Tweet Classification for OSR Topics

Submitted on August 24, 2023
in partial fulfillment of
the conditions for the award of the degree
MSc Scientific and Data Intensive Computing.

Longwen Hu
22071908

Department of Physics & Astronomy
University College London

I, Longwen Hu, confirm that the work presented in this dissertation is my own. Where information has been derived from other sources, I confirm that this has been indicated in the dissertation.

Acknowledgements

Initially, I would like to express my gratitude to my supervisor Theo Jolliffe for his great support on this project. He gave me the constructive guidance on weekly meetings and emails during the entire project. The completion of this project is inseparable from his insight and experienced knowledge.

Secondly, my great thank are also extended to co-supervisor, Nikos Nikolaou. It is his insight and detailed guidance help this project have further progress.

Finally, I also thank my family and my friends for their love and support the whole time.

Abstract

Twitter content has been served as an increasingly critical approach to spread and access information. Due to the short length, informal form and ambiguous content of tweets, tweet classification tasks are still challenging. In this project, we approach this task by topic modeling and topic classification. For topic modeling, we experiment on both probabilistic methods and transformer-based models. For topic classification, we apply compression-based methods and fine-tune pre-trained language models. Furthermore, we present sentence-level analysis when evaluating the prediction results from both two types of approaches and collect a GPT generated dataset for both fine-tuning and semantic analysis. Experiments show that topic classification methods are more accurate and reliable, with a simple Gzip + KNN classifier, it can reach 60% accuracy. Further experiments on ensemble model of the combination of DistilBert model, BERT-base model and BERT-large model reaches approximately 95% accuracy, which is the best-performed model among all and is suggested for deployment considerations. Moreover, further analysis demonstrates that tweets tend to be in multiple classes, multi-label classification is suggested as the main future direction for this project. The repository of the code is available at https://github.com/uclHU/Tweet-Classification-for-OSR-Topics.

Contents

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers

DAN Deep Average Network

DeBERTa Decoding-enhanced BERT with disentangled attention

DistilBert Distilled BERT

GPT Generative Pre-training Transformer

HDBSCAN Hierarchical Density-Based Spatial Clustering of Applications with Noise

KNN K Nearest Neighborhood

LDA Latent Dirichlet Allocation

LSA Latent Semantic Analysis

LSTM Long Short-Term Memory networks

MLM Masked Language Modeling

MLPs Multi-Layer Perceptrons

NCD Normalized Compression Distance

NN Neural Network

OSR Office for Statistics Regulation

TC Tweet Classification

TF-IDF Term Frequency - Inverse Document Frequency

UMAP Uniform Manifold Approximation and Projection

ZSC Zero-Shot Classification

Chapter 1

Introduction

Twitter, as a social media platform, enables a wide and rapid way to share information. It is estimated that there are around 500 million tweets posted everyday [4]. As it is considerably tied with people's daily life, it has been served as a critical approach to spread and access information, including statistical information from all sources. Tweet Classification (TC) serves as an essential task to analyze user's behavior and attitude, which covers the tasks for stance detection, hate speech detection and sentimental analysis as well as category classification [5]. Due to the various forms of content and the short length for each tweets, classification task on tweets is not a trivial task, especially for multi-class classification [6].

According to Antypas et al. [7], tweet classification task can be achieved through two ways, topic modeling and topic classification. For topic modeling, it is the study of words distribution for each topic and the topic distribution for each under classified documents. For a successful classification using topic modeling, it requires the assumption of the presence of all topics and the terms in one cluster to be capable of representing one topic. In the past twenty years, various models have been proposed and studied, including traditional models such as Latent Dirichlet Allocation (LDA) proposed by Blei et al. [1] in 2003 and the most recent large language model-based approach, like BERTopic [2]. As the nature of unsupervised learning, it does not suffer from requiring manual annotation. However, it heavily depends on ad-hoc analysis for understanding each cluster, which brings limits to its generalization for a new set of topics. For topic classification, it is seen as a supervised learning task. Compared to topic modeling, it suffers from the requirement on annotations, but adds potential interpretability to the model.

Existing studies on tweet classification mainly focus on 2 or 3 class tasks, such as sentiment analysis and stance detection, as it serves to provide sufficient information for user behavior analysis. However, this project focuses on tweets classification on 9 topics, aiming at providing a comprehensive review on how different models perform, from tradition methods, like K Nearest Neighborhood (KNN), to the state of the art models, such as Bidirectional Encoder Representations from Transformers (BERT) and BERTopic. In addition, this project covers the evaluation on a wide range of candidate models with a detailed discussion on the effect of data preprocessing and model choosing. Moreover, we also introduce comparison with the models trained on Generative Pre-training Transformer (GPT) generated dataset which a sentence level analysis on predictions' reasonableness. Finally, we suggest an ensemble model, which is a combination on three language models, for deployment consideration.

The rest of the paper is structured as follows. Chapter 2 covers the illustrations as well

as discussions of the existing schemes for tweet classification task. Chapter 3 presents the implementation details of this project including dataset analysis, data preprocessing, and the architecture of the candidate models. Chapter 4 compares and analyzes the performance of each model. Chapter 5 concludes the project work and highlights future study directions.

Chapter 2

Background & Related Work

This chapter offers a detailed review on existing methods for solving text classification tasks. These methods are categorized in two groups, topic modeling methods and topic classification methods.

2.1 Topic Modeling

Latent Dirichlet Allocation, also known as LDA, was first introduced in 2003 by Blei et al. [1] to generate a mixture of latent topic distribution for each document, where each topic is a distribution over words. The general process for calculating the topic distribution is achieved by repeatedly calculating the probability over Dirichlet distribution following the formula:

$$p(\theta, z, w, |\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{2.1}$$

where α sets the prior on the per-document topic distribution while β sets the Dirichlet prior on the per-topic word distribution. Figure 2.1 is the graphical representation, where each rectangle box denotes the replicated process of calculation. Moreover, θ, z, w in both formula and graphical representation stand for the topic distribution for each document, the topic for each word in the same document from θ as well as the observed words respectively. As one of the most popular algorithms, LDA has shown its potential and strength in downstream tasks, like text classification. However, the performance can be largely affected by the document length, especially for short and unstructured text like tweet [8]. Moreover, LDA is limited by the assumption that any potential word should be covered by the corpus already, where Twitter data often involves evolving text, making it less preferred for this project.

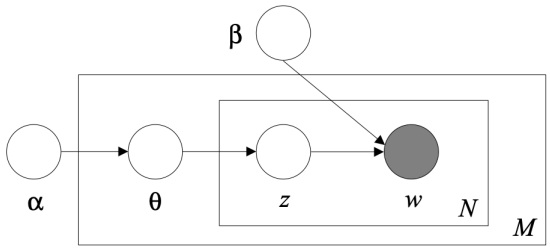


Figure 2.1: Graphical Model Representation of LDA [1]

In addition to traditional LDA, several variations have been proposed to be applied on social media text tasks. Rosen-Zvi et al. [9] combined author-topic modeling with LDA, which includes an extra step to use topic-based representation for both the content of documents and the interests of authors. It is claimed that by introducing extra document-level representation, the adapted model is able to provide significantly improved predictive power in terms of perplexity. Moreover, Steinskog et al. [10] applied pooling techniques to aggregate similar tweets into individual documents in order to alleviate the disadvantage of short length for a single tweet. More specifically, they applied Rosen-Zvi et al. [9] author-topic model and analyzed the co-occurrence of hashtag to improve the overall topic coherence.

However, the above LDA-based methods suffer from a valid evaluation on text classification task. As stated by Rosen-Zvi et al. [9], it often requires human judgements in topic evaluation. In addition, qualitative analysis is also a hard task, as LDA-based methods are lack of topic interpretability, especially with a large number of topics.

Compared to traditional LDA approaches, Grootendorst [2] proposed a topic clustering algorithm based on pre-trained language model. As shown in Figure 2.2, the whole training process includes three stages. Firstly, a pre-trained language model extracts an embedding representation of documents. The sentence embedding normally is represented as a vector with hundreds of elements. For example, the output BERT-base pre-trained model is a 768 vector. In the second stage, the extracted embedding is processed by UMAP [11], which is a dimension reduction algorithm. The reduced embeddings are further clustered by HDBSCAN [12] to group similar documents and each cluster is assigned with one topic. In order to generate topic-word distributions for each cluster, a class-level TF-IDF is computed using:

$$W_{t,c} = tf_{t,c} \times \log(1 + \frac{A}{tf_t}) \tag{2.2}$$

Where $tf_{t,c}$ represents the term frequency of term t in class c and A is the average number of words per class. Compared to the classic TF-IDF calculation [13], it adds an additional one to the logarithm to keep the result positive. The calculation of maximal marginal relevance [14] is designed to eliminate the redundant words in each cluster. Although it is not applied in the original paper, we include this extra step to improve the overall interpretation for each topic.

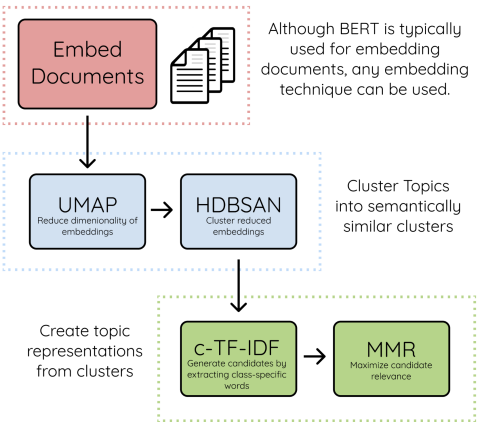


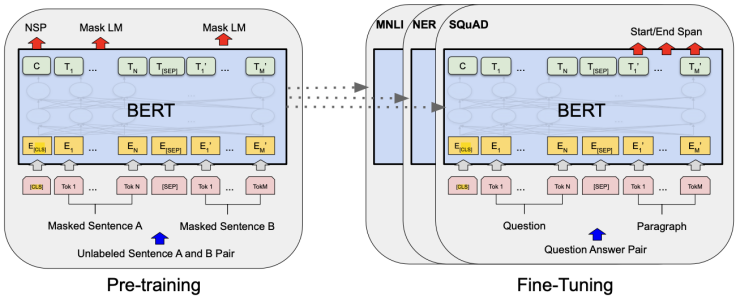
Figure 2.2: BERTopic Model Training Process [2]

2.2 Topic Classification

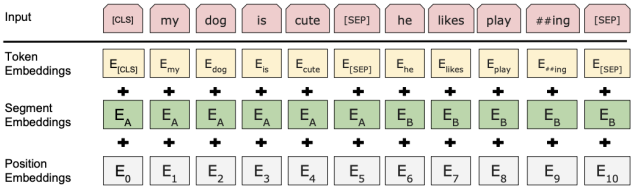
Compared to traditional classification tasks, topic classification involves an extra step for feature extraction. For some simple methods like feed-forward neural networks, they view text as a bag of words, where each word is represented by vector embeddings like word2vec [15]. Deep Average Network (DAN) was proposed in 2015 by Iyyer et al. [16], it applies word2vec to extract vector embeddings for each word and take the average sum of all words embeddings in a sentence. The final vector is then passed to the input layer of a Multi-Layer Perceptrons (MLPs). For a simple method like this, it was able to reach the state of the art performance at that time. Compared to DAN, Jiang et al. [17] propose a combination method of compressor with k nearest neighbor (KNN) classifier, which has an even simpler architecture. They apply gzip as the compression algorithm combined with Normalized Compression Distance (NCD) [18] to compute the distance between instances in KNN. The general intuition follows that compressors are good at capturing regularity and the difference in regularity denotes those objects are from different categories. The performance of such a model is compared to deep neural network models like LSTM, and pre-trained language models on seven datasets for text classification tasks. The model can reach the state of the art performance and even outperform BERT on out-of-distribution dataset.

