Detecting Interests in Social Media

Joshua Fitzmaurice
Department of Computer Science
University of Warwick

Supervised by Arshad Jhumka
Year of Study: 3rd

28 March 2023

WARWICK
THE UNIVERSITY OF WARWICK

Abstract

Your abstract goes here. This should be about 2-3 paragraphs summarising the motivation for your project and the main outcomes (software, results, etc.) of your project.

Contents

Chapter 1

Introduction

1.1 Motivation

The content seen on social media varies greatly from user to user. This is because social media use recommender systems to show users content they are likely to be "interested" in . These recommender systems are based on a users past interactions Typically, if a user views a certain type of post, they are likely to view similar posts in the future. This project aims to find the interests of a user by analysing their social media posts.

Take a look at the following example:

This set of social media posts are taken from a user's social media feed. As a human, we can easily identify what the user is interested in; they are interested in Formula 1 and Food (specifically, steak). We make this assumption based on the content of the posts.

Looking at the posts, there rises a couple questions:

- What are the posts about?
- Are the posts seen representative of all posts across the social media platform?

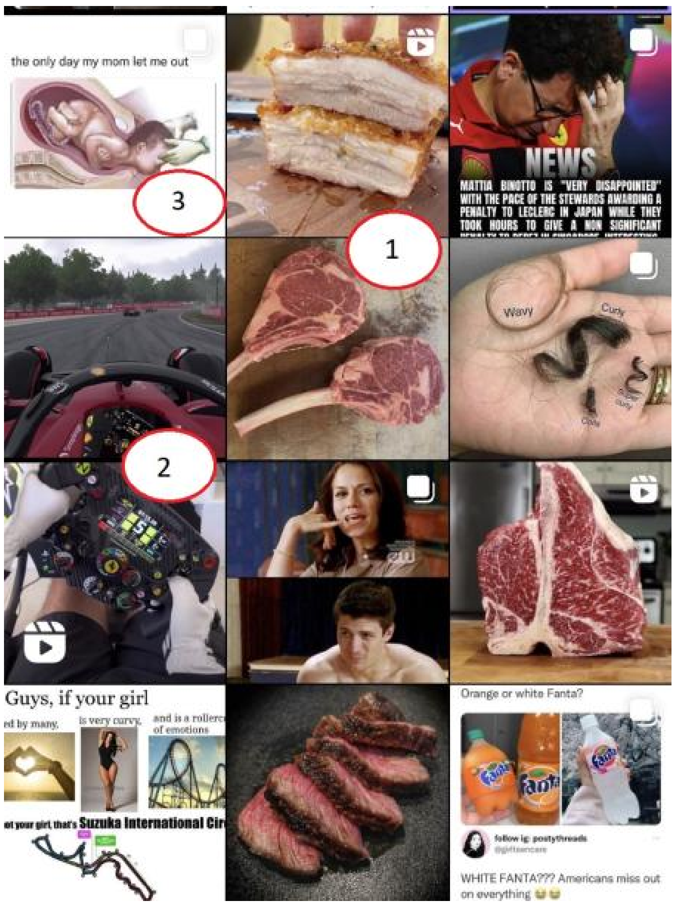

the only day my mom let me out
3
1
NEWS
MATTIA BINOTTO IS "VERY DISAPPOINTED" WITH THE PACE OF THE STEWARDS AWARDING A PENALTY TO LECLERC IN JAPAN WHILE THEY TOOK HOURS TO GIVE A NON SIGNIFICANT
Wavy
Curly
2
Guys, if your girl
is very curvy,
and is a rollerc
of emotions
Suzuka International Cir
Orange or white Fanta?
WHITE FANTA??? Americans miss out on everything

Figure 1.1: Example tweet

- How can users find other posts that are dissimilar to the posts they are shown?

1.1.1 Topics

This project will use the notion of topics to identify what the posts are about.

a subject that is discussed, written about, or studied

Figure 1.2: Topic Definition - **?**

This definition of a topic is a good starting point for this project. This definition is a bit too specific. If this definition were to be used there would be no meaningful output from this project - due to the fact you could classify each post into its own unique topic. To overcome this, we will use a more general definition of a topic. Essentially, a topic comprises of a set of words that are related to each other.

1.1.2 Problem Statement

The goal of this project is to be able to classify social media posts. This will allow us to identify interests through comparing similarities between posts shown to the user. This will be followed by quantifying a set of posts to compare the similarity between a users posts and posts from the entire social media site. Finally, this project aims to create a user interface that allows users to discover what their social media feed says about their interests and how they compare to the rest of the social media site. The user interface will also allow users to discover posts that are dissimilar to the posts they are shown.

1.2 Related work

1.2.1 Pythia - Litou and Kalogeraki (2017)

Pythia is an automated system for short text classification. It makes use of Wikipedia structure and articles to identify topics of posts. Essentially, "Wikipedia contains articles organized in various taxonomies, called categories". Pythia then goes on to use this information as their training data as well as handling sparseness in posts on social media.

Pythia also demonstrates a method to overcome the lack of context in short texts - This is a large problem in identifying smaller social media posts like tweets, and will be further worked on in this project. They use a method called "Post Enrichment", which performs i) Named Entity Recognition then ii) Lemmatization and stop word removal. We then use the named entities to query wikipedia for similar articles that are then appended to the post.

Although this method works well in cases where keywords are used, there are cases where no keywords are used, and more context is needed. Take for example the following tweet:

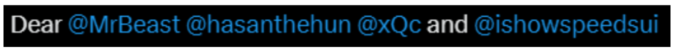
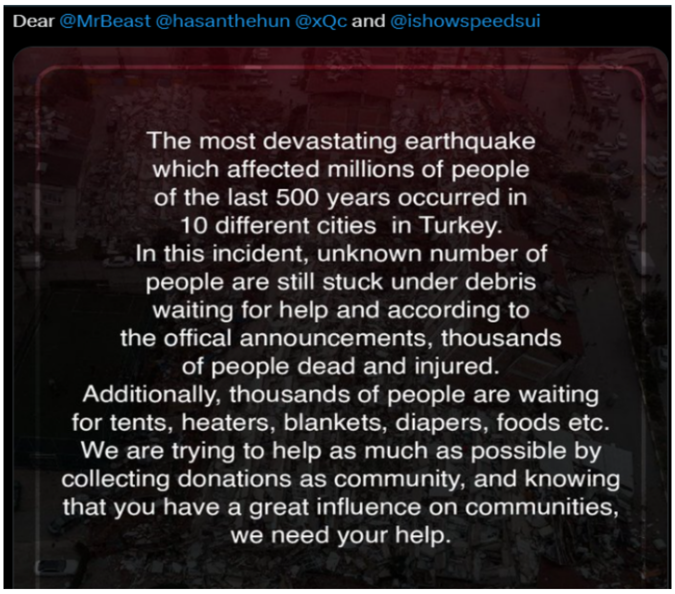

