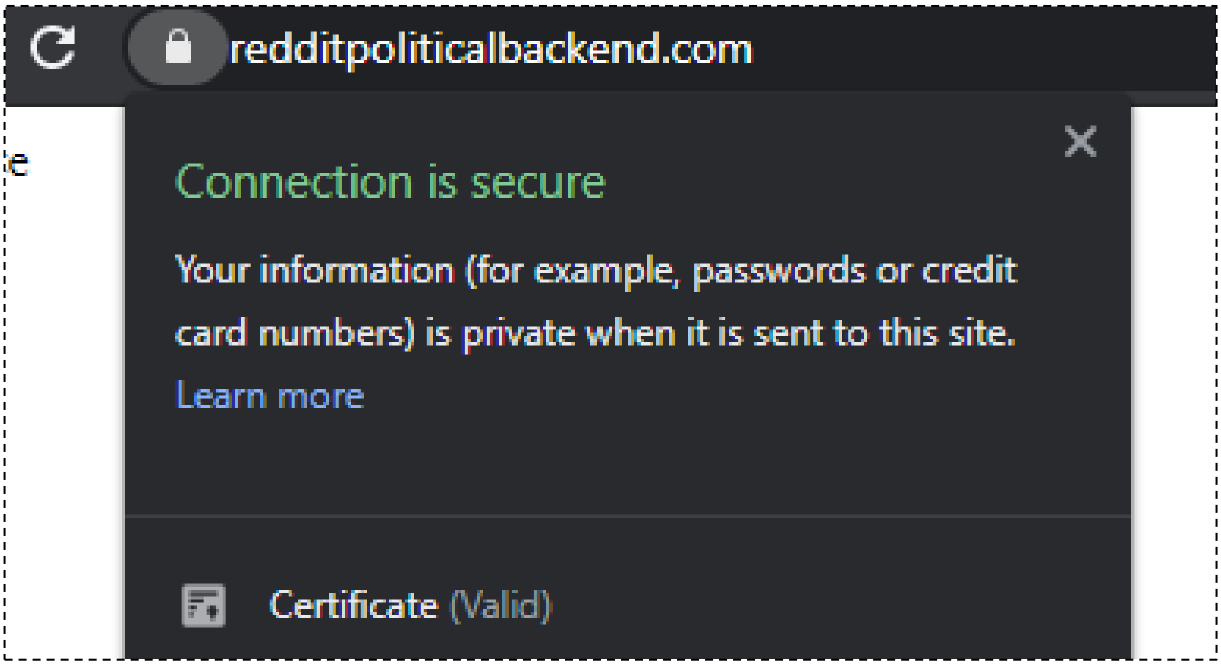


Custom metadata

http	8080
startup-script	docker run -e CL_SECRET=[redacted]-e CL_ID=[redacted]-e USERNAME=[redacted]-e PASSWORD=[redacted] -p 8080:8080 rockwillor/flask_gunicorn_reddit

Figure 40: Docker run script passed into the instance template. (variables removed)

One other issue noted with using Docker meant uploading the backend code to a public repository. PRAW has an API key and Secret key that must not be shared, and publishing the code without it will not work. The solution was using Docker environment variables in the docker starter script. They will now remain hidden only to a private Google Cloud platform account.



redditpoliticalbackend.com

e

Connection is secure

Your information (for example, passwords or credit card numbers) is private when it is sent to this site.

Learn more

Certificate (Valid)

Figure 41: Backend Hosted with FQDN and Valid SSL Certificates.

With three running instances and two workers per instance, the model can serve predictions simultaneously to 6 users with the ability through Load Balancing to scale to serve even more users to meet demands. Having SSL security also enables secure access for the user between the backend hosted at https://redditpoliticalbackend.com/ and the Front-End, https://reddit-political-analysis.com/.

3.4 Creating and Hosting Front-End

3.4.1 – Problem Definition.

The frontend was developed using React with Gatsby, a static site generator that enables rapid performance, domain routing, and built-in security features.

Developing the frontend to interact with the already developed backend meant ensuring data was handled correctly in every aspect of the application. This section covers developing the components and connecting them to the backend through Axios.

3.4.2 – Developing Components.

Similar Subreddit Visualiser:

Creating the Venn diagram visual component in React utilising the backend data *referenced in section 3.3.2* was made using the D3 visualisation library for data-driven documents. D3 has a selection of graphs, although a Venn diagram is not one of them. Implementing an additional library for D3 called VennJS gave a crude visualisation tool.

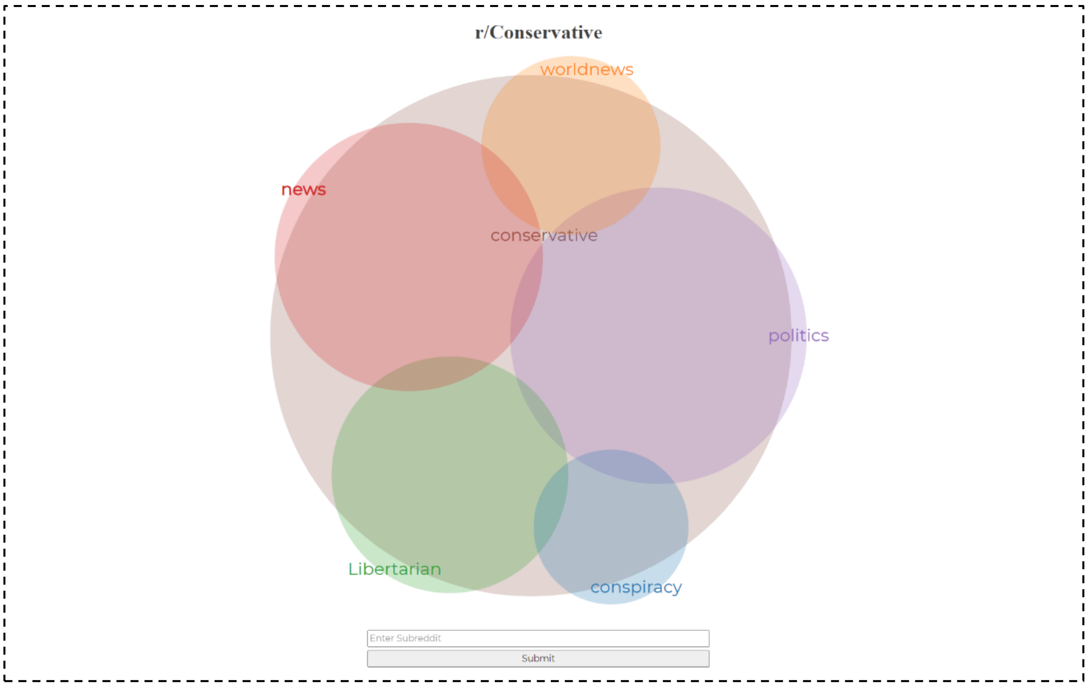
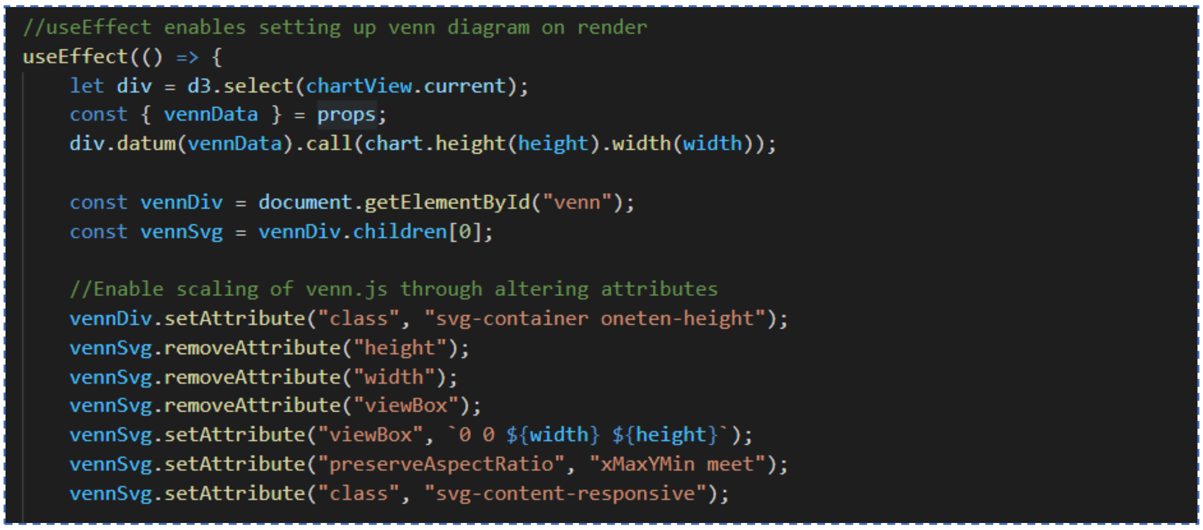


Figure 42: React Venn-Diagram implementation for subreddits.

Processing the data, updating the state, and adding additional features to VennJS was difficult due to the lack of parameters and options available through the library.
This meant it was not initially possible to scale the Venn diagram through CSS, call functions upon clicking different areas within the Venn diagram, and no way to interact with the diagram itself through React.



```
//useEffect enables setting up venn diagram on render
useEffect(() => {
    let div = d3.select(chartView.current);
    const { vennData } = props;
    div.datum(vennData).call(chart.height(height).width(width));

    const vennDiv = document.getElementById("venn");
    const vennSvg = vennDiv.children[0];

    //Enable scaling of venn.js through altering attributes
    vennDiv.setAttribute("class", "svg-container oneten-height");
    vennSvg.removeAttribute("height");
    vennSvg.removeAttribute("width");
    vennSvg.removeAttribute("viewBox");
    vennSvg.setAttribute("viewBox", `0 0 ${width} ${height}`);
    vennSvg.setAttribute("preserveAspectRatio", "xMaxYMin meet");
    vennSvg.setAttribute("class", "svg-content-responsive");
```

Figure 43: Code that enables venn.js to scale to given screen size.

Word Frequency Visualisation:

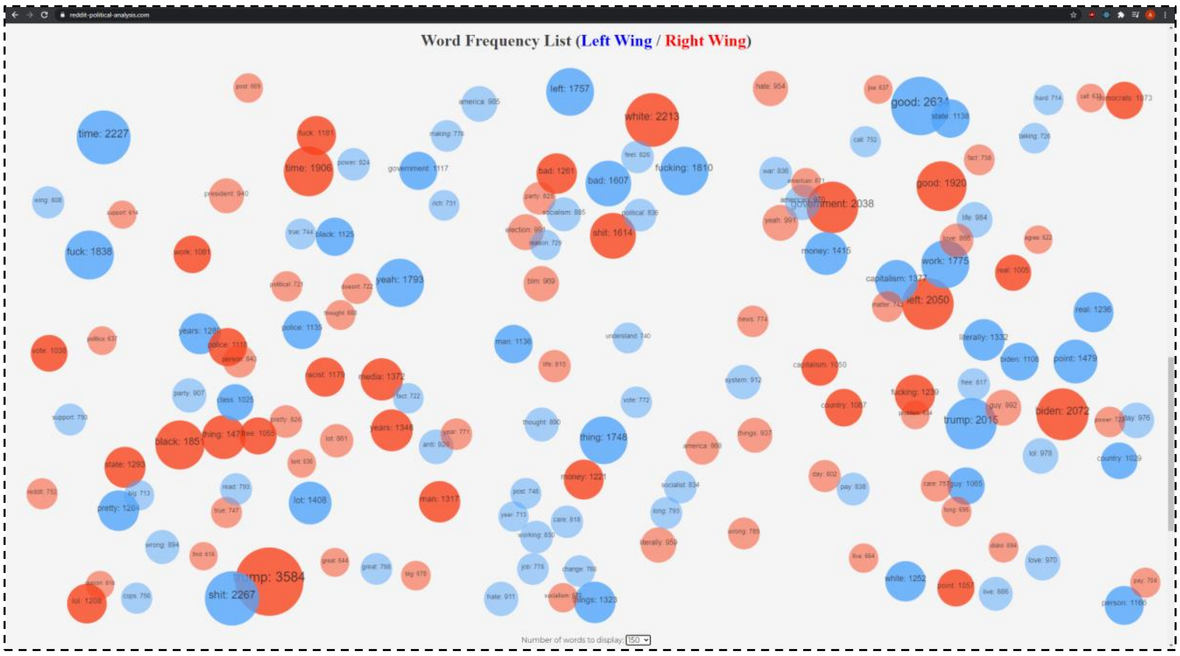
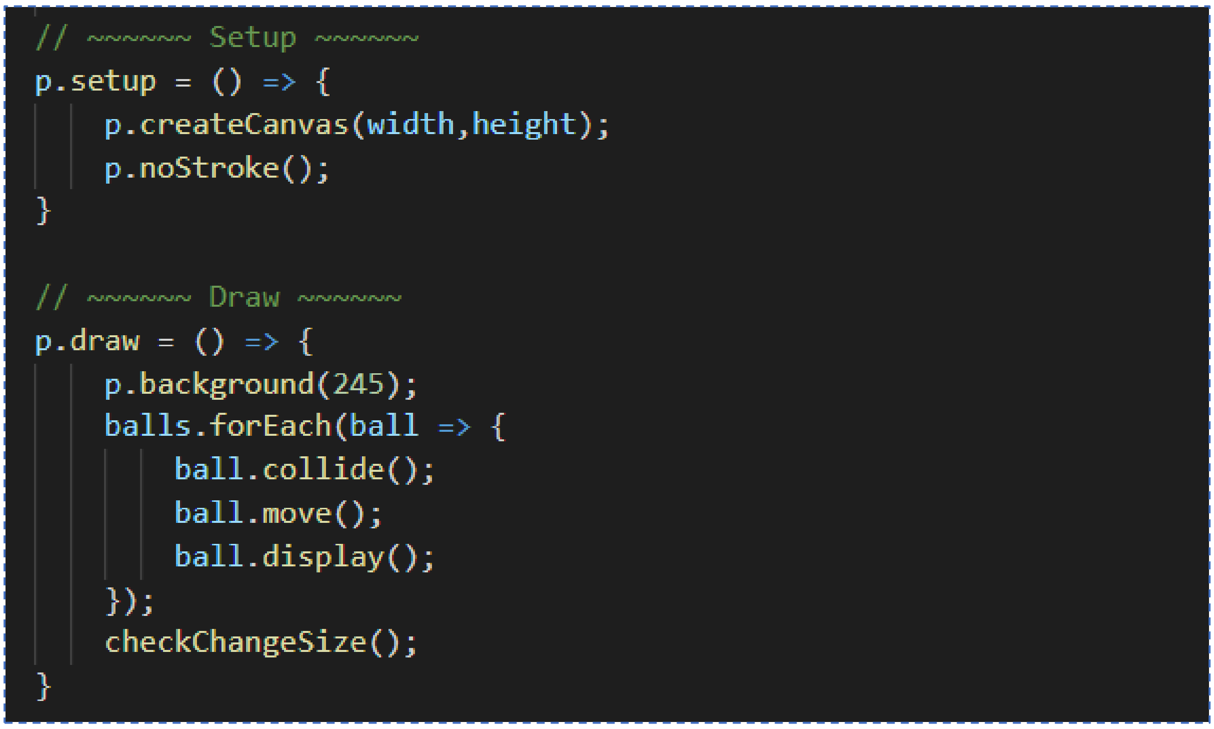


Figure 44: Word Frequency Sketch Visualiser, visible on the platform.

The word frequency sketch was built by implementing the p5js library into React, a processing library built for vanilla JavaScript, p5js initially lacks support for React, but a wrapper library called *react-p5-wrapper* enables sketches to run in React.



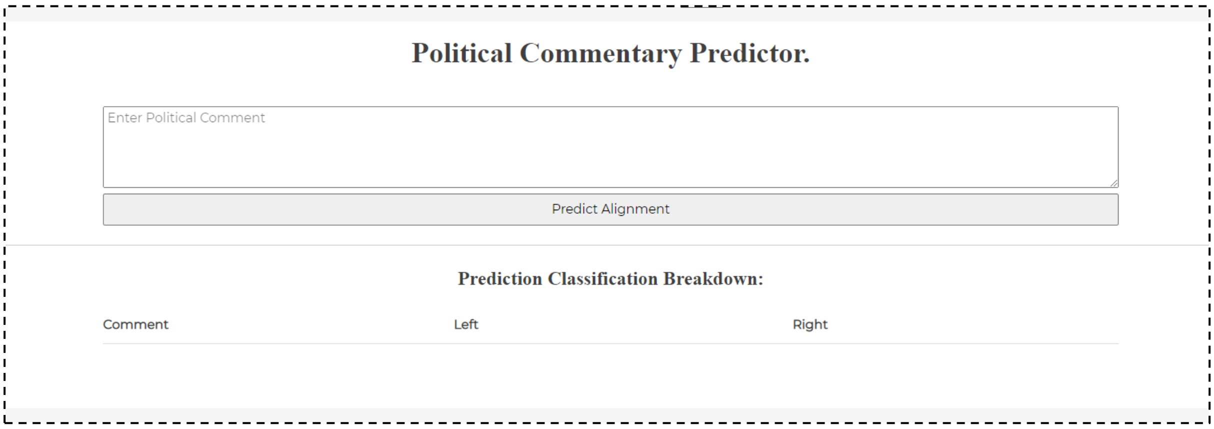
```
// ~~~~~~~ Setup ~~~~~~
p.setup = () => {
    p.createCanvas(width,height);
    p.noStroke();
}

// ~~~~~~~ Draw ~~~~~~~
p.draw = () => {
    p.background(245);
    balls.forEach(ball => {
        ball.collide();
        ball.move();
        ball.display();
    });
    checkChangeSize();
}
```

Figure 45: p5js inherited methods to draw canvas to screen.

