AN EXPLORATION INTO SHORT FORM TEXT CLASSIFICATION METHODS

Candidate Number: 246504

Statement of Originality

This report is submitted as part of the requirements for the degree of Computer Science and Artificial Intelligence. It is a product of my own labour except where specified in this report. The work is an exploration of different methods of classifying short form text with the aim of finding an optimum method. The report may be freely copied and distributed provided that the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years.

Professional and Ethical considerations

This project is a meta-analysis of a pre-existing dataset. Therefore, a Secondary Data Analysis form was completed to show that I have properly considered any and all professional and ethical issues that could arise from carrying out this project with this particular dataset. The result of said form confirmed that the dataset as well as the scope of the project was within regulations.

I have also familiarised myself, and will be working in accordance with, the BCS code of conduct, particularly sections 1 and 2: Public Interest, and Professional Competence and Integrity.

Acknowledgements

I would like to thank Prof. Jeff Mitchell for his guidance, and overall immeasurable assistance in helping me conduct the best research I can. I would also like to thank Prof. Ian Mackie and the University of Sussex for facilitating this project.

Abstract

In this project, the aim was to explore the effectiveness of two different methods for text classification: Naïve Bayes (NB), and a fine-tuned BERT. Specifically, the dataset used is a collection of anonymised tweets which are all paired with a label that indicates the primary emotion conveyed in each tweet. The primary metrics used to evaluate the techniques experimented with are Accuracy and macro-averaged F1-Score (Grandini, M et.al, 2020). Precision, recall and F1-Score were also measured on a class-by-class basis.

Naïve Bayes is one of the simpler yet still efficient methods for text classification. Its probabilistic model is based on Bayes' theorem (Raschka, S. 2014). The *naïve* adjective comes from its assumption that the features in a dataset are mutually independent. More formally, they would be considered to be *independent and identically distributed (i.i.d)*. In practice this is rarely true, however respectable results can still be achieved.

Bidirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) is a language representation model that is designed to pre-train deep bidirectional representations from unlabelled corpora by conditioning on both left and right context throughout all layers of the transformer. This allows for the pre-trained BERT model to be fine-tuned with just one additional output layer to create state-of-the-art models (Devlin, J. et al. 2018). The main advantage of this model, compared to Naïve Bayes, is that it can retrieve sequence-based data, as opposed to the bag-of-words model in which all word order information is lost before the learning algorithm is executed.

While the results between the two were closer than expected, there was a clear (and expected) winner between the two. BERT conclusively outperformed Naïve Bayes (NB) in every metric possible on both validation and test data.

Contents

1 Introduction

As we venture further into what could be described the golden age of Artificial Intelligence (AI) and Machine Learning (ML), the number of applications for it is ever increasing, and new improvements are always being made. One of the applications is natural language processing (NLP). This can range from classifying text into different categories (Kowsari, K. et al. 2019), to predicting missing words in a sentence (Devlin, J. et al. 2018) to a fully interactive chat bot like Chat-GPT (OpenAI, 2024).

In this project, the aim was to explore the effectiveness of two different methods for text classification: Naïve Bayes (NB), and a fine-tuned BERT. Specifically, the dataset used is a collection of anonymised tweets which are all paired with a label that indicates the primary emotion conveyed in each tweet. The emotions that the dataset consists of are: Sadness, Joy, Love, Anger, Fear and Surprise.

1.1 Background

1.1.1 Text Classification

Text classification can be **binary,** or **multi-class** in this case. It starts with identifying text in a document, tagging it, and categorising the text on the insights derived from the text (Tripathi, P. 2023). The methods for performing text classification can vary widely, but they all follow relatively similar general steps. The components of a text classifier are as follows:

1. Preprocessing: Preparing the text for the learning algorithm which could include
    a. Tokenisation
    b. Word processing such as stop-word removal (Silba, C Ribeiro, B. 2003), Stemming and Lemmatisation (Balakrishnan, V. Lloyd-Yemoh, E. 2014)
    c. Feature representation and Extraction
2. Learning Algorithm: Training and optimising a model for classification

As part of the endeavour to explore the options for text classification, various methods of preprocessing were tested i.e. omitting/including stop word removal, different types of feature representation etc.

1.1.2 Machine Learning

Under the umbrella of artificial intelligence, one of the most commercially developed fields of study is machine learning. This concerns the study and development of statistical models/algorithms which can then be used to perform tasks without being given explicit instructions. Machine learning has been applied to a variety of tasks where it is too costly, in both money and time, to develop algorithms to complete such tasks directly. Examples include: large language models – examples of which are ChatGPT (OpenAi, 2024); computer vision; facial/speech recognition (Apple Face ID) etc.

There are many types of statistical models that are used, but all of them follow a core principle. They are fed a large set of varying data, often accompanied with a correct answer/outcome associated with each piece of data in the dataset. This is called **supervised learning** (Cunningham, Cord, Delaney. 2008) and is the variant used in this investigation. An alternative would be unsupervised learning, the difference being that there are no answers provided with the data set. The objective of unsupervised learning is to find trends and use them to group data.

As stated earlier, this investigation experimented with two main types of classification models: NB, and a fine-tuned BERT. The performance was measured using two primary metrics: Accuracy, and F1-Score (Grandini, et al. 2020).

1.2 Related Literature

This section reviews some of the literature related to this project. The information and findings within this literature were used to influence decisions made throughout the investigation.

1.2.1 Naïve Bayes and Text Classification I

Raschka (2014) provided an in-depth description of the theory behind and purpose of an NB classifier, particularly within the context of text classification. It contains a comprehensive description of the basic concepts behind an NB classifier, including Bayes' theorem, which the method is based on.

Furthermore, a selection of preprocessing methods one might decide to use when using Naïve Bayes to classify text was given. These included:

- Tokenisation: breaking down a corpus into individual elements that can serve as an input for NLP tasks

- Stop words (removal): Stop words are particularly common words either within the language or the text corpus itself that are therefore considered relatively uninformative

- Stemming and Lemmatisation: stemming involves transforming a word into its root form e.g. swimming -> swim. This can create fake words, such as leaves -> leav. Another option that avoids this is lemmatisation. This involves obtaining the canonical forms of words, also known as the *lemma*, e.g. leaves -> leaf.

- N-grams: This refers to how many words each feature consists of in a feature vector. Single word features are called unigrams, two words are called bigrams, then trigrams and so on.

1.2.2 Feature extraction for document classification

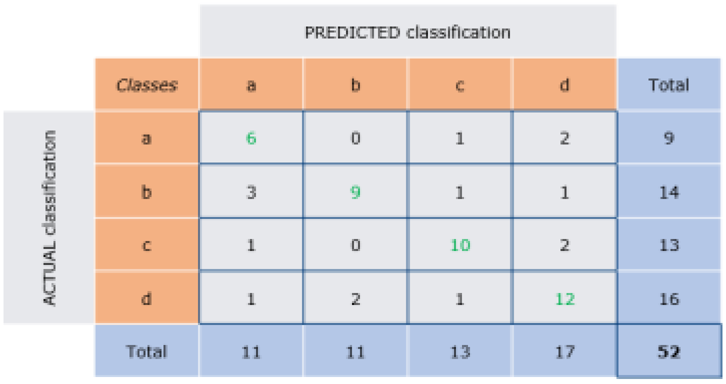
Vidhya, Singh and Leavline (2015) proposed a system for **feature extraction,** where the word frequencies in a document are scaled to their relative frequency in other documents in the same class, as well as the corpus overall. Term frequency (TF) is the relative importance of a term in a document, while Inverse Document Frequency (IDF) is a numerical statistic representing how important a word is in a certain corpus. TF-IDF is mathematically represented as follows:

$TF\text{–}IDF(t,d) = TF(t,d) \times IDF(t),$ in which $IDF(t) = log2\ |D|\{\ d: t \in d\ \}, |D|$ being the total number of documents in the corpus.

They then used a J48 algorithm to test this method of extraction, comparing features extracted with a null stemmer: a dummy stemmer that performs no stemming at all. J48 had an accuracy increase of 5.5 percentage points from 93% to 98.5%.

1.2.3 Metrics for Multi-Class Classification: An Overview

Grandini, Bagli and Visani (2020) presented a comprehensive overview of the ways in which to measure the performance of a multi-class classifier. They cover key concepts and metrics needed to properly evaluate a multi-class classifier, which include:



		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Figure 1 (Grandini, et al. 2020)

- Confusion matrix: A table that contains the quantities of correct and incorrect predictions for each class. A confusion matrix can be used to calculate:
    - True Positives (TP): Correct predictions of samples from a particular class
    - True Negatives (TN): Correct predictions that a sample is not in a particular class
    - False Positives (FP): Incorrect predictions that a sample is in a particular class
    - False Negatives (FN): Incorrect predictions that a sample is not in a particular class

- Accuracy: The percentage of predictions the model gets correct

- Precision: The percentage of correct predictions of a class out of the total predictions for that class i.e. how precisely it attributes said class. $\frac{TP}{TP+FP}$

- Recall: The percentage of correct predictions of a class out of the total instances of that class i.e. how fully it predicts a class $\frac{TP}{TP+FN}$

- F1-Score: A weighted average of precision and recall using the concept of harmonic mean
$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Balanced Accuracy: A model wide average of the recalls for each class

- Macro averaged F1-Score: A model wide average of F1-Score for each class

- Micro averaged F1-Score: An alternative way to calculate F1-Score, by summing all TP, TN, FP, FN to calculate model wide precision, recall and F1. In practice, this is equivalent to Accuracy

1.2.4 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

This paper, by Devlin, Chang, Lee, and Toutanova, details the theory behind and process of training BERT. This deep transformer was trained to be able to produce meaningful bidirectional representations of language.