Dear @MrBeast @hasanthehun @xQc and @ishowspeedsui

Figure 1.3: Example tweet

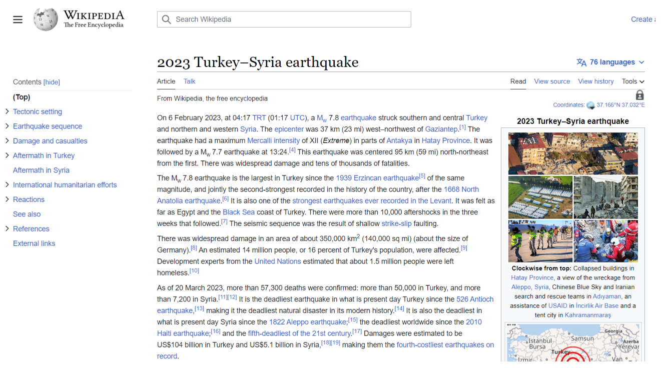
The text gives us very little context; What is this tweet about? the best guess I could give is it is a message to other users. If these users had wikipedia articles, we could use them. But in reality this post is about something else. Lets add some other form of context; add the media the tweet contains. This gives us a lot more context; the tweet is discussing the earthquake that hit Turkey and Syria, if we use this for our query we get more relevant articles:



Dear @MrBeast @hasanthehun @xQc and @ishowspeedsui

The most devastating earthquake which affected millions of people of the last 500 years occurred in 10 different cities in Turkey. In this incident, unknown number of people are still stuck under debris waiting for help and according to the offical announcements, thousands of people dead and injured. Additionally, thousands of people are waiting for tents, heaters, blankets, diapers, foods etc. We are trying to help as much as possible by collecting donations as community, and knowing that you have a great influence on communities, we need your help.

Figure 1.4: Example tweet - media



2023 Turkey–Syria earthquake

From Wikipedia, the free encyclopedia

Coordinates: 37.166°N 37.032°E

On 6 February 2023, at 04:17 TRT (01:17 UTC), a M_w 7.8 earthquake struck southern and central Turkey and northern and western Syria. The epicenter was 37 km (23 mi) west–northwest of Gaziantep.[1] The earthquake had a maximum Mercalli intensity of XII (*Extreme*) in parts of Antakya in Hatay Province. It was followed by a M_w 7.7 earthquake at 13:24.[4] This earthquake was centered 95 km (59 mi) north-northeast from the first. There was widespread damage and tens of thousands of fatalities.

The M_w 7.8 earthquake is the largest in Turkey since the 1939 Erzincan earthquake[5] of the same magnitude, and jointly the second-strongest recorded in the history of the country, after the 1668 North Anatolia earthquake.[6] It is also one of the strongest earthquakes ever recorded in the Levant. It was felt as far as Egypt and the Black Sea coast of Turkey. There were more than 10,000 aftershocks in the three weeks that followed.[7] The seismic sequence was the result of shallow strike-slip faulting.

There was widespread damage in an area of about 350,000 km^2 (140,000 sq mi) (about the size of Germany).[8] An estimated 14 million people, or 16 percent of Turkey's population, were affected.[9] Development experts from the United Nations estimated that about 1.5 million people were left homeless.[10]

As of 20 March 2023, more than 57,300 deaths were confirmed: more than 50,000 in Turkey, and more than 7,200 in Syria.[11][12] It is the deadliest earthquake in what is present day Turkey since the 526 Antioch earthquake,[13] making it the deadliest natural disaster in its modern history.[14] It is also the deadliest in what is present day Syria since the 1822 Aleppo earthquake;[15] the deadliest worldwide since the 2010 Haiti earthquake;[16] and the fifth-deadliest of the 21st century.[17] Damages were estimated to be US$104 billion in Turkey and US$5.1 billion in Syria,[18][19] making them the fourth-costliest earthquakes on record.

2023 Turkey–Syria earthquake

Clockwise from top: Collapsed buildings in Hatay Province, a view of the wreckage from Aleppo, Syria, Chinese Blue Sky and Iranian search and rescue teams in Adıyaman, an assistance of USAID in İncirlik Air Base and a tent city in Kahramanmaraş

Figure 1.5: Wikipedia articles related to the hashtag "BLASTPremier"

We now have a lot of relevant information to append to the post. We could use other information for context as well: Tweet author, images/media in tweets, retweet information, like information, etc.

1.2.2 Topic tracking of student-generated posts - Peng et al. (2020)

This paper proposes a solution for determining valuable information/topics discussed in student forums on online courses. It uses a model called "Time Information-Emotion Behaviour Model" or otherwise called "TI-EBTM" to detect key topics discussions , keeping in mind the progress of time throughout the forum.
Although this paper specializes in academic online forums, the approaches made could be relevant and useful for this project.

1.2.3 Topic classification of blogs - Husby and Barbosa (2012)

This paper uses Distant Supervision - ′an extension of the paradigm used by (Snow et al. (2004)) for exploiting WordNet to extract hypernym (is-a) relations between entitities′ - to get training data via Wikipedia articles. Then trains their own designed model on this data to be able to classify topics via a multi-class recognition model (69% accuracy) and via a binary classification model (90% accuracy).

1.3 Objectives

To achieve the problem statement, the following objectives must be met:

- Generate a list of topics for classification
- Implement methods for identifying the topics in social media posts
- Compare and contrast the results of the different methods
- Create a user interface
  - Allow users to compare their interests to the rest of the social media site

  - Allow users to discover posts that are dissimilar to the posts they are shown

The largest problem with this project is the second objective of finding and creating methods for identifying topics.

Chapter 2

Background

2.1 Topic Modelling

As discussed in Chapter 1, this project uses topics to identify what the posts are about. This section will discuss how topics are created and how they are used in this project.
There are 2 methods of topic modelling that were considered for this project. The first method is Unsupervised Learning, and the second method is Supervised Learning.

2.1.1 Unsupervised Learning

Unsupervised learning is a method of machine learning that does not require labelled data. This method is used to identify patterns in data. For this project, the patterns are topics.
Using LDATopic **?** and BERTTOPIC **?**, topics were identified from a set of unlabelled documents.

The benefits of using unsupervised learning are that it does not require labelled data, it is easy to implement, and the classification of posts with the generated topics should be accurate - as the topics are generated to be seperable. The downsides of using unsupervised learning are that the topics generated may

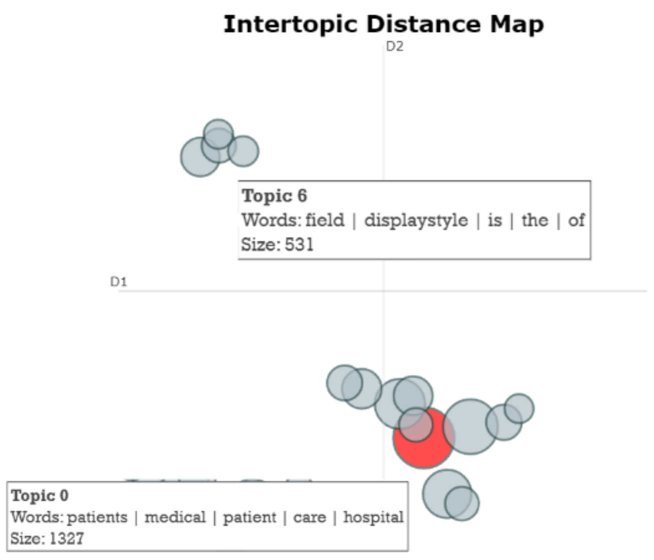


Figure 2.1: LDATopic creating topics from a set of unlabelled documents

not be meaningful; Take for example Topic 6 from Figure 2.1. This topic is made up of the words 'displaystyle', 'is', 'the', and 'of'. It is quite hard for a human to understand what this topic is about.