P5js, like similar sketch libraries, have a wide variety of built-in functions for creating objects on the screen and comes with a few inherited methods that must run for the sketch to be drawn to the screen. These functions handle frame by frame logic and initialising the canvas.

Initially, the option to move the balls around and interact with them using the mouse was implemented. However, this feature struggled with mobile devices scrolling and increased computation for the sketch. This is not a central feature of the application, and therefore these features were removed.

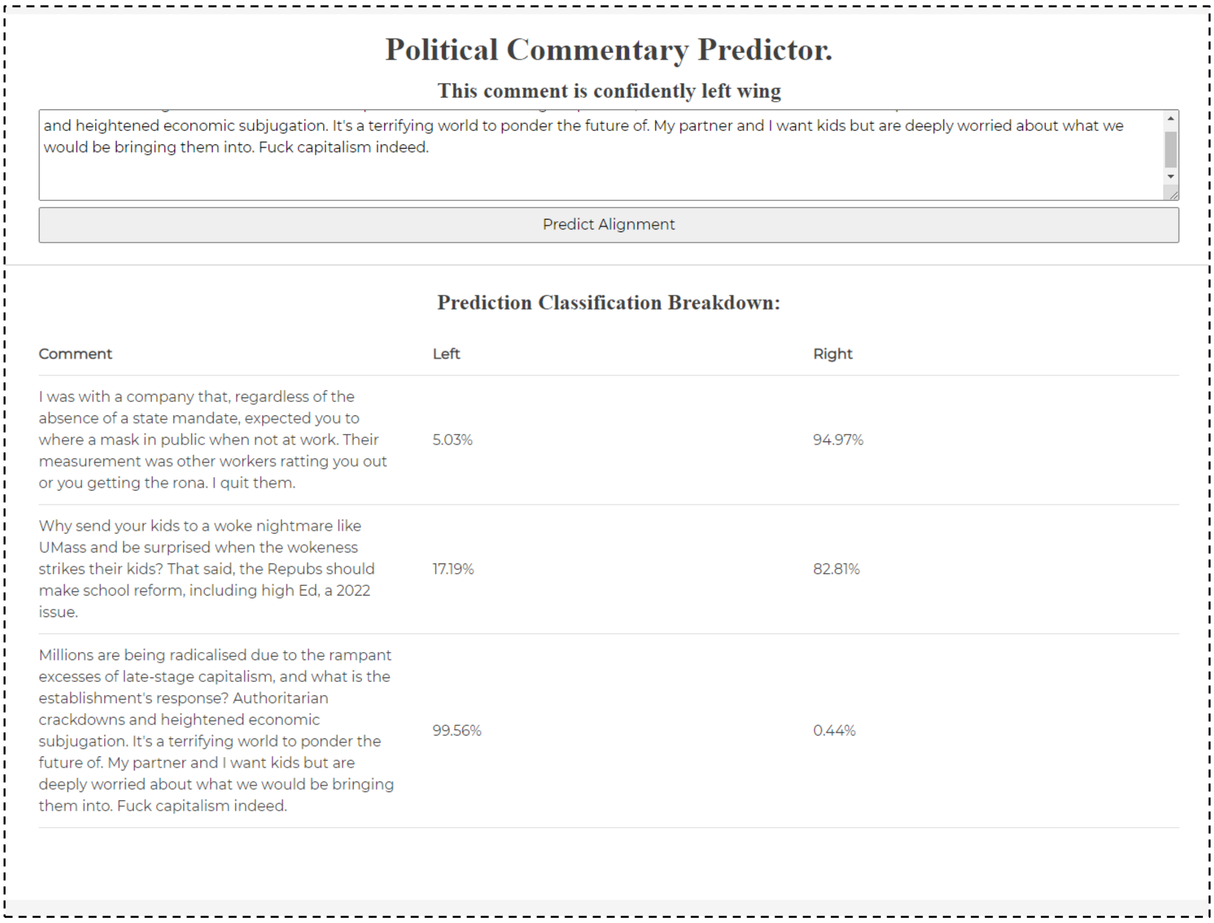
Political Commentary Predictor:


Political Commentary Predictor.
Enter Political Comment
Predict Alignment
Prediction Classification Breakdown:

Comment	Left	Right

Figure 46: Reddit Political Commentary Predictor Form taken from website.

The political commentary predictor was an essential component to implement; it required a form that sends a post request to the backend for a model prediction that updates multiple states on the page upon the request's completion.


Political Commentary Predictor.
This comment is confidently left wing

and heightened economic subjugation. It's a terrifying world to ponder the future of. My partner and I want kids but are deeply worried about what we would be bringing them into. Fuck capitalism indeed.

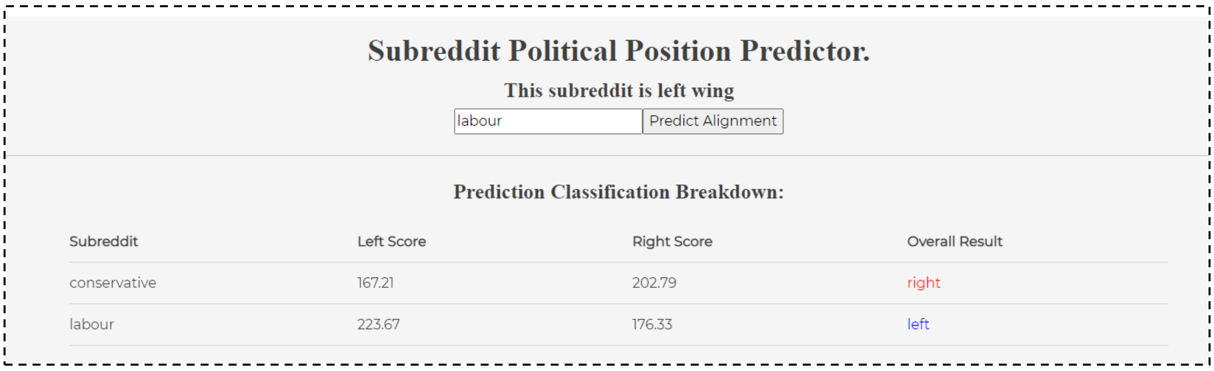
Predict Alignment
Prediction Classification Breakdown:

Comment	Left	Right
I was with a company that, regardless of the absence of a state mandate, expected you to where a mask in public when not at work. Their measurement was other workers ratting you out or you getting the rona. I quit them.	5.03%	94.97%
Why send your kids to a woke nightmare like UMass and be surprised when the wokeness strikes their kids? That said, the Repubs should make school reform, including high Ed, a 2022 issue.	17.19%	82.81%
Millions are being radicalised due to the rampant excesses of late-stage capitalism, and what is the establishment's response? Authoritarian crackdowns and heightened economic subjugation. It's a terrifying world to ponder the future of. My partner and I want kids but are deeply worried about what we would be bringing them into. Fuck capitalism indeed.	99.56%	0.44%

Figure 47: Showing state changes to the webpage upon form submission.

Subreddit Political Position Predictor:

The subreddit predictor utilises the backend functionality to scrape hundreds of comments and predict each of them. Like with the commentary predictor, an important aspect is that the accuracy of the model is shown. This will allow users to understand how the model thinks as they can change their text and see how the prediction might be affected.



Subreddit Political Position Predictor.

This subreddit is left wing

Prediction Classification Breakdown:

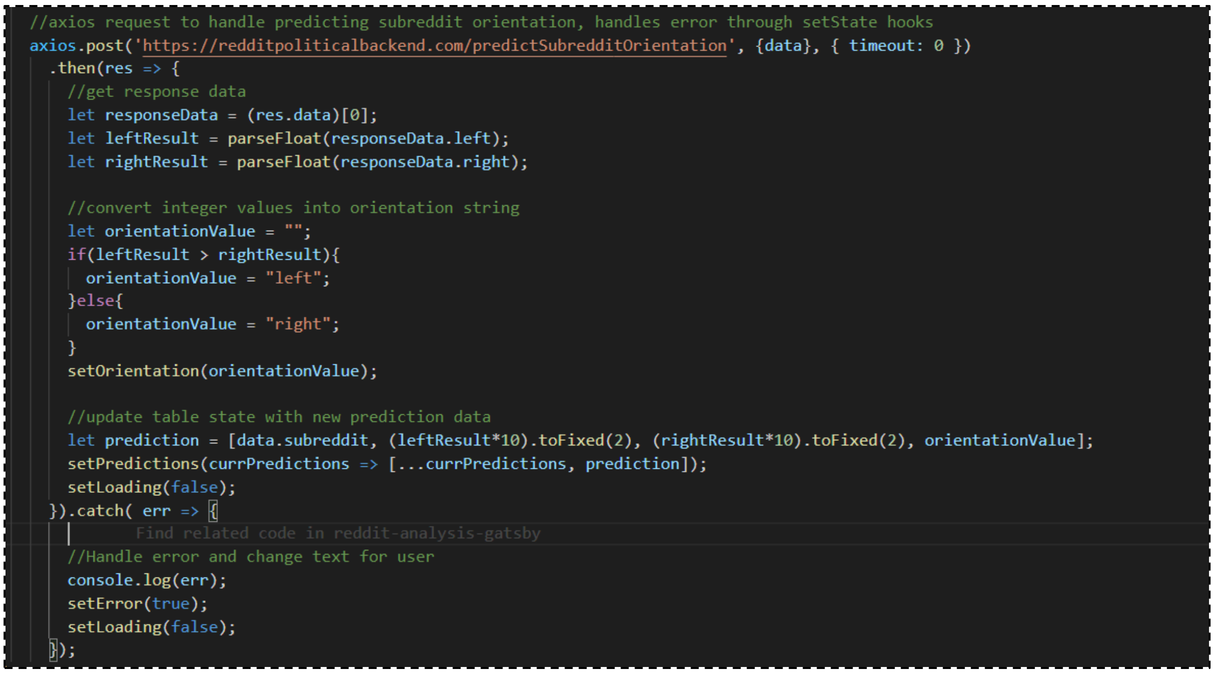
Subreddit	Left Score	Right Score	Overall Result
conservative	167.21	202.79	right
labour	223.67	176.33	left

Figure 48: Subreddit Political Predictor, screenshot taken from the website.

The subreddit predictor is based on the same component as the commentary predictor. However, it has a different interaction with the backend and additional data in the classification breakdown data. The backend does not return a single percentage but a sum of all SoftMax output. This tool shows how left/right the overall sentiment score of a subreddit is. This score can determine how extreme a subreddit is politically or if it may represent a political echo chamber within its community.

3.4.3 – Connecting the Backend.

Using the library Axios to enable asynchronous requests in React, requests to the backend are asynchronous, not to inhibit the user's ability to continue interacting with the frontend while a request is carried out. This is particularly important as for some requests the response time can exceed 10+ seconds.



```
//axios request to handle predicting subreddit orientation, handles error through setState hooks
axios.post('https://redditpoliticalbackend.com/predictSubredditOrientation', {data}, { timeout: 0 })
  .then(res => {
    //get response data
    let responseData = (res.data)[0];
    let leftResult = parseFloat(responseData.left);
    let rightResult = parseFloat(responseData.right);

    //convert integer values into orientation string
    let orientationValue = "";
    if(leftResult > rightResult){
      orientationValue = "left";
    }else{
      orientationValue = "right";
    }
    setOrientation(orientationValue);

    //update table state with new prediction data
    let prediction = [data.subreddit, (leftResult*10).toFixed(2), (rightResult*10).toFixed(2), orientationValue];
    setPredictions(currPredictions => [...currPredictions, prediction]);
    setLoading(false);
  }).catch( err => {
    |       Find related code in reddit-analysis-gatsby
    //Handle error and change text for user
    console.log(err);
    setError(true);
    setLoading(false);
  });
```

Figure 49: handling requests through Axios, including error handling.

Handling requests, in React, the DOM will not update unless the state of that element changes. Therefore updating each components state as responses or errors are returned allows the frontend to update that element for the user. A timeout of 0 is used as timeout responses from the backend are handled through *gunicorn* due to standard long request times.

4. Testing & Evaluation

4.1 Testing Components Strategies

4.1.1 – Model Testing

Throughout this project, multiple models of different sizes were trained. As a result, 11 final models were determined for testing based upon initial training performance.

Four BERT models each trained with different learning rates and batch sizes.

Two DistilBERT models, one trained through SimpleTransformers library and another through HuggingFace with TensorFlow 2.0.

One Count-Vectoriser model and finally a linear regression model trained through the SKLearn library.

The models were tested through 5-fold cross-validation, an evaluation method through which the dataset is split into five different stratified 80/20 splits of data used to train the model 5 separate times. Each time trained and tested on a different dataset fold, this is repeated until all the dataset has been used to test the model's accuracy. These results are then summed and divided by the number of folds to find a realistic accuracy of the model on new data.

The result of evaluating these models is found below:

Model Name	***Batch Size***	***Epoch Count***	***Training Accuracy***	***Testing Accuracy***	***5-Fold Validation Accuracy***
BERT	*8*	*8*	*50.2%*	*50.1%*	*50.0%*
BERT	*16*	*8*	*92.8%*	*62.0%*	*59.8%*
BERT	*32*	*8*	*83.2%*	*69.2%*	*66.3%*
BERT	*64*	*8*	*78.3%*	*60.7%*	*60.0%*
DistilBERT (simpleTransformers)	*32*	*6*	*88.0%*	*70.3%*	*68.5%*
DistilBERT (HuggingFace)	*32*	*6*	*82.4%*	*72.6%*	*72.2%*
Count-Vectoriser	*NA*	*50*	*70.2%*	*68.2%*	*63.1%*
Linear Regression	*NA*	*NA*	*62.6%*	*51.1%*	*52.4%*

Figure 50: Table showing Evaluation of multiple models.

From *Fig 47.* it shows that on new data DistilBERT trained through the HuggingFace library with batch size 32 performed best. However, the model trained for six epochs due to a learning rate scheduler with early stopping parameters; this stopped the training before validation loss began increasing.

The difference between the HuggingFace and SimpleTransformers implementation is most likely due to the additional layers implemented through TensorFlow and freezing the layers. This can help prevent overfitting the parameters, which will improve generalisation of new data.

4.1.2 – Backend Testing

Each endpoint within the backend was tested through Python's *unittest* library, which enables performing requests to the flask server and asserting the returned responses are as expected.

The following endpoints exist in the final backend Flask server:

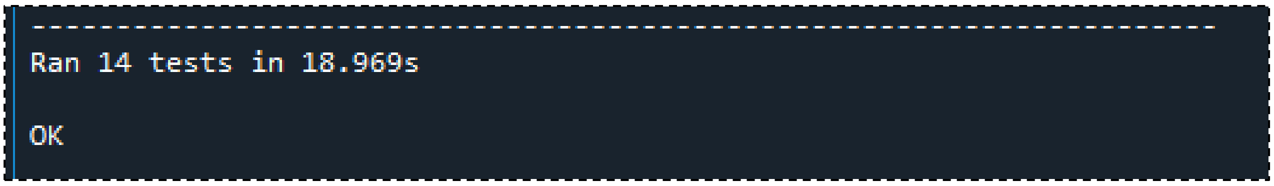
Endpoint 1: *"/"*
Description*:* The main index endpoint that is used for validating the server is running successfully.
Request Methods: [GET]

Endpoint 2: *"/pullSubreddits"*
Description: Used to extract Venn Diagram data of frequently related subreddits.
Request Methods: [POST]

Endpoint 3: *"/predictOrientation"*
Description: Utilises the Machine Learning Model to predict commentary.
Request Methods: [POST]

Endpoint 4: *“/predictSubredditOrientation”*
Description: Analyses subreddit comments and create a sum of political alignment.
Request Methods: [POST]

Running the tests in Spyder takes between 15-20 seconds to complete. If all tests complete successfully, the server can be pushed to Docker Repository to be served in the Google Cloud instance.



```
----------------------------------------------------------------------
Ran 14 tests in 18.969s

OK
```

Figure 51: Completed unit test run in Spyder for all 14 tests passing successfully.