As inspired by transformer architecture [19] and masked language model [20], Devlin et al. [3] propose Bidirectional Encoder Representations from Transformers (BERT) to alleviate the limitation of uni-direction. BERT assumes a bidirectional model can produce a more comprehensive representation than a single left-to-right or right-to-left model. It is achieved by introducing masked language modeling (MLM). More specifically, a proportion of words in each sentence are replaced by '[MASK]', and then the model is trained to predict those masked tokens. Furthermore, it also considers the downstream tasks on sentence pair-wise relationships like question answering and classification. Hence, BERT also includes special tokens like '[SEP]' and '[CLS]', representing sentence separation and sentence classification. The BERT framework includes model pre-training and fine-tuning two steps. As presented in Figure 2.3(a), the model applies MLM on unlabeled corpus, where the sentence representation includes special tokens. The corpus consists of sentences from BooksCorpus (800M words) and English Wikipedia (2,500M words). In addition,

as shown in Figure 2.3(b), the input representation also includes position embeddings, which alleviate the limits on words inherent understanding. The fine-tuning procedure is further trained on pre-trained parameters and is designed to fit different downstream tasks, like language inference and text classification.



(a) Overall Pre-training and Fine-tuning Procedures



(b) BERT Input Representation

Figure 2.3: BERT Architecture [3]

BERT is evaluated on 11 NLP tasks in the original paper and it can perform competitively on each task, especially for BERT-large model. However, both BERT-base and BERT-large models are large in size, containing 110M parameters and 340M parameters respectively. Sanh et al. [21] propose a distilled version of BERT model with 60% in parameter size, but retains 97% of its language understanding capabilities. It applies knowledge distillation, which is a compression technique aiming at training a student model to reproduce the behavior of a teacher model. In DistilBert model architecture, the student model is initialized from the teacher BERT model by taking one layer out of two. Moreover, the training loss for the student model combines a distillation loss and supervised training loss. A distillation loss measures the difference between the prediction probability of the student model and the teacher model, which is computed as:

$$L_d = \sum_i t_i \times \log(s_i) \tag{2.3}$$

where t_i and s_i are the estimated probability from the teacher model and the student model respectively. In addition, Sanh et al. [21] experiment on different variations of training loss, and find that including cosine embedding loss helps align the directions of the student and teacher hidden states vectors.

In this project, we experiment on most of the above models. A simple model, like Gzip + KNN, is considered to provide a baseline performance. As shown in many studies, current the state of the art performance is retained by fine-tuning pre-trained language models, so the project mainly considers fine-tuning on BERT based models, instead of deep neural

networks. Moreover, DistilBert is also fine-tuned and evaluated in order to showcase the difference in performance under constrained computational training.

Chapter 3

Methodology

3.1 Dataset Analysis

In this project, there are in total two single-label datasets. One dataset is from OSR, which is collected from Twitter API and labeled by keyword-based model. The dataset contains 10288 tweets with 20 topics. The other one is generated by GPT. The GPT generated set is labeled in 9 classes following the given number of topics in project brief, with 1000 tweets for each class (details on this dataset are illustrated in Section 3.3).

3.1.1 Imbalance

Figure 3.1 presents the topic distribution of the dataset. It is evident that the dataset is heavily imbalanced, as the size for some topics like 'Ukraine' and 'Statistical Literacy' only contains 9 instances. We consider two methods to solve the imbalance problem.

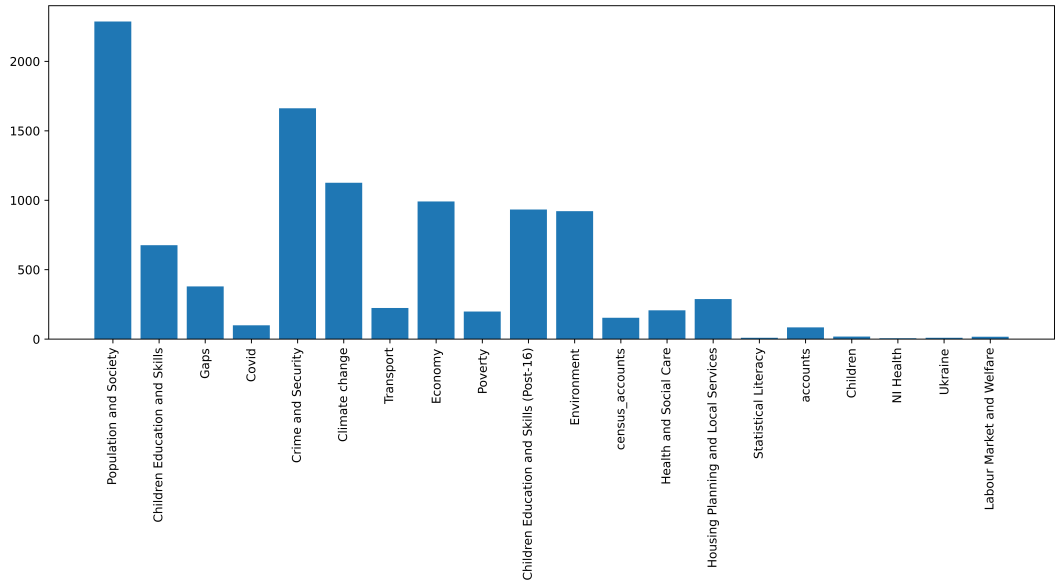


Figure 3.1: Original Dataset Topic Distribution

Regroup topics

As given in the project brief, there are in total 9 classes. Regroup the topics from 20 to 9 can relatively reduce the gap on topics size, but it is still imbalanced as shown in Figure 3.2. This method is combined with other imbalance processing approaches, like calculating inverse weighted loss and oversampling.

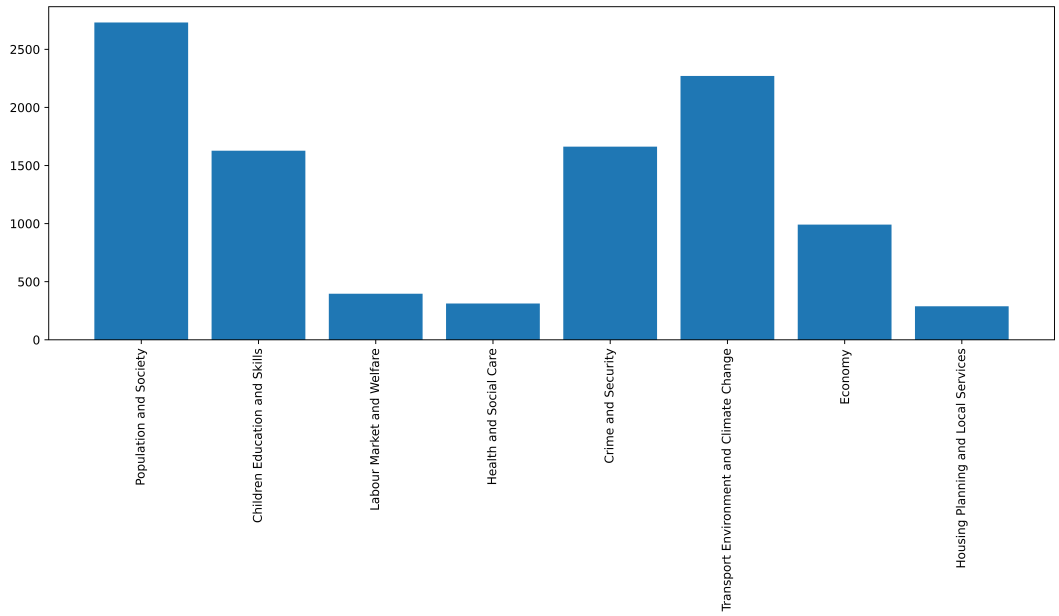


Figure 3.2: Regrouped Dataset Topic Distribution

Inverse weighted loss

In an imbalanced dataset, inverse weighted loss refers to calculating the distribution of classes in the training data and assigning examples of each class a weight that is inversely proportional to the frequency of that class in the loss function. Note that to fine-tune a pre-trained model, Cross Entropy is applied as the loss function, written as

$$L(\hat{y}, y) = -\sum_{k}^{K} y^{(k)} \log \hat{y}^{k} \tag{3.1}$$

When considering the weight, the equation is transformed to:

$$L(\hat{y}, y) = -\sum_{k}^{K} w^{(k)} y^{(k)} \log \hat{y}^{k} \tag{3.2}$$

$$w^{(k)} = \frac{S_t}{N_k} \tag{3.3}$$

where S_t, N_K, $w^{(k)}$ represent the total number of instances, the number of instances in class k and the inverse frequency weight of class k respectively. As we also apply instance based methods, like KNN, the calculation of inverse class frequency weighted distance is

considered to be:

$$D_i = w^k E_{i,j} \tag{3.4}$$

or

$$D_i = E_{i,j} \log(1 + w^{(k)}) \tag{3.5}$$

where $E_{i,j}$ represents the Euclidean distance between instance i and j. Both Equation (3.4) and (3.5) are examined in our experiments (detailed discussion is covered in Section 3.4.2).

3.1.2 Ground truth

The original dataset is provided with label being predicted by a keyword-based model. In this project, two datasets are used in model training. One of them uses the label from keyword-based model as the ground truth and the other considers the label from a GPT. For the GPT generated set, we use the prompt, such as 'Produce a tweet about healthcare containing statistical information', where the generated tweet is then labeled as 'Healthcare'. The GPT generated dataset is further utilized to evaluate the correctness and reasonableness of keyword-based model.

3.1.3 Topics

The original data set contains 20 topics, which is further merged into 8, according to the target topic given in the OSR brief. Table 3.1 represents how 20 labels are matched to the target labels. It can be seen in Table 3.1 that a proportion of labels have a direct match to the merged ones. For the remaining topics, we randomly select 10 to 20 tweets based on their original size, then each tweet is manually annotated. The final transformed label is the one has the maximum number. In addition to transform topics by group, we also consider regroup at tweet-level, which is to use the fine-tuned model trained by GPT generated dataset to classify the topics without a direct match.