2.1.2 Supervised Learning

Supervised learning is a method of machine learning that requires labelled data. This method is used to identify patterns in data. For this project, a set of topics can be manually created and then used to label the posts. This method is much easier to implement than unsupervised learning, as the topics are already created. The downside of this method is that it requires labelled data, which can be time consuming to create.

For this project, supervised learning was used to classify the posts into the topics that were created. The original set of topics were:

- **Culture**
- **Entertainment**
- **News**
- **Philosophy**
- **Religion**
- **Science**
- **Sports**
- **Technology**
- **Law**
- **History**
- **Geography**
- **Video Games**
- **Music**
- **Medicine**
- **Business**
- **Foods**
- **Disasters**
- **Nature**
- **Education**
- **Politics**
- **Economics**
- **Computer Science**
- **Mathematics**

Pythia Litou and Kalogeraki (2017) gave inspiration for most of the topics. Some extra topics were included to deal with uncovered topics, such as 'Video Games' and 'Foods'.

Once the topics were created, labelled data was created using a distant supervision method. Using Subreddits and Wikipedia categories as ground truth labels, and posts within those Subreddits and Wikipedia categories as the data. The labelled data was then used to train a supervised learning model. Although distant supervision allows us to quickly create labelled data, it is not perfect. The ground truth labels may not be accurate, and the posts may not be relevant to the topic. This is due to the fact that the posts can be made by anyone and it is possible that someone may post something that is not relevant to the subreddit or wikipedia category they are posting in.

2.2 Text Classification

As mentioned in Chapter 1 there exists research on topic identification in short texts such as social media posts. Topic analysis on short posts is harder than on longer texts because of the lack of context ; Twitter posts are limited to 280 characters, so users tend to attach images, or reference other tweets (via retweeting) to add context that would not be obvious from the text alone.

2.2.1 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are Neural Networks that are used to model sequential data. They are good at modelling sequential data because they can take into account the previous inputs in the sequence when making a prediction. This makes them good for text as the text input can be seen as a sequence of words and we can leverage the previous words to make a prediction about the whole sentence.

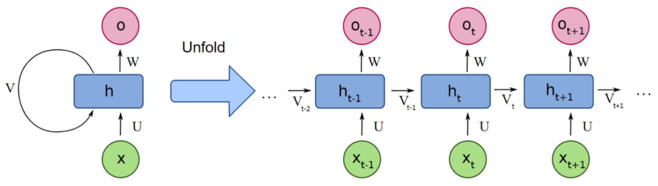


Figure 2.2: RNN Architecture

The RNN architecture in Figure 2.2 is a simple RNN. It takes in a sequence of words (X_i), and outputs a prediction. The RNN takes input words one at a time, and passes a hidden state (V_i) to the next word. The hidden state contains information about the previous words in the sequence. At the end of the sequence, the final hidden state of the RNN is fed through a softmax layer to get a probability distribution over the possible classes.

The main problem with RNNs is the affect of short-term memory. For every new word in the sequence, the RNN ‘forgets’ parts of the previous words in the sequence. This is because the hidden state is updated by each new word.

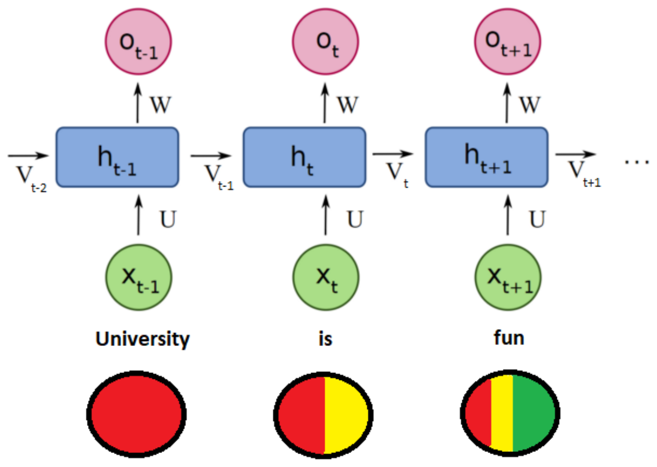


Figure 2.3: Problem with RNNs

In figure 2.3, we can see how much each 'word' in the sequence affects the hidden state over time. For long sequences, the hidden state is predominantly affected by the last words in the sequence. This means we lose information about the earlier words in the sequence. This could be a problem with our text classification problem as the key information in the text may be in the beginning of the sentence. For example: "Manchester United lost the match 2-1, it was a poor performance but the atmosphere in the theatre of dreams was astounding." The key information in this sentence is that Manchester United lost the match (meaning the text is about sport). However, this information could be lost due to the short-term memory problem. In the case above it is likely we classify the text as entertainment due to the later references of 'performance' and 'theatre'

2.2.2 Long Short-Term Memory (LSTM)

The Long Short-Term Memory (LSTM) model was developed in 1997 to solve the short-term memory problem in RNNs. It does this by using 3 memory gates: the input gate, the forget gate and the output gate. These gates perform different functions:

- **Input Gate:** The input gate decides which values from the current input should be added to the cell state.
- **Forget Gate:** The forget gate decides which values from the cell state should be kept.
- **Output Gate:** The output gate decides which values from the cell state should be used to make a prediction.

TODO: add lstm diagram and explain how the changes solve the short-term memory problem.

LSTM's (as well as RNNs) have another problem: they can only process information in one direction at a time. This means that they can only use previous words to understand the current word, or the next words to understand the current word. Bidirectional versions of these models aim to solve this problem by processing the sequence in both directions at the same time. However, this is not a perfect solution as it only 'learns' the context from both directions and not as a whole. In section 2.2.3 we will discuss a model that solves this problem.

2.2.3 Transformers

Architecture

In 2017 Google released a paper called *Attention is all you need* **?**. This paper introduced the Transformer architecture.

Figure 2.5 shows the architecture of a Transformer model. Focussing on the encoder, the input is passed through a 'Multi-Head Attention' layer. This layer takes in the input and creates a matrix of attention weights. These attention weights essentially say how much of an impact each word has on each other.