The tests and their output are found in *Appendix B*, which shows the endpoint tested, a description of the test ran, the data passed, the expected output, and if the test passed successfully. The code

4.1.3 – Frontend Testing

The Front-End intends to implement the data and Backend in a visualised way. Since the frontend is only a means to send and handle responses, unit testing is not appropriate as there are no functions that need verification.

This means the testing for the Front-End is a manual evaluation of input fields to ensure they handle data correctly and evaluating that responses from the server are being handled and outputted correctly as well

The table in *Appendix C* describes the tests performed, their output, and screenshots of the results.

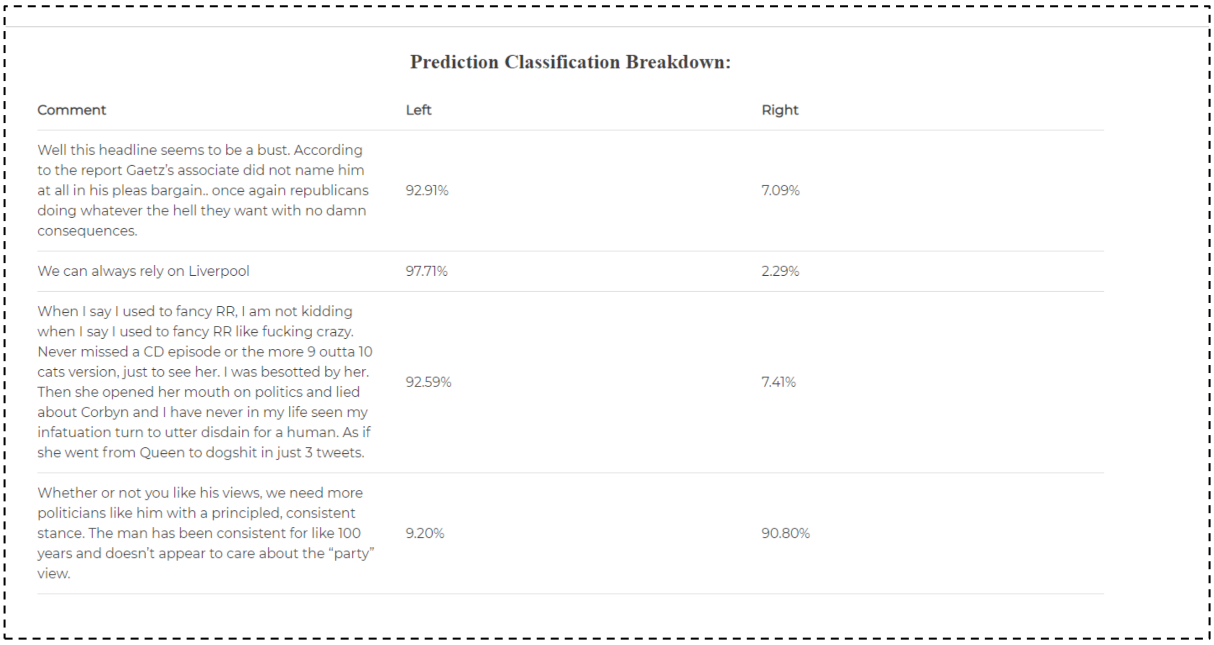
4.2 Evaluation

4.2.1 – System Aims & Objectives Evaluation

Evaluating the system on the aims and objectives initially described in the *Project Proposal* helps understand if the project has reached the expectation initially set out to be achieved.

1. *Develop a model that can categorise left-wing and right-wing commentary sentiment*

This aim has been achieved. The DistilBERT model hosted on the Backend can correctly categorise left- and right-wing commentary sentiment with 72%+ accuracy on new data. Evaluating the model against the aim, it appears to classify left and right sentiment with a high enough degree of accuracy that it can be considered a successful classifier of political commentary.



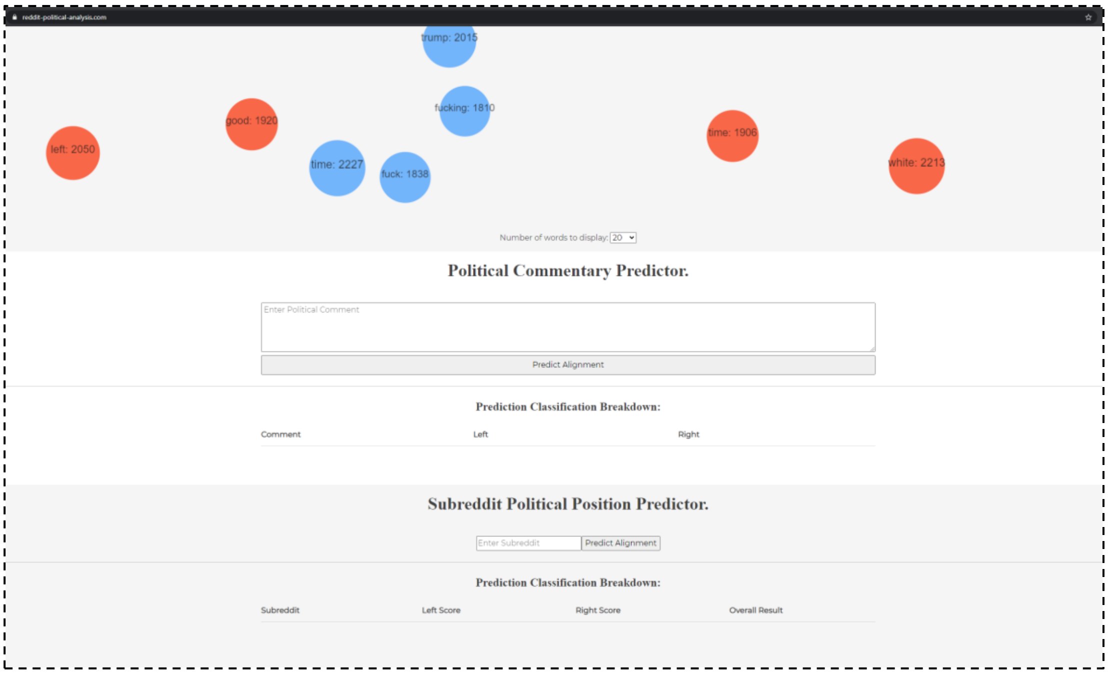
Prediction Classification Breakdown:

Comment	Left	Right
Well this headline seems to be a bust. According to the report Gaetz's associate did not name him at all in his pleas bargain.. once again republicans doing whatever the hell they want with no damn consequences.	92.91%	7.09%
We can always rely on Liverpool	97.71%	2.29%
When I say I used to fancy RR, I am not kidding when I say I used to fancy RR like fucking crazy. Never missed a CD episode or the more 9 outta 10 cats version, just to see her. I was besotted by her. Then she opened her mouth on politics and lied about Corbyn and I have never in my life seen my infatuation turn to utter disdain for a human. As if she went from Queen to dogshit in just 3 tweets.	92.59%	7.41%
Whether or not you like his views, we need more politicians like him with a principled, consistent stance. The man has been consistent for like 100 years and doesn't appear to care about the "party" view.	9.20%	90.80%

Figure 52: Example Correct Predictions of Comments from r/Conservative and r/Labour.

2. *Build a platform for users to interact with the model*

The platform that users can use to interact with the model is live at [*https://reddit-political-analysis.com/*](https://reddit-political-analysis.com/). It enables the ability to interact with the model and visualise the dataset and perform subreddit analysis utilising the model. Exceeding the expectations of this aim and enables greater interaction with the model than initially expected.



Number of words to display: 20

Political Commentary Predictor.

Enter Political Comment

Predict Alignment

Prediction Classification Breakdown:

Comment	Left	Right

Subreddit Political Position Predictor.

Enter Subreddit

Predict Alignment

Prediction Classification Breakdown:

Subreddit	Left Score	Right Score	Overall Result

Figure 53: Developed Platform with commentary predictor that interacts with model.

3. Develop a react frontend that allows users to connect with live data

The Venn Diagram tool accesses live Reddit data, and this aim has been reached as the tool allows users to experiment with live Reddit data. In addition, the subreddit prediction tool also uses live data in that it scrapes currently popular comments from input subreddits live.

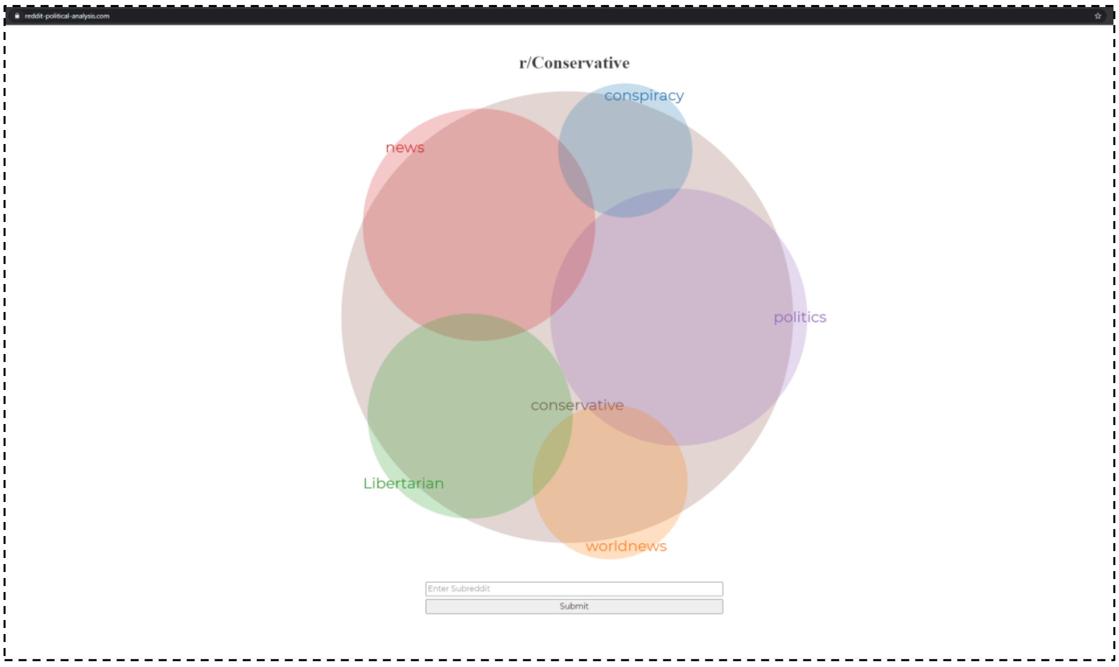


Figure 54: Venn Diagram Visualisation Too interacting with Live Reddit Data.

4. *Define and Categorise left-wing and right-wing subreddits into a list.*

This objective was completed using the created Venn Diagram tool that generates left and right-wing subreddits into a list; due to Ethical considerations, this list has not been published. A sample version of this list is available to *see fig. 4*

5. *Mine the data using PRAW into a CSV file*

Extracting the data into PRAW was completed successfully and discussed in *section 3.1.2,* showing the implementation of this objective. The CSV files were extracted and stored by subreddit name before combining them into a flattened CSV file, *see fig. 24*

6. *Train a model to categorise the left-wing and right-wing sentiment*

Completing this objective relates to *aim 1*. The model was not trained from scratch, instead fine-tuned, which trains the lower layers of the pre-trained model. This objective was completed successfully. In the proposal, the discussion of the level of accuracy of the classification on new data to consider the model successful was evaluated between 70-80%. The final model with 5-fold cross-validation achieved 72% accuracy within the pre-defined threshold of a successful model.

7. *Host the model as a RESTful API on a cloud instance*

The model was successfully hosted in Google Cloud as an API. However, to define if the API is RESTful, it must adhere to the architectural bounds of a REST API [23].

The interface must be separate from the backend and storage. For this API, the frontend interface is hosted entirely separately through Netlify.

There must be no client context stored on the server between requests. This server does not store or cache user information within it.

The server must explicitly state whether it can be cached or not, through GET requests Cacheability is standard. However, the server utilises POST requests for submitting and receiving data; therefore, data is explicitly not cached by the server.

Finally, *The API must work whether it is communicating with a server directly or through an intermediary such as a load balancer*. The server itself is hosted behind a load balancer, so all communication occurs by default through an intermediary.

This shows that the API meets all requirements of the architectural guidelines to be considered RESTful.

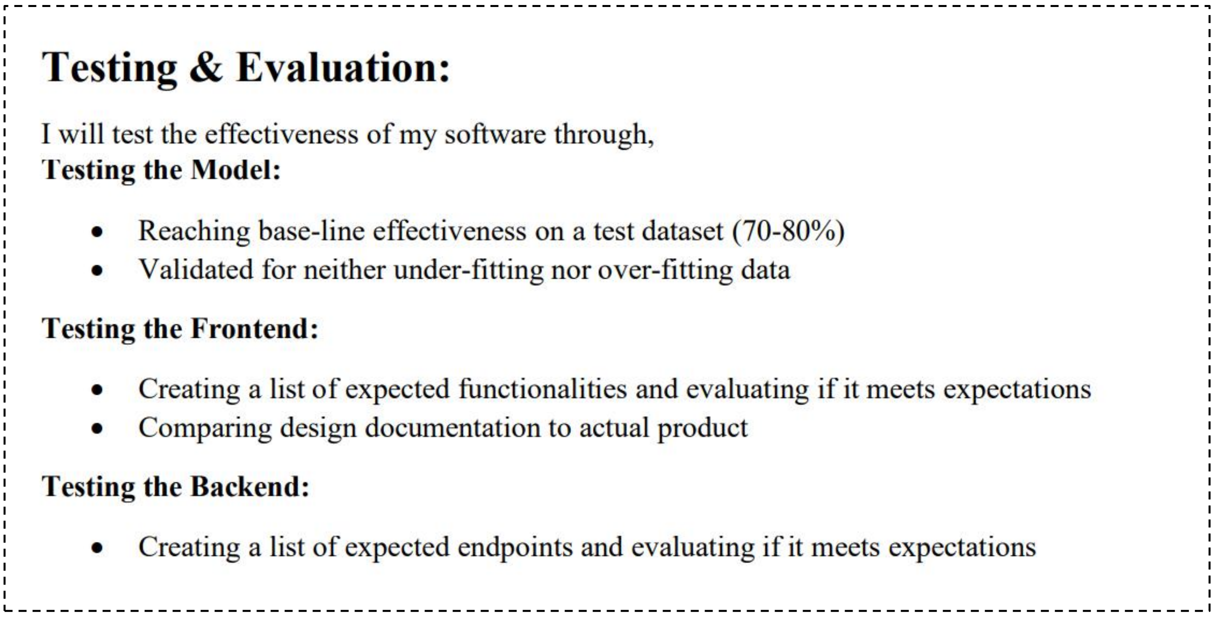
8. Create a microservice that enables interacting with live Reddit Data

Initially, this was intended to be a separate service to the hosted model API. However, when it came to implementation, since the live Reddit data interaction made frequent calls to the model's predictive endpoint, it made more sense to package this functionality into a single server. Thus, although this service was not achieved to the expectation of the objective, the functionality was still implemented successfully.

9. Develop an interactive React Front-End that can make calls to the hosted services.

An interactive frontend was developed and hosted using React with the Gatsby framework. This enabled interacting with live Reddit data and using the model to predict sentiment through calls to the hosted services. This objective was achieved and hosted at *http://reddit-political-analysis.com/*

The project proposal stated the following quantitative measures would be considered to test the effectiveness of the software in evaluation:


Testing & Evaluation:

I will test the effectiveness of my software through,

Testing the Model:

- Reaching base-line effectiveness on a test dataset (70-80%)
- Validated for neither under-fitting nor over-fitting data

Testing the Frontend:

- Creating a list of expected functionalities and evaluating if it meets expectations
- Comparing design documentation to actual product

Testing the Backend:

- Creating a list of expected endpoints and evaluating if it meets expectations

Figure 55: Testing and Evaluation section from The Project Proposal

The model has achieved baseline effectiveness of 72% through 5-fold cross-validation, with 82% average training accuracy across each fold. This meets the models' criteria effectiveness, ensuring a lack of over-fitting the data as the model is generalising successfully without reducing the validation accuracy.

The frontend's expected functionality is specified through the aims and objectives of the component, and this can be found to be successful through the earlier evaluation of these metrics.

Comparing the design documentation alongside the implemented solution, *see section 2.4.2*, the frontend implementation shares a striking resemblance to the proposed design solution.