It creates these representations through a masked language modelling (MLM) task, by creating meanings for a masked word based on the words that surround it i.e. its **context**. The reason for this is, unlike left-to-right language modelling, the bidirectionality of the model would allow it to 'see' the target word and therefore predict it trivially. By masking the target word, the model is forced to learn robust semantic representations based on a fusion of the left and right context of the word that enable it to predict the hidden work based on these learned representations.

During the training, 80% of the time the target word was masked. Another 10% of the time it was replaced with a random word, and the final 10% of time the word remained untouched (for the purpose of adding bias towards the actual word).

BERT is trained on two large unlabelled corpora: BooksCorpus (800M words), and Wikipedia (2,500M words).

1.2.5 Text-based emotion classification using emotion cause extraction

Li and Xu (2014) presented and showcased a method for classifying 'short informal microblog' posts from a Chinese social network, Weibo. The posts are very similar in format to the tweets that were classified in this project. For this task, they had six classes of emotion: Happiness, Anger, Disgust, Fear, Sadness, and Surprise.

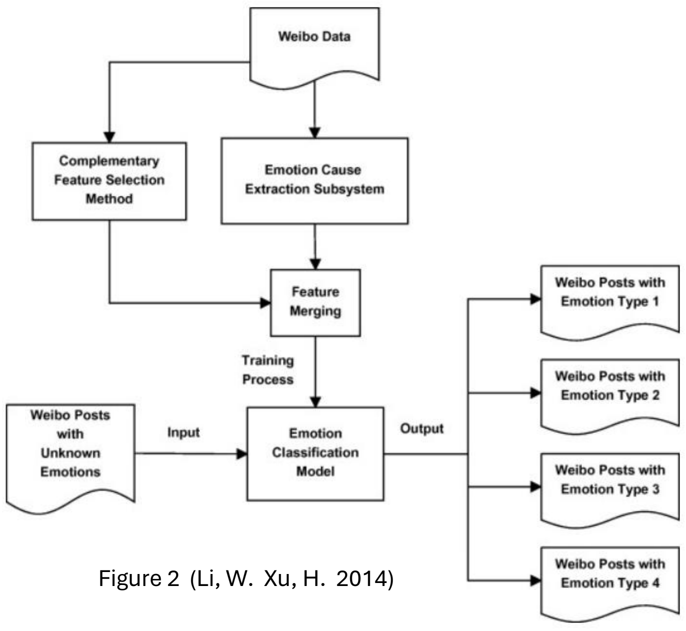

Weibo Data
Complementary Feature Selection Method
Emotion Cause Extraction Subsystem
Feature Merging
Training Process
Weibo Posts with Unknown Emotions
Input
Emotion Classification Model
Output
Weibo Posts with Emotion Type 1
Weibo Posts with Emotion Type 2
Weibo Posts with Emotion Type 3
Weibo Posts with Emotion Type 4

Figure 2 (Li, W. Xu, H. 2014)

The idea behind the technique that they employed, 'emotion cause detection', is looking for features that are "meaningful" to emotions instead of simply choosing words with high co-occurrence degree. The workflow for the entire system is shown in figure 2.

In the classification process, they used a Support Vector Machine (SVM) (Cortes, C., Vapnik, V. 1995) which has proven to be an effective model for text-driven classification. The particular variation they use is called Support Vector Regression (SVR), which returns a real number, and is a better choice for an imbalanced dataset like the one they chose to use.

Apart from for the Sadness class, classification performed with Emotion Cause (EC) detection marginally outperformed that which was not performed with EC detection. The improvement was minor in each case, but they argued that this improvement demonstrated ‘great potential’ in the technique of Emotion Cause Detection.

1.3 Project Objectives

Within this investigation, the aim was to conduct a thorough evaluation of the performance of an NB classifier and a fine-tuned BERT classifier on the given dataset, which consists of tweets paired with a label indicating the emotion conveyed in the tweet. Different implementations of these classifiers were tried to see in what form they work the best, and by the end a clear winner was found out of all of the tested implementations, based on the performance metrics detailed in 2.1.

2 Method

2.1 Performance Metrics

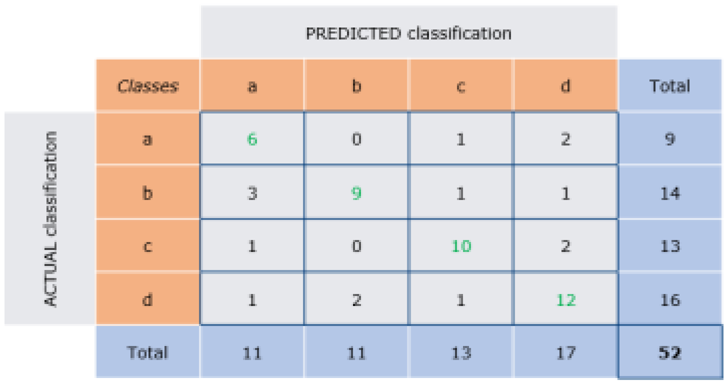
As mentioned in 1.2, the primary metrics used to evaluate the techniques experimented with are Accuracy and macro-averaged F1-Score (Grandini, M et.al, 2020). Precision, recall and F1-Score were also measured on a class-by-class basis to inform decisions on hyperparameter optimisations and/or preprocessing decisions.

2.1.1 Accuracy

This metric is the most basic form of evaluation for any classifier. It is calculated simply as the percentage of the predictions that the model gets correct:

$$Accuracy = \frac{Correct\ predictions}{Total\ predictions}$$

2.1.2 Precision and Recall



		PREDICTED classification				
	Classes	a	b	c	d	Total
ACTUAL classification	a	6	0	1	2	9
	b	3	9	1	1	14
	c	1	0	10	2	13
	d	1	2	1	12	16
	Total	11	11	13	17	**52**

Figure 3 (Grandini, M. et al. 2020)

Precision and Recall act as a foundation for calculating F1-Score. They are calculated using the True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN). In this context ‘positive’ means part of the given class, and ‘negative’ means not part of the given class. To visualise these quantities, a confusion matrix can be built.

In reference to figure 3:

- TP of class ‘a’ is the intersection column and row ‘a’.
- TN of class ‘a’ can be calculated as the sum of the fields not in column or row ‘a’
- FP of class ‘a’ is the sum of column ‘a’ minus the TP of class ‘a’
- FN of class ‘a’ is the sum of row ‘a’ minus the TP of class ‘a’

Naturally, in the context of multi-class classification, these scores will be calculated individually for each class. The formulae are as follows:

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

In practice, precision represents the proportion of all positive predictions made by the model that are actually positive, whereas recall represents the proportion of all positives in the dataset that have been predicted as such by the model. (Grandini, M et.al, 2020).

To get the model’s precision and recall for the entire dataset there are two options. The first, ‘Macro-averaged’, is where both scores are calculated for each class respectively, followed by the mean of each score between all classes.

$$P_i = Precision\ of\ class\ i \qquad R_i = Recall\ of\ class\ i \qquad N = Number\ of\ classes$$

$$Macro\ Precision = \frac{P_1 + P_2 + \cdots P_N}{N} \qquad Macro\ Recall = \frac{R_1 + R_2 + \cdots R_N}{N}$$

The second option is ‘Micro-averaged’, where all the TP, FP, TN, FN scores are summed, then precision and recall are calculated with these totals.

$$N = Number\ of\ Classes \qquad TP_i = True\ positives\ of\ class\ i$$

$$Micro\ Precision = \frac{TP_1 + TP_2 + \ldots TP_N}{(TP_1 + TP_2 + \ldots TP_N) + (FP_1 + FP_2 + \cdots FP_N)}$$

$$Micro\ Recall = \frac{TP_1 + TP_2 + \ldots TP_N}{(TP_1 + TP_2 + \ldots TP_N) + (FP_1 + FP_2 + \cdots FP_N)}$$

When using micro-averaging, precision and recall become equal to Accuracy as the method is effectively summing all the predictions together anyway. Therefore, macro-averaged scores were used in this project as Accuracy was already being measured.

2.1.3 F1-Score

F1-Score acts as a measure of overall performance by aggregating Precision and Recall through the concept of harmonic mean (Grandini, M et.al, 2020). It can be calculated as follows:

$$F1 = \frac{2}{Precision^{-1} \times Recall^{-1}} = 2 \times \frac{Precision + Recall}{Precision + Recall}$$

F1-Score can be understood as a weighted average between Precision and Recall, and acts as a valuable alternative to Accuracy when evaluating a model’s performance. When evaluating a multi-class classifier, F1-Score can be calculated for each class individually, or it can be calculated for the entire model.

2.2 Dataset overview

The chosen dataset, as previously mentioned, contains approximately 400,000 tweets. All of which are accompanied with a number that indicates the primary emotion conveyed in the text. The 6 labels are:

Sadness : 0, Joy : 1, Love : 2, Anger : 3, Fear : 4, Surprise : 5

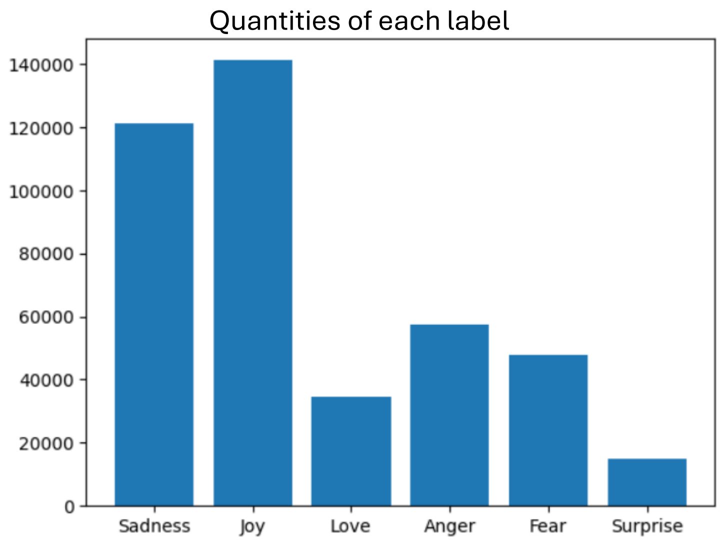


Figure 4

Figure 4 shows the total number of tweets ascribed to each emotion. There is a large imbalance between the different classes, which was taken into consideration particularly during the NB portion of the investigation.