Self-Attention

Before looking into Multi-Head Attention, let's discuss Self-Attention.
Self-Attention is a Sequence-to-Sequence model that calculates the attention

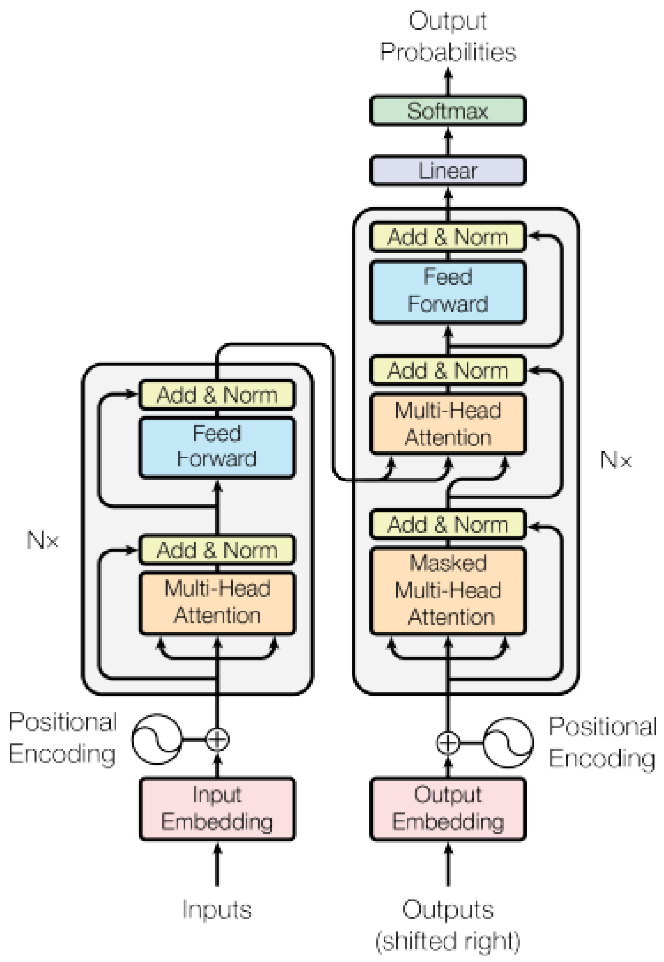


Figure 2.4: Transformer Architecture - **?**

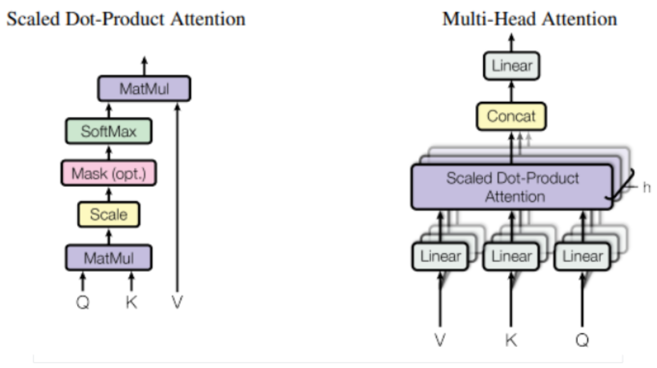


Figure 2.5: Self-Attention - **?**

weights between each word in the sequence. The attention weights are how important each word is to the other words in the sequence. The attention weights are calculated using 3 vectors: Query, Key, and Value. All of these vectors are made from passing the input through a linear layer. Each vector is made using a different linear layer.

Figure 2.5 shows how the attention weights are calculated. First, the Query and Key vectors are multiplied together. This is then passed through a softmax layer to convert all weights into probabilities. These probabilities act as the attention weights. The attention weights are then multiplied by the Value vector to get the final output.

Multi-Head Attention

The paper *Attention is all you need* **?** also introduces the Multi-Head Attention layer. This layer uses self-attention but splits the Query, Key and Value vectors into multiple heads. Each head calculate the attention weights independently, and their outputs are concatenated together.

TODO: benefits

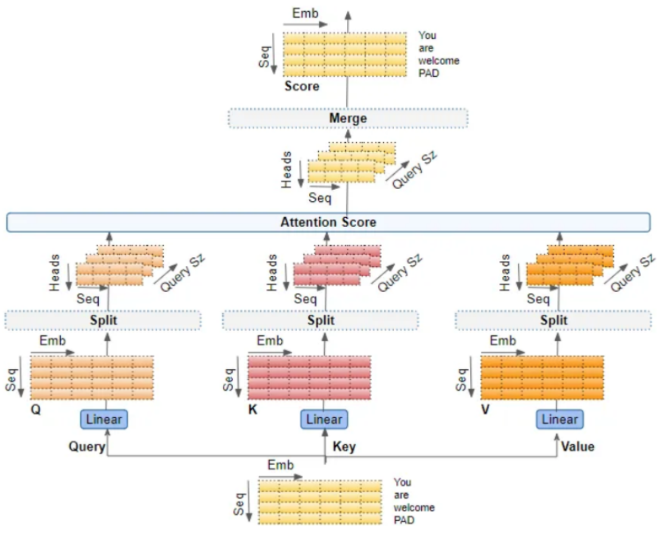


Figure 2.6: Multi-Head Attention

2.2.4 Bidirectional Encoder Representation of Transformers (BERT) - ?

BERT is an acronym for Bidirectional Encoder Represtations from Transformers. It's architecture uses several Transformer encoders put together. Transformers make use of self-attention to learn contextual representation of words. This solves the problem discussed in **??** where the RNNs only look at the previous words in the sentence, or both directions independently.

A BERT classification model is trained in 2 steps: pre-training and fine-tuning. The pre-training step is done on a large corpus by Google. The fine-tuning step is done on a smaller dataset specific to the task at hand.

pre-training

The pre-training step of the BERT model consists of learning 2 objectives: masked language modelling (MLM) and next sentence prediction (NSP).
Masked Language Modelling (MLM) is a task where the model is given a sentence with some words masked out. The model is then expected to predict the masked words.

../images/bert.png

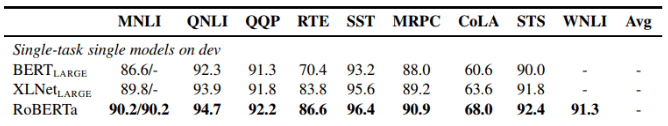
Figure 2.7: BERT Architecture

Next Sentence Prediction (NSP) is a task where the model is given 2 sentences, a and b. The model is then expected to predict whether sentence b is the next sentence following a.

The pre-training step is done on a large corpus of unlabelled data. This makes it easy to train the model, as it does not require collecting a large amount of labelled data. Using Transformers makes the model more efficient, as it can process the entire sentence at once instead of processing the sentence word by word.

fine-tuning

To use BERT for our task of topic classification, we need to fine-tune the model using a labelled dataset - posts labelled with the corresponding topic they belong to (as figured out in **??**).