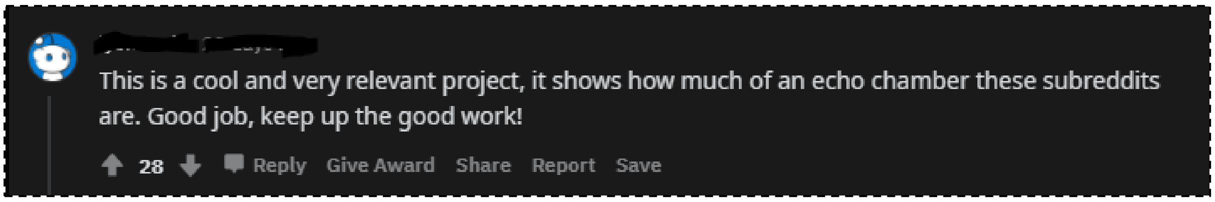
The expected endpoints for the initial model were to enable a single commentary prediction endpoint and one to enable extraction of live Reddit data. The final backend implements both endpoints and goes beyond to have additional functionality through scraping and predicting data with the */predictSubredditOrientation* endpoint. Again, this exceeds the initial evaluation criteria.

4.2.3 – End User Evaluation

Evaluating the User Experience is difficult for this project as it represents a niche platform of potential use. The target audience is Reddit users, helping them better understand the communities they are in engaging in and helping identify echo chambers.

The best approach to target these users and get their opinion on the usefulness and functionality of the platform was to post the project on Reddit. Posting the project resulted in it reaching the top of multiple different subreddits for the day.
The posts aggregated 789 upvotes and 108 comments offering feedback on their use of the project.

Most of the comments demonstrated a particular short statement that could break the model. However, the rest offered valuable and helpful feedback on parts of the project that could see feature improvements.



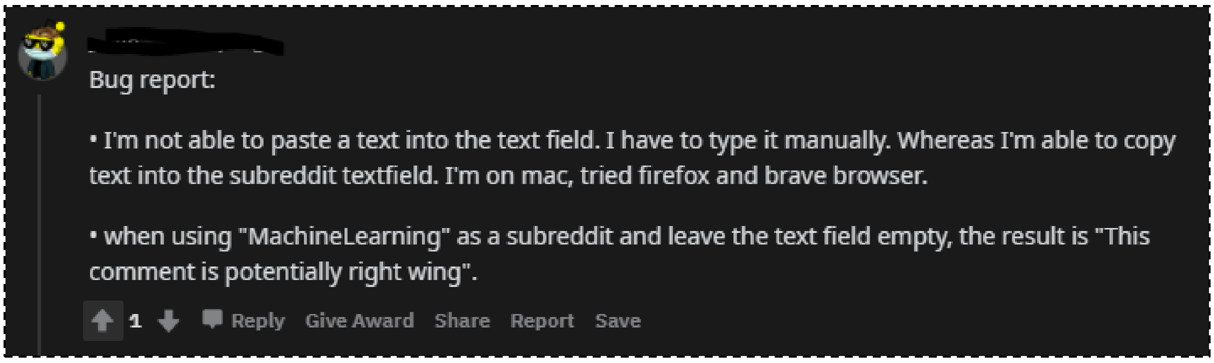
[redacted]

This is a cool and very relevant project, it shows how much of an echo chamber these subreddits are. Good job, keep up the good work!

Figure 56: User recognising the use for identifying echo chambers on the platform.

The project's intentions were not published on Reddit, so identifying that Reddit users understood the potential use of this project awards the development. Users were able to understand how to use the project without being told explicitly.

Some Feedback was constructive and enabled changes to be made. Several comments were listed and saved that offered further exploration into changes that could be made to the project.


[redacted]

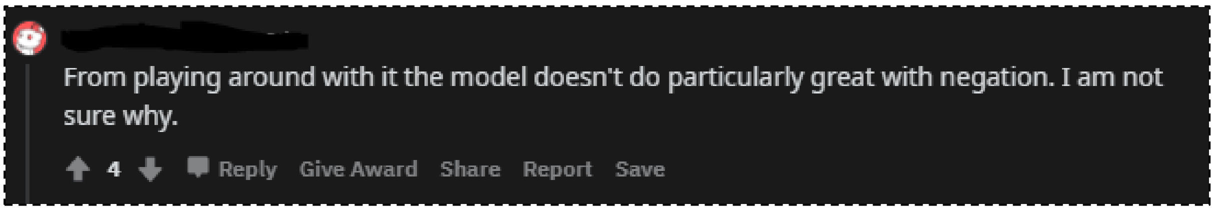
Bug report:

• I'm not able to paste a text into the text field. I have to type it manually. Whereas I'm able to copy text into the subreddit textfield. I'm on mac, tried firefox and brave browser.

• when using "MachineLearning" as a subreddit and leave the text field empty, the result is "This comment is potentially right wing".

Figure 57: User reports bug in Predictor components.

This user found a bug in the text field validation of both the commentary and subreddit predictor. The necessary changes were made to amend these issues and push changes to the website.

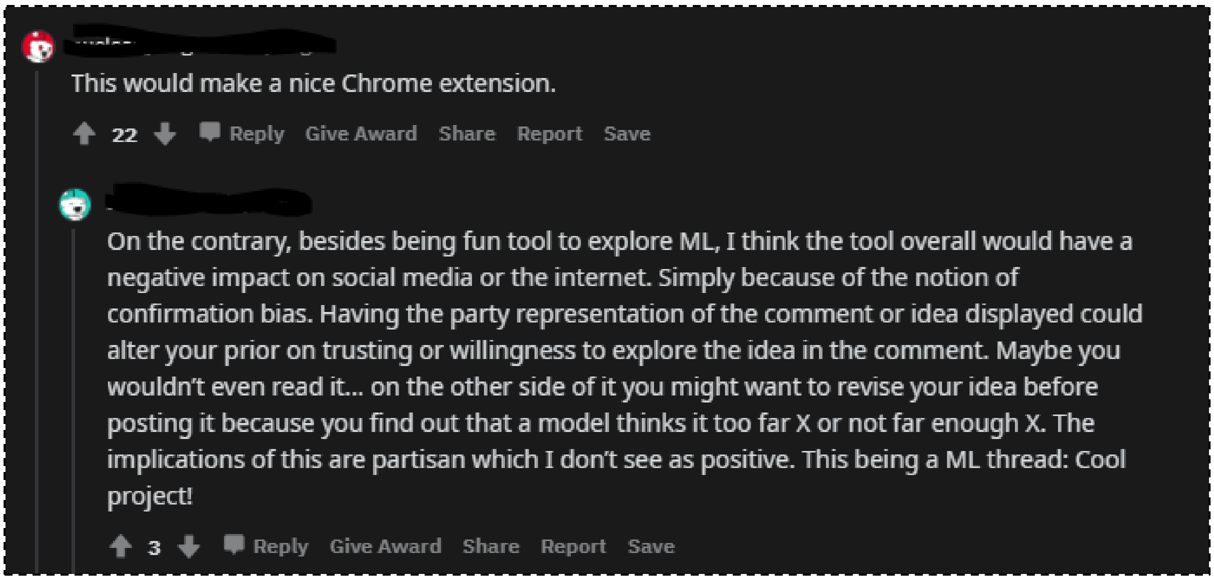

[redacted]

From playing around with it the model doesn't do particularly great with negation. I am not sure why.

Figure 58: User recognises the difficulty in negation with Machine Learning Model

This user discovered the identification of potential issues with negation in the DistilBERT model. A look into the effects of negation in pre-trained language models found a recent paper that discussed PLM's inability to distinguish between negated and non-negated text, what they have coined as *Mispriming*. [24]

This further research has made for more significant insights into the future improvement of the platform and the model as it evolves.


[redacted]

This would make a nice Chrome extension.

22 Reply Give Award Share Report Save

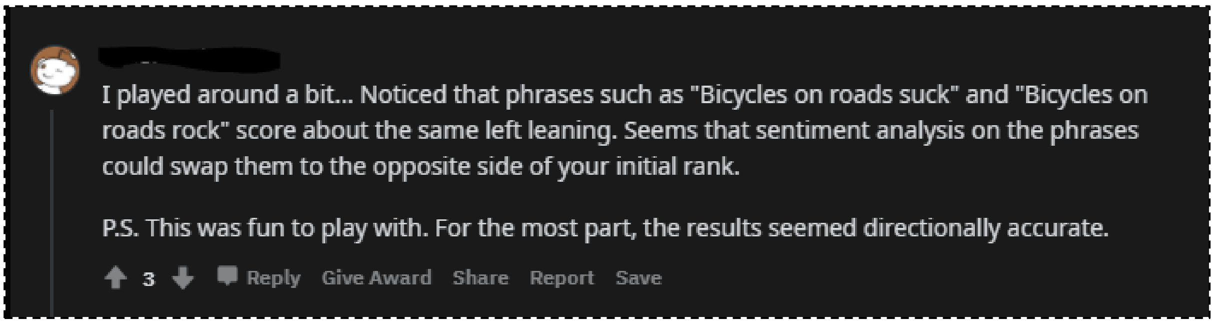
[redacted]

On the contrary, besides being fun tool to explore ML, I think the tool overall would have a negative impact on social media or the internet. Simply because of the notion of confirmation bias. Having the party representation of the comment or idea displayed could alter your prior on trusting or willingness to explore the idea in the comment. Maybe you wouldn't even read it... on the other side of it you might want to revise your idea before posting it because you find out that a model thinks it too far X or not far enough X. The implications of this are partisan which I don't see as positive. This being a ML thread: Cool project!

3 Reply Give Award Share Report Save

Figure 59: Discussion on the future model implementation on external sources

The discussion between two users on the future of this project and the possible effects it may have on user's behaviour and activity on social media platforms in the future. This is an interesting discussion and one that helped enforce ethical considerations made with this project. The ability to not classify a specific user or post was made to ensure the project could not automate the removal of any specific user or post based upon confirmation bias.


[redacted]

I played around a bit... Noticed that phrases such as "Bicycles on roads suck" and "Bicycles on roads rock" score about the same left leaning. Seems that sentiment analysis on the phrases could swap them to the opposite side of your initial rank.

P.S. This was fun to play with. For the most part, the results seemed directionally accurate.

3 Reply Give Award Share Report Save

Figure 60: User recognition of specific phrases that impact sentiment.

Feedback from one user noted how different words affect the prediction of the model. This is the intention of the whole platform, to allow users to interact with the model and experiment with how different words affect the political prediction. This shows that the project is being used as it was intended when initially designing the project.

4.3 Future Improvements

Based upon the evaluation of the project and the feedback received from Reddit users, there are a few notable improvements I would make to this project to increase its effectiveness and its use as a project for the political analysis of Reddit.

1. *Evolution of the Machine Learning Model*

The model performs very effectively given the time that was available for training. However, the future of this model would be in developing one that is catered specifically to the online political commentary domain. This means instead of fine-tuning a pre-trained model which has been trained on a corpus such as Wikipedia, the model would be trained from scratch on a large, mined corpus of online political commentary.

One of the main limitations of this model was the language barrier and the fact that the pre-trained model had not seen a lot of political internet vocabulary. Attempts were made to alter the vocabulary file of the pre-trained model. However, it would still miss the vital context between the new vocabulary and the previous Wikipedia vocabulary, which could be much more formal than social media's language. Furthermore, training a model from scratch is an expensive, computationally heavy task. Therefore, it was not possible for this project, although something that may be more feasible in the future as GPU prices decrease and hardware capability increases.

2. *Develop more tools to analyse the live Reddit data.*

Developing more tools that allow live interaction with Reddit data is undoubtedly the next stage of this project. For example, the subsequent tool developed would be a live word map of different subreddits. The user can select a subreddit and generate their own word frequency list visualiser to see the most popular words from that subreddit. This would enable identifying how language changes for each subreddit and how different communities interact.

3. Integrate system to monitor subreddit's orientation

During backend development, research was done into running Cron jobs; these enable functions to run on repeat in set intervals. The idea of analysing a range of subreddits and comparing their sentiment score over time to identify subreddits that become politically extreme was an exciting improvement on the current features. Implementing this tool could allow subreddit moderators to determine if their subreddits are becoming echo chambers of political opinion over time. This would create a long-term usage of the platform for a single user, which will keep them engaged with the project.

5. Conclusion

Overall, this project has been successful. It has fulfilled all the project's initial aims; it has a model that can categorise left-wing and right-wing commentary sentiment with over 72% accuracy on new data. The project has a platform that enables users to interact with the model. Users on Reddit interacted and left great feedback on the different use cases for this project and future implementations they wish to see. The React frontend allows users to connect with live data, as evidenced by *section 3.4*. Through multiple Reddit API's users can interact with interesting live data through visualisation tools built into the frontend and backend of this project.

Every aim and objective initially created was completed for this project, except for hosting a separate microservice for interacting with live Reddit data. This service was integrated into the machine learning model hosting service, *as evidenced in section 3.3.* This is one of the few changes made during the development of this project that intended to deliver a complete platform for users within the allotted timeframe.

In hindsight, if one aspect of this system could be done differently, it would be augmenting the dataset to generate more quality data from the limited number of concise political subreddits. Extracting the data multiple times due to poor-quality data*, as discussed in section 3.2.2, could have been missed by* better research of chosen subreddits. This would have saved a large portion of time and allowed greater focus on the model and developing the user experience.

This project has overcome huge hurdles in terms of the deadline to completion and the number of iterations it took to extract data from Reddit and train the machine learning models. It was a great learning experience to help further understand the field of Natural Language Processing and the process of building a modern full-stack application for production, developing a cutting-edge machine learning model, hosting a scalable, future-proof backend microservice and a high-performance frontend. These have all increased the knowledge of developing production-quality systems using modern architecture.

6. BCS Project Criteria & Self-Reflection

This project has met the six outcomes expected by the Chartered Institute for IT. The following section outlines the six criteria and gives a reasoned response as to why each has been met or reference a relevant section that shows meeting those criteria.

1. *An ability to apply practical and analytical skills gained during the degree programme.*

   The application of practical skills can be seen in *section 2.3* in using algorithms such as K-Nearest Neighbour to implement sorting arrays by frequency. I learned these algorithms through *COMP202 – Complexity of Algorithms.*

   The ability to develop a model and evaluate the model using evaluation techniques such As k-fold cross-validation was gained through *COMP111 - Introduction to Artificial Intelligence* and *COMP219 – Advanced Artificial Intelligence*. These modules also taught Python and the libraries *Pandas* and *NumPy,* which are used to analyse the machine learning models to compare effectiveness. The use of these techniques can be seen in *Fig. 50,* which shows the analytical evaluation of the different machine learning models.

2. *Innovation and/or creativity.*

   The innovation and creative component of this project is the naïve approach to visualising live Reddit data. Through my research of similar work in *section 1.2.2,* there is no such tool to visualise subreddits in this way. I then improved upon this and created the ability to click on each Venn bubble and dive deeper to create this experience of going down a rabbit hole of subreddits. This shows the creativity and innovative thought put into each component of the project.

   Although projects do exist that enable a user to predict their political orientation, no such platform exists that enables political analysis in this way or can be used as a tool to detect possible echo chambers in communities. This is where other such work falls short, and my project innovates on previous ideas. This tool is described in *section 3.4.2, Fig. 48.*

3. *Synthesis of information, ideas, and practices to provide a quality solution together with an evaluation of that solution.*

   This project provided a quality solution through rigorous testing and evaluation procedures during the development before the platform went into production.

   Section 4.1.1 shows the time-consuming but quality practice of performing k-fold cross-validation to ensure the model met the high standards of this project, as defined in *Fig. 55, w*hich shows the requirements needed to be met before the project would be considered successful.

   Appendices B and C show the detailed testing on each component to ensure the end-user experience was reliable and secure. *Section 4.2.3* highlights the trust of quality in the product, serving it to hundreds of target users to gain critical feedback in the form of over 100 comments. Each considered in the evaluation of the final product, which is concluded in *Section 5.* The conclusion discusses an evaluation of the final product by measuring its performance against the aims and objectives.

4. *The project meets a real need in a wider context.*

   Section 1.1.2 discusses the project's motivations; I discuss the risks of rising reliance on social media for a single source of news. For example, Reddit has the largest percentage of users who rely on the platform for its news and the potential for dangerous biases to form in the content they consume called echo chambers.

   This project helps meet the real need of allowing users to detect if they are taking part in a political echo chamber or if a community they are involved in shares a significant political bias. *Fig. 56* displays one of the many target user feedback surrounding the real need for this project.

5. *Ability to self-manage a significant piece of work.*

This work was self-managed, Appendix D shows the initial Gantt chart for planning out this significant piece of work. This work is significant as it represents many different aspects of my degree.

Extracting data and training the model represents multiple modules such as algorithmic design, data mining, artificial intelligence, and advanced artificial intelligence. However, this only represents one-third of this project's undertaking.