Original topic	Merged topic
Population and Society	Population and Society
Children Education and Skills	Children, Education and Skills
Gaps	Labour Market and Welfare
Covid	Health and Social Care
Crime and Security	Crime and Security
Climate change	Transport, Environment and Climate Change
Transport	Transport, Environment and Climate Change
Economy	Economy
Poverty	Population and Society
Children Education and Skills (Post-16)	Children, Education and Skills
Environment	Transport, Environment and Climate Change
census_accounts	Population and Society
Health and Social Care	Health and Social Care
Housing Planning and Local Services	Housing, Planning and Local Services
Statistical Literacy	Population and Society
accounts	Population and Society
Children	Children, Education and Skills
NI Health	Health and Social Care
Labour Market and Welfare	Labour Market and Welfare

Table 3.1: Topic Transformation Table

3.2 Data Preprocessing

For a Natural Language Processing(NLP) task, training a machine learning model requires the step of encoding a text feature into a feature that can be recognized by computer. In this section, it covers data processing steps on text in our dataset.

3.2.1 Tokenization

Tokenization refers to the stage of splitting text, like sentences or paragraph into small units, like words or characters. In the project, we consider using word-level tokenizer, which separates each sentence by words. More specifically, we consider the downstream task is to fine-tuning a pre-trained model, like BERT. The tokenizer we choose for this project include two other stages. After splitting into individual words, each words is mapped into a pre-set vocabulary, where a unique id is given to each words. Moreover, for each sentence, there are two additional tokens added at both front and back, labeled as '[CLS]' and '[SEP]', where '[CLS]' representing the start of each sentence and is further used to classify the label of corresponding sentence in downstream classification tasks, and '[SEP]' is the special separator token used for separating sentence pair. Sentence pair is used in tasks like natural language inference or question answering, as we use BERT pre-trained model, such type of token is also included. In addition to a mapped id, a vector of integers representing the attention mask state is also generated. The attention mask vector is used to represent whether corresponding token can be replaced by a special

mask token '[MASK]'. As the dataset is processed in batches, there are different length for each sentence. For the shorter ones, they are padded to the longest length and the padded tokens are labeled to be 0, denoting they cannot be replaced by '[MASK]', where the remaining elements in the vector are 1s. A traditional BERT-based tokenizer also include a vector of token type id. It is a vector to identify the tokens in two separate sentences, where 0s stand for the tokens in the first sentence and 1s for the tokens in the second sentence. As our project is a classification task, type identification is not required. One possible consideration is to use DistilBert model, which does not require type id identification. It is designed to perform a faster learning with a slight compromise to score.

3.2.2 Token Removal

In this project, token removal refers to the process of removing common words and common tweets symbols, where different process fits to different model training.

Removal of hashtags, username, emojis and url

For tweet content, it is common to have hashtags, username, emojis as well as url. According to Sun et al.[22], hashtag or url may contain certain information in text classification. Hashtags play a critical role in web content search, and have been observed to help categorize ambiguous tweets [5]. URL consists of a protocol and an IP address, where the IP address is also known as domain name, which may contain topic information. For example, a tweet includes "http://sports.sohu.com" can be categorized to "sport" class. However, not all the urls are using a formal domain name as the IP address. In our tweet dataset, there exists examples like, "https://t.co/S7jebI3fp2" which does not add any information for classification. Therefore, we consider both cases in data preprocessing phase, the dataset with hashtags and url, and the dataset without them.

Removal of stop words

Stop words in NLP refer to the common words, such as"the", "a", "and". For fine-tuning a pre-trained model, these words are kept to improve semantic understanding, while can be removed for topic modeling task and compression-based model.

3.2.3 Lemmatization

Lemmatization is the process of reducing the inflectional form of a word to a common base. For example, after terms lemmatization, "corpora" is mapped to "corpus" and "better" is mapped to "good". According to May et al. [23], lemmatization is a critical factor for their experiments on improving topic modeling interpretability. Schofield and Mimno [24] suggest that lemmatization also plays the role as grammar correction in NLP tasks. Same to stop words removal, the dataset with lemmatization is used for topic

modeling as well as in compression-based model.

3.3 Dataset generated from GPT

As discussed in former section, the datasets in this project include the ground truth from keyword-based model as well as from GPT. The dataset generated from GPT is used as a comparison dataset for measuring the reasonableness of prediction from keyword-based model, as well as helping regrouping imbalanced classes. The dataset is generated by calling GPT 3.5-turbo API with the prompt, such as "Create a two-column CSV of 100 tweets about Population and Society. These tweets should contain statistical information." We generate 1000 tweets for each topic using the similar prompt. List below presents one of the example instance from each topic.

- Population and Society: The world's population is projected to reach 9.9 billion by 2050. #population #society
- Health and Social Care: The global average life expectancy has increased from 56 to 72 years in the last century.
- Crime and Security: According to the FBI,there were 1.2 million violent crimes reported in the US in 2019.
- Children Education and Skills: 2 out of 3 children cannot read proficiently by the end of 4th grade,leading to long-term educational disadvantages. #ChildrenEducationAndSkills
- Business Trade and International Development: The United Kingdom is negotiating new trade agreements with countries around the world following its departure from the EU.
- Housing Planning and Local Services: Efforts to increase the number of community gardens have resulted in a 20% decrease in food insecurity rates.
- Economy: Foreign direct investment outflows reached a record high of $30 billion.
- Labour Market and Welfare: Minimum wage violations resulted in fines totaling £1.2 million in the last year.
- Transport Environment and Climate Change: The transportation sector consumes about 55% of all oil used globally. It's time to promote renewable energy sources and reduce our dependence on fossil fuels! #RenewableTransportation

3.4 Model Training

3.4.1 Topic Modeling

For topic modeling, we use both probabilistic methods and transformer-based models to create dense clusters.

LDA

As a probabilistic method, the preprocessing process for LDA includes removal of stop words, punctuation and all the special characters to reduce the corpus size. In addition, the final corpus also include bigram words, which are two adjacent words connected through an underline. Examples for bigram in the corpus include "new_york", "id_number", "united_state". For fine-tuning on the hyper-parameter k, we grid search on the list of 10 to 100 with step size of 5, and each model is measured by computing corresponding perplexity and coherence score.

BERTopic

As a transformer-based model, the preprocessing data only exclude special words, like "@username". The model applies "paraphrase-MiniLM-L6-v2", which is an English sentence-transformer model, mapping each sentence to a 384 dimensional dense vector space. The embedded representation is processed using UMAP[11] to reduce dimension and HDBSCAN [12] to cluster the embeddings and generate semantically similar groups. To find the label for each cluster, we use the prediction from key-word based model as ground truth, and each cluster is labeled by the topic with the maximum appearance frequency.

3.4.2 Topic Classification

Topic classification can be seen as involving two phases, the feature extraction phase and the model tuning phase. In this project, we apply a simple feature extraction method like Gzip compression method combined with KNN to provide a performance baseline. This is then compared to zero-shot models as well as fine-tuned models, like BERT.

Gzip + KNN

Gzip is one type of compression algorithm, allowing regularity extraction [25]. In an NLP task, applying Gzip to the text dataset expects the similar content to be compressed to the same length. This is achieved by calculating the Normalized Compression Distance(NCD):

$$NCD = \frac{C(xy) - min\{C(x), C(y)\}}{max\{C(x, C(y)\}} \tag{3.6}$$

where $C(xy)$ is the compressed length of x concatenated with y. By using above formula, it is expected that when text x is similar to y, the compressed length $C(xy)$ is close to $min\{C(x), C(y)\}$, hence it generates a small value of normalized compression distance. In Jiang et al.[17]'s work, the k for KNN is set to 2 and when there is a tie in the neighborhood set, it always chooses the correct label. In our project, as the dataset is imbalanced, we also consider using 1-NN model as to eliminate the potential incorrect classification caused by the distribution. Moreover, the distance to each training instance

is calculated by original NCD multiplied with a weighted value, where the value is the distribution of corresponding class. Considering the potential of overfitting, the hyper-parameter of k is set to range from 1 to 8.

Zero-shot classification model

Zero-shot Classification(ZSC) model is a pre-trained model used to classify new examples from previously unseen classes. According to Alhoshan et al. [26], ZSC model is able to achieve an acceptable performance on classification task for the ones that are lack of data. In their work, ZSC model managed to achieve 82% recall and F1-score using 60 instances for a binary classification task. Inspired by their work, ZSC model is considered to provide a baseline on performance to evaluate further fine-tuned models. In the project, we choose fine-tuned DeBERTa model [27] with a cutting edge performance on tasksource [28] from HuggingFace.

BERT model

To fine-tune a BERT model for a classification task, it only requires an additional classifier layer, where the input is a vector of encoded sentence representation and the output is connected to a softmax layer. The output of the softmax layer represents the desired probabilities of each class and is further used to compute cross entropy loss using the Equation (3.1). Since our dataset is imbalanced, we introduce inverse weights when calculating the loss as presented in Equation (3.2) and (3.3). In our project, we also fine-tune on the hyper-parameters of BERT-base model in terms of batch size and learning rate. Based on suggested settings from the original manuscript [3], we grid search the choice of batch size 16 or 32 with the list of learning rate, [5e-5, 3e-5, 2e-5]. Due to computation power, the grid search on hyper-parameters for BERT-large is not included, but we fine-tuned one model with batch size set to 16 and learning rate to be 2e-5. The training epoch for both base and large models is set to 10, since the validation loss starts increasing after 4 or 5 epochs, and the training loss tends to be quite small (less than 0.1).

DistilBert model

For DistilBert model, we apply the same setting for loss calculation and grid search for hyper-parameter tuning. Moreover, we also fine-tune a DistilBert model using GPT generated dataset which is further used to analyze reasonableness of keyword-based model.

Chapter 4

Evaluation & Analysis

In this section, it covers the performance of each model in terms of accuracy, recall, precision and f1-score. There are two categories of models, using topic modeling method and topic classification method. In addition, the performance of models trained using different datasets is also compared.

4.1 Topic Modeling Performance

4.1.1 LDA Model

In the project, we tune the hyper-parameter of the number of clusters for a LDA model from 10 to 100 with step size of 5. To later compare with BERTopic model, the whole dataset is shuffled with a fixed random seed, and the test set takes the last 20% of data. The label of each cluster is determined by counting the majority number of topics in that cluster for all documents in the corpus. Hence, there could be some topics that do not appear in the candidate topics options. We record the number of candidate topics for each model and evaluate its performance on the test set in terms of accuracy. Furthermore, the perplexity and coherence score are also presented to measure the interpretability of each model.

Number of clusters	Number of predicted topics	Accuracy	Perplexity	Coherence score
10	6	0.0306	-8.6276	0.3052
15	7	0.0269	-13.6118	0.3091
20	8	0.0306	-17.4437	0.3092
25	7	0.0310	-19.9018	0.3107
30	8	0.0289	-22.3353	0.4173
35	8	0.0313	-24.7572	0.4466
40	8	0.0306	-27.1964	0.4738
45	8	0.0295	-29.6771	0.4896
50	8	0.0000	-32.1332	0.4903
55	8	0.0458	-34.6134	0.4966
60	8	0.1169	-37.0545	0.5088
65	8	0.0335	-39.5261	0.5123
70	8	0.0421	-42.3624	0.5149
75	8	0.0326	-44.4668	0.5144
80	8	0.0337	-47.3055	0.5173
85	8	0.0399	-49.8483	0.5127
90	7	0.0408	-52.7955	0.5160
95	8	**0.1229**	-55.3608	0.5055
100	8	0.0382	**-57.9952**	**0.5149**

Table 4.1: LDA performance table. The number of clusters affects the perplexity of model, but there is no guarantee on being able to make a more accurate prediction.