	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT$_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet$_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

Figure 2.8: Comparing RoBERTa and BERT Liu et al. (2019)

Before fine-tuning, the model needs a final output layer to be added. This output layer will be a linear layer with x neurons, where x is the number of topics (so currently 23). The fine-tuning step is trained using categorical cross-entropy loss, and the Adam optimizer. The model is trained for 6 epochs, with a batch size of 32. The model is then evaluated on the test set, achieving an accuracy of $\approx 42\%$. Please refer to **??**for a more detailed analysis of the model's performance.

2.2.5 Robustly Optimised BERT Approach - RoBERTa

RoBERTa is a variant of BERT that is more efficient and robust. It uses the same architecture as BERT, but with some modifications to the pre-training step. RoBERTa only performs the masked language modelling task, and does not perform the next sentence prediction task. On top of this, RoBERTa uses dynamic masking, where during runtime the model randomly masks out words in the sentence, instead of masking out words statically (masking the same words for a given sentence).
TODO: add more details from papers about RoBERTa.
The pre-training of RoBERTa was done on a larger corpus of text than BERT. This improved the model's performance.

The figure above shows how RoBERTa performs better than BERT on most NLP benchmark tests. This led to RoBERTa being tested for this project. RoBERTa was fine-tuned using the same method as BERT, and achieved an accuracy of $\approx 71\%$ on the test set. Evaluation of the model is discussed in **??**.

Chapter 3

Design

In this chapter, we describe the overall design of our solution to the problem identified in Chapter 1, building on work described in Chapter 2.

3.1 Topic Classification

Chapter 2 discussed 4 different models for classifying text into topics: RNN, LSTM, BERT, and RoBERTa. The chapter also used intuition to determine which model would be expected to perform best - RoBERTa. The reason being, RNNs and LSTMs suffer from being unable to understand context of a whole sentence; they are limited to understanding context in a single direction (left to right or right to left). BERT and RoBERTa both use a technique called self-attention to overcome this limitation. From observing results in Liu et al. (2019) RoBERTa outperforms BERT in most cases.
Building off of the RoBERTa model described in Chapter 2, we use the model to classify posts into topics. The design of this model is relatively simple due to the fact we are performing transfer learning on a pre-trained model.

As seen in the diagram, the RoBERTa model has been altered with a new classification layer.

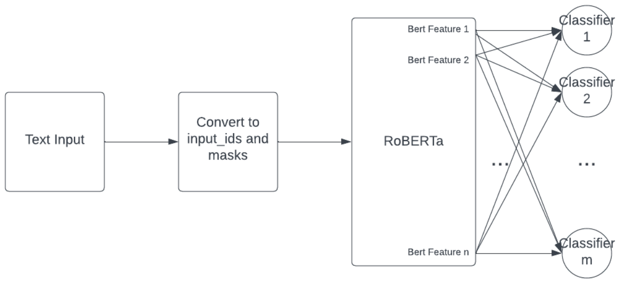


Figure 3.1: RoBERTa model

3.2 Adding Context

3.2.1 What is context? and why is it important?

As discussed in section sec:pythia, context is an important factor in classifying posts. In figure 1.3 there is a tweet that would be hard to classify with the text alone. The section then goes onto show how adding the image that the tweet was posted with (figure **??**) adds more information to the post and makes it easier to classify.

3.2.2 Methods for adding context

There are many ways to add context to a post. Pythia uses Named Entity Recognition (NER) for adding context. In this project the use of Optical Character Recognition, Audio Transcription, and Threads/Retweets are explored.

Named Entity Recognition - NER

Optical Character Recognition - OCR

Images in posts may contain text. This text adds context to the post and can be helpful in classifying the post. Using OCR we can extract any text from an image and use it in addition to the text of the post. This should improve the accuracy of the model when infographics/text based images are used.

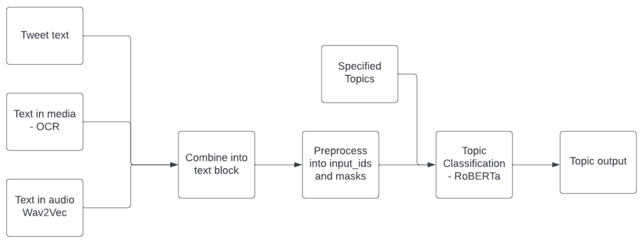


Figure 3.2: Context Aware Model

Audio Transcription - Wav2Vec

Some posts may contain videos. The audio in the videos may contain useful context. Using Wav2Vec we can extract the audio from the video in the form of text. This text can be used in addition to the text of the post. This should improve the accuracy of the model when audio descriptions/explanations are used in a post.

Retweets and Threads

Although a post alone may not contain enough context to classify it, there may be a conversation around the post. This conversation would be found in the retweets and threads of the post. If the retweets and threads are discussing the same topic as the post, then the extra context given by the retweets and threads can be used to help classify the post.

3.2.3 Context Aware Model

The context aware model created in this project will use OCR, Wav2Vec, and retweets/threads to add context to posts. The extra text extracted from these methods will be added to the text of the post. This will be done before the post is classified. The model will then classified with all the additional text.

The RoBERTa classification model will be the same as the one designed figure 3.1. The only difference is the input to the model.

3.3 Python Application

The Python application is made to show users the topics they are interested in and allows them to compare their interests to the social media site.
The first step of design for the application was to decide what features the application should have. This led to the process of requirements analysis.

3.3.1 Requirements Analysis

User Stories

Before building the application, user stories were created to help guide what features are required. The user stories created were:

1. As a user, I want to be able to see what topics I see the most on social media/see what topics I am interested in.
2. As a user, I want to be able to compare what I see on social media to what other people see on social media.
3. As a user, I want to be able to reach out and find posts on topics I am not interested in.
4. As a user, I want to be able to see what topics are trending on social media.

These stories help set up the requirements for the application.

Business Cases

The next step is to outline the business cases (methods of solving the problem) for the application. For this project, All business cases will be made by the author of this dissertation. The business cases created were:

1. **Do Nothing** - This does not improve the problems users face. It is a baseline case.
2. **Chrome Extension** - Allow users to see what percentage of their social media feed is made up of each topic, as well as compare this to live data from the social media platform.
3. **Python Application** - Allow users to see what percentage of their social media feed is made up of each topic, as well as compare this to live data from the social media platform.

The difference between the chrome extension and the python application is the framework they are built in as well as how they are interacted with. The chrome extension would be built in JavaScript, whereas the python application would be built in Python. The chrome extension would be accessible from a chrome browser (via the extensions store), whereas the python application would be run from a python script.
Although, a chrome extension would be more accessible to users and easier to distribute, it would be more difficult to implement as I would have to learn JavaScript and the chrome extension API. I am already familiar with Python and the python libraries used in this project.

Requirements

Using the user stories and chosen business case, the requirements for the application can be determined. **C** - User Requirement **D** - System Requirements

1. Functional Requirements
   (a) C) The user should be able to see what topics they see the most on social media/see what topics they are interested in.
      i. D) The application should be able to get access to the users social media feed via the social media API.
      ii. D) The application should be able to classify the posts in the users social media feed.

   iii. D) The application should be able to calculate the percentage of posts in the users social media feed that are about each topic.
   iv. D) The application should display the top 5 topics the user is interested in as well as their percentage impact on the users social media feed.