Developing the backend required knowledge in server development, cloud architecture and container technology such as Docker. I learned cloud technology architecture through AWS during an internship to further my degree prospects. The theoretical knowledge from the degree allowed me to quickly transfer that knowledge to Google Cloud to set up and host autoscaling instances that are served behind a TLS terminating load balancer.

The frontend took information learned from a scripting languages module; it used React integrated with Gatsby for static site generation. All these components represent part of a sizeable significant piece of work for my final year project, to be able to manage the flow of development that each process was not impeded by the previous. In addition, it required self-management through utilising a Kanban board to keep track of each stage of development and track deadlines relating to the Gantt chart in *Appendix D.*

I believe my ability to manage this work was only capable due to a solid foundation built upon by the experience I had before it. Therefore, I did not take many risks selecting this project as each component built for the platform I had developed in or seen before.

One of my weak areas on this project was the harsh critique of my work. At each stage of training the model, I would evaluate it and should it not reach the pre-defined requirements defined in *fig. 55* I would often re-evaluate the entire model and data before continuing, this took a large portion of the time in creating this piece of work.

This project was developed through self-management, although I would implement stricter deadlines on each component in the future. This would help ensure other components do not get less time to be worked on.

6. *Critical self-evaluation of the process.*

I am proud of the platform I have developed. It shows a broad range of knowledge in Computer Science. However, the aspect of this project that I am particularly proud of, my machine learning model that went through over ten different iterations over months, getting better and improving. Each iteration taught me a new aspect of Natural Language Processing and artificial intelligence that I did not know before.

The backend application and the pitfalls of hosting the backend for a production server were nothing I expected. I initially struggled with enabling the server to work. However, through perseverance, I managed to set up an architecture that is future-proof and can handle increasing demand which is a great stepping stone in learning more about networking and cloud architecture.

The frontend visualisation tools took creativity and thought to develop and not just build a simplistic multi-page design but create something that is functional but also a good user experience. However, I wish I had managed my time better on building and training the model that I could have developed additional tools on the Frontend. I had many ideas that I could implement within the given time, although referenced in *Section 4.3 Future Improvements.*

One aspect of creating a project such as this that I would change in the future is not taking on such a large undertaking when initially coming up with this project. I had a good foundation of knowledge of each component, but creating a seamless interface that enables each of these components to communicate was a large task that was not included in my Gantt Chart. In the future must be considered before taking on a task like this again.

7. References

[1] J. Gottfried and E. Shearer. (2016). News Use Across Social Media Platforms 2016. Available at: https://www.journalism.org/wp-content/uploads/sites/8/2016/05/PJ_2016.05.26_social-media-and-news_FINAL-1.pdf

[2] M. Cinelli, G. Morales, A. Galeazzi, W. Quattrociocchi, M. Starnini. (2021). The echo chamber effect on social media. *Proceedings of the National Academy of Sciences*, Available at: https://www.pnas.org/content/118/9/e2023301118

[3] Gottfried, J. and Shearer, E. (2016). *Reddit, Facebook and Twitter users most likely to get news on each site*. Available at: https://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/pj_2016-05-26_social-media-and-news_0-02

[4] M. Alstyne, E. Brynjolfsson. (1997). *Electronic Communities: Global Village or Cyberbalkans?* Available at: http://web.mit.edu/marshall/www/papers/CyberBalkans.pdf

[5] *r/The_Donald*. (2019). *Subreddit Archival* [online] Available at: https://web.archive.org/web/20190618023144/https://www.reddit.com/r/The_Donald

[6] C. Lima. (2020). *Twitch, Reddit crack down on Trump-linked content as industry faces reckoning, June 2020*. Available at: https://www.politico.com/news/2020/06/29/reddit-bans-pro-trump-forum-in-crackdown-on-hate-speech-344698

[7] N. Kligler-Vilenchik, C. Baden, M. Yarchi (2020). *Interpretative Polarization across Platforms: How Political Disagreement Develops Over Time on Facebook, Twitter, and WhatsApp.* Available at: https://journals.sagepub.com/doi/pdf/10.1177/2056305120944393

[8] u/tigeer. (2020). *Reddit stance classifier.* Available at: www.reddit-lean.com

[9] T. Brown. (2020). *Language Models are Few-Shot Learners*. Available at: https://arxiv.org/pdf/2005.14165.pdf

[10] M. Aßenmacher, C. Heumann. (2020). *On the comparability of pre-trained language models*. Available at: http://ceur-ws.org/Vol-2624/paper2.pdf

[11] J. Devlin, M. Chang, K. Lee, K. Toutanova. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Available at: https://arxiv.org/abs/1810.04805

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov. (2019). *RoBERTa: A Robustly Optimised BERT Pretraining Approach*. Available at: https://arxiv.org/pdf/1907.11692.pdf

[13] V. Sanh, L. Debut, J. Chaumond, T. Wolf. (2020). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. Available at: https://arxiv.org/pdf/1910.01108.pdf

[14] A. Soliman, J. Hafer, F. Lemmerich. (2019). *A Characterisation of Political Communities on Reddit*. Available at: https://www.researchgate.net/publication/335878887_A_Characterization_of_Political_Communities_on_Reddit

[15] M. Iqbal. (2019). *Determining Political Inclination in Tweets Using Transfer Learning*. Available at: https://digitalcommons.georgiasouthern.edu/cgi/viewcontent.cgi?article=3074&context=etd

[16] A. Acosta, S. Ciurea-ilcus, M. Wegrzynski. (2016). Predicting users' political support from their Reddit comment history. Available at: http://cs229.stanford.edu/proj2016/report/AcostaIlcusWegrzynski-Predicting%20user's%20political%20support%20from%20their%20Reddit%20comment%20history-report.pdf

[17] Jeff Reback et al. (2021). *pandas-dev/pandas: Pandas 1.2.4*. Zenodo. Available at: https://zenodo.org/record/4681666

[18] Spyder IDE. (2021). *Welcome to Spyder— Spyder 4 documentation*. Available at: https://docs.spyder-ide.org/current/index.html

[19] B. Boe. (2021). *PRAW: The Python Reddit API Wrapper — documentation*. Available at: https://praw.readthedocs.io

[20] Marx, D. (2021). *Python Pushshift.io API wrapper — GitHub*. Available at: https://github.com/dmarx/psaw

[21] S. Datta, E. Adar. (2019). *Extracting Inter-Community Conflicts in Reddit.* Available at: https://ojs.aaai.org/index.php/ICWSM/article/view/3217/3085

[22] Reddit. (2015) *Reddit API Guidelines — GitHub* Available at: https://github.com/reddit-archive/reddit/wiki/API

[23] Stoplight (2021). *Types of API | Types Of API Calls & REST API Protocol*. Available at: https://stoplight.io/api-types

[24] N. Kassner, H. Schutze. (2020). *Negated and Misprimed Probes for Pretrained Language Models: Birds Can Talk, But Cannot Fly*. Available at: https://arxiv.org/pdf/1911.03343.pdf/

Appendices

Appendix A:

Python Code showing the Data Extractor used for extracting comments from Reddit.

```
"""
Class: DataExtractor
Usage: Used to pull and pre-process data from Reddit for dataset
Input (chosen_subr): The subreddit to pull data from
"""
class DataExtractor:
    def __init__(self, chosen_subreddits):
        self.reddit = praw.Reddit() #Initialise PRAW API Object
        self.subreddit_list = chosen_subreddits

    #Pulls top data from past year from reddit given set comment guidelines reached
    def extract_comment(self, subreddit):
        comments_arr = []
        subreddit_data = self.reddit.subreddit(subreddit)
        hot_data = subreddit_data.top("year", limit = None)
        count = 0
        for submission in hot_data:
            submission.comments.replace_more(limit=None)
            for comment in submission.comments.list():
                if(comment.score > 10 and len(comment.body) > 15 and len(comment.body) < 500):
                    comment_clean = self.text_preproc(comment.body)
                    comments_arr.append(comment_clean)
            count += 1
            print(count, len(comments_arr))
        return comments_arr

    #Pre-process data stream coming in from Reddit, removes a lot of reddit specific syntax
    def text_preproc(self, x):
        x = x.lower()
        x = x.encode('ascii', 'ignore').decode()
        x = re.sub(r'https*\S+', ' ', x)
        x = re.sub(r'@\S+', ' ', x)
        x = re.sub(r'#\S+', ' ', x)
        x = re.sub(r'\'\w+', '', x)
        x = re.sub('[%s]' % re.escape(string.punctuation), ' ', x)
        x = re.sub(r'\w*\d+\w*', '', x)
        x = re.sub(r'\s{2,}', ' ', x)
        return x

    #Data combiner used to combine and extract comments from csv file
    def pull_data(self):
        comment_list = []
        for subreddit in self.subreddit_list:
            comments = self.extract_comment(subreddit)
            comment_list.extend(comments)
            dataFrame = pd.DataFrame(comments, columns=['comment'])
            dataFrame.to_csv(subreddit+'.csv', index=False)
```

Appendix B:

Tested Endpoints and Output

Table showing unit tests performed on backend endpoints.

EndPoint No.	*Test Description*	*Data Submitted*	*Test Assertion*	*Test Successful?*
1	Verify working server response.	None	Status = 200	Yes.
2	Verify receives working response.	Subreddit: "Gaming" Size: "5"	Status = 200	Yes.
2	Test data is returned.	Subreddit: "Conservative" Size: 5	Data Length > 0	Yes.
2	Test response when data is missing.	{} empty data	Status = 400 (bad request)	Yes.

2	Test response when subreddit does not exist.	Subreddit: "fake_subreddit_not_exist"
		Size: "5"
			Status = 404 (not found error)
			Data = "Subreddit Not Found"
				Yes.

2	Test response when no subreddit is entered.	Subreddit: ""
		Size "5"
			Status = 400
			Data = "Empty Subreddit Value"
				Yes.

3	Verify working server response.	Comment: "Hello World"	Status = 200	Yes.
3	Test data is returned from server.	Comment: "Test Returns Values"	Data Length > 0	Yes.
3	Test endpoint handles special characters	Comment: "test special, chars! #12345-=_+{}~:@<>?/.,"	Status = 200	Yes.

3	Test response when no comment entered	Comment: ""	Status = 400
			Data = "Empty Comment Value"
				Yes.

3	Test response when string exceeds 512 length string upper limit.	Comment: "a" * 513	Status = 400
			Data = "Comment String Too Long"
				Yes.

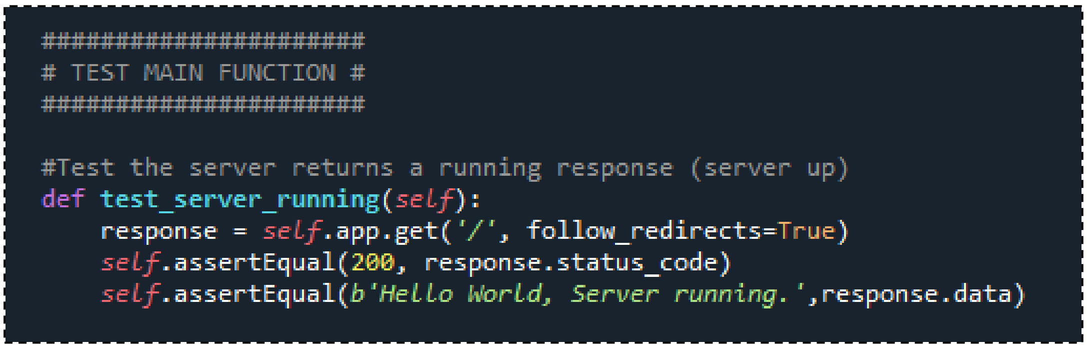
4	Verify working server response.	Subreddit: “Gaming”	Status = 200	Yes.

4	Test Subreddit that does not exist	Subreddit: ”fake_subreddit_not_exist”	Status = 404
			Data = “Subreddit Not Found”
				Yes.

4	Test response when subreddit data left empty	Subreddit: “”	Status = 400
			Data = “Empty Subreddit Value”
				Yes.

Main Endpoint Test's

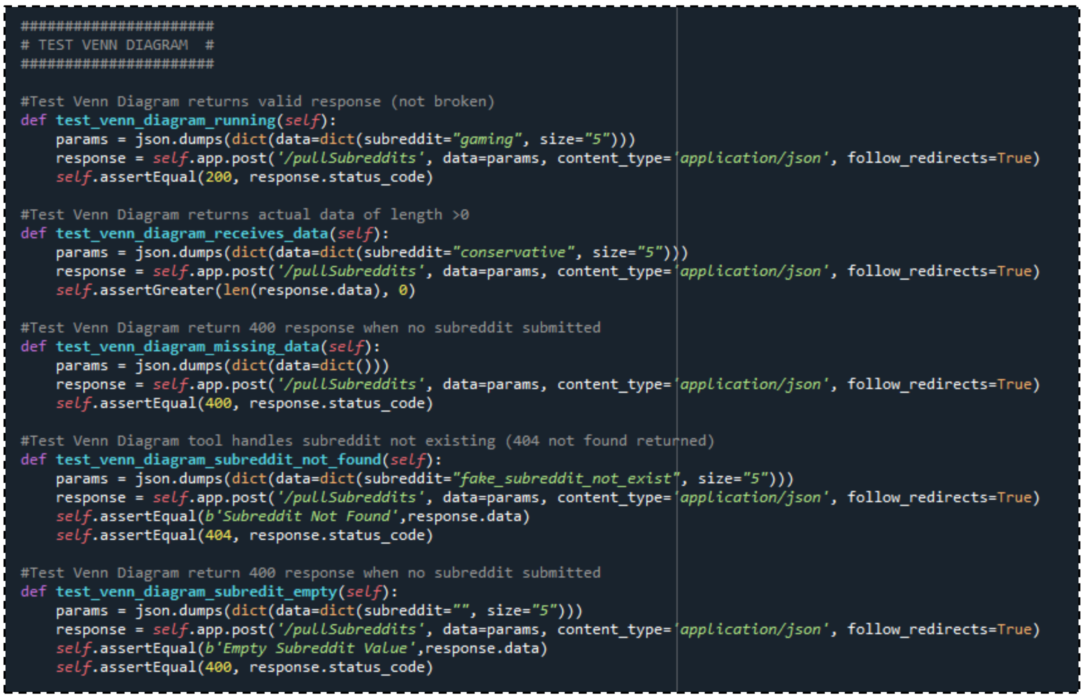
Main Endpoint Unit Test Code.



```
########################
# TEST MAIN FUNCTION #
########################

#Test the server returns a running response (server up)
def test_server_running(self):
    response = self.app.get('/', follow_redirects=True)
    self.assertEqual(200, response.status_code)
    self.assertEqual(b'Hello World, Server running.',response.data)
```