The data was taken from the Hugging Face Library (Chen, Y. S. et al. 2018). The immense size of the dataset did have its benefits, but due to the number of tweets, the computational power needed to use the entire dataset (or a train-test split) for training a model was far too great. Therefore, a small portion of the dataset was used and still provided amicable results.

2.3 Multinomial Naïve Bayes Classifier

Inspired by the paper described in 1.2.1, the first model experimented with was Naïve Bayes. This is one of the simpler yet still efficient methods for text classification. Its probabilistic model is based on Bayes' theorem (Raschka, S. 2014). The *naïve* adjective comes from its assumption that the features in a dataset are mutually independent. More formally, they would be considered to be *independent and identically distributed (i.i.d)*. In practice this is rarely true, however respectable results can still be achieved. The goal of the model is to attribute the class that, when paired with the given set of features from the sample, maximises the **posterior-probability**.

2.3.1 Bayes' Rule

Bayes' rule is the definition of **posterior probability** (Raschka, S. 2014):

$$Posterior\ Probability = \frac{conditional\ probability\ \times prior\ probabiilty}{evidence}$$

Let

- x_i be the feature vector of the sample i
- w_j be the notation of class j
- $P(x_i|\, w_j)$ be the probability of observing sample x_i given that it belongs to class w_j

The general notation for posterior probability can be written as

$$P(w_j|x_i) = \frac{P(x_i|\, w_j)\ \times P(w_j)}{P(x_i)}$$

Conditional probability is defined as the probability of the feature vector occurring in the given class. Prior probability is the probability of said class occurring at all, and the evidence is the probability of the feature vector occurring at all. The aim of a Naïve Bayes classifier is to assign to a feature vector the class that maximises its posterior probability.

2.3.2 Pre-processing

2.3.2.1 Tokenisation

In order to create a feature representation, the text must first be tokenised. This involves taking a document/piece of text and transforming it into a list of tokens that typically represent a single word. For example, consider this sentence:

"Everyone loves Naïve Bayes."

Tokenising the sentence above with no additional processing would produce this list of tokens:

'Everyone', 'loves', 'Naïve', 'Bayes.'

There are two things to take note of here. First of all, the word 'everyone' has a capital 'E' which, if left unchecked, could lead the classifier to incorrectly identify it as different to the same word in all lower case. Similarly, there is a full stop attached to the word 'Bayes', which could allow a similar error to manifest.

It is natural, then, that the next step would be to convert all the tokens to lower/upper case and remove punctuation. Creating this uniformity will not only make patterns in the data easier for the algorithm to recognise, but also reduce the dimensionality (Palo, H et. al. 2021) of the feature vector that will be constructed for the data samples, thus reducing the computational expense of training the model.

2.3.2.2 Stop-word removal

In the pursuit of minimising the dimensionality of the data, it can be beneficial to only prioritise more informative words and remove the most common words from the equation entirely (Raschka, 2014). One approach would be to find the *n* most frequent words and remove them from the vocabulary. An alternative to that would be to use a pre-existing, language specific stop-word dictionary available from various sources. Removing these uninformative words can decrease the dimensionality of the data without foregoing critical information needed for classification.

2.3.2.3 Stemming and Lemmatisation

At this stage, plurals and alternate tenses of words will be tokenised as different, which can lead to words that are practically the same semantically being flagged as separate. Stemming involves crudely chopping off the morphological affixes of words in an attempt to reveal the root word. While this can lead to some success, it can often lead to errors, for example, when dealing with irregular plurals: ‘Leaf’, ‘Leaves’. In this instance, ‘Leaves’ would be stemmed to ‘Leav’ which would not solve the issue of falsely separate tokens. Alternatively, lemmatisation uses morphological analysis to return to the base dictionary form of the word, known as the *lemma*. This does require parts-of-speech (POS) tagging (Martinez, A. R. 2012) to be performed which leads to greater computational expense.

2.3.2.4 Bag-of-words

After performing the desired preprocessing techniques on the input data, the next step is to create a bag-of-words model. This allows the creation of an unordered representation of the words that each data sample contains. In order to generate these representations, it is necessary to first define a *vocabulary*. This involves identifying all the unique words that occur in the training data, and assigning an index that will represent each word in the **feature vectors**. With a vocabulary *V* of length *D,* with *D* being the number of unique words in the dataset, all generated feature vectors will be of length *D*.

Then, for each sample of text, a feature vector can then be generated. Each number within the feature vector will represent the frequency of the corresponding word within said sample. Consider this vocabulary to demonstrate:

{i : 0, love : 1, walking : 2, hate : 3, running : 4}

Above is a dictionary containing a vocabulary of length 5. It contains all the words in the vocabulary assigned to a number representing their index within the generated feature vectors. Below is a sentence and its corresponding feature vector.

“I love running fast, I hate running slow.”   [2, 1, 0, 1, 2]

In this example, notice that despite the sentence containing the word ‘fast’, the vocabulary does not contain that word, so there is no way for the generated feature vector to represent it. Furthermore, as the words ‘I’ and ‘running’ occur twice, the number in their respective indexes represent that frequency.

2.3.2.4 TF-IDF values

When using bag-of-words representations, there is the option of replacing the frequency value for each word in a sample with term-frequency inverse-document-frequency (TF-IDF) values (Vidhya, S. et al. 2015).

$$Term\ Frequency\ (TF)\ =\frac{Number\ of\ times\ given\ word\ occurs\ in\ document}{Total\ number\ of\ words\ in\ document}$$

$$Inverse\ Document\ Frequency\ (IDF)\ =\ log_2(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ given\ word})$$

$$TFIDF\ =\ TF\ \times IDF$$

TF-IDF values can be interpreted as a value of weighted importance for each word, as it is inversely proportional to how often the word appears throughout the training document. Frequency and TF-IDF values were experimented with in this investigation for comparison within an NB model.

2.4 Fine-tuned BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers (BERT) is a language representation model that is designed to pre-train deep bidirectional representations from unlabelled corpora by conditioning on both left and right context throughout all layers of the transformer. This allows for the pre-trained BERT model to be fine-tuned with just one additional output layer to create state-of-the-art models (Devlin, J. et al. 2018). The main advantage of this model, compared to Naïve Bayes, is that it can retrieve sequence-based data, as opposed to the bag-of-words model in which all word order information is lost before the learning algorithm is executed.

2.4.1 Pre-training

BERT learns the aforementioned 'bidirectional representations' through sample sentences in which 15% of the words are 'masked' i.e. hidden. This must be done because, unlike models in which representations are learned unidirectionally, bidirectionality would allow each word to indirectly 'see itself' (Devlin, J. et al. 2018), therefore the model would trivially predict the target word in a multi-layered context. Masking the target words during training forces to the model to create robust representations of their meaning that allow them to be predicted accurately. The entire process is called a 'masked language model' (MLM).

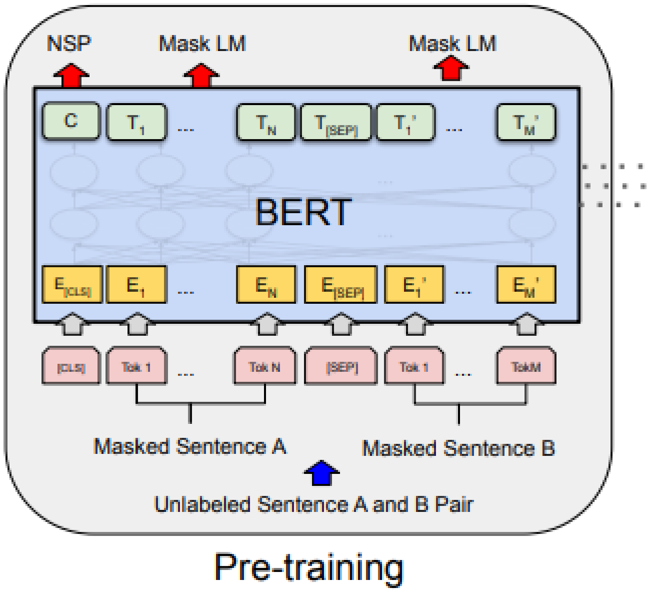


Figure 5 (Devlin, J. et al. 2018)

2.4.2 Feature representation and extraction

2.4.2.1 Tokenisation

The HuggingFace library (from which BERT is accessible) provides their own tokeniser that encodes the tokens ready for model training. Unlike the method of tokenisation presented in 2.3.2.1, BERT Tokeniser often creates sub-word tokens from more uncommon words in an attempt to extract morphological information. E.g. the word 'igneous' would be split into the tokens 'ign' and '#neous' with the hash being a tag for a sub word token. It then generates character-based (letter-based) representations for the sub-token.

2.4.2.2 Contextualised Word embeddings

The tokens generated are then turned into IDs that correspond to a word embedding representing the original word. A word embedding can be described as a vector representation of a word in *n* dimensional space, with the characteristic that the embeddings for semantically similar words exist close by in said space.