   (b) C) The user should be able to compare what they see on social media to what other people see on social media.

   i. D) The application should be able to get access to live posts from the social media API.
   ii. D) The application should be able to classify the live posts from the social media API.
   iii. D) The application should be able to calculate the percentage of live posts that are about each topic.
   iv. D) The application should display the top 5 topics that are trending on social media as well as their percentage impact on the social media platform.
   v. D) The application should display a similarity metric between the users social media feed and the live posts.

   (c) C) The user should be able to reach out and find posts on topics they are not interested in.

   i. D) The application should store all posts that are classified as a topic to be able to search through them.
   ii. D) The application should store alongside the post the top topic it was classified as.
   iii. D) The application should allow the user to search for posts by topic.
   iv. D) The application should display a random selection of posts that are classified as the topic the user searched for.

2. Non-Functional Requirements

   (a) C) The User Interface should be easy to use within 5 minutes of use.

(b) C) The User Interface should not be unresponsive for more than 5 seconds.

(c) C) The User Interface should be suitable for users with no technical experience.

3.3.2 Frontend

User Interface Design

3.3.3 Backend

API Design

Database Design

Chapter 4

Implementation

In this chapter, we describe the implementation of the design we described in Chapter 3. You should **not** describe every line of code in your implementation. Instead, you should focus on the interesting aspects of the implementation: that is, the most challenging parts that would not be obvious to an average Computer Scientist. Include diagrams, short code snippets, etc. for illustration.

4.1 Data Collection

4.2 BERT

4.3 RoBERTa

4.4 Python Application

Chapter 5

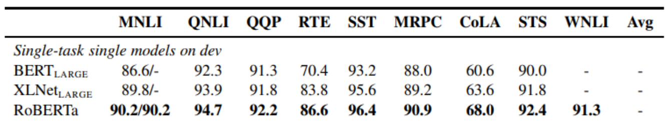
Evaluation

Describe the approaches you have used to evaluate that the solution you have designed in Chapter 3 and executed in Chapter 4 actually solves the problem identified in Chapter 1.

While you can discuss unit testing etc. you have carried here a little bit, that is the minimum. You should present data here and discuss that. This might include *e.g.* performance data you have obtained from benchmarks, survey results, or application telemetry / analytics. Tables and graphs displaying this data are good.

5.1 BERT vs RoBERTa

During development, both BERT and RoBERTa were used to classify the posts into topics. BERT and RoBERTa have been compared before in . In this paper the results shown were:



	MNLI	**QNLI**	**QQP**	**RTE**	**SST**	**MRPC**	**CoLA**	**STS**	**WNLI**	**Avg**
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

Figure 5.1: TODO

The results of the paper show RoBERTa beats BERT in all given Natural Language Processing tasks. In this project similar tests were carried out on the specific topic classification task. The results of the classification were compared to see which model performed better.

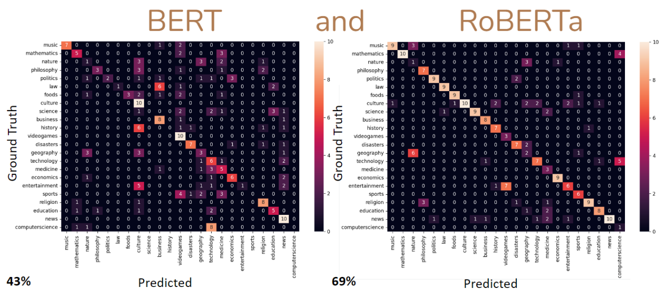


Figure 5.2: BERT vs RoBERTa

Figure 5.2 shows the confusion matrices of BERT and RoBERTa respectively for the test set - the test set was a random 10% sample of the training data. The columns of the confusion matrix represent the predicted labels and the rows represent the ground truth labels. It is obvious from the confusion matrices that BERT did not perform as well as RoBERTa; there are a lot of misclassifications. A misclassification can be identified as a cell in the confusion matrix that is not on the diagonal. Comparing this to the confusion matrix of RoBERTa, it is clear that RoBERTa performed much better. In fact, RoBERTa is 60% more accurate than BERT (43% compared to 69%).

5.2 Principal Component Analysis (PCA) of data

One thing to note in the RoBERTa confusion matrix in Figure 5.2 is that the model has high concentration of misclassifications; the model makes the same misclassification often. After noticing this trend, it was believed that the topics that were commonly misclassified were similar to each other - could be grouped together.

Principal Component Analysis (PCA) was used to reduce the dimensionality of the data and to allow for visualisation of the data. PCA was performed on the output features of RoBERTa.

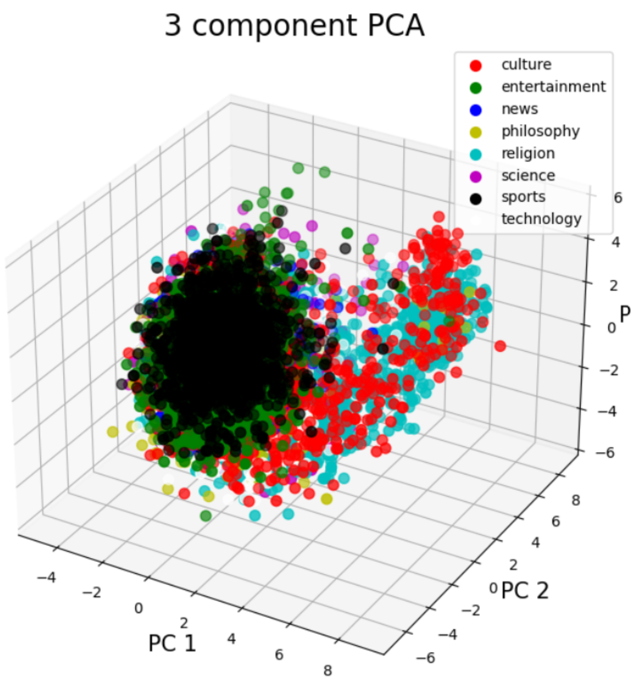


Figure 5.3: PCA of data

Figure 5.3 shows the PCA of the data. The data was reduced to 3 dimensions using PCA. The data was then plotted on a 3D graph. The colours of the points represent the topics (of which some are identified in the legend). The graph is not easy to interpret; the only visible trend is that ‘Religion’ and ‘Culture’ seem to be very dissimilar to the other topics.
To help interpret the data, the analysis was performed on 2 topics at a time. This time the data was reduced to 2 dimensions using PCA. The data was then plotted on a 2D graph.

5.2.1 Findings from PCA

In this section, the notable findings from the PCA are discussed.