As shown in Table 4.1, LDA models are not capable of making a precise prediction for an 8-class classification task. According to the perplexity score for each model, it is observed that most of the models are well described. However, it can only reach approximately 12% accuracy for the best-performed one. One possible reason is that for LDA model, it does not consider the position relationship between words, which affects sentence understanding. Moreover, we apply majority vote to determine the topic of each cluster with the assumption that each tweet is classified to be a single label. Given an example tweet, lie "Data reveals a strong link between community health and crime rates. Neighborhoods with better access to healthcare services experience up to 25% reduction in certain types of crimes. Investing in health can truly build safer communities.", it is only labeled as "Crime and Security ", but when considering multi-labeling, other topics like "Healthcare", "Society" can also be applied. It is also observed that as the number of clusters increases, the interpretability also improves. However, the data shows that it does not have a proportional relation with model performance. More specifically, for the model with 50 clusters, the accuracy is 0 and for the model with 60 clusters, the accuracy reaches 11%. We use the above tweet as an example and present the predictions from three outlier models in Table 4.2 .

Number of cluster	Evaluated accuracy	Prediction
50	0.0000	Health and Social Care
60	0.1169	Economy
95	**0.1229**	Crime and Security

Table 4.2: Prediction Comparison Among Three Outlier Models

As analyzed before, the example tweet is classified as "Crime and Security" in the dataset, but also reasonable to be labeled as "Health and Social Care". Based on this, both 50-cluster model and 95-cluster model should be considered as making the correct prediction. Therefore, a 0% accuracy for 50-cluster model further proves the reasonableness of considering tweet classification as a multi-labeling task.

4.1.2 BERTopic Model

As shown in Figure 4.3, BERTopic model also does not have a good performance, with accuracy only between 1.5% to 3.2%. Although the model's performance fluctuates as the number of cluster increases, the overall accuracy for 55 to 100 clusters model is higher than 10 to 50's. It is likely to see an improvement as the number of cluster further increases. Limited by computing power, it can be included in future work.

Number of clusters	Number of predicted topics	Accuracy
10	7	0.0185
15	8	0.0237
20	8	0.0233
25	8	0.0206
30	8	0.0182
35	8	0.0185
40	8	0.0157
45	8	0.0196
50	7	0.0182
55	8	0.0221
60	8	0.0262
65	8	0.0272
70	8	0.0237
75	8	0.0202
80	8	0.0201
85	8	0.0268
90	8	0.0315
95	8	0.0271
100	8	**0.0319**

Table 4.3: BERTopic performance table. There is a trend that when the number of clusters increases, the accuracy score also tends to rise.

4.2 Topic Classification Performance

4.2.1 Zero-shot Classification Model

The Zero-shot classification model is evaluated by using two datasets, the one with hashtag and url, and the one without. As shown in Table 4.4, the dataset with hashtag and url included outperforms the one without by around 1%, which is consistent with Diao et

al.[5] study on the effectiveness of hashtag in tweet classification task. We use this result aiming at providing a performance baseline and helps to benchmark other fine-tuned models discussed in later section.

Dataset	Accuracy
Dataset with hashtag and url	**0.4047**
Dataset without hashtag and url	0.3958

Table 4.4: Zero-shot classification performance table.

4.2.2 Gzip + KNN

Table 4.5 presents the accuracy of different KNN models. We measure the performance of KNN model with different settings, which includes tuning on the number of neighborhoods and different distance calculation scheme. Among all these settings, 2-NN model without weighted distance generates the best-performed model with the accuracy score of 61.61%. Compared to 2-NN model, as the input dataset is imbalanced, 1-NN can easily overfit, which may cause a 2% gap with 2-NN. For the models considering weighted distance, it is observed to have a clear gap with the conventional ones. Since the content of a tweet is usually quite short, and we are using Gzip as the compression method, the output length after compression is quite small. Hence, even after combining two different tweets, there is not a big difference on length. In addition, there is a relative large gap on topic distribution, which may further mislead the classification results. However, this method considers the length of compressed content as the only feature and directly applies instance-based method for a 9-class classification task, which generates a better performance than zero-shot classification model on accuracy score. It is believed that further study on compression algorithm selection and distance calculation scheme will improve the performance.

Model Setting	Accuracy
1-NN	0.5907
2-NN	**0.6161**
2-NN with weighted distance	0.4378
3-NN with weighted distance	0.3446
4-NN with weighted distance	0.3402
5-NN with weighted distance	0.3146
6-NN with weighted distance	0.3136
7-NN with weighted distance	0.2994
8-NN with weighted distance	0.2994

Table 4.5: Performance table of gzip + KNN.

4.2.3 Fine-tuned Models

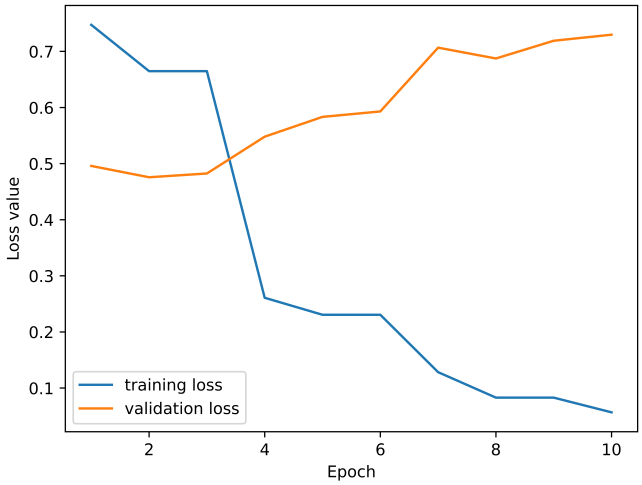
In this project, we fine-tune the DistilBert model, BERT-base model and BERT-large model. For both DistilBert and BERT-base model, we grid search its optimal settings on training batch size and learning rate. The three models are also trained using the dataset

containing hashtag and url, and the one without. In addition, the GPT generated dataset is also used to train a DistilBert model, and the hyper-parameters are also tuned in terms of batch size and learning rate. Tables below presents the accuracy, precision, recall as well as f1-score on the different datasets. Apart from finding the best performed model among all, training models on different datasets also provide the insight on whether to include hashtag and url filtering in data preprocessing pipeline. Figure 4.1 and Figure 4.2 present the training and validation loss curve for 6 models. Each of them is trained on 10 epochs. It is evident that each model overfits the training set after 3-5 epochs. Conventional approaches to address this problem includes performing early stopping during training, increasing training set size, reducing models complexity and using an ensemble of combined models. In this project, we apply early stopping to avoid overfitting on the training set. The model chosen for evaluation is the models around the first turning point on the validation loss curve. The average performance score of the chosen three models is recorded as the score on the performance table.

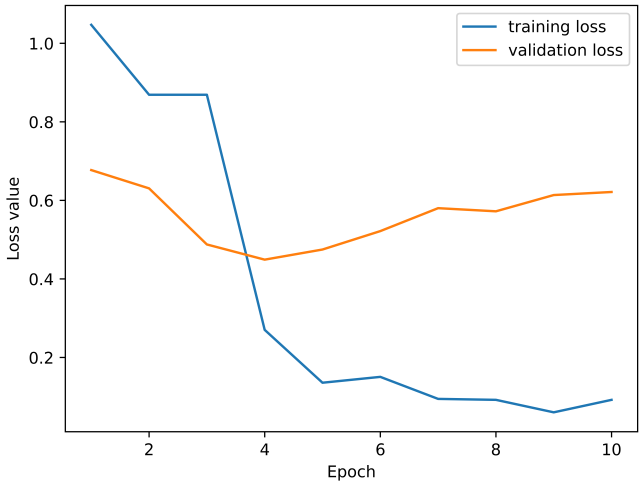
Model	Accuracy	Precision	Recall	F1-score
DistilBert[1]	0.8764	0.8346	0.8494	0.8413
DistilBert[2]	0.9317	0.9013	0.9241	0.9169
BERT-base[1]	0.9121	0.8847	0.9072	0.8945
BERT-base[2]	0.9337	0.9102	0.9328	0.9205
BERT-large[1]	0.9244	0.9097	0.9097	0.9086
BERT-large[2]	**0.9485**	**0.9372**	**0.9398**	**0.9383**

Table 4.6: Performance table on dataset without hashtag and url. BERT-large[2] outperforms other models in all four metrics. [1] denotes the model is trained using the dataset with hashtag and url, [2] denotes the model is trained using the dataset without hashtag and url.

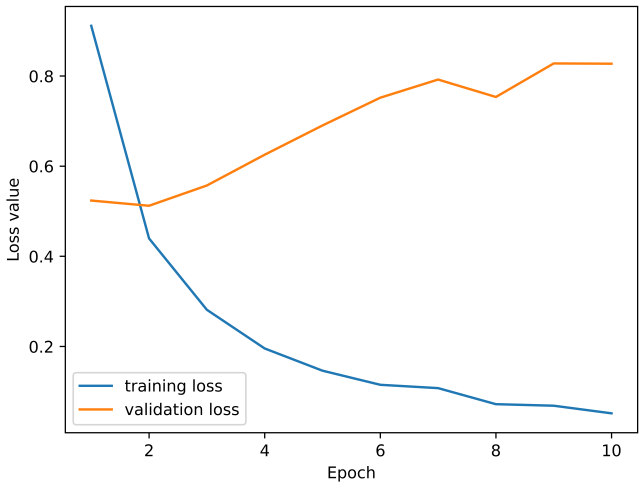
According to Table 4.6, the model trained on filtered dataset performs better in all of the four metrics, which denotes that these models are better at predicting the tweet without hashtag and url. One possible reason is that after filtering, two datasets do not have the same amount of tweets without hashtag and url. It is observed that the fine-tuning task is proportion to the size of training dataset. With a larger size of dataset, it is more likely for the model to perform better [3]. More specifically, the dataset is estimated to have 6243 tweets with hashtag or url, which is around 60% of the dataset. Furthermore, it is clear that BERT-large model outperforms the rest models. As BERT-large model has 1024 vector space for sentence representation and is three times more on parameter amount, such performance is expected. In addition, it is also observed that DistilBert[2] (DistilBert without hashtag & url) and BERT-base[2] (BERT-base without hashtag & url) do not have a big difference among the four metrics, even though the DistilBert is only 60% the size of BERT-base model. It is evaluated by Sanh et al.[21] that DistilBert model retains 97% of BERT-base model's language understanding capability. With the same setting on sentence presentation (a vector of 768), DistilBert and Bert-base model are expected to have a relatively similar performance.