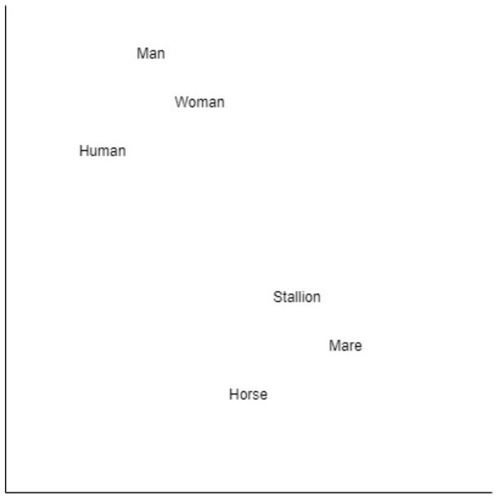


Figure 6 Word embeddings

Furthermore, as you can see in figure 6, pairs/groups of words with a particular semantic relationship will be similar distances away from each other compared to other groups of words with the same semantic relationship. These word embeddings are based on the distributional hypothesis:

"Words that occur in similar contexts tend to have similar meanings." (Harris, Z. S. 1954)

As part of pre-training, BERT has already learned these contextualised word embeddings through training on large unannotated corpora (Delvin, J. et al. 2018). Unlike previous trained embeddings like Radford et al. (2018), which only learn the embeddings through left to right context i.e. unidirectionally, BERT trains its contextualised embeddings bi-directionally. This allows for pre-requisite as well as subsequent context to a word to influence its embedding i.e. represented meaning.

2.4.2.3 Encoding

Once all of the words have been converted to IDs, the entire dataset is then converted into a **batch encoding** (BERT, n.d.). This is a dictionary of lists, arrays, or tensors (Comon, P. 2014) representing the contextualised 'meaning' contained within each data sample.

2.4.3 Fine-tuning

Once the batch encodings of the data have been generated, the model is ready to be fine-tuned. This involves randomising the parameters within the network, and repeatedly conducting forward passes through the network to generate a set of predictions. From these predictions, loss can be calculated. When initialising the pre-trained version to be fine-tuned for classification, the size of the output layer must be decided i.e. the number of neurons. This must be equal to the number of classes in the dataset being used, as each one will output the probability that the current sample is part of the class that the neuron represents.

2.4.3.1 Loss and Gradient Descent

To calculate loss, the specified loss function applied to the predictions to generate a continuous number that represents the error in the predictions. In this case, cross entropy loss was used (Mao, A. et al. 2023). Then, the partial derivative of the loss with respect to each parameter is calculated through **backpropagation**, allowing for **stochastic gradient descent** (Amari, S. I. 1993) to be performed on these parameters, meaning that they are updated in such a way that should decrease the loss.

2.4.3.2 Pre-existing layers: Tune or Freeze

When fine-tuning BERT, unless specified otherwise, the entire model and all its parameters will be tuned/updated to fit the given task. While this may sound like a good idea, this can lead to *catastrophic forgetting*: where a model 'forgets' how to perform the first task when learning how to perform the second task (Goodfellow, I. J. et al. 2013). On one hand, this could lead to better results on the new task. However, it could also indicate *overfitting* to the training data, which can then lead to poor performance on unseen data.

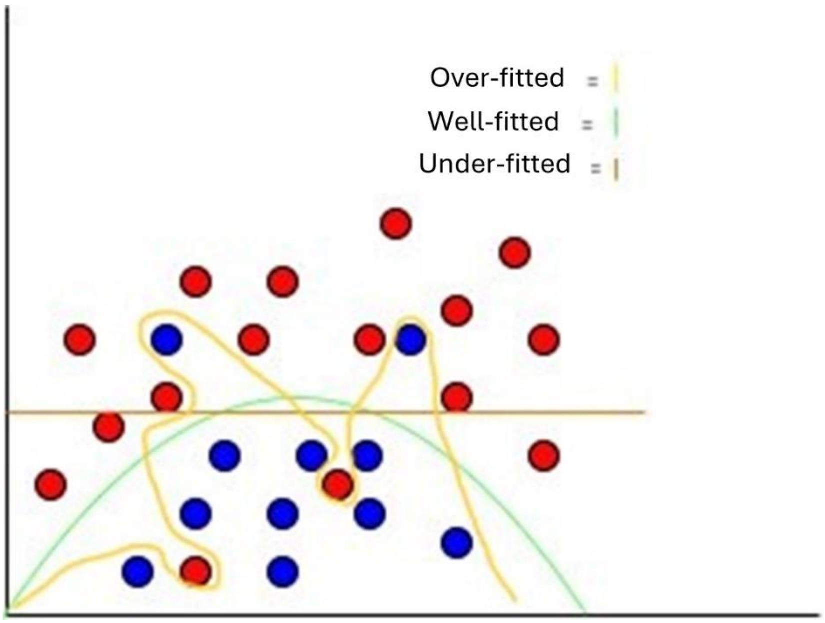


Figure 7

To combat this issue, all the layers of the base model can be frozen, leaving only the classification head to be trained. This means that the only parameters that would get updated would be that of the classification head. This could combat catastrophic forgetting and overfitting, as well as significantly reducing the time complexity of training the model. Experiments were conducted with and without this method in the project.

2.4.4 Hyper-parameters for Training

When preparing BERT, or any neural network, for training. There are certain hyper-parameters that must be defined.

These include:

- Number of **Epochs**: This determines how many times the entire training dataset is passed through the network during the training process.
- **Batch Size** (Per device): Batch size determines how many samples are used for each forward pass. This is used because when working with large datasets, it is

computationally difficult to use the entire dataset every time. However, if the system can handle it, larger batch sizes equate to faster training times.

- **Weight Decay**: This is a form of regularisation which encourages the weights to stay small, so as to not overfit to 'noise' in the data.
- **Learning Rate**: This specifies how far the weights should shift along the gradient of the loss.

2.5 Python Libraries

Within the investigation, various python libraries were used to implement the techniques described in this section. They are not the only libraries that were utilised, rather those which are necessary for the techniques described above.

2.5.1 Natural Language Toolkit (NLTK)

The NLTK provides functionality for a wide range of pre-processing techniques. These include, but are not limited to: tokenisation; stop-word removal; POS tagging; stemming and lemmatisation (Bird, S. et al. 2009). The techniques listed are all that were used from this library.

2.5.2 Scikit-learn

When conducting relatively basic methods of machine learning, almost all techniques are defined and provided through this module. For this investigation, Scikit-learn was utilised for implementing a multinomial NB and for calculating model-wide performance metrics (Pedregosa, F. et al. 2011).

2.5.3 Transformers

This module provides access to a plethora of transformers and transformer architectures that have been developed and/or pre-trained for a selection of complex tasks from HuggingFace, Inc. This investigation imported and experimented with BERT in particular (BERT, n.d.).

2.5.4 Pytorch

In order to build neural networks, as well as create tensors and organise datasets into formats that complex neural networks (e.g. transformers) can use, one of the best options is Pytorch. This module was chosen for its ability to build neural networks, turn encodings into tensors, and create dataset objects (Paszke, A. et al. 2019). This was also used for defining a custom classification head on top of a frozen BERT.

3 Results

3.1 Naïve Bayes

The first iteration tested was just with tokenisation, punctuation and stop-word removal before creating the vocabulary in training. Initially, an attempt to create a training set, which was made up of 80% of the entire dataset, was conducted. Unfortunately, the attempt to create feature vectors for every sample, in a dataset that size, proved too computationally expensive. This meant that the code failed at runtime, requiring over twenty-three gigabytes of memory.

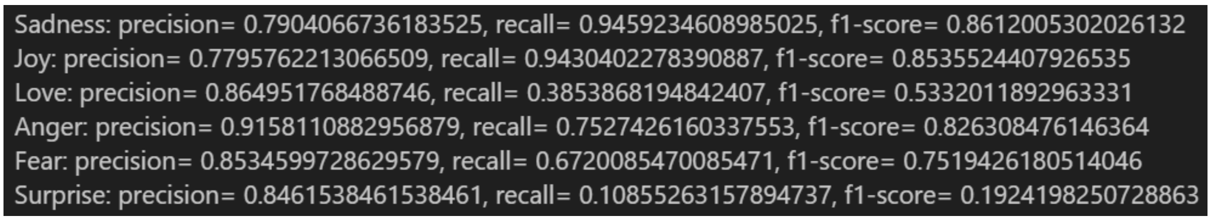
This was unsurprising considering the program was attempting to represent 320,000 tweets at once with dense vectors. These vectors were of a size proportional to the number of tweets used to create the vocabulary. To make the process manageable, a decision was made to use a much smaller sample of about 10% of the full dataset. This still resulted in over 40,000 tweets for training which proved to be more than enough data. Furthermore, leaving some data unused proved useful during later iterations (See 3.1.2)

3.1.1 Punctuation & Stop-word removal

The first iteration that was trained used data that had punctuation and stop-words removed. For the stop-words, the English stop-word list provided by NLTK was used. The scores this yielded were as follows:

Accuracy = 0.809 *F1-Score = 0.670 (3.s.f)*

It was observed that there was a large discrepancy between the two scores. The F1-score being so much lower than Accuracy should indicate that the model is underperforming on one or more individual classes. These classes can be discerned from the class specific precision, recall and F1-Scores.



```
Sadness: precision= 0.7904066736183525, recall= 0.9459234608985025, f1-score= 0.8612005302026132
Joy: precision= 0.7795762213066509, recall= 0.9430402278390887, f1-score= 0.8535524407926535
Love: precision= 0.864951768488746, recall= 0.3853868194842407, f1-score= 0.5332011892963331
Anger: precision= 0.9158110882956879, recall= 0.7527426160337553, f1-score= 0.826308476146364
Fear: precision= 0.8534599728629579, recall= 0.6720085470085471, f1-score= 0.7519426180514046
Surprise: precision= 0.8461538461538461, recall= 0.10855263157894737, f1-score= 0.1924198250728863
```