Venn Diagram Endpoint Test's

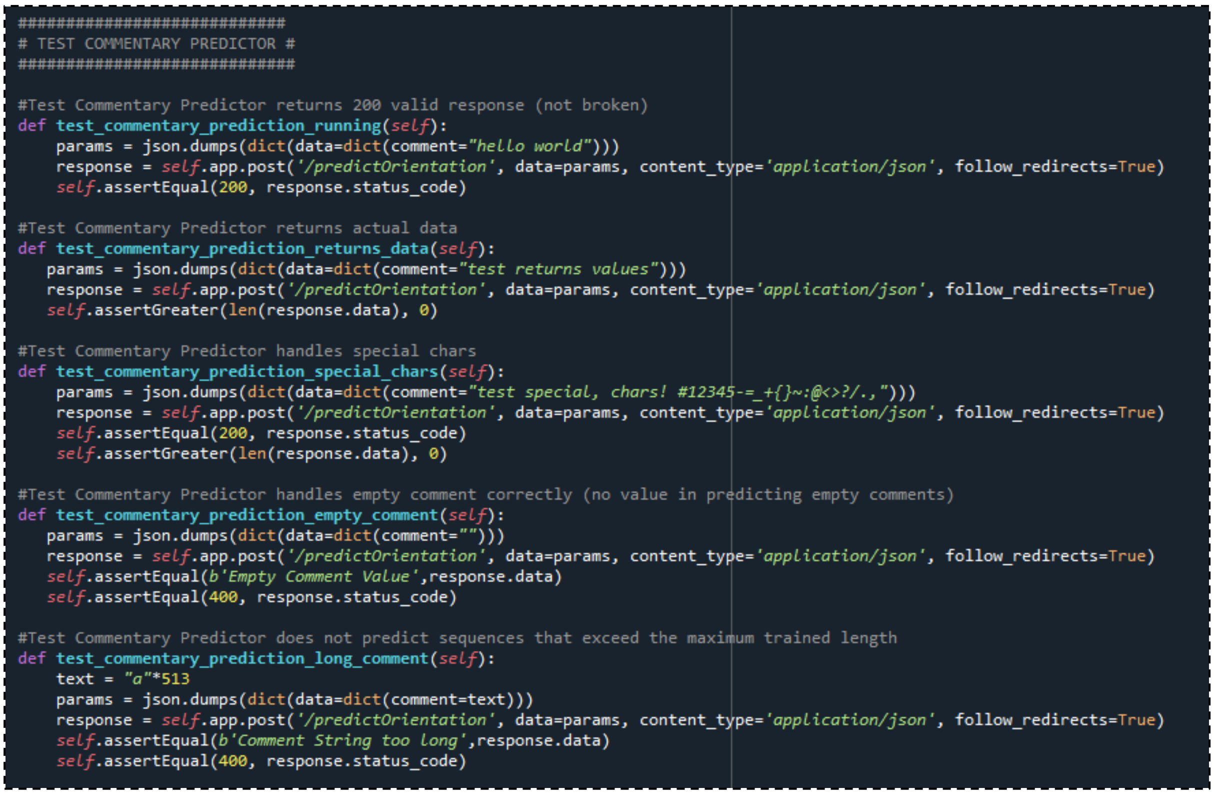
Venn Diagram Endpoint Unit Test's Code.



```
######################
# TEST VENN DIAGRAM  #
######################

#Test Venn Diagram returns valid response (not broken)
def test_venn_diagram_running(self):
    params = json.dumps(dict(data=dict(subreddit="gaming", size="5")))
    response = self.app.post('/pullSubreddits', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(200, response.status_code)

#Test Venn Diagram returns actual data of length >0
def test_venn_diagram_receives_data(self):
    params = json.dumps(dict(data=dict(subreddit="conservative", size="5")))
    response = self.app.post('/pullSubreddits', data=params, content_type='application/json', follow_redirects=True)
    self.assertGreater(len(response.data), 0)

#Test Venn Diagram return 400 response when no subreddit submitted
def test_venn_diagram_missing_data(self):
    params = json.dumps(dict(data=dict()))
    response = self.app.post('/pullSubreddits', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(400, response.status_code)

#Test Venn Diagram tool handles subreddit not existing (404 not found returned)
def test_venn_diagram_subreddit_not_found(self):
    params = json.dumps(dict(data=dict(subreddit="fake_subreddit_not_exist", size="5")))
    response = self.app.post('/pullSubreddits', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(b'Subreddit Not Found',response.data)
    self.assertEqual(404, response.status_code)

#Test Venn Diagram return 400 response when no subreddit submitted
def test_venn_diagram_subredit_empty(self):
    params = json.dumps(dict(data=dict(subreddit="", size="5")))
    response = self.app.post('/pullSubreddits', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(b'Empty Subreddit Value',response.data)
    self.assertEqual(400, response.status_code)
```

Commentary Predictor Endpoint Test's

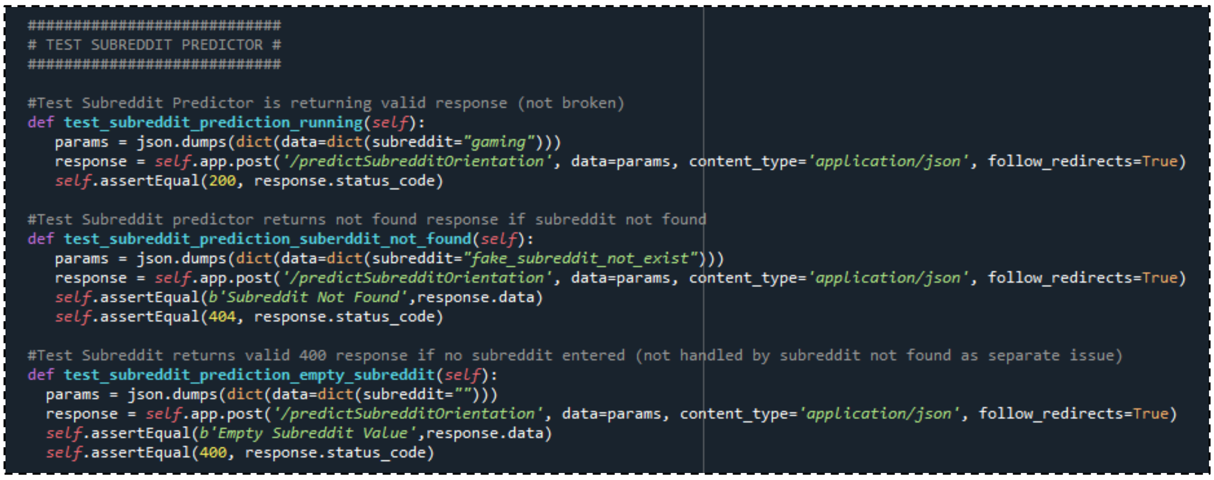
Commentary Predictor Endpoint Unit Test's Code.



```python
##############################
# TEST COMMENTARY PREDICTOR #
##############################

#Test Commentary Predictor returns 200 valid response (not broken)
def test_commentary_prediction_running(self):
    params = json.dumps(dict(data=dict(comment="hello world")))
    response = self.app.post('/predictOrientation', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(200, response.status_code)

#Test Commentary Predictor returns actual data
def test_commentary_prediction_returns_data(self):
   params = json.dumps(dict(data=dict(comment="test returns values")))
   response = self.app.post('/predictOrientation', data=params, content_type='application/json', follow_redirects=True)
   self.assertGreater(len(response.data), 0)

#Test Commentary Predictor handles special chars
def test_commentary_prediction_special_chars(self):
    params = json.dumps(dict(data=dict(comment="test special, chars! #12345-=_+{}~:@<>?/.,")))
    response = self.app.post('/predictOrientation', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(200, response.status_code)
    self.assertGreater(len(response.data), 0)

#Test Commentary Predictor handles empty comment correctly (no value in predicting empty comments)
def test_commentary_prediction_empty_comment(self):
   params = json.dumps(dict(data=dict(comment="")))
   response = self.app.post('/predictOrientation', data=params, content_type='application/json', follow_redirects=True)
   self.assertEqual(b'Empty Comment Value',response.data)
   self.assertEqual(400, response.status_code)

#Test Commentary Predictor does not predict sequences that exceed the maximum trained length
def test_commentary_prediction_long_comment(self):
    text = "a"*513
    params = json.dumps(dict(data=dict(comment=text)))
    response = self.app.post('/predictOrientation', data=params, content_type='application/json', follow_redirects=True)
    self.assertEqual(b'Comment String too long',response.data)
    self.assertEqual(400, response.status_code)
```

Subreddit Orientation Predictor Endpoint Test's

Subreddit Orientation Predictor Endpoint Unit Test's Code.



```python
############################
# TEST SUBREDDIT PREDICTOR #
############################

#Test Subreddit Predictor is returning valid response (not broken)
def test_subreddit_prediction_running(self):
   params = json.dumps(dict(data=dict(subreddit="gaming")))
   response = self.app.post('/predictSubredditOrientation', data=params, content_type='application/json', follow_redirects=True)
   self.assertEqual(200, response.status_code)

#Test Subreddit predictor returns not found response if subreddit not found
def test_subreddit_prediction_suberddit_not_found(self):
   params = json.dumps(dict(data=dict(subreddit="fake_subreddit_not_exist")))
   response = self.app.post('/predictSubredditOrientation', data=params, content_type='application/json', follow_redirects=True)
   self.assertEqual(b'Subreddit Not Found',response.data)
   self.assertEqual(404, response.status_code)

#Test Subreddit returns valid 400 response if no subreddit entered (not handled by subreddit not found as separate issue)
def test_subreddit_prediction_empty_subreddit(self):
  params = json.dumps(dict(data=dict(subreddit="")))
  response = self.app.post('/predictSubredditOrientation', data=params, content_type='application/json', follow_redirects=True)
  self.assertEqual(b'Empty Subreddit Value',response.data)
  self.assertEqual(400, response.status_code)
```

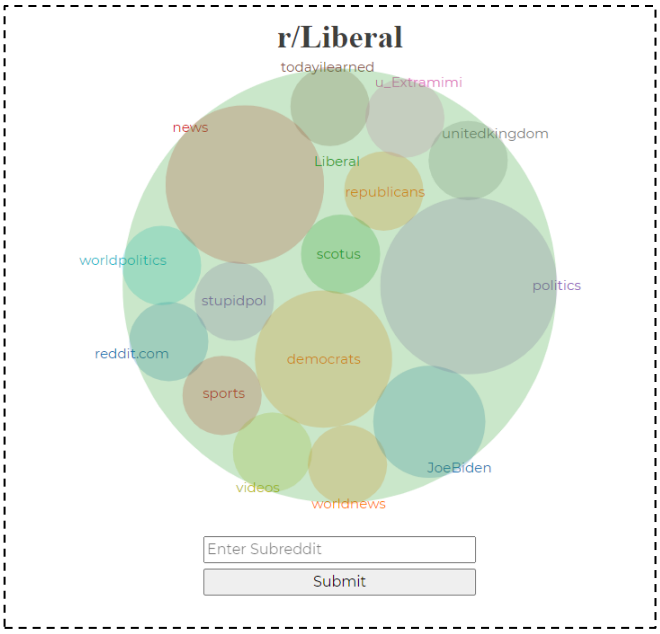
Appendix C:

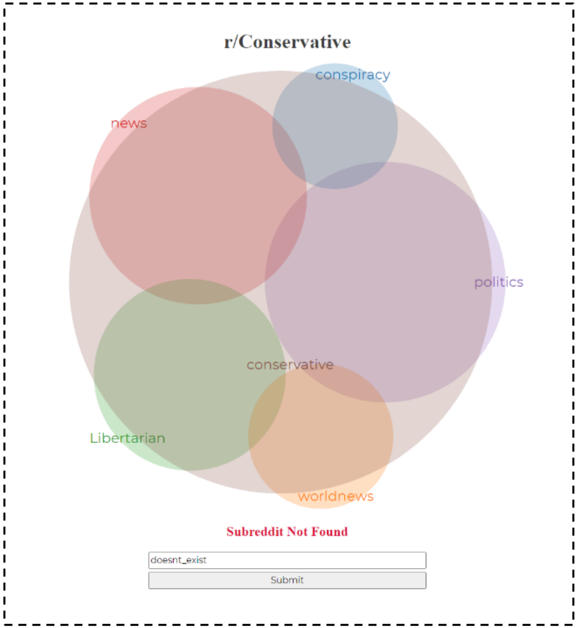
Front-End Testing Table

Test	*Test Description*	*Data Entered*	*Expected Output*	*Test Passed?*
1.1	Test Valid Text Input	Subreddit: "Liberal"	Venn Diagram to change to contain similar subreddits *r/Liberal*	Yes.
1.2	Test Subreddit Name that does not exist	Subreddit: "Doesnt_Exist"	Venn Diagram should not change current view, but display "Subreddit Not Found" text on screen.	Yes.
1.3	Test No Subreddit Name	Subreddit: ""	The built in HTML input required should trigger, although attempting to bypass should result in a "Subreddit Input Required" text	Yes.
1.4	Test Subreddit Name Greater than 50 characters	Subreddit: “ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ”	The React *onChange* function should not allow more than 50 characters, even with copy & paste, it should stop at W. Attempts to bypass result in "Subreddit Not Found" text	Yes.
1.5	Attempt to click/search subreddit during current search	Subreddit: "Politics"	The frontend should disable clicking the Venn diagram / searching the form during search to prevent simultaneous searches.	Yes.
2.1	Change number of words to display	Word Value: 50	It should update the sketch to show 50-word bubbles.	Yes.
2.2	Attempt to change value on form through inspect element, too low.	Word Value: 0	It should catch the extreme case and default the sketch to 20-word bubbles.	Yes.
2.3	Attempt to change value on form through inspect element, too high	Word Value: 1000	Again, it should catch this extreme value and default to 20-word bubbles	Yes.

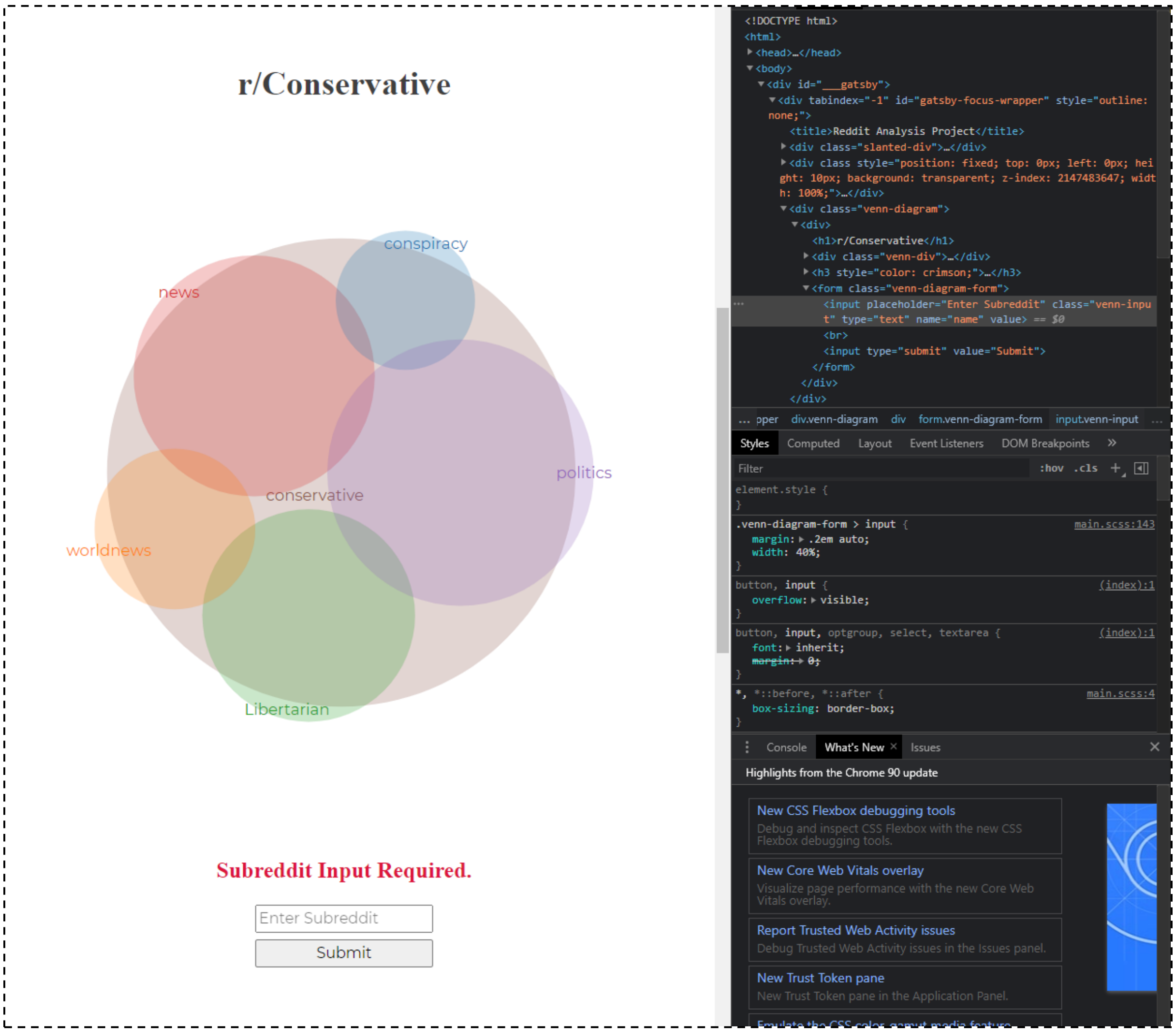
3.1	Enter valid comment for prediction	Comment: "Hello World"	It should attempt to classify the comment and output to the table the left and right wing %	Yes.
3.2	Enter empty comment and attempt to predict	Comment: ""	Should receive an error: "Comment cannot be empty"	Yes.
3.3	Enter comment over 512-character limit	Comment: “a” * 513	*onChange* function will not allow more than 512 characters entered into TextArea / state, although attempting to circumvent will result in "Comment Length must be < 512"	Yes.
3.4	Attempt to classify special characters	Comment: “!"£$%^&*()_+{}:@<>?-=[];'#,./”	Should process and return classification result.	Yes.
4.1	Enter valid subreddit for prediction	Subreddit: "Conservative"	It should classify the subreddit correctly and output the result.	Yes.
4.2	Leave Subreddit value empty	Subreddit: ""	It should return for the user to enter a subreddit.	No.
4.3	Enter Subreddit that does not exist	Subreddit: "Doesn't_Exist"	Should return "Failed to predict Subreddit" to the user to alert them of non-existing subreddit	Yes.
4.4	Enter Subreddit with > 50 characters	Subreddit: “a” * 51	Form should not allow to enter more than 15 characters or submit more without altering state.	Yes.

