Explanation-Driven Text Classification and its Application to Emotion Detection

Shen Chen

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2024

Abstract

Improving the Neural Network's performance has been one of the main focus of the Machine Learning fields. Due to the "black box" nature of machine learning models, i.e. the hidden inner states and the undisclosed reasoning process, it's a challenge for researchers to locate the root of performance bottleneck.

In this project, I focus on explainable methods for the Neural Network classifier and benchmark the efficiency of a model updating pipeline based on explanations that reveal the inference process. For the test purpose, the project specifically focuses on Emotion Classification, however, the very approach can be easily extended to sentiment or entailment.

The main contributions of the projects are:

- An explainability method which hierarchically detects the feature interaction scores on the sentence level and generates interpretable explanations for emotion classification.
- Data augmentation based on generated explanations and examine the efficiency compared to regular augmentation methods.
- Propose an Emotion Classification metric that considers the similarity between emotions and better captures the errors while considering the nature of emotions.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

The data and model used in this project are publicly available.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Shen Chen*)

Acknowledgements

I would like to express my gratitude to my supervisor Mirella Lapata, who constantly pointed me towards the correct direction and offered insightful hints which helped to gradually form a complete picture of a self-contained project. She overlooked the project's progress and gave me extremely valuable feedback throughout all the stages of my work. I'm especially thankful for her accepting my self-proposed project and walking through the idea together, raising issues that might impact the outcome and stirring the direction of the project to a more meaningful area. This has been extremely important regarding efficient time usage and resource investment. She also provided university-funded tooling that enabled me to finish my project.

I would also thank my second marker Pasquale Minervini, who came up and proposed an interesting sub-topic to investigate further into the direction of my project. His advice was critical in terms of avoiding time waste. I hereby express my gratitude.

Finally, I would like to thank all my friends who supported each other during this stressful time.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

”AI”, a word that is getting more and more attention in modern life, is gradually making its way from fictional works into our daily lives. It’s no longer surprising to see products that have existed for centuries embrace and embed themselves with the power of AI. From AI-powered self-driven automobiles to the human-alike ChatBot such as ChatGpt, our world is entering a new era where human is no longer the only entity in the process of decision-making. Sometimes we even let the AI make the decisions for us entirely due to our trust in its ability to analyse data that are several magnitude larger than the amount humans can process therefore reaching a better decision after seeing a ”greater picture”. However, how could we know if they are making the correct decision, what is the process of AI concluding and whether the process is troublesome or not, how can we **trust** AI?

To answer this question, we need to demystify ”AI”. What exactly is ”AI”? It is not a creature that resides in the world of the internet and thinks in the same way as humans as portrayed in the movies or novels. Underneath, it is powered by Machine Learning Models, which were given an extremely huge amount of data to study, then uncovering the patterns and making reasonable decisions when given unseen data by leveraging the observations they found from the training data. These models are, simply putting, many layers of complex mathematical computations involving coefficients that are called **weight**, that are applied to the input we gave to the model. Therefore it’s difficult to trace back and understand the whole process before reaching the output as the human brain simply can’t link a series of matrices of numbers with the task that was assigned to the model itself. And with the increasing complexity of ML models, it’s getting even more impossible to eyeball the raw model parameters and make sense of them. This is the reason why ML models are considered ”black-box” as humans can only interpret it as some magical process that takes in the input and returns an output.

Without the means to understand the reasoning process, we can’t evaluate the model properly with only the pairing between the input and output, therefore no trust should be given since all the predictions made by the model can just be luck. We need a human-comprehensible way to showcase the reasoning process of models to examine it

thoroughly. This is where the topic of ”Explanable AI” comes into play.

The main focus of this project is the Emotion Classification Task, where the input is sentences and the corresponding output is the emotion that may be conveyed. The current Explainable AI research is across various topics, while the research conducted on Image Explanation may be more accessible to the general public, those concerning Natural Language are also growing steadily.

Despite the existence of various media online nowadays, text is still the dominating medium that is used to relay information. Language models are walking into the spotlight and present in different corners of our lives, especially after the development of transformer architecture and subsequent powerful language models. Therefore, it is becoming more important for humans to understand and verify these language models are doing the correct thing. Much research has been conducted around this topic.