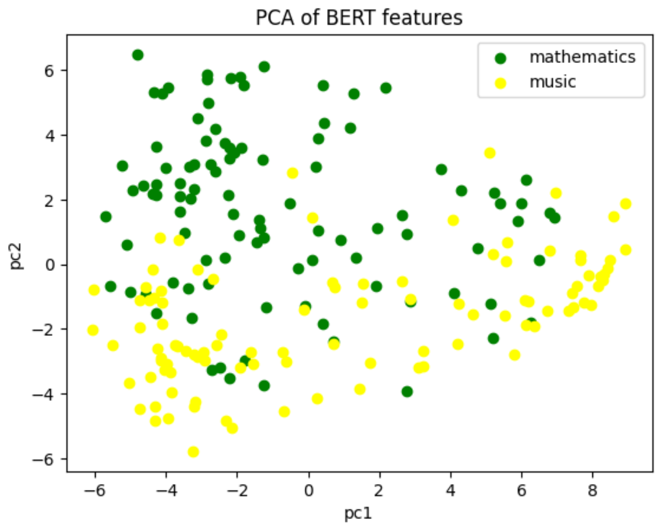


Figure 5.4: Mathematics vs Music

Mathematics vs Music

This comparison was made to act as a baseline for the rest of the comparisons. The hypothesis was that these topics would be dissimilar to each other.

Figure 5.4 shows the PCA of the data for the 'Mathematics' and 'Music' topics. There is a clear separation between the two topics. Although the data is not linearly separable, there is a strong trend that the data points of the 'Mathematics' topic are on the bottom of the graph and the data points of the 'Music' topic are on the top of the graph. Using a perceptron classifier on the 2 principal components, led to an accuracy of 73%. This is a strong indicator that the 2 topics are dissimilar to each other.

Geography vs Nature

Looking back at the confusion matrix of RoBERTa in Figure 5.2, the model predicts 'Geography' instead of 'Nature' 6 times out of 10. This is a very high concentration of misclassifications. The hypothesis was that the topics are very similar.

The figure shows the PCA of the data for the 'Geography' and 'Nature' top-

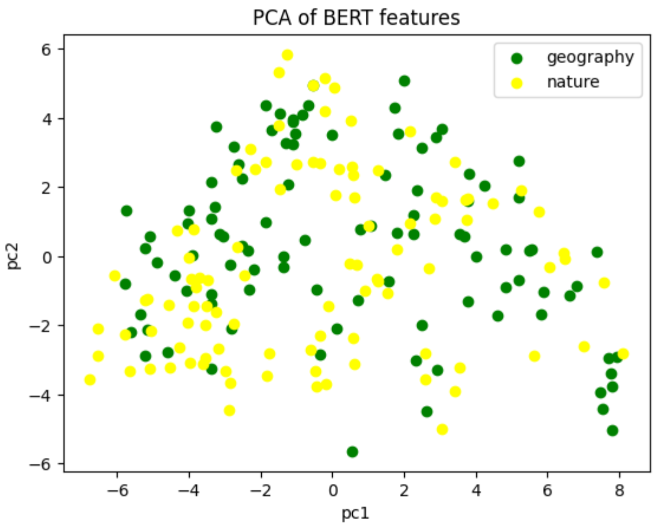


Figure 5.5: Geography vs Nature

ics. In comparison to Figure 5.4, there is no clear separation between the two topics. In fact, visually, the two topics seem to be almost identical. Using a perceptron classifier on the 2 principal components, led to an accuracy of 52%. This is a strong indicator that the 2 topics are similar; the fact the classifier is only a minor improvement on random guessing shows that the 2 topics are similar.
From this finding it was concluded that the model was not able to distinguish between the two topics. This led to the merging of the 'Geography' and 'Nature' topics into a single topic. It was decided this topic would be called 'Geography'.

Technology vs Computer Science

Similarly to the previous comparison, the confusion matrix in Figure 5.2 shows that the model predicts 'Technology' instead of 'Computer Science' 5 times out of 10. The hypothesis was that the topics are very similar. However, the results from the perceptron classifier on the 2 principal components showed a very high accuracy of (TODO). This is not the result expected and caused confusion as to why the model was unable to distinguish between the two topics.

To further investigate this, the data was manually inspected to attempt to reach a new hypothesis. This led to the discovery of the unbalanced nature of the data. The 'Technology' topic had 10 times more data points than the 'Computer Science' topic. This answers why the perceptron classifier performed so well but the model was unable to distinguish between the two topics.
The inbalance of the data was a surprise as the script used to scrape the data was designed to scrape an equal number of data points. It was found that the search query for the 'Computer Science' topic yielded no results on wikipedia leading to a small number of data points.
It was decided to remove the computer science topic from the dataset due to the limited dataset.

News

TODO

5.3 Context

5.3.1 Media - Images and Videos

After implementing OCR and Wav2Vec, the model was tested on a set of 60 hand picked posts (3 for each topic). The updated context aware model was compared to the original model.

Figure 5.6 shows the confusion matrix of the original model. Figure 5.7 shows the confusion matrix of the updated context aware model. The addition of the media context improved the accuracy of the model by 10%. This is a notable increase in accuracy. However, due to the nature of gathering this test data, it must be noted some form of bias is present. The test data was hand picked, so some unconscious bias may have been introduced to provide posts that would be hard to classify without the media, but easy to classify with the media. Although, this could be the case this test does show that the context from media can still be useful in classifying posts.