Front-End Testing Output Screenshots





Test 1.1 *Test 1.2*



```
<!DOCTYPE html>
<html>
 ▸<head>…</head>
 ▾<body>
   ▾<div id="___gatsby">
     ▾<div tabindex="-1" id="gatsby-focus-wrapper" style="outline:
      none;">
         <title>Reddit Analysis Project</title>
       ▸<div class="slanted-div">…</div>
       ▸<div class style="position: fixed; top: 0px; left: 0px; hei
        ght: 10px; background: transparent; z-index: 2147483647; widt
        h: 100%;">…</div>
       ▾<div class="venn-diagram">
         ▾<div>
            <h1>r/Conservative</h1>
          ▸<div class="venn-div">…</div>
          ▸<h3 style="color: crimson;">…</h3>
          ▾<form class="venn-diagram-form">
...          <input placeholder="Enter Subreddit" class="venn-inpu
             t" type="text" name="name" value> == $0
             <br>
             <input type="submit" value="Submit">
           </form>
         </div>
       </div>
```

```
... pper   div.venn-diagram   div   form.venn-diagram-form   input.venn-input   ...
Styles   Computed   Layout   Event Listeners   DOM Breakpoints   »
Filter                                                    :hov  .cls  +
element.style {
}
.venn-diagram-form > input {                              main.scss:143
   margin: ▸ .2em auto;
   width: 40%;
}
button, input {                                              (index):1
   overflow: ▸ visible;
}
button, input, optgroup, select, textarea {                  (index):1
   font: ▸ inherit;
   ~~margin: ▸ 0;~~
}
*, *::before, *::after {                                    main.scss:4
   box-sizing: border-box;
}
```

Console   What's New ×   Issues

Highlights from the Chrome 90 update

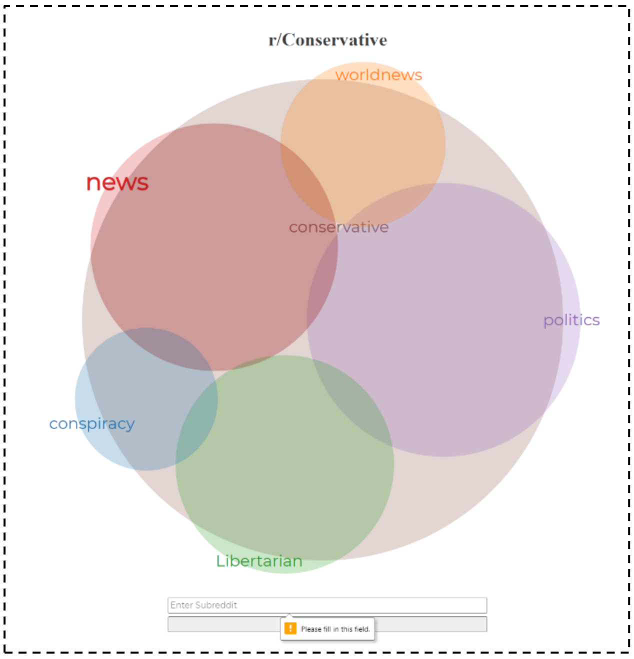
New CSS Flexbox debugging tools
Debug and inspect CSS Flexbox with the new CSS Flexbox debugging tools.

New Core Web Vitals overlay
Visualize page performance with the new Core Web Vitals overlay.

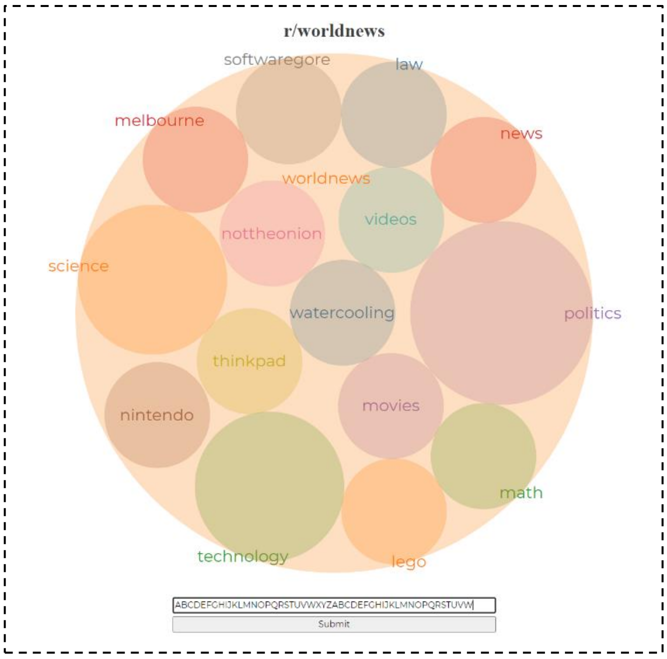
Report Trusted Web Activity issues
Debug Trusted Web Activity issues in the Issues panel.

New Trust Token pane
New Trust Token pane in the Application Panel.

Test 1.3

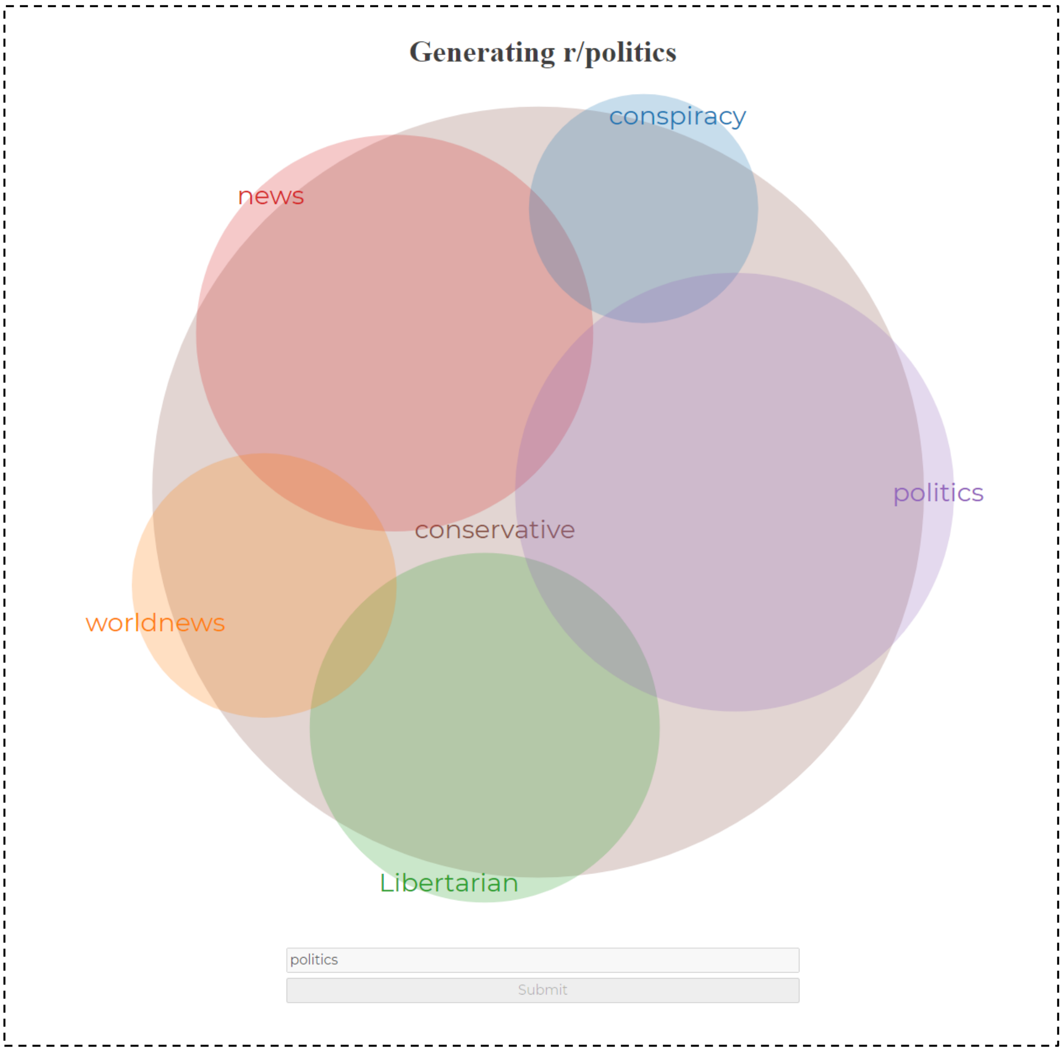


Test 1.4



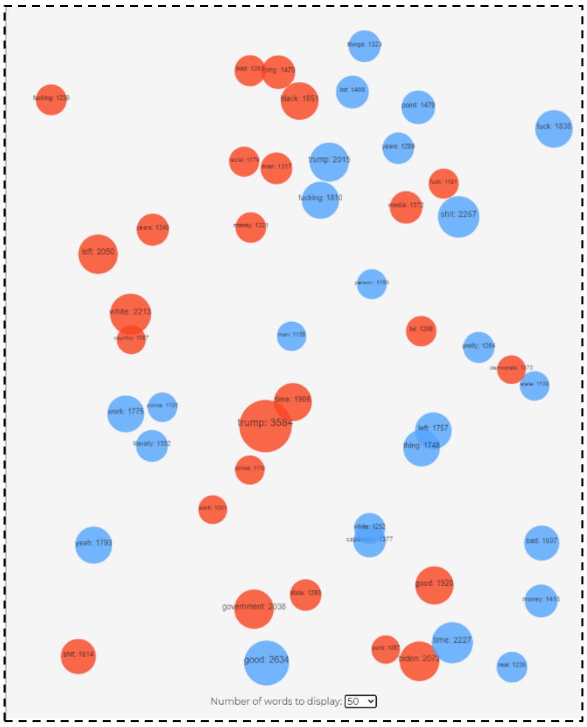
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVW

Test 1.5

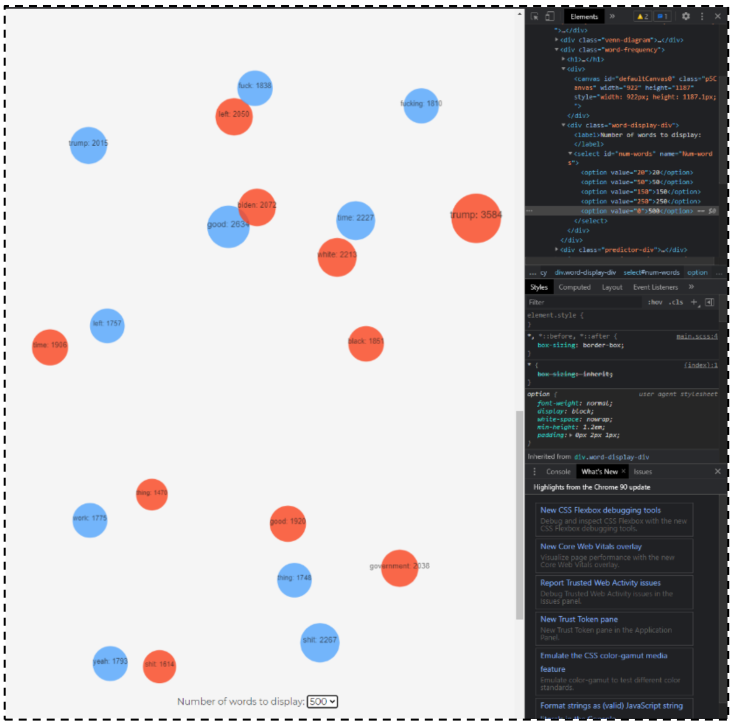


politics

Test 1.6



Test 2.1



```
">…</div>
▸<div class="venn-diagram">…</div>
▾<div class="word-frequency">
  ▸<h1>…</h1>
  ▾<div>
    <canvas id="defaultCanvas0" class="p5Canvas" width="922" height="1187"
    style="width: 922px; height: 1187.1px;
    ">
  </div>
  ▾<div class="word-display-div">
    <label>Number of words to display:
    </label>
   ▾<select id="num-words" name="Num-words">
      <option value="20">20</option>
      <option value="50">50</option>
      <option value="150">150</option>
      <option value="250">250</option>
      <option value="0">500</option> == $0
    </select>
  </div>
</div>
▸<div class="predictor-div">…</div>
```

```
element.style {
}
*, *::before, *::after {                 main.scss:4
  box-sizing: border-box;
}
* {                                      (index):1
  ~~box-sizing: inherit;~~
}
option {                      user agent stylesheet
  font-weight: normal;
  display: block;
  white-space: nowrap;
  min-height: 1.2em;
  padding: ▸ 0px 2px 1px;
}
Inherited from div.word-display-div
```

Highlights from the Chrome 90 update

New CSS Flexbox debugging tools
Debug and inspect CSS Flexbox with the new CSS Flexbox debugging tools.

New Core Web Vitals overlay
Visualize page performance with the new Core Web Vitals overlay.

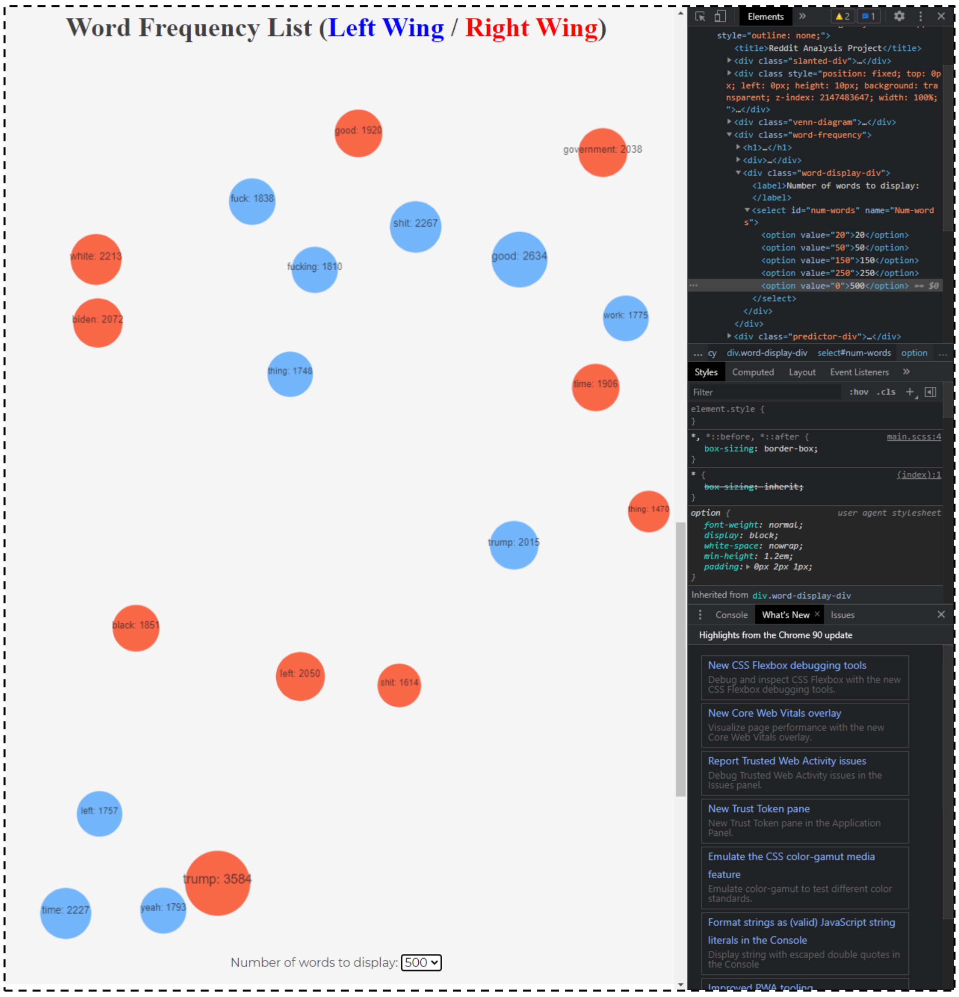
Report Trusted Web Activity issues
Debug Trusted Web Activity issues in the Issues panel.

New Trust Token pane
New Trust Token pane in the Application Panel.

Emulate the CSS color-gamut media feature
Emulate color-gamut to test different color standards.