Figure 8

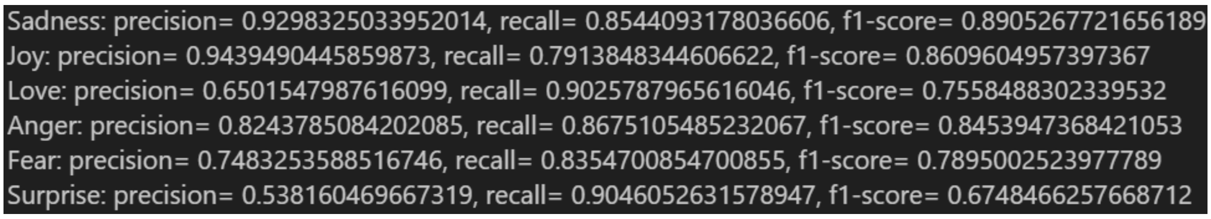
From the results in figure 8, it is clear that the drastically low recall scores on ‘Surprise’ and ‘Love’ are the main factors that brought down the F1-Score. Recalling figure 4, it is known that ‘Love’ and ‘Surprise’ were the smallest classes in terms of the samples that they contain within the dataset. When using a Naïve Bayes model, this results in the ‘prior probability’ being lower, which in turn means that a higher degree of certainty in the feature vector is required to yield the highest ‘posterior probability’ with that class (refer to section 2.3.1). This is proven further in the investigation by the high precision scores for those two classes, which indicate that the model was only attributing said classes to a sample when it was all but certain of that attribution.

3.1.2 Equalising the proportion of classes for training

One way to combat the unhelpfully low ‘prior probabilities’ of the smaller classes would be to use a training dataset of a similar size but has an even split of each class. This would equalise the ‘prior probability’ between all the classes without losing out on training data. Thanks to the immense size of the full dataset, compared to the fraction of it that was used for training and validation, this could be done without creating a crossover between the training and validation set. Here are the results of training with a dataset of said formation.

Accuracy = 0.839 *F1-Score = 0.803 (3.s.f)*

Accuracy has improved by 3 percentage points since the previous iteration. More importantly, however, the F1-Score has improved significantly by 13.3 percentage points, just by changing the proportion of all the classes contained in the training data.



```
Sadness: precision= 0.9298325033952014, recall= 0.8544093178036606, f1-score= 0.8905267721656189
Joy: precision= 0.9439490445859873, recall= 0.7913848344606622, f1-score= 0.8609604957397367
Love: precision= 0.6501547987616099, recall= 0.9025787965616046, f1-score= 0.7558488302339532
Anger: precision= 0.8243785084202085, recall= 0.8675105485232067, f1-score= 0.8453947368421053
Fear: precision= 0.7483253588516746, recall= 0.8354700854700855, f1-score= 0.7895002523977789
Surprise: precision= 0.538160469667319, recall= 0.9046052631578947, f1-score= 0.6748466257668712
```

Figure 9

The F1-Score has increased across every class by varying degrees. As expected, the difference is the greatest in the classes that were under-represented in the original training data. Their F1-Scores are still the lowest out of all the classes. This is most likely due to how low their precision scores became, but there was still significant improvement overall.

3.1.3 Lemmatisation

Here, the aim was to test the impact that Lemmatisation has on both the performance of the model, but also the size of the vocabulary and resulting feature vectors. The results were:

Accuracy = 0.828   F1-Score = 0.791 (3.s.f)

Feature Vector length without Lemmatisation: 25194

Feature Vector length with Lemmatisation: 20728

Compared to the model trained without a lemmatised set of data, this did perform slightly worse by about 1 percentage point in both of the main metrics. Interestingly, the lemmatisation resulted in a feature vector about 18% smaller than before, indicating that a relatively large amount of morphological information had been lost. This is likely the cause of the slightly decreased Accuracy and F1-Score. In view of these findings, lemmatisation was not used for the next two iterations of NB.

3.1.4 TF-IDF Vectors

Rather than using feature vectors that simply measured term frequency, an experiment was conducted using TF-IDF vectors (refer to 2.3.2.4). The intention being to attribute more importance to features that occur more rarely, in turn using those rarer features as a sign of the classes they occur in.

Accuracy = 0.838   F1-Score = 0.796 (3.s.f)

The Accuracy was just slightly worse than the current record by about 0.1 percentage points, whereas the F1-Score decreased by about 0.5 percentage points. This outcome was surprising, as it was expected that the TF-IDF values would allow the classifier to pay more attention to the features that are more important.

3.1.5 Log Smoothed class proportions

Referring back to Figure 8, it can be noted that the main culprit that brought down the performance of the current best iteration was the low precision i.e. over-classification of the small classes. As mentioned earlier, this was due to the inflated 'prior-probabilities' put in place to combat the previous under-classification (low recall). In an attempt to find an acceptable middle ground, a decision was made to take the true ratio present in the true dataset, and apply the natural log function to smooth it. Then, a multiplier was applied to reach a total similar to the original dataset. In practice, the ratio of the new training dataset looked like this:

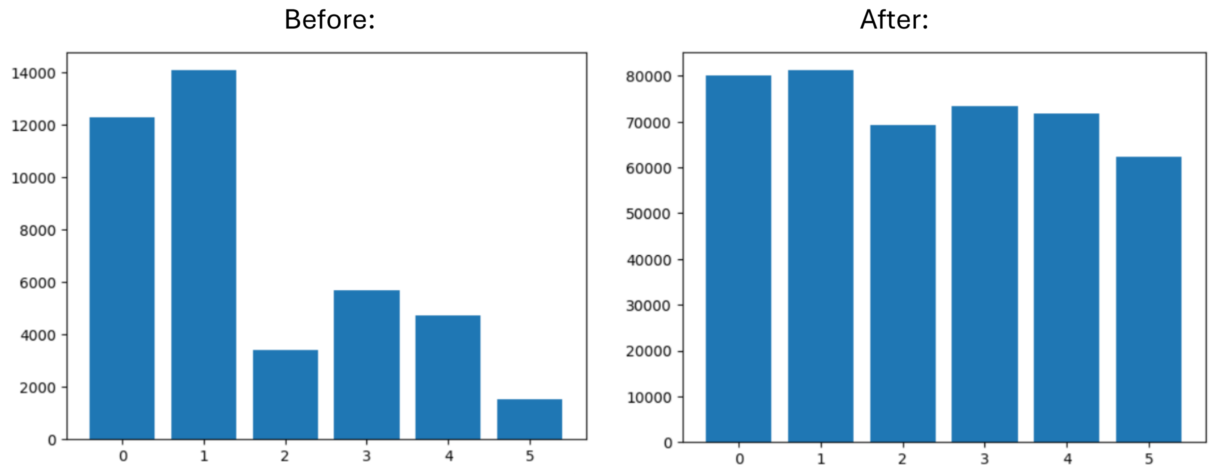
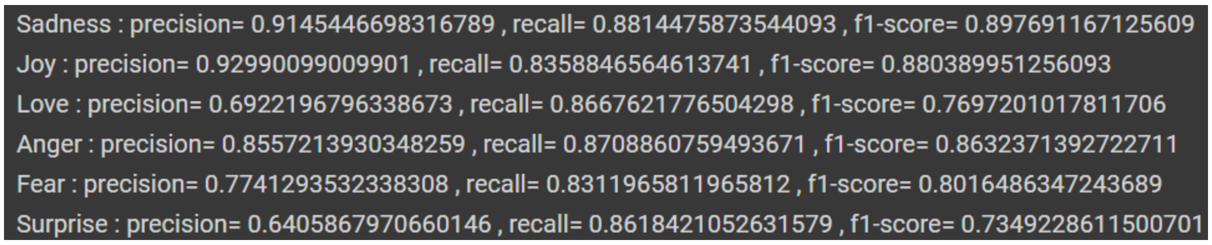


Figure 10

The total of the former training dataset was 41681, while that of the latter was 43813. The scores of training the classifier with this training dataset were as follows.

Accuracy = 0.857   *F1-Score = 0.825 (3.s.f)*

As you can see, this technique has a had a positive effect and garnered the best iteration of Naïve Bayes in this investigation. To understand why, the class specific scores are shown in figure 11.



```
Sadness : precision= 0.9145446698316789 , recall= 0.8814475873544093 , f1-score= 0.897691167125609
Joy : precision= 0.92990099009901 , recall= 0.8358846564613741 , f1-score= 0.880389951256093
Love : precision= 0.6922196796338673 , recall= 0.8667621776504298 , f1-score= 0.7697201017811706
Anger : precision= 0.8557213930348259 , recall= 0.8708860759493671 , f1-score= 0.8632371392722711
Fear : precision= 0.7741293532338308 , recall= 0.8311965811965812 , f1-score= 0.8016486347243689
Surprise : precision= 0.6405867970660146 , recall= 0.8618421052631579 , f1-score= 0.7349228611500701
```

Figure 11

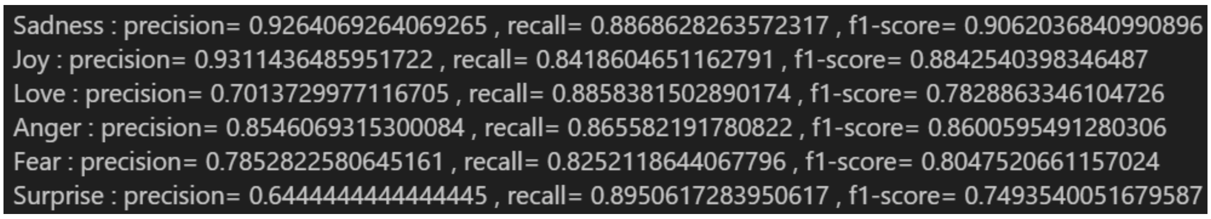
Compared to the previous best iteration, the precision has increased in the smaller classes (‘Love’ and ‘Surprise’). The recall has suffered slightly, but the F1-Score for these classes has benefited overall.