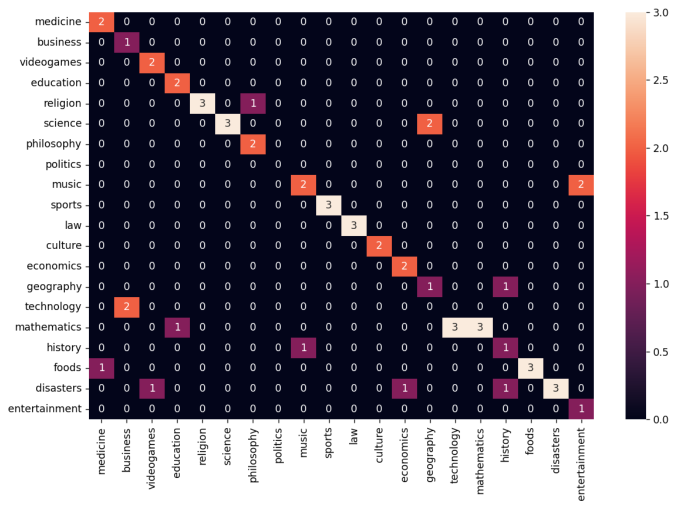


Figure 5.6: Confusion Matrix without Media Context

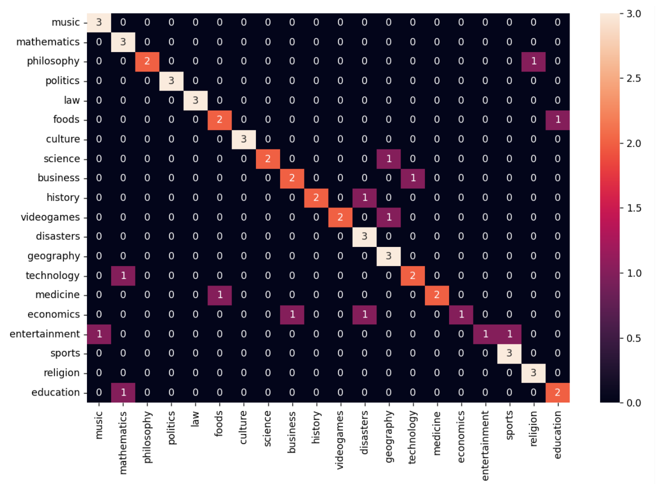


Figure 5.7: Confusion Matrix with Media Context

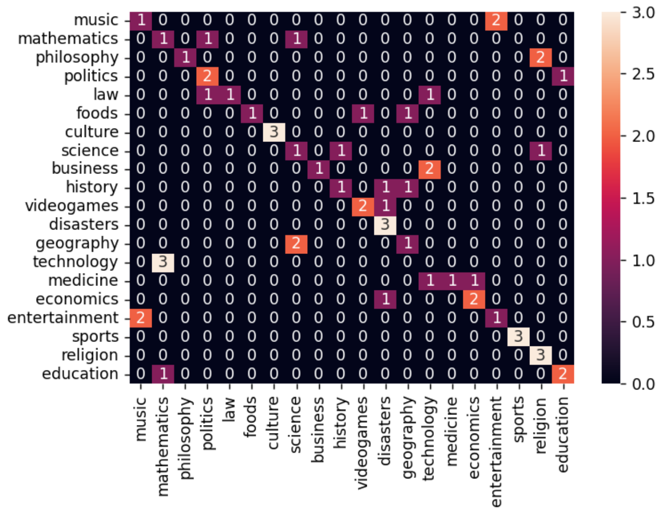


Figure 5.8: Confusion Matrix without Thread/Retweet Context

5.3.2 Retweets and Threads

Separately to implementing OCR and Wave2Vec, another model was created that added the context of retweets and threads. This model was made separately to allow us to see the performance impact of the retweet and thread context. If we had tested them together, we would have seen an improvement but would not have been able to identify the impact of each feature.
The model was tested on a set of 60 hand picked posts, that were different to the posts used in the previous test. The reason behind this is simply the API used to access the tweets was unable to access retweets and threads from posts older than 7 days - The posts used in the previous test were all older than 7 days when this test was performed.

Figure 5.8 shows the confusion matrix of the original model. Figure 5.9 shows the confusion matrix of the thread/retweet context aware model. Similarly to the previous test, the addition of the thread/retweet context improved the accuracy of the model by 10%. Again, this is a notable increase in accuracy although the same bias as the previous test is present.

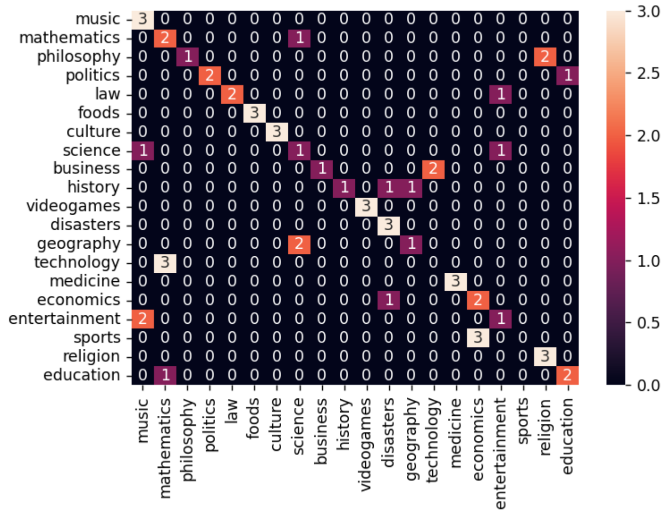


Figure 5.9: Confusion Matrix with Thread/Retweet Context

5.3.3 Context Aware Conclusion

The context aware models both improved the accuracy of the model by 10%. This shows how this extra context can be useful in classifying posts. However, there are cases where the extra context is not useful. For example, some videos are paired with songs for their audio. This audio is not related to the topic of the post. Which could cause the model to misclassify the post. In its current state the model is not able to identify whether the audio is related to the topic of the post or not.

This project shows that the context of the post can be useful in classifying posts. However, due to the naive approach taken in this project, the context is not always useful. In Chapter 6 we will discuss how this context can be used in a more sophisticated way to improve the accuracy of the model.

Chapter 6

Conclusions

The project aimed to:

- Classify social media posts
- Quantify and compare a users social media feed to the rest of the social media site to identify interests
- Build a user interface that allows users to discover what their social media feed says about their interests and how they compare to the rest of the social media site
- Build a user interface to allow users to find posts that are dissimilar to the posts they are shown

It is clear that all of these aims were met. RoBERTa was fine-tuned for the topic classification task. The fine-tuning was done using labelled data from Reddit and Wikipedia. Quantification of a set of tweets was done using the mean of the probabilistic outputs from our model. A user interface was built that is able to show a comparison of topics between a users social media feed and the rest of the social media site. The user interface also allows users to discover posts that are dissimilar to the posts they are shown.
From this, we can conclude that the project was a success.

6.1 Future work

6.1.1 Advanced Context Input

TODO: discuss how we can include context from images and videos better using other Transformer models. Can also include other aspects of context including: Author, data from linked articles, etc.

6.1.2 Chrome Extension

TODO: take the current Python User interface and turn it into a Chrome extension.

6.1.3 Bias Analysis

TODO: develop a definition of bias, and then explain how we can further our model to analyse bias in social media posts.

Bibliography

Husby, Stephanie & Barbosa, Denilson. Topic classification of blog posts using distant supervision. In *Proceedings of the Workshop on Semantic Analysis in Social Media*, pages 28–36, 2012.

Litou, Iouliana & Kalogeraki, Vana. Pythia: A system for online topic discovery of social media posts. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2497–2500, 2017. doi: 10.1109/ICDCS.2017.289.

Liu, Yinhan & Ott, Myle & Goyal, Naman & Du, Jingfei & Joshi, Mandar & Chen, Danqi & Levy, Omer & Lewis, Mike & Zettlemoyer, Luke & Stoyanov, Veselin. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL `http://arxiv.org/abs/1907.11692`.

Peng, Xian & Han, Chengyang & Ouyang, Fan & Liu, Zhi. Topic tracking model for analyzing student-generated posts in spoc discussion forums. *International Journal of Educational Technology in Higher Education*, 17 (1):35, Sep 2020. ISSN 2365-9440. doi: 10.1186/s41239-020-00211-4. URL `https://doi.org/10.1186/s41239-020-00211-4`.

Snow, Rion & Jurafsky, Daniel & Ng, Andrew. Learning syntactic patterns for automatic hypernym discovery. *Advances in neural information processing systems*, 17, 2004.