(a) DistilBert with hashtag & url

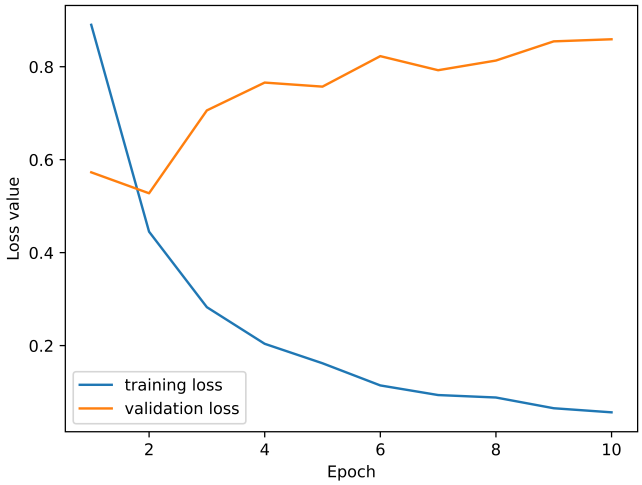


(b) BERT-base with hashtag & url

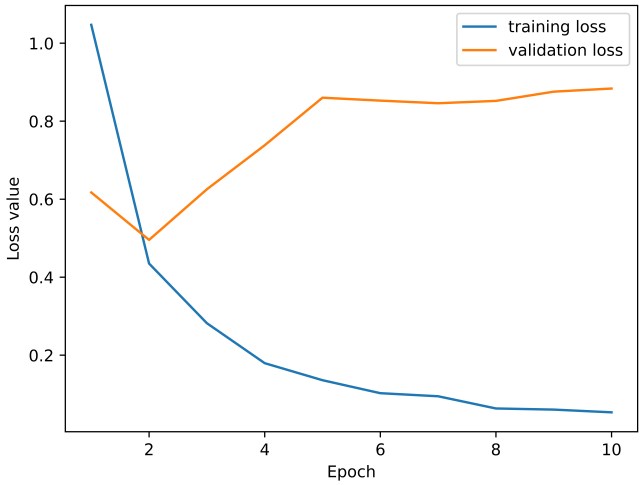


(c) BERT-large with hashtag & url

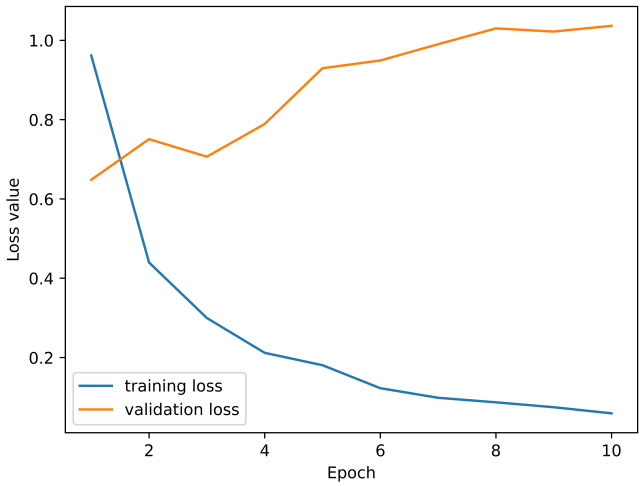
Figure 4.1: Training and validation loss on dataset with hashtag and url for 10 epochs.



(a) DistilBert without hashtag & url



(b) BERT-base without hashtag & url



(c) BERT-large without hashtag & url

Figure 4.2: Training and validation loss on dataset without hashtag and url for 10 epochs.

Model	Accuracy	Precision	Recall	F1-score
DistilBert[1]	0.8764	0.8346	0.8495	0.8413
DistilBert[2]	0.8861	0.8391	0.8934	0.8610
BERT-base[1]	0.8817	0.8402	0.8694	0.8535
BERT-base[2]	**0.8988**	0.8444	**0.9207**	**0.8755**
BERT-large[1]	0.8749	0.8243	0.8545	0.8375
BERT-large[2]	0.8913	**0.8771**	0.8745	0.8729

Table 4.7: Performance table on dataset with hashtag and url. BERT-base[2] shows the best performance on filtered test set, and slightly outperforms BERT-large[2] in terms of accuracy and F1-score. [1] denotes the model is trained using the dataset with hashtag and url, [2] denotes the model is trained using the dataset without hashtag and url.

As shown in Table 4.7, DistilBert model and BERT-base model both have a better performance when they are trained on the filtered dataset. One reason could be when the dataset contains the hashtag and url, the model tends to overfit the number or the name. For example, some of the tweets in the dataset contains url like "https://t.co/vQiAVbDC6a https://t.co/MNWe0pD5gk", which does not represent any valid information. Instead of extracting the information, the model may learn the pattern of such random number and characters, which leads to a drop in performance. The same situation applies to BERT-large model, which could be the reason why BERT-large model produces the worst performance among all of the three models as it may overfit more.

In table 4.8, it presents the performance of a DistilBert model trained on the GPT generated dataset. For the evaluation results on the GPT generated test dataset, all four of the metrics reach 98% score, which denotes the pattern for each topic is well learned. However, it performs poorly on the original dataset with only around 12% accuracy and 11% f1-score. One of the major reasons we consider is the multi-labeling case.

Dataset	Accuracy	Precision	Recall	F1-score
GPT generated dataset	**0.9856**	**0.9855**	**0.9854**	**0.9855**
Original dataset without hashtag & url	0.1237	0.2017	0.1377	0.1229
Original dataset with hashtag & url	0.1105	0.1884	0.1101	0.1094

Table 4.8: Performance table of the DistilBert model trained on the GPT generated dataset measured on three datasets.

The following list represents three example tweets from the original dataset.

1. Latest job in Nottingham : Nottingham Trent University is a HR Data amp Insights Analyst
    - Prediction from the keyword-based model: Children, Education and Skills
    - Prediction from the model trained by original dataset: Children, Education and Skills
    - Prediction from the model trained by GPT generated dataset: Labour Market and Welfare

2. In practical terms when personal data is collected, the controller could draft a privacy consent form which discloses only the categories of recipients amp invoke the lack of knowledge of the recipients actual identity.
    - Prediction from the keyword-based model: Population and Society
    - Prediction from the model trained by original dataset:Population and Society
    - Prediction from the model trained by GPT generated dataset: Crime and Security
3. Would need to analyse this. Start with how many risk their lives on shifts whilst supporting dependents and have to sort out their transport to work together with paying taxes and NI. Can you pop back when you ve got that starting data
    - Prediction from the keyword-based model: Transport, Environment and Climate Change
    - Prediction from the model trained by original dataset:Transport, Environment and Climate Change
    - Prediction from the model trained by GPT generated dataset: Crime and Security

The first example is labeled as Children, Education and Skills by both keyword-based model and the fine-tuned model from the original dataset. As the text contains the word 'University' and is talking about university event, it is acceptable to be labeled as Education. However, the text is about a job announcement, it is also reasonable to classify it related to labour market making the prediction from GPT dataset model also acceptable. The second example includes the words of 'personal data' and 'recipient', as the model trained on the original dataset is using the result from keyword-based model as the ground truth. It is expected for them to have the same label prediction. But the sentence also can be interpreted as a security related issue, which makes the prediction from GPT dataset-based model also acceptable. For the third example, we can see there is the word 'transport' included, which is a direct match to the keywords in the topic class. Furthermore, the text expresses worries on security, which falls within the topic "Crime and Security". The above three examples are random picked ones in which the model trained by GPT generated dataset does not make a matched prediction with the keyword-based model. However, the mismatch does not mean the model is making an incorrect prediction. Instead, it further proves that tweets tend to be in multiple classes, i.e. the task is actually a multi-label classification task, rather than a single-class classification one. This is a possible future direction for this project.

4.2.4 Ensemble Model

Based on above results, we propose an ensemble model, combining DistilBert, BERT-base and BERT-large. As shown in Table 4.9, we compare two ensemble models on a filtered test set. The ensemble model with DistilBert, BERT-base, BERT-large reaches the current best performance on the filtered test set in terms of accuracy. It follows the majority vote when making a prediction, which is the same to the ensemble model with an

additional DistilBert model trained on GPT generated set. Moreover, when there is a tie in decision, it predicts the same label as the BERT-large model's output for the ensemble model of the four models. As presented in above sections, the DistilBert model trained on GPT generated set has an outstanding performance on GPT set, but only has an accuracy score around 12% on real-world dataset. It is discussed that the model trained on GPT set has a comprehensive and reasonable understanding of text, the biggest limitation on its performance is due to the deterministic label. Therefore, it is observed that there is an approximately 12% drop on accuracy when include DistilBert trained on GPT set in the ensemble model.

Model	Accuracy
DistilBert, BERT-base, BERT-large	**0.9495**
DistilBert, BERT-base, BERT-large, DistilBert on GPT set	0.8233

Table 4.9: Performance table of ensemble model. The ensemble model of DistilBert, BERT-base and BERT-large is the best performed model among all.

Chapter 5

Summary

In this project, we start with dataset analysis and data preprocessing. We consider the potential problem from imbalanced dataset and reliability of ground truth, and provide corresponding solutions. Moreover, we set the pipeline for data preprocessing and evaluate its impact on downstream model training. We approach this tweet classification task in two ways, topic modeling and topic classification. For topic modeling, we evaluate the performance of the probabilistic model LDA and the pre-trained language model BERTopic. We analyze their prediction on specific prompts and discuss the factors that could impact its performance. For topic classification, we start from a simple method, like Gzip + KNN, and use it as a performance baseline to compare with large language models. Moreover, to provide a comprehensive analysis and produce a well-performed model, we explore different model architectures, including DistilBert, BERT-base and BERT-large, with a grid search on hyper-parameters in terms of learning rate and batch size. Furthermore, considering the ground truth is coming from a keyword-based model, we use GPT to generate a 9000-instance dataset and fine tune a DistilBert model on top of it. The model is further evaluated on both the GPT dataset and the original set. From above results, it is observed that topic classification models outperform topic modeling models. However, the error analysis concludes that poor performance partially comes from only considering the task as a single-classification task. Hence, it is suggested that the future direction of this project can consider multi-label classification and using manually annotated datasets to evaluate the performance for the keyword-base model as well as the models presented above. In addition, due to the computing power, several experiments are limited. For example, the number of clusters for topic modeling models is set from 10 to 100 and there could be a grid search on hyper-parameters for BERT-large models, which could be involved in future study.

Finally, by applying ensemble learning, the model of combining DistilBert, BERT-base and BERT-large reaches the best-performance on accuracy and is suggested as the best candidate for the future deployment consideration.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Gaurav Kanojia and Aditya Rastogi. Disaster tweets classification. *Iconic Research And Engineering Journals*, 6:295–298, 2023.

[5] Shizhe Diao, Sedrick Scott Keh, Liangming Pan, Zhiliang Tian, Yan Song, and Tong Zhang. Hashtag-guided low-resource tweet classification. In *Proceedings of the ACM Web Conference 2023*, pages 1415–1426, 2023.

[6] Mena Badieh Habib Morgan and Maurice van Keulen. Information extraction for social media. In *Proceedings of the Third Workshop on Semantic Web and Information Extraction*, pages 9–16, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University.

[7] Dimosthenis Antypas, Asahi Ushio, Jose Camacho-Collados, Leonardo Neves, Vítor Silva, and Francesco Barbieri. Twitter topic classification. *arXiv preprint arXiv:2209.09824*, 2022.

[8] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval: 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland, April 18-21, 2011. Proceedings 33*, pages 338–349. Springer, 2011.