3.1.6 Best NB on test data

For a final evaluation, the best implementation of Naïve Bayes was used on some completely unseen test data i.e. data that had not been used to inform optimisation decisions. The scores were as follows:

Accuracy = 0.862   *F1-Score = 0.831 (3.s.f)*

On this particular spread of data, the model seems to perform even better than on the validation data. This is likely due to the test data happening to contain more features that were present in the training data, meaning that the model is more prepared to make predictions based on said features.



```
Sadness : precision= 0.9264069264069265 , recall= 0.8868628263572317 , f1-score= 0.9062036840990896
Joy : precision= 0.9311436485951722 , recall= 0.8418604651162791 , f1-score= 0.8842540398346487
Love : precision= 0.7013729977116705 , recall= 0.8858381502890174 , f1-score= 0.7828863346104726
Anger : precision= 0.8546069315300084 , recall= 0.865582191780822 , f1-score= 0.8600595491280306
Fear : precision= 0.7852822580645161 , recall= 0.8252118644067796 , f1-score= 0.8047520661157024
Surprise : precision= 0.6444444444444445 , recall= 0.8950617283950617 , f1-score= 0.7493540051679587
```

Figure 12

Like before, the main areas that are bringing the score down are the precision of the two smaller classes: Love and Surprise. This is to be expected as the prior-probabilities of these two classes have been artificially increased to remedy their under-classification. While this has obviously resulted in a net-positive, one side effect is bound to be low precision.

3.2 Fine-Tuned BERT

As training BERT takes a significant amount of time and computational resources, only two main different iterations were trained. One where all parameters were updated, and the other where only the output layer was updated.

3.2.1 Unfrozen iteration

The first iteration implemented was one where no layer was frozen. This meant all the parameters within the network, including that which have been pre-trained, were free to update to fit the task at hand. While this could have allowed the model to gain more optimal performance, it was possible that this could cause not only over-fitting, but also catastrophic forgetting.

3.2.1.1 Full dataset

Initially, the attempt was to train with a full-sized training dataset i.e. 60% of the full dataset or roughly 320,000 tweets. After leaving it to train for over five hours, it had not even reached half-way to completion. Due to this, something similar was done as with Naïve Bayes, taking 15% of the full dataset. This still amounted to 62521 tweets. The validation and test datasets were composed of 3% or 12504 tweets each.

3.2.1.2 Hyper-parameters

During the first training session, the following hyper-parameters were used:

- Epochs: 3
- Batch size: 128
- Weight Decay: 0.01
- Learning rate: 5e-5 (default)

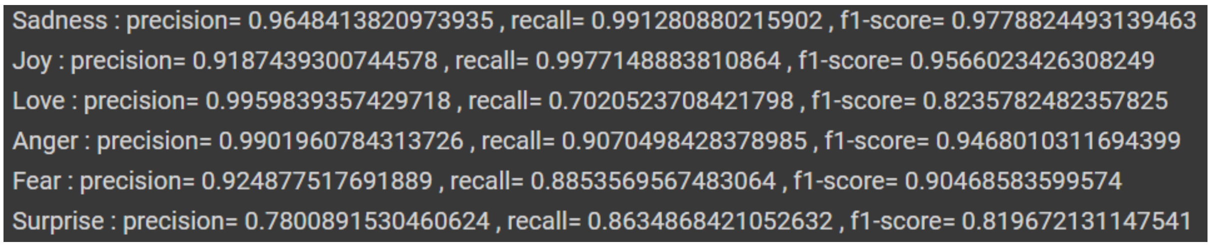
As stated in 2.4.4, Epochs are full passes that the training set will make through the network during training. Whereas batch size is how many samples can pass through at a time i.e. how quickly a full Epoch will occur. Regarding weight decay, it is a regularisation measure that stops the parameters overfitting to the training data. Finally, learning rate details how much the parameters will update during each forward pass.

3.2.1.2 Scores

As expected, this model supremely outperformed the arguably 'simple' Naïve Bayes model. The scores were as follows:

Accuracy = 0.941   *F1-Score = 0.905 (3.s.f)*

Compared to the best NB implementation, the Accuracy has increased by over 9 percentage points, whereas the F1-Score has increased by 8 percentage points. Interestingly, according to the training logs, the optimal version of the model was already reached at some point during the second epoch. When reproducing this model, the 'Number of Epochs' could therefore be set to 2.



```
Sadness : precision= 0.9648413820973935 , recall= 0.991280880215902 , f1-score= 0.9778824493139463
Joy : precision= 0.9187439300744578 , recall= 0.9977148883810864 , f1-score= 0.9566023426308249
Love : precision= 0.9959839357429718 , recall= 0.7020523708421798 , f1-score= 0.8235782482357825
Anger : precision= 0.9901960784313726 , recall= 0.9070498428378985 , f1-score= 0.9468010311694399
Fear : precision= 0.924877517691889 , recall= 0.8853569567483064 , f1-score= 0.90468583599574
Surprise : precision= 0.7800891530460624 , recall= 0.8634868421052632 , f1-score= 0.819672131147541
```

Figure 13

Above are the class specific scores. As with before, the lowest performing classes are 'Love' and 'Surprise'. It seems that, due to their relative rarity, they are harder for these models to predict as efficiently as the other classes. That being said, the model is still performing well on them even if they are the worst of the bunch.

3.2.2 Classification Head on Frozen BERT

The last experimentation that was conducted was with a classification head above the pre-trained BERT, where the BERT layer was prevented from updating its parameters leaving only the classification head to be updated.

3.2.2.1 3 Epochs

The first two attempts used a ReLU and Tanh (Apicella, A. et al. 2021) activation function respectively for the final layer of the classification head. Neither iteration garnered an Accuracy above 40%, so the scores were not analysed any further. The loss did, however, seem to be decreasing ever so slightly, as in figure 14. This could indicate that due to the restricted flexibility of the model, compared to the previous, I need to give it more time to train itself.

Step	Training Loss	Validation Loss	Accuracy
400	1.712	1.594	0.346
800	1.601	1.564	0.378
1200	1.587	1.557	0.393

Figure 14

3.2.2.2 15 Epochs

Here, 3 things were changed:

- Number of Epochs: 3 -> 15
  This was to give the model significantly more time to optimize itself
- Batch Size: 128 -> 256
  To quicken the training process
- Final Activation: Leaky ReLU (Apicella, A. et al. 2021)
  To allow outputs below zero to still have a continuous value, therefore allowing the gradient of the loss to be computed in these cases.
- Learning Rate: 5e-5 -> 1e-3
  This increase in learning rate should allow the model to update the parameters by a greater margin each at each step.

While the model did definitely achieve more than the previous attempt, it still pales in comparison to the fully fine-tuned BERT.

Step	Training Loss	Validation Loss	Accuracy
800	1.542	1.406	0.478
1600	1.456	1.376	0.491
2400	1.447	1.372	0.489
3200	1.444	1.365	0.493

Figure 15

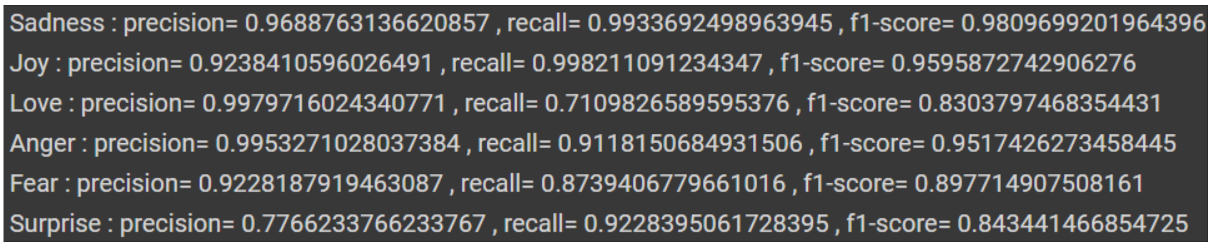
As seen in figure 15, it has still failed to reach an accuracy of even 50%. However, as the loss did not seem to reach a minimum, it is safe to assume that with more epochs that this could have reached a better performance. However, this was not experimented with this further. Because the amount of time it would take to train the model at this rate to even get close to the fully fine-tuned BERT would be too great. This would completely defeat the point of attempting this method in the first place.

3.2.3 Best BERT on test data

Like with Naïve Bayes (NB), the best implementation of BERT was evaluated on test data.

Accuracy = 0.944 *F1-Score = 0.911*

In a similar vain to NB, the performance on this dataset was even better than on the validation. This was to be expected considering the NB results, as it indicated that the test data had more features in common with the training data than the validation data does.



```
Sadness : precision= 0.9688763136620857 , recall= 0.9933692498963945 , f1-score= 0.9809699201964396
Joy : precision= 0.9238410596026491 , recall= 0.998211091234347 , f1-score= 0.9595872742906276
Love : precision= 0.9979716024340771 , recall= 0.7109826589595376 , f1-score= 0.8303797468354431
Anger : precision= 0.9953271028037384 , recall= 0.9118150684931506 , f1-score= 0.9517426273458445
Fear : precision= 0.9228187919463087 , recall= 0.8739406779661016 , f1-score= 0.897714907508161
Surprise : precision= 0.7766233766233767 , recall= 0.9228395061728395 , f1-score= 0.843441466854725
```

Figure 16

As expected, considering the overall performance, the scores for the individual classes were all quite high. Interestingly, while the two smallest classes were yet again those with the lower scores, they do not seem to be faltering in the same way. ‘Love’ for instance, had a relatively low recall score with a nearly perfect precision score, indicating that there were a considerable number of samples within the Love class that aren’t getting classified as such. But when the classifier does attribute Love to a sample, it is rarely wrong. For ‘Surprise’ however, the opposite is true. The low score was precision, whereas recall was much higher, indicating that Surprise is often getting over classified.