Format strings as (valid) JavaScript string

Test 2.2



```
style="outline: none;">
  <title>Reddit Analysis Project</title>
 ▸<div class="slanted-div">…</div>
 ▸<div class style="position: fixed; top: 0px; left: 0px; height: 10px; background: transparent; z-index: 2147483647; width: 100%;
  ">…</div>
 ▸<div class="venn-diagram">…</div>
 ▾<div class="word-frequency">
   ▸<h1>…</h1>
   ▸<div>…</div>
   ▾<div class="word-display-div">
      <label>Number of words to display:
      </label>
     ▾<select id="num-words" name="Num-words">
        <option value="20">20</option>
        <option value="50">50</option>
        <option value="150">150</option>
        <option value="250">250</option>
        <option value="0">500</option> == $0
      </select>
    </div>
  </div>
 ▸<div class="predictor-div">…</div>
```

```
… cy  div.word-display-div  select#num-words  option  …

element.style {
}
*, *::before, *::after {                 main.scss:4
  box-sizing: border-box;
}
* {                                      (index):1
  ~~box-sizing: inherit;~~
}
option {                      user agent stylesheet
  font-weight: normal;
  display: block;
  white-space: nowrap;
  min-height: 1.2em;
  padding: ▸ 0px 2px 1px;
}
Inherited from div.word-display-div
```

Highlights from the Chrome 90 update

New CSS Flexbox debugging tools
Debug and inspect CSS Flexbox with the new CSS Flexbox debugging tools.

New Core Web Vitals overlay
Visualize page performance with the new Core Web Vitals overlay.

Report Trusted Web Activity issues
Debug Trusted Web Activity issues in the Issues panel.

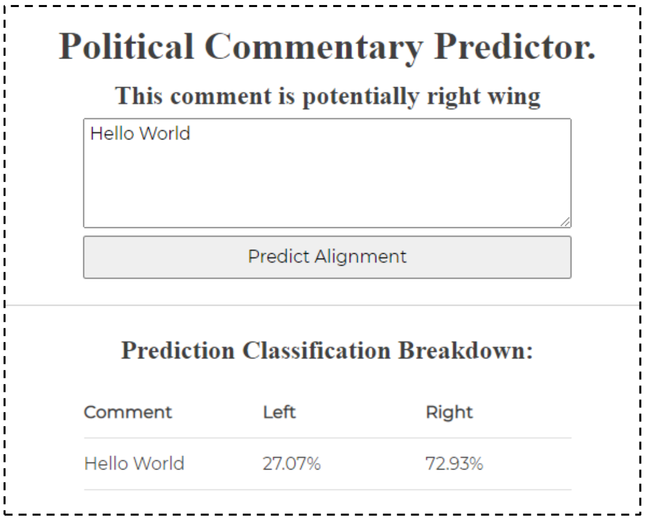
New Trust Token pane
New Trust Token pane in the Application Panel.

Emulate the CSS color-gamut media feature
Emulate color-gamut to test different color standards.

Format strings as (valid) JavaScript string literals in the Console
Display string with escaped double quotes in the Console

Improved PWA tooling

Test 2.3



Political Commentary Predictor.

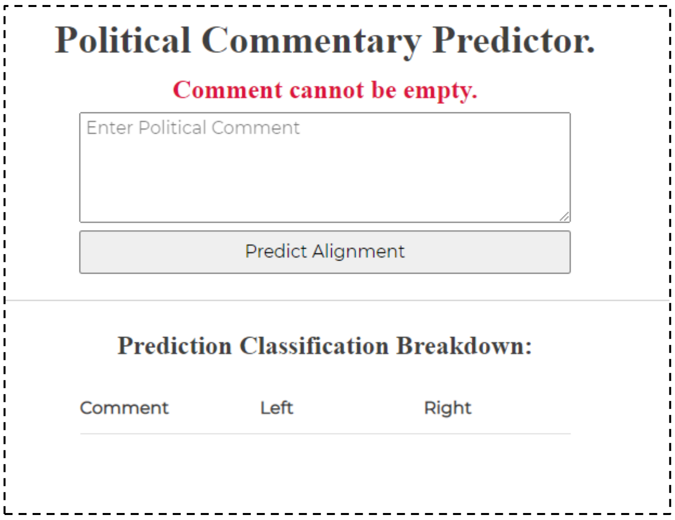
This comment is potentially right wing

Hello World

Prediction Classification Breakdown:

Comment	Left	Right
Hello World	27.07%	72.93%

Test 3.1



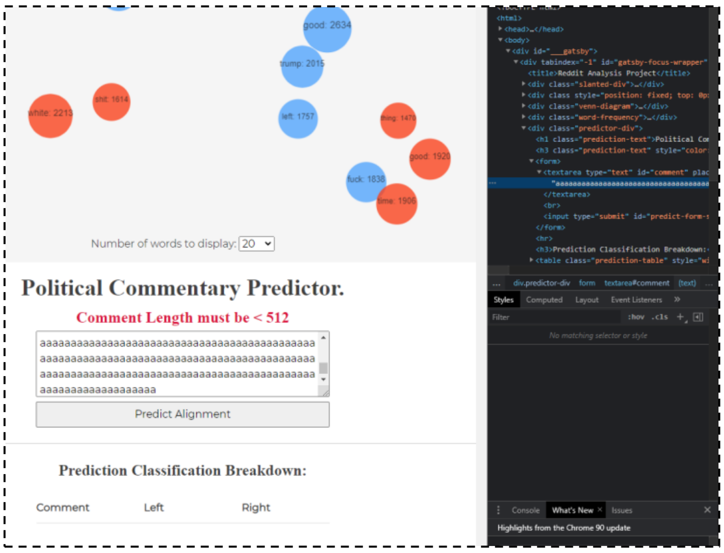
Political Commentary Predictor.

Comment cannot be empty.

Prediction Classification Breakdown:

Comment	Left	Right

Test 3.2



good: 2634
trump: 2015
white: 2213
shit: 1614
left: 1757
thing: 1470
good: 1920
fuck: 1838
time: 1906
Number of words to display: 20
Predict Alignment
div.predictor-div form textarea#comment (text)
Styles Computed Layout Event Listeners
Filter :hov .cls
No matching selector or style
Console What's New Issues
Highlights from the Chrome 90 update

Political Commentary Predictor.

Comment Length must be < 512

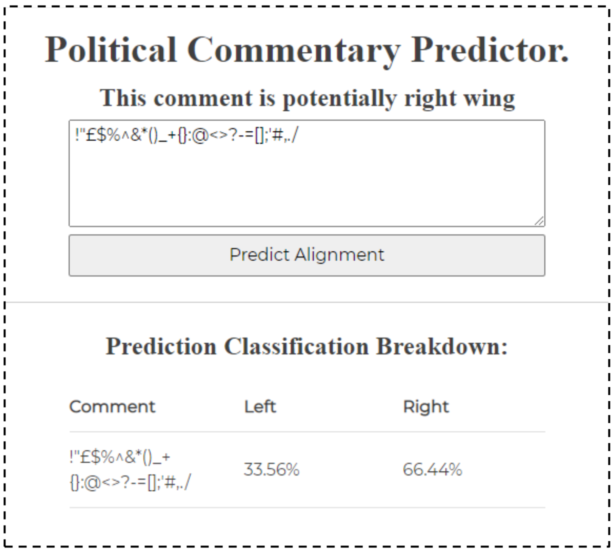
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa

Prediction Classification Breakdown:

Comment	Left	Right

```
<html>
 ▸<head>…</head>
 ▾<body>
  ▾<div id="___gatsby">
   ▾<div tabindex="-1" id="gatsby-focus-wrapper"
     <title>Reddit Analysis Project</title>
    ▸<div class="slanted-div">…</div>
    ▸<div class style="position: fixed; top: 0p
    ▸<div class="venn-diagram">…</div>
    ▸<div class="word-frequency">…</div>
    ▾<div class="predictor-div">
      <h1 class="prediction-text">Political Co
      <h3 class="prediction-text" style="color:
     ▾<form>
      ▾<textarea type="text" id="comment" plac
         "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
       </textarea>
       <br>
       <input type="submit" id="predict-form-s
      </form>
      <hr>
      <h3>Prediction Classification Breakdown:
     ▸<table class="prediction-table" style="wi
```

Test 3.3



Political Commentary Predictor.

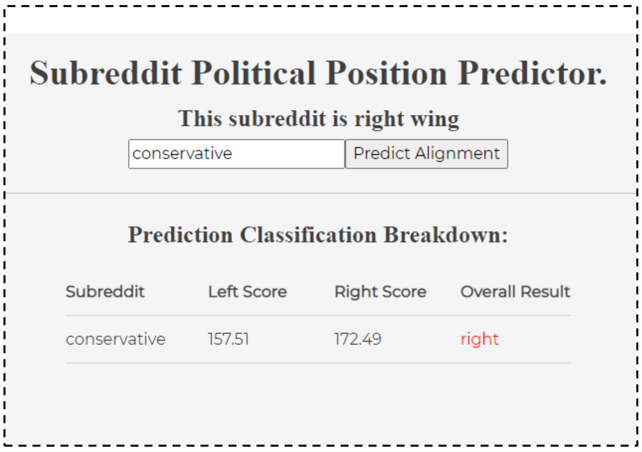
This comment is potentially right wing

!"£$%^&*()_+{}:@<>?-=[];'#,./

Prediction Classification Breakdown:

Comment	Left	Right
!"£$%^&*()_+ {}:@<>?-=[];'#,./	33.56%	66.44%

Test 3.4



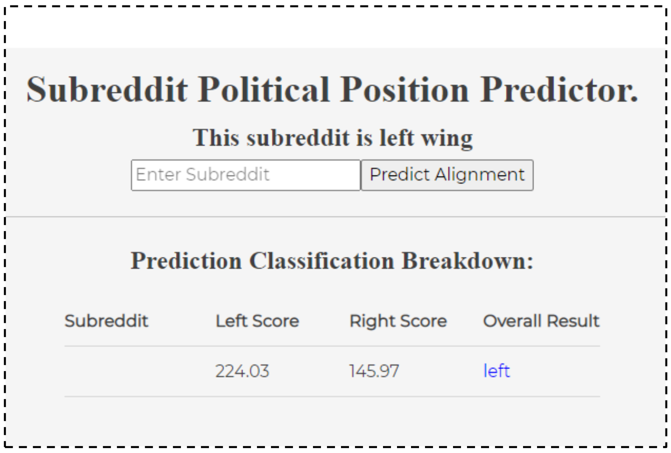
Subreddit Political Position Predictor.

This subreddit is right wing

conservative

Prediction Classification Breakdown:

Subreddit	Left Score	Right Score	Overall Result
conservative	157.51	172.49	right



Subreddit Political Position Predictor.

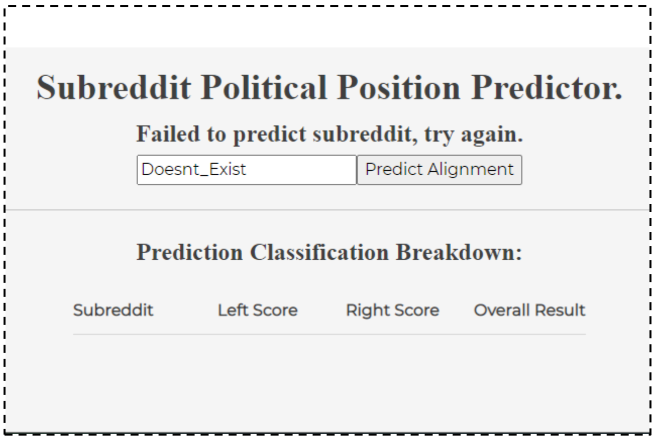
This subreddit is left wing

Prediction Classification Breakdown:

Subreddit	Left Score	Right Score	Overall Result
	224.03	145.97	left

Test 4.1

Test 4.2 –
Note: praw assumed the empty subreddit is /r/all as defined in praw's API


Subreddit Political Position Predictor.

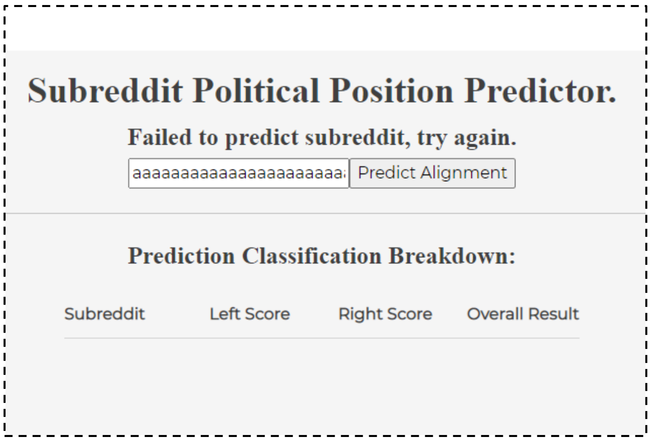
Failed to predict subreddit, try again.

Doesnt_Exist | Predict Alignment

Prediction Classification Breakdown:

Subreddit	Left Score	Right Score	Overall Result

Test 4.3


Subreddit Political Position Predictor.

Failed to predict subreddit, try again.

aaaaaaaaaaaaaaaaaaaaaa | Predict Alignment

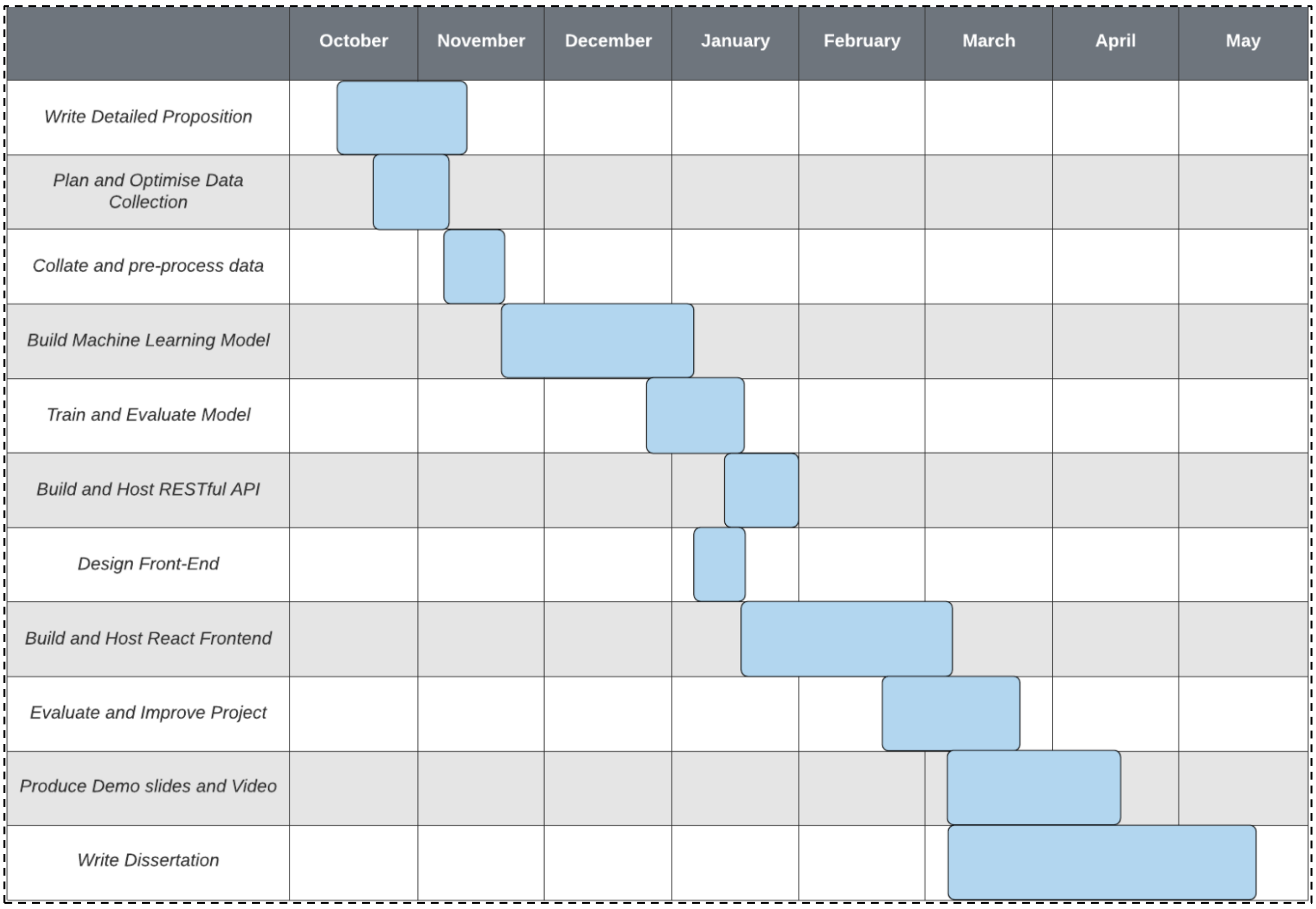
Prediction Classification Breakdown:

Subreddit	Left Score	Right Score	Overall Result

Test 4.4

Appendix D:

Proposed Gantt Chart Design.


October
November
December
January
February
March
April
May
Write Detailed Proposition
Plan and Optimise Data Collection
Collate and pre-process data
Build Machine Learning Model
Train and Evaluate Model
Build and Host RESTful API
Design Front-End
Build and Host React Frontend
Evaluate and Improve Project
Produce Demo slides and Video
Write Dissertation