[9] Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The author-topic model for authors and documents. *arXiv preprint arXiv:1207.4169*, 2012.

[10] Asbjørn Steinskog, Jonas Therkelsen, and Björn Gambäck. Twitter topic modeling by tweet aggregation. In *Proceedings of the 21st nordic conference on computational linguistics*, pages 77–86, 2017.

[11] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[12] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013.

[13] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text classification. In *14th International Conference on Machine Learning*, pages 143–151, 1996.

[14] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

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

[16] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 1681–1691, 2015.

[17] Zhiying Jiang, Matthew Yang, Mikhail Tsirlin, Raphael Tang, Yiqin Dai, and Jimmy Lin. "low-resource" text classification: A parameter-free classification method with compressors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6810–6828, 2023.

[18] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul MB Vitányi. The similarity metric. *IEEE transactions on Information Theory*, 50(12):3250–3264, 2004.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[20] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[21] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[22] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

[23] Chandler May, Ryan Cotterell, and Benjamin Van Durme. An analysis of lemmatization on topic models of morphologically rich language. *arXiv preprint arXiv:1608.03995*, 2016.

[24] Alexandra Schofield and David Mimno. Comparing apples to apple: The effects of stemmers on topic models. *Transactions of the Association for Computational Linguistics*, 4:287–300, 2016.

[25] Edleno Silva de Moura, Gonzalo Navarro, Nivio Ziviani, and Ricardo Baeza-Yates. Fast and flexible word searching on compressed text. *ACM Transactions on Information Systems (TOIS)*, 18(2):113–139, 2000.

[26] Waad Alhoshan, Liping Zhao, Alessio Ferrari, and Keletso J Letsholo. A zero-shot learning approach to classifying requirements: A preliminary study. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 52–59. Springer, 2022.

[27] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[28] Damien Sileo. tasksource: A dataset harmonization framework for streamlined nlp multi-task learning and evaluation, 2023.

Appendix A

Model Training Tables

A.1 Performance Table of DistilBert Trained on Original Dataset

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	257	0.6542	0.7446	0.8052	0.7657	0.8350
0.8635	2.0	514	0.5548	0.7961	0.8277	0.8056	0.8540
0.8635	3.0	771	0.4839	0.7912	0.8427	0.8115	0.8589
0.3097	4.0	1028	0.5256	0.8148	0.8544	0.8315	0.8667
0.3097	5.0	1285	0.5657	0.8346	0.8494	0.8413	0.8764
0.1839	6.0	1542	0.6005	0.8208	0.8430	0.8304	0.8710
0.1839	7.0	1799	0.6580	0.8319	0.8349	0.8314	0.8706
0.1254	8.0	2056	0.6348	0.8342	0.8515	0.8423	0.8774
0.1254	9.0	2313	0.6601	0.8314	0.8394	0.8348	0.8745
0.0935	10.0	2570	0.6731	0.8400	0.8427	0.8407	0.8779

Figure A.1: DistilBert on original dataset with learning rate: 2e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	257	0.5790	0.7659	0.8225	0.7875	0.8472
0.7473	2.0	514	0.5007	0.8115	0.8503	0.8264	0.8647
0.7473	3.0	771	0.4903	0.8007	0.8418	0.8174	0.8594
0.2608	4.0	1028	0.5370	0.8249	0.8491	0.8350	0.8657
0.2608	5.0	1285	0.6034	0.8424	0.8514	0.8455	0.8803
0.1543	6.0	1542	0.5988	0.8396	0.8565	0.8466	0.8788
0.1543	7.0	1799	0.6736	0.8486	0.8453	0.8458	0.8769
0.0981	8.0	2056	0.6476	0.8400	0.8605	0.8492	0.8788
0.0981	9.0	2313	0.6837	0.8443	0.8510	0.8469	0.8788
0.0713	10.0	2570	0.6943	0.8467	0.8562	0.8509	0.8793

Figure A.2: DistilBert on original dataset with learning rate: 3e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	257	0.4958	0.7757	0.8526	0.8027	0.8526
0.6647	2.0	514	0.4756	0.8336	0.8480	0.8386	0.8701
0.6647	3.0	771	0.4823	0.8197	0.8588	0.8360	0.8730
0.2305	4.0	1028	0.5479	0.8314	0.8618	0.8439	0.8735
0.2305	5.0	1285	0.5832	0.8295	0.8542	0.8401	0.8779
0.1282	6.0	1542	0.5929	0.8251	0.8627	0.8404	0.8745
0.1282	7.0	1799	0.7066	0.8476	0.8496	0.8472	0.8774
0.0828	8.0	2056	0.6873	0.8392	0.8510	0.8448	0.8764
0.0828	9.0	2313	0.7189	0.8410	0.8524	0.8461	0.8788
0.0566	10.0	2570	0.7297	0.8417	0.8510	0.8460	0.8793

Figure A.3: DistilBert on original dataset with learning rate: 5e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
1.0001	1.0	514	0.6163	0.7620	0.8133	0.7790	0.8394
0.4832	2.0	1028	0.5556	0.8131	0.8284	0.8166	0.8623
0.3307	3.0	1542	0.5381	0.8168	0.8425	0.8254	0.8691
0.2429	4.0	2056	0.6014	0.8289	0.8455	0.8353	0.8720
0.1849	5.0	2570	0.6600	0.8367	0.8408	0.8375	0.8740
0.1564	6.0	3084	0.6724	0.8219	0.8491	0.8333	0.8696
0.1316	7.0	3598	0.7511	0.8536	0.8481	0.8501	0.8808
0.1037	8.0	4112	0.7284	0.8438	0.8494	0.8461	0.8798
0.0946	9.0	4626	0.7584	0.8452	0.8470	0.8457	0.8798
0.0731	10.0	5140	0.7803	0.8448	0.8438	0.8437	0.8783

Figure A.4: DistilBert on original dataset with learning rate: 2e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.908	1.0	514	0.5671	0.7815	0.8256	0.7960	0.8511
0.4351	2.0	1028	0.5301	0.8371	0.8439	0.8400	0.8715
0.2993	3.0	1542	0.5461	0.8250	0.8605	0.8401	0.8754
0.2186	4.0	2056	0.6724	0.8348	0.8517	0.8417	0.8745
0.168	5.0	2570	0.6923	0.8410	0.8441	0.8417	0.8754
0.1302	6.0	3084	0.6834	0.8301	0.8600	0.8432	0.8740
0.1094	7.0	3598	0.7413	0.8400	0.8515	0.8453	0.8774
0.0876	8.0	4112	0.7654	0.8383	0.8529	0.8452	0.8788
0.0833	9.0	4626	0.7748	0.8474	0.8530	0.8499	0.8798
0.0593	10.0	5140	0.7859	0.8423	0.8487	0.8452	0.8769

Figure A.5: DistilBert on original dataset with learning rate: 3e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.8922	1.0	514	0.5350	0.7953	0.8363	0.8092	0.8628
0.4521	2.0	1028	0.5359	0.8214	0.8385	0.8282	0.8652
0.2928	3.0	1542	0.5876	0.8264	0.8504	0.8367	0.8798
0.2099	4.0	2056	0.6974	0.8288	0.8435	0.8351	0.8764
0.1531	5.0	2570	0.8245	0.8367	0.8125	0.8232	0.8710
0.1124	6.0	3084	0.7553	0.8349	0.8543	0.8435	0.8764
0.1045	7.0	3598	0.7912	0.8452	0.8538	0.8492	0.8822
0.0716	8.0	4112	0.7909	0.8422	0.8529	0.8471	0.8788
0.0746	9.0	4626	0.8364	0.8462	0.8458	0.8458	0.8779
0.0533	10.0	5140	0.8262	0.8491	0.8536	0.8511	0.8837

Figure A.6: DistilBert on original dataset with learning rate: 5e-5 and batch size: 16

A.2 Performance Table of DistilBert Trained on Filtered Original Dataset

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	255	0.6963	0.7398	0.7965	0.7599	0.8292
0.934	2.0	510	0.5516	0.7593	0.8244	0.7846	0.8390
0.934	3.0	765	0.5481	0.7913	0.8211	0.8040	0.8513
0.3454	4.0	1020	0.5285	0.8089	0.8388	0.8216	0.8616
0.3454	5.0	1275	0.6033	0.8223	0.8361	0.8279	0.8635
0.1887	6.0	1530	0.6543	0.8231	0.8291	0.8250	0.8621
0.1887	7.0	1785	0.6787	0.8321	0.8373	0.8331	0.8655
0.1329	8.0	2040	0.6760	0.8144	0.8348	0.8225	0.8571
0.1329	9.0	2295	0.7171	0.8279	0.8348	0.8300	0.8616
0.0948	10.0	2550	0.7125	0.8239	0.8327	0.8269	0.8581

Figure A.7: DistilBert on filtered dataset with learning rate: 2e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	255	0.6439	0.7420	0.8096	0.7623	0.8297
0.8228	2.0	510	0.4992	0.7872	0.8436	0.8077	0.8537
0.8228	3.0	765	0.4997	0.8182	0.8433	0.8296	0.8655
0.2682	4.0	1020	0.5653	0.8218	0.8495	0.8330	0.8665
0.2682	5.0	1275	0.6542	0.8342	0.8381	0.8353	0.8630
0.1412	6.0	1530	0.7490	0.8299	0.8269	0.8273	0.8611
0.1412	7.0	1785	0.7309	0.8353	0.8385	0.8353	0.8635
0.1028	8.0	2040	0.6964	0.8184	0.8419	0.8283	0.8601
0.1028	9.0	2295	0.7635	0.8382	0.8425	0.8396	0.8684
0.07	10.0	2550	0.7657	0.8369	0.8419	0.8389	0.8655

Figure A.8: DistilBert on filtered dataset with learning rate: 3e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	255	0.5673	0.7481	0.8261	0.7672	0.8267
0.7272	2.0	510	0.4910	0.7874	0.8529	0.8078	0.8483
0.7272	3.0	765	0.5940	0.8135	0.8374	0.8228	0.8635
0.2406	4.0	1020	0.6715	0.8334	0.8309	0.8305	0.8650
0.2406	5.0	1275	0.6597	0.8237	0.8436	0.8324	0.8625
0.1264	6.0	1530	0.8143	0.8411	0.8300	0.8335	0.8665
0.1264	7.0	1785	0.7223	0.8423	0.8429	0.8415	0.8670
0.0932	8.0	2040	0.7135	0.8276	0.8485	0.8358	0.8650
0.0932	9.0	2295	0.7700	0.8408	0.8489	0.8439	0.8704
0.0591	10.0	2550	0.7825	0.8380	0.8450	0.8410	0.8689