4 Discussion

4.1 Naïve Bayes Comparison

Overall, the Naïve Bayes model significantly outperformed expectations. It was not expected that it would reach much higher than 70%, especially considering that the problem is multinomial rather than binary.

Naïve Bayes results

	3.1.1	3.1.2	3.1.3	3.1.4	3.1.5
Accuracy	0.809	0.839	0.828	0.838	0.857
F1-Score	0.670	0.803	0.791	0.791	0.825

Figure 17

As you can see from the table above, the only iterations that garnered any improvement were 3.1.2 and 3.1.5, both of which implemented simple changes in the content of the training dataset in order to manipulate the 'prior probabilities' to be calculated in the training process. The findings that the process of using TF-IDF vectors impaired the results were surprising, considering Vidhya et al. (2015) success.

4.1.1 Failures of Lemmatisation and TF-IDF

For lemmatisation, the assumption was, that due to how short the data samples were (tweets have a 280-character limit on free accounts, 4000 on premium), the morphological information (Steele, S. 1995) that was lost due to lemmatisation may have carried more weight/information. This is in contrast to tasks where the data samples are more long-form e.g. articles, reviews etc.

Regarding TF-IDF vectors, there is a pitfall (amongst others) where this technique can suffer. The 'inverse-document-frequency' (IDF) portion of the formula does not take into account how often a word occurs within each of the other documents i.e. the 'intensive margin'. Instead, it only considers whether it occurs at all in said documents i.e. the 'extensive margin' (Scott, J. 2021). This means that it can cause the model to be overly sensitive to the extensive margin, and under sensitive to the intensive. Another pitfall of TF-IDF would be its inability to represent sequential semantic data, but this is true for all of the Naïve Bayes iterations in this investigation, therefore it cannot be the point of failure in this case.

4.1.2 Prior-probability manipulation

To demonstrate the effect that changing the proportions of each class within the training dataset had on the resulting model, the different values of the prior probabilities for each model are shown in figure 18. First, recall the formula.

$$Posterior\ Probability = \frac{conditional\ probability\ \times prior\ probabiilty}{evidence}$$

Prior Probability Values

	Sadness	Joy	Love	Anger	Fear	Surprise
3.1.1	0.294	0.337	0.082	0.136	0.113	0.037
3.1.2	0.167	0.167	0.167	0.167	0.167	0.167
3.1.5	0.182	0.185	0.157	0.168	0.164	0.142

Figure 18

As can be seen in the first iteration, the prior probabilities for each class vary greatly. The largest class, Joy, was approximately eleven times larger than the smallest, Surprise. The consequence of this inequality was demonstrated through the extremely low recall scores for Surprise and Love.

While equalising the priors in iteration 3.1.2 did yield much better performance, the smaller classes still lacked. However, unlike before where they lacked in recall, they now lacked in precision. Smoothing the ratios in the final iteration, and the priors by extension, allowed the priors to loosely reflect the higher frequency of the larger classes without making it too improbable to assign the smaller classes to a sample.

4.2 Fine-Tuned BERT Comparison

It was unsurprising that BERT significantly outperformed Naïve Bayes, primarily because it is able to represent and take into account sequential information, as well as the semantic similarity between words. What was surprising, however, was how poorly BERT performed when all the parameters were frozen apart from those within the classification head. The expectation was that the outputs of the pre-trained pooling layer would be sufficient for a classifier to work with, but clearly that was not the case with the given hyper-parameters.

NOTE: For the two iterations with a frozen BERT layer, F1-Score was not measured, as their accuracies were too low for them to be worth considering.

BERT Results

	3.2.1	3.2.2.1	3.2.2.2
Accuracy	0.941	0.393	0.493
F1-Score	0.905	N/A	N/A

Figure 19

The failure of the frozen BERT iterations could indicate that the masked language modelling (MLM) task, that BERT is pre-trained on, is too far removed from the relatively simple task of emotion classification. The values of the parameters learned for said task seem not to be suited to responding to the information that was critical to the task.

4.3 Conclusion

The aim of this investigation was to discern which of the two models, Naïve Bayes and BERT, were better for performing text classification on the dataset. This dataset consisted of over 400,000 tweets, all of which were paired with a number that indicates the emotion conveyed in the given tweet. The six emotions were: Sadness, Joy, Love, Anger, Fear, and Surprise. Specifically, the goal was to see what implementations of these two methods performed the best on this task.

Naïve Bayes performed best with a simple bag-of-words implementation with log-smoothed class proportions in the training dataset. This iteration outperformed methods containing preprocessing techniques such as TF-IDF vectors and lemmatisation, or that with a dataset containing standard class proportions. Regarding BERT, the best iteration was when all of the parameters were allowed to freely update and optimize for the given task. The iterations consisting of a frozen BERT layer under a classification head failed to generate any respectable result.

While the results between the two were closer than expected, there was a clear (and expected) winner between the two. BERT conclusively outperformed Naïve Bayes (NB) in every metric possible on both validation and test data.

BERT	Validation	Test
Accuracy	0.941	0.944
F1-Score	0.905	0.911

NB	Validation	Test
Accuracy	0.857	0.862
F1-Score	0.825	0.831

Figure 20

BERT's observed victory could indicate that its ability to extract sequential information, as well as represent semantic similarity between different words, allows it to spot more complex patterns in the dataset. BERT seemed to be able to use those patterns to make more accurate predictions. That being said, NB's performance was not bad by any means, and indicates that, for a task like emotion classification, it can be sufficient where access to powerful computation is limited. In conclusion, BERT is the superior method by any measurement for this task, but Naïve Bayes is still a viable option.

4.4 Further work

Based on the investigation outlined in this report, there are many directions where further research could be conducted. There are a multitude of classification methods other than those which were experimented with here. Evaluating their performance on the same dataset could provide further insight into their strengths and weaknesses.

For example, Sharma et al. (2019) evaluated the performance of a Convoluted Neural-Network (CNN) when classifying Bangla documents as satirical or not. Before input to the CNN, they used a 3D feature vector representation of the shape 1000x10x2. Their implementation achieved an accuracy of 96% on test data. This method could be applied to the dataset used in this investigation. However, there may be issues with the feature representation that they used when applied to my dataset. This is because the documents used, in this project, are much smaller than those used by Sharma et al (2019).

As discussed in 3.2.2, the implementation of a classification head on top of BERT with frozen parameters severely underperformed in this investigation. One direction of further research could include developing a more complex classification head to use on top of BERT e.g. a CNN. It may be the case that, due to the inability for the pre-trained parameters to optimise themselves for the new task, there needs to be more processing done, after the final pooling layer, to be able to extract the relevant information for the classification task. However, judging by the fully fine-tuned model's performance on unseen data, overfitting did not seem to be an issue, so catastrophic forgetting did not seem to have a negative effect on the final outcome. This would indicate that an implementation with a frozen BERT layer is unlikely to outperform its unfrozen counterpart.

Another direction could be to reevaluate the proposed methods in this investigation on different types of data. In this case, the length of the data samples is generally quite short. It would be interesting to determine how differently the models trained in this project would perform on longer forms of text that are still categorised into the same six emotions. Or, the same implementations could be retrained on a new dataset of text with a new group of classes entirely, to see if the performance is comparable.

Appendices

Code

I submitted two notebooks as well as the full dataset all within a zip folder. Nobo.ipynb contains all experimentation done with Naïve Bayes, and bert.ipynb contains all the experimentation done with BERT. There are some instances where code may need to be altered to access the current directory if and when it is being tested, which are clearly marked.

Bibliography

Amari, S. I. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5), 185-196.

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, 138, 14-32.

Balakrishnan, V., & Lloyd-Yemoh, E. (2014). Stemming and lemmatisation: A comparison of retrieval performances.

BERT. (n.d.). Huggingface.co. https://huggingface.co/docs/transformers/en/model_doc/bert

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python: analyzing text with the natural language toolkit. " O&#x27;Reilly Media, Inc."

Chen, Y. S., Huang, Y. H., Liu, H. C. T., Wu, J. (2018). Contextualized Affect Representations for Emotion Recognition", *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,* 3687-3697, Association for Computational Linguistics.

Comon, P. (2014). Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3), 44-53.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20, 273-297.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*.

Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3), 146-162.

Kalyan, K. S. (2023). A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal*, 100048.

Kowsari, K., Jafari Meimandi, K., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text classification algorithms: A survey. *Information*, 10(4), 150.

Mao, A., Mohri, M., & Zhong, Y. (2023, July). Cross-entropy loss functions: Theoretical analysis and applications. *In International Conference on Machine Learning* (pp. 23803-23828). PMLR.

Martinez, A. R. (2012). Part-of-speech tagging. *Wiley Interdisciplinary Reviews*: Computational Statistics, 4(1), 107-113.

Li, W., & Xu, H. (2014). Text-based emotion classification using emotion cause extraction. *Expert Systems with Applications*, 41(4), 1742-1749.

OpenAI. (2024). ChatGPT [Large language model]. https://chat.openai.com

Palo, H. K., Sahoo, S., & Subudhi, A. K. (2021). Dimensionality reduction techniques: Principles, benefits, and limitations. *Data Analytics in Bioinformatics: A Machine Learning Perspective*, 77-107.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12, 2825-2830.

Raschka, S. (2014). Naive bayes and text classification I-introduction and theory. *arXiv preprint arXiv:1410.5329*.

Scott, J. (2021). What's in a word? Retrieved from https://towardsdatascience.com/whats-in-a-word-da7373a8ccb#:~:text=TL%3BDR%3A%20Term%20Frequency%2D,resistant%20on%20the%20intensive%20margin.

Selva Birunda, S., & Kanniga Devi, R. (2021). A review on word embedding techniques for text classification. *Innovative Data Communication Technologies and Application: Proceedings of ICIDCA 2020*, 267-281.

Silva, C., & Ribeiro, B. (2003, July). The importance of stop word removal on recall values in text categorisation. *In Proceedings of the International Joint Conference on Neural Networks*, 2003. (Vol. 3, pp. 1661-1666). IEEE.

Soyalp, G., Alar, A., Ozkanli, K., & Yildiz, B. (2021, September). Improving text classification with transformer. In 2021 *6th International Conference on Computer Science and Engineering* (UBMK) (pp. 707-712). IEEE.

Steele, S. (1995). Towards a theory of morphological information. *Language*, 260-309.

Tripathi, P. (2023) 'Everything you need to know about document classification [Complete Guide]', *docsumo*, 19 October. Available at: https://www.docsumo.com/blog/document-classification (Accessed: 17 November 2023).

Vidhya, S., Singh, D. A. A. G., & Leavline, E. J. (2015). Feature extraction for document classification. *International Journal of Innovative Research in Science, Engineering and Technology*, 4(6), 50-56.

School of Engineering and Informatics

SECONDARY DATA ANALYSIS COMPLIANCE FORM FOR UG AND PGT PROJECTS

This form must be completed before any analysis of secondary data may begin.

Introduction

This form should be read in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects (V4.0)".

This form is for project that will use *secondary data*, that is pre-existing data. For projects that will collect new data, either the *User-testing Compliance Form* must be completed, or a full Low-risk or High-risk ethics application made to the Science and Technology Research Ethics Committee.

There are four types of secondary data:

- 1A: Meta-analysis – data obtained directly from published peer-reviewed articles
- 1B: Publicly available anonymised data sets uploaded to open access repositories or databases
- 1C: Secondary datasets available from previous School or Sussex University (ethics approved) projects
- 1D: Secondary datasets not publicly available but which may be accessed under certain conditions/agreements

Figure 1 is a decision flow chart of when this form can be used. The four types of secondary data listed at the top of the flow chart: boxes 1A-1D.

Both **meta-analyses** (1A) and **publicly available anonymized datasets** (1B) do not require a University ethics review. Instead, you should complete the Secondary Data Analysis Compliance Form (Box 3A).

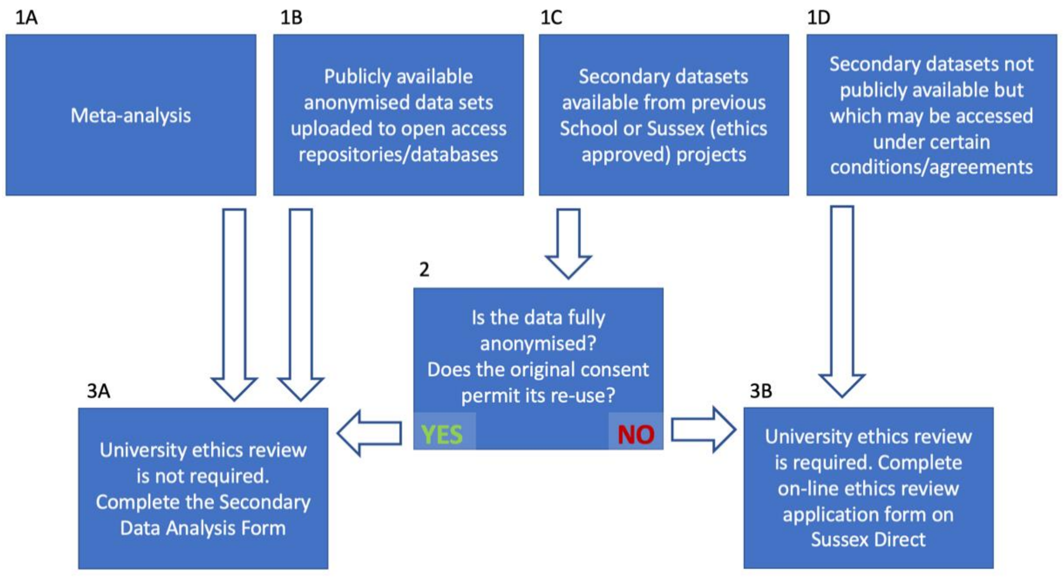

1A
Meta-analysis
1B
Publicly available anonymised data sets uploaded to open access repositories/databases
1C
Secondary datasets available from previous School or Sussex (ethics approved) projects
1D
Secondary datasets not publicly available but which may be accessed under certain conditions/agreements
2
Is the data fully anonymised? Does the original consent permit its re-use?
YES
NO
3A
University ethics review is not required. Complete the Secondary Data Analysis Form
3B
University ethics review is required. Complete on-line ethics review application form on Sussex Direct

Figure 1. Secondary Data Analysis Flow Chart

For secondary **datasets available from previous School or Sussex (ethics approved) projects** (1C), you need to determine whether the data is anonymized and if the original consent form permits its re-use (Box 2).

- If the answer to both in Box 2 is 'Yes', then a University ethics review is not required, and you should complete the Secondary Data Analysis Compliance Form (Box 3A).
- If the answer to both in Box 2 is 'No', then you must complete an online ethics review application form on Sussex Direct (Box 3B).

For secondary **datasets not publicly available** but which may be accessed under certain conditions/agreements, you will need to make an online ethics review application (Box 3B).

Part 1: Determining whether you can use this Secondary Data Analysis Compliance form or whether you need to submit a full ethics application via Sussex Direct.

You can use this form if your project involves (1) AND/OR (2) below (double click the check box to tick):	
1. A meta-analysis.	☐
2. Using publicly available anonymized datasets (e.g., data obtained from open access repositories or databases). URL of dataset: https://www.kaggle.com/datasets/nelgiriyewithana/emotions	☒

You can use this form if your project involves research described in (3), **and** meets **BOTH** criteria A and B below:	
3. Analysing datasets that were originally collected by researchers at the University of Sussex, according to a previously approved ethics application. The data to be analysed also meets both of the following criteria: A. The data is fully anonymised. Yes / No (delete the inapplicable) B. The original ethics application permits the re-use of data collected. Yes / No Sussex Ethics Application Number (e.g., ER/XYZ99/1): Add Number Sussex Ethics Application Project Title: Add Title If the dataset does not meet BOTH criteria A and B, then you will need to apply for an ethics review on Sussex Direct.	☐

You CANNOT use this form if your project involves the following (4):	
4. Analysing datasets that are not publicly available and not originally collected by Sussex researchers. The datasets can only be accessed according to conditions/agreements arranged by your supervisor or the University. In this scenario, you will need to apply for ethics approvals on Sussex Direct, regardless of how the data is anonymised. The ethics review will also consider whether the re-use of the data is permitted according to the original ethics application.	☐

Based on your answers above, please either complete Parts 2-4 of this form or discuss with your supervisor about submitting an ethics application via Sussex Direct.

Part 2: Project information

Project Title:	An exploration into short form text classification methods
Student:	Conrad Lovegrove 246504
Supervisor:	Jeff Mitchell
Start & End Dates:	02/10/2023-09/05/2024

Part 3: Project description and ethical considerations

I am aiming to explore and test different methods for classifying short form text. The dataset I want to use is publicly available, and consists of a large collection of anonymised tweets attached to a label indicating the emotion conveyed in the tweet. The main ethical consideration to be made is that the data was collected in such a way that the identity of the original authors of each tweet cannot be determined. The data is fully anonymised and it is impossible to discern the identity of the participants.
Reflection on ethics of data collection (up to 2000 characters)
Consider the procedures performed by the individuals who originally collected the data you'll analyse. We recommend discussing how the researchers ensured participant wellbeing, consent, & confidentiality. Reflect on the four ethical principles for psychologists that are outlined in the *BPS Code of Ethics*: (1) Respect, (2) Competence, (3) Responsibility, and (4) Integrity. You can also refer to specific *BPS guidelines and policies*, including the *Code of Human Research Ethics*. All contributors to the dataset were expected to abide by the Hugging face code of conduct. The standards they were expected to abide by are: "Examples of behavior that contributes to a positive environment for our community include: Demonstrating empathy and kindness toward other people Being respectful of differing opinions, viewpoints, and experiences Giving and gracefully accepting constructive feedback Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience Focusing on what is best not just for us as individuals, but for the overall community Examples of unacceptable behavior include: The use of sexualized language or imagery, and sexual attention or advances of any kind Trolling, insulting or derogatory comments, and personal or political attacks Public or private harassment Publishing others' private information, such as a physical or email address, without their explicit permission Other conduct which could reasonably be considered inappropriate in a professional setting" They enforced these guidelines through a ban system, ensuring that integrity was maintained.

Part 4: Approvals

Role	Name	Signature	Date
Student	Conrad Lovegrove	Conrad Lovegrove	09/05/2024
Supervisor	Jeff Mitchell	Jeff Mitchell	10/05/24

Eng&Inf SREO, V1.0, Oct 2023. Adapted from the School of Psychology "Ethics form for UG/PGT projects involving secondary data analysis (not NHS-related)"