Figure A.9: DistilBert on filtered dataset with learning rate: 5e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
1.1013	1.0	510	0.6074	0.7485	0.8173	0.7742	0.8385
0.5303	2.0	1020	0.5424	0.8019	0.8290	0.8132	0.8532
0.3519	3.0	1530	0.6154	0.8197	0.8207	0.8183	0.8586
0.2665	4.0	2040	0.6552	0.8265	0.8195	0.8222	0.8606
0.1919	5.0	2550	0.6558	0.8155	0.8406	0.8264	0.8630
0.1451	6.0	3060	0.7264	0.8327	0.8407	0.8360	0.8689
0.1195	7.0	3570	0.8053	0.8260	0.8242	0.8227	0.8601
0.1189	8.0	4080	0.8071	0.8226	0.8346	0.8263	0.8606
0.0976	9.0	4590	0.8324	0.8269	0.8261	0.8249	0.8596
0.076	10.0	5100	0.8260	0.8355	0.8353	0.8348	0.8650

Figure A.10: DistilBert on filtered dataset with learning rate: 2e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.9492	1.0	510	0.5973	0.7572	0.8287	0.7836	0.8434
0.4661	2.0	1020	0.5080	0.8146	0.8535	0.8311	0.8567
0.2954	3.0	1530	0.6910	0.8283	0.8231	0.8245	0.8591
0.2263	4.0	2040	0.7367	0.8448	0.8293	0.8363	0.8635
0.1749	5.0	2550	0.7399	0.8402	0.8373	0.8383	0.8650
0.1273	6.0	3060	0.7759	0.8352	0.8414	0.8377	0.8689
0.1051	7.0	3570	0.8864	0.8375	0.8271	0.8308	0.8616
0.0877	8.0	4080	0.8407	0.8327	0.8360	0.8335	0.8625
0.0781	9.0	4590	0.8586	0.8345	0.8362	0.8345	0.8645
0.0627	10.0	5100	0.8637	0.8392	0.8339	0.8360	0.8630

Figure A.11: DistilBert on filtered dataset with learning rate: 3e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.8901	1.0	510	0.5727	0.7730	0.8217	0.7887	0.8439
0.445	2.0	1020	0.5276	0.7930	0.8453	0.8123	0.8444
0.2825	3.0	1530	0.7059	0.8374	0.8205	0.8256	0.8606
0.2037	4.0	2040	0.7658	0.8562	0.8265	0.8399	0.8660
0.1618	5.0	2550	0.7571	0.8332	0.8438	0.8377	0.8640
0.1141	6.0	3060	0.8227	0.8499	0.8409	0.8444	0.8694
0.0934	7.0	3570	0.7924	0.8377	0.8415	0.8378	0.8665
0.0881	8.0	4080	0.8132	0.8365	0.8434	0.8387	0.8699
0.065	9.0	4590	0.8545	0.8402	0.8430	0.8403	0.8670
0.0562	10.0	5100	0.8590	0.8444	0.8474	0.8454	0.8709

Figure A.12: DistilBert on filtered dataset with learning rate: 5e-5 and batch size: 16

A.3 Performance Table of BERT-base Trained on Original Dataset

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	257	0.6305	0.7254	0.8018	0.7512	0.8180
0.8689	2.0	514	0.4877	0.8120	0.8500	0.8245	0.8667
0.8689	3.0	771	0.4490	0.7911	0.8590	0.8148	0.8599
0.2702	4.0	1028	0.4748	0.8291	0.8689	0.8457	0.8730
0.2702	5.0	1285	0.5217	0.8326	0.8543	0.8413	0.8783
0.1505	6.0	1542	0.5288	0.8351	0.8650	0.8481	0.8754
0.1505	7.0	1799	0.5801	0.8417	0.8585	0.8487	0.8769
0.092	8.0	2056	0.5721	0.8402	0.8694	0.8535	0.8818
0.092	9.0	2313	0.6135	0.8453	0.8618	0.8522	0.8808
0.0723	10.0	2570	0.6213	0.8399	0.8622	0.8498	0.8798

Figure A.13: BERT-base on original dataset with learning rate: 2e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	257	0.5962	0.7414	0.8132	0.7626	0.8268
0.7597	2.0	514	0.5120	0.8170	0.8507	0.8292	0.8652
0.7597	3.0	771	0.4818	0.7975	0.8565	0.8202	0.8652
0.2391	4.0	1028	0.5223	0.8220	0.8613	0.8377	0.8652
0.2391	5.0	1285	0.5516	0.8172	0.8599	0.8347	0.8706
0.1316	6.0	1542	0.5747	0.8139	0.8593	0.8333	0.8710
0.1316	7.0	1799	0.6290	0.8332	0.8483	0.8386	0.8701
0.0773	8.0	2056	0.6089	0.8312	0.8620	0.8450	0.8764
0.0773	9.0	2313	0.6633	0.8384	0.8532	0.8448	0.8774
0.0633	10.0	2570	0.6705	0.8452	0.8581	0.8510	0.8818

Figure A.14: BERT-base on original dataset with learning rate: 3e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	257	0.5628	0.7527	0.8299	0.7774	0.8336
0.6883	2.0	514	0.4727	0.8332	0.8530	0.8419	0.8740
0.6883	3.0	771	0.4391	0.8188	0.8702	0.8411	0.8725
0.2393	4.0	1028	0.5871	0.8243	0.8513	0.8327	0.8676
0.2393	5.0	1285	0.6000	0.8211	0.8466	0.8323	0.8706
0.1341	6.0	1542	0.6232	0.8244	0.8543	0.8376	0.8735
0.1341	7.0	1799	0.6928	0.8334	0.8464	0.8390	0.8779
0.0786	8.0	2056	0.7089	0.8337	0.8479	0.8400	0.8735
0.0786	9.0	2313	0.7268	0.8363	0.8464	0.8404	0.8774
0.0539	10.0	2570	0.7387	0.8354	0.8466	0.8405	0.8754

Figure A.15: BERT-base on original dataset with learning rate: 5e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
1.0364	1.0	514	0.5489	0.7727	0.8238	0.7924	0.8457
0.457	2.0	1028	0.4867	0.8441	0.8592	0.8497	0.8779
0.2751	3.0	1542	0.4484	0.8281	0.8620	0.8390	0.8788
0.2038	4.0	2056	0.5486	0.8322	0.8630	0.8441	0.8740
0.1559	5.0	2570	0.6556	0.8461	0.8429	0.8433	0.8779
0.1288	6.0	3084	0.6332	0.8328	0.8587	0.8444	0.8759
0.1095	7.0	3598	0.7137	0.8413	0.8468	0.8432	0.8754
0.0786	8.0	4112	0.6784	0.8367	0.8648	0.8497	0.8793
0.079	9.0	4626	0.6957	0.8468	0.8657	0.8552	0.8837
0.0603	10.0	5140	0.7175	0.8459	0.8604	0.8521	0.8822

Figure A.16: BERT-base on original dataset with learning rate: 2e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.9683	1.0	514	0.5484	0.7789	0.8277	0.7976	0.8506
0.4471	2.0	1028	0.4842	0.8306	0.8553	0.8415	0.8725
0.2923	3.0	1542	0.5338	0.8097	0.8446	0.8215	0.8701
0.2007	4.0	2056	0.6236	0.8332	0.8463	0.8378	0.8735
0.1614	5.0	2570	0.7156	0.8265	0.8387	0.8304	0.8715
0.1188	6.0	3084	0.6625	0.8360	0.8496	0.8422	0.8759
0.1149	7.0	3598	0.7649	0.8359	0.8512	0.8420	0.8774
0.076	8.0	4112	0.7407	0.8327	0.8528	0.8418	0.8720
0.0689	9.0	4626	0.7659	0.8354	0.8511	0.8428	0.8759
0.0567	10.0	5140	0.7846	0.8391	0.8525	0.8448	0.8783

Figure A.17: BERT-base on original dataset with learning rate: 3e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.8575	1.0	514	0.5390	0.7886	0.8376	0.8061	0.8511
0.447	2.0	1028	0.5180	0.8143	0.8471	0.8259	0.8662
0.269	3.0	1542	0.5750	0.8077	0.8420	0.8195	0.8667
0.2012	4.0	2056	0.7642	0.8214	0.8231	0.8195	0.8599
0.1522	5.0	2570	0.7314	0.8198	0.8394	0.8283	0.8681
0.1155	6.0	3084	0.6636	0.8238	0.8610	0.8400	0.8754
0.1073	7.0	3598	0.7997	0.8250	0.8413	0.8320	0.8686
0.0685	8.0	4112	0.7830	0.8208	0.8449	0.8315	0.8667
0.0581	9.0	4626	0.7975	0.8245	0.8432	0.8330	0.8691
0.0456	10.0	5140	0.8208	0.8254	0.8481	0.8356	0.8730

Figure A.18: BERT-base on original dataset with learning rate: 5e-5 and batch size: 16

A.4 Performance Table of BERT-base Trained on Filtered Original Dataset

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	255	0.7382	0.7058	0.7789	0.7260	0.7894
0.907	2.0	510	0.5794	0.7838	0.8189	0.7976	0.8444
0.907	3.0	765	0.5382	0.8084	0.8375	0.8200	0.8547
0.2877	4.0	1020	0.5942	0.8301	0.8319	0.8301	0.8670
0.2877	5.0	1275	0.6349	0.8198	0.8384	0.8280	0.8655
0.1513	6.0	1530	0.7735	0.8384	0.8144	0.8250	0.8635
0.1513	7.0	1785	0.7221	0.8456	0.8333	0.8385	0.8704
0.1059	8.0	2040	0.7615	0.8390	0.8312	0.8349	0.8684
0.1059	9.0	2295	0.7841	0.8362	0.8253	0.8306	0.8684
0.0811	10.0	2550	0.7918	0.8374	0.8246	0.8306	0.8645

Figure A.19: BERT-base on filtered dataset with learning rate: 2e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	255	0.6597	0.7225	0.7990	0.7429	0.7968
0.8033	2.0	510	0.5609	0.8155	0.8378	0.8247	0.8596
0.8033	3.0	765	0.5589	0.8119	0.8388	0.8231	0.8591
0.2454	4.0	1020	0.6598	0.8314	0.8273	0.8279	0.8625
0.2454	5.0	1275	0.6541	0.8103	0.8393	0.8229	0.8625
0.1332	6.0	1530	0.8259	0.8424	0.8213	0.8304	0.8665
0.1332	7.0	1785	0.7644	0.8298	0.8335	0.8312	0.8650
0.0907	8.0	2040	0.7939	0.8298	0.8255	0.8274	0.8660
0.0907	9.0	2295	0.8244	0.8310	0.8207	0.8255	0.8655
0.061	10.0	2550	0.8335	0.8310	0.8213	0.8256	0.8640

Figure A.20: BERT-base on filtered dataset with learning rate: 3e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	255	0.6320	0.7746	0.8197	0.7918	0.8360
0.7073	2.0	510	0.6156	0.7967	0.8232	0.8055	0.8473
0.7073	3.0	765	0.6028	0.8104	0.8381	0.8201	0.8552
0.2389	4.0	1020	0.6896	0.8296	0.8296	0.8290	0.8655
0.2389	5.0	1275	0.7462	0.8279	0.8353	0.8310	0.8694
0.1264	6.0	1530	0.9275	0.8488	0.8112	0.8271	0.8684
0.1264	7.0	1785	0.8244	0.8393	0.8313	0.8347	0.8729
0.0851	8.0	2040	0.8776	0.8281	0.8226	0.8249	0.8655
0.0851	9.0	2295	0.8838	0.8440	0.8278	0.8346	0.8675
0.0546	10.0	2550	0.8911	0.8371	0.8239	0.8296	0.8665

Figure A.21: BERT-base on filtered dataset with learning rate: 5e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
1.0741	1.0	510	0.6328	0.7761	0.8093	0.7820	0.8292
0.465	2.0	1020	0.5751	0.8326	0.8237	0.8265	0.8625
0.2979	3.0	1530	0.5442	0.8285	0.8482	0.8370	0.8719
0.2312	4.0	2040	0.6811	0.8434	0.8298	0.8350	0.8665
0.1609	5.0	2550	0.6873	0.8216	0.8338	0.8271	0.8635
0.14	6.0	3060	0.8476	0.8386	0.8175	0.8265	0.8640
0.1135	7.0	3570	0.8456	0.8302	0.8202	0.8249	0.8630
0.0973	8.0	4080	0.8595	0.8307	0.8186	0.8243	0.8625
0.0758	9.0	4590	0.8828	0.8306	0.8201	0.8251	0.8655
0.0669	10.0	5100	0.9072	0.8332	0.8192	0.8259	0.8660

Figure A.22: BERT-base on filtered dataset with learning rate: 2e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
1.047	1.0	510	0.6171	0.7493	0.8057	0.7716	0.8336
0.4348	2.0	1020	0.4954	0.8056	0.8646	0.8296	0.8714
0.2818	3.0	1530	0.6252	0.8181	0.8323	0.8212	0.8660
0.1793	4.0	2040	0.7381	0.8216	0.8258	0.8227	0.8733
0.1356	5.0	2550	0.8601	0.8161	0.8219	0.8165	0.8660
0.1023	6.0	3060	0.8526	0.8363	0.8299	0.8307	0.8758
0.0944	7.0	3570	0.8459	0.8234	0.8298	0.8251	0.8729
0.0631	8.0	4080	0.8519	0.8212	0.8325	0.8252	0.8714
0.0602	9.0	4590	0.8756	0.8200	0.8267	0.8226	0.8719
0.0532	10.0	5100	0.8836	0.8262	0.8258	0.8249	0.8724

Figure A.23: BERT-base on filtered dataset with learning rate: 3e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.9875	1.0	510	0.5592	0.7906	0.8234	0.8042	0.8463
0.4428	2.0	1020	0.6291	0.8177	0.8243	0.8198	0.8606
0.2998	3.0	1530	0.6034	0.8133	0.8342	0.8209	0.8586
0.2088	4.0	2040	0.7925	0.8365	0.8152	0.8231	0.8562
0.1563	5.0	2550	0.7368	0.8096	0.8301	0.8190	0.8571
0.1302	6.0	3060	0.9748	0.8449	0.8036	0.8208	0.8591
0.1024	7.0	3570	0.9088	0.8364	0.8264	0.8300	0.8645
0.0897	8.0	4080	0.9440	0.8340	0.8254	0.8283	0.8635
0.067	9.0	4590	0.9582	0.8370	0.8169	0.8260	0.8635
0.0526	10.0	5100	0.9805	0.8330	0.8192	0.8255	0.8630

Figure A.24: BERT-base on filtered dataset with learning rate: 5e-5 and batch size: 16

A.5 Performance Table of DistilBert Trained on GPT generated Dataset

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	225	0.2531	0.9361	0.9359	0.9346	0.935
No log	2.0	450	0.1835	0.9514	0.9520	0.9512	0.9511
0.4372	3.0	675	0.1798	0.9543	0.9546	0.9539	0.9539
0.4372	4.0	900	0.2059	0.9499	0.9500	0.9497	0.9494
0.0575	5.0	1125	0.2002	0.9563	0.9567	0.9561	0.9561
0.0575	6.0	1350	0.2019	0.9557	0.9552	0.9553	0.955
0.0231	7.0	1575	0.2152	0.9548	0.9550	0.9546	0.9544
0.0231	8.0	1800	0.2156	0.9554	0.9556	0.9554	0.955
0.0116	9.0	2025	0.2240	0.9559	0.9561	0.9557	0.9556
0.0116	10.0	2250	0.2221	0.9563	0.9566	0.9562	0.9561

Figure A.25: DistilBert on GPT generated dataset with learning rate: 2e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	225	0.2213	0.9434	0.9430	0.9419	0.9422
No log	2.0	450	0.1806	0.9512	0.9516	0.9508	0.9506
0.3512	3.0	675	0.1927	0.9515	0.9518	0.9512	0.9511
0.3512	4.0	900	0.2410	0.9490	0.9494	0.9490	0.9489
0.044	5.0	1125	0.2280	0.9554	0.9556	0.9550	0.955
0.044	6.0	1350	0.2199	0.9611	0.9609	0.9606	0.9606
0.0176	7.0	1575	0.2272	0.9562	0.9565	0.9560	0.9561
0.0176	8.0	1800	0.2321	0.9574	0.9576	0.9572	0.9572
0.0067	9.0	2025	0.2397	0.9590	0.9593	0.9588	0.9589
0.0067	10.0	2250	0.2381	0.9596	0.9599	0.9594	0.9594

Figure A.26: DistilBert on GPT generated dataset with learning rate: 3e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	225	0.2093	0.9466	0.9460	0.9448	0.945
No log	2.0	450	0.1837	0.9581	0.9578	0.9575	0.9572
0.289	3.0	675	0.2127	0.9540	0.9533	0.9527	0.9528
0.289	4.0	900	0.2200	0.9558	0.9560	0.9556	0.9556
0.0448	5.0	1125	0.2501	0.9565	0.9568	0.9562	0.9561
0.0448	6.0	1350	0.2577	0.9561	0.9559	0.9557	0.9556
0.0118	7.0	1575	0.2600	0.9559	0.9552	0.9554	0.955
0.0118	8.0	1800	0.2770	0.9555	0.9552	0.9552	0.955
0.0044	9.0	2025	0.2838	0.9548	0.9549	0.9545	0.9544
0.0044	10.0	2250	0.2838	0.9548	0.9549	0.9545	0.9544

Figure A.27: DistilBert on GPT generated dataset with learning rate: 5e-5 and batch size: 32

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	450	0.2320	0.9410	0.9412	0.9399	0.94
0.5426	2.0	900	0.2227	0.9465	0.9472	0.9460	0.9461
0.1125	3.0	1350	0.2242	0.9456	0.9446	0.9444	0.9439
0.0642	4.0	1800	0.2368	0.9557	0.9556	0.9550	0.955
0.0368	5.0	2250	0.2539	0.9515	0.9512	0.9513	0.9506
0.024	6.0	2700	0.2570	0.9543	0.9546	0.9539	0.9539
0.0106	7.0	3150	0.2576	0.9554	0.9547	0.9549	0.9544
0.0121	8.0	3600	0.2783	0.9538	0.9540	0.9534	0.9533
0.0047	9.0	4050	0.2817	0.9538	0.9540	0.9534	0.9533
0.003	10.0	4500	0.2754	0.9537	0.9539	0.9534	0.9533

Figure A.28: DistilBert on GPT generated dataset with learning rate: 2e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	450	0.2169	0.9437	0.9445	0.9432	0.9433
0.4523	2.0	900	0.1979	0.9486	0.9486	0.9479	0.9478
0.109	3.0	1350	0.2404	0.9545	0.9539	0.9533	0.9533
0.0659	4.0	1800	0.2330	0.9559	0.9555	0.9550	0.955
0.0301	5.0	2250	0.2434	0.9580	0.9583	0.9580	0.9578
0.0201	6.0	2700	0.2462	0.9572	0.9569	0.9570	0.9567
0.0089	7.0	3150	0.2618	0.9581	0.9585	0.9581	0.9578
0.0074	8.0	3600	0.2717	0.9616	0.9618	0.9612	0.9611
0.0025	9.0	4050	0.2805	0.9597	0.9601	0.9596	0.9594
0.0014	10.0	4500	0.2808	0.9608	0.9612	0.9607	0.9606

Figure A.29: DistilBert on GPT generated dataset with learning rate: 3e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
No log	1.0	450	0.2178	0.9434	0.9437	0.9426	0.9428
0.3833	2.0	900	0.2535	0.9449	0.9435	0.9424	0.9422
0.1094	3.0	1350	0.2235	0.9551	0.9551	0.9548	0.9544
0.0692	4.0	1800	0.2818	0.9545	0.9540	0.9534	0.9533
0.0299	5.0	2250	0.2676	0.9569	0.9559	0.9562	0.9556
0.019	6.0	2700	0.2761	0.9586	0.9584	0.9580	0.9578
0.0074	7.0	3150	0.2832	0.9589	0.9590	0.9587	0.9583
0.0063	8.0	3600	0.3009	0.9585	0.9583	0.9579	0.9578
0.0029	9.0	4050	0.2968	0.9610	0.9610	0.9607	0.9606
0.0013	10.0	4500	0.3040	0.9588	0.9588	0.9585	0.9583

Figure A.30: DistilBert on GPT generated dataset with learning rate: 5e-5 and batch size: 16

A.6 Performance Table of BERT-large

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
1.0054	1.0	510	0.6756	0.7428	0.8093	0.7561	0.8095
0.4708	2.0	1020	0.5588	0.7870	0.8473	0.8101	0.8419
0.2992	3.0	1530	0.5928	0.8114	0.8595	0.8286	0.8660
0.2202	4.0	2040	0.6855	0.8465	0.8468	0.8452	0.8729
0.149	5.0	2550	0.6949	0.8136	0.8565	0.8324	0.8689
0.1246	6.0	3060	0.7984	0.8290	0.8518	0.8393	0.8719
0.0958	7.0	3570	0.7819	0.8326	0.8478	0.8391	0.8694
0.0922	8.0	4080	0.8141	0.8263	0.8486	0.8358	0.8724
0.0658	9.0	4590	0.8614	0.8472	0.8508	0.8476	0.8797
0.0662	10.0	5100	0.8575	0.8355	0.8484	0.8401	0.8753

Figure A.31: BERT-large on original dataset with learning rate: 2e-5 and batch size: 16

Training Loss	Epoch	Step	Validation Loss	Precision	Recall	F1	Accuracy
0.9616	1.0	510	0.6482	0.7704	0.8009	0.7781	0.8360
0.4395	2.0	1020	0.7507	0.8422	0.7993	0.8157	0.8552
0.2995	3.0	1530	0.7064	0.8445	0.8213	0.8287	0.8684
0.2117	4.0	2040	0.7889	0.8262	0.8325	0.8245	0.8679
0.1805	5.0	2550	0.9295	0.8406	0.8161	0.8271	0.8670
0.1225	6.0	3060	0.9491	0.8429	0.8260	0.8333	0.8758
0.0983	7.0	3570	0.9901	0.8444	0.8299	0.8359	0.8773
0.0869	8.0	4080	1.0300	0.8377	0.8278	0.8319	0.8719
0.0745	9.0	4590	1.0220	0.8439	0.8341	0.8379	0.8773
0.0591	10.0	5100	1.0365	0.8410	0.8308	0.8352	0.8753

Figure A.32: BERT-large on filtered dataset with learning rate: 2e-5 and batch size: 16