1.2 Objective

In this project, I investigated a method that is model-agnostic to present the reasoning process of the Language Classification Task. The goal is to output a comprehensible visualization that illustrates the stages of reasoning, demonstrating how our model handles the input and reaches a decision. We refer to the outputs of this algorithm **explanations** of our emotion classifier.

Furthermore, I tried to incorporate explanations to further improve the performance of the model by augmenting training data to cancel the model biases spotted during the verification stage. The goal is to expose mistakenly learnt patterns and intentionally fix the bias of the model rather than dumping resources and feeding more training data blindly to the model, which may pose little to no improvement, or even deteriorate the performance of the model. After incorporating the explanations we generated in the previous step, our model is expected to improve more effectively when fed the same amount of data as other data augmentation methods. I also take a slight detour during this step and take this opportunity to superficially examine the feasibility of replacing human feedback with GPT-generated synthetic data.

Next, I highlighted the nature of emotion and questioned the efficiency of the standard metric used to evaluate the Emotion Classification task. Naturally, I examined a new performance metric that considered the inner nature of emotion and captured the similarity between them.

Finally, I evaluated the performance of the classifier which is trained on data augmented based on explanations against ordinary data augmentation techniques, confirming if the approach is beneficial.

Chapter 2

Background

The goal of this project is to explore ways that can be used to reveal the inner process of machine learning models and demystify the ”black-box” nature of the models themselves. In addition, I will try to test the feasibility of using explanations as guidance to generate training data that can correct the model bias and mistakes in a more resource-friendly way.

2.1 Emotion

2.1.1 Definition

Emotions are states of mind controlled by neurophysiological changes, which lead to different thoughts, behaviours and feelings. It’s a form of conscious mental reaction that an individual experiences as a strong feeling normally directed towards a certain event, accompanied by physiological and behavioural changes.

People tend to show different reactions to the same topic based on the emotion they produce from it, different people generate distinct emotional responses and even the same person will react differently towards the same event at different times. The most common expressions of emotions are facial expressions, tone of conversation, body language etc.

To better illustrate and group emotions, a psychological PhD, Robert Plutchik, proposed a circular diagram illustrating the spectrum of human emotions and their interconnections, called the **Emotion Wheel**. It is obvious from the graph that the categorization of emotions is fine-grained hierarchically, the emotion embedded in a sentence can be intensive and strong, or vaguely and hard to perceive.

The Emotion Classification is built under the assumption that emotions that belong to a core exhibit a common pattern underneath. For example, ”Ecstatic over my stellar grade! Hard work pays off!” and ”I’m quite satisfied with the grade I got after all the hard work.”, while the former is more intensive as it uses a stronger tone than the latter, it’s not hard to notice that they are both conveying the emotion of ”joy” due to the presence of words such as ”ecstatic” and ”satisfied”.

Figure 2.1: Emotion Wheel. Core emotions reside at the centre, with less intensive variants of them radiating towards the outer edge.

Another example without the presence of explicit words associated with emotion would be ”You are pissing me off” and ”This is unacceptable!”, the former explicitly expresses the feeling of ”anger” by using the phrase ”pissing me off” and it’s easy for a human to perceive it correctly. In contrast, the latter doesn’t use terms that are explicitly associated with ”anger”, but due to the nature of the phrase, as it is normally used in undesirable situations, our brain can associate the term ”unacceptable” with a negative emotion. We associate the emotion ”anger” with the second example sentence according to the context it is normally used in.

These are some examples we humans noticed. While there are certainly other clues we use to perceive the emotion, the point is that text under same core emotion exhibits certain patterns.

2.1.2 Text Emotion Classification Challenges

Emotions are conveyed through multiple mediums, human perceive others’ emotions by observing their gesture, tone and the content of the conversation if they are engaged in an exchange of information. Combining all such information leads to a mostly precise perception of emotions.

Most of the mediums humans use in the physical environment are unfortunately not available on the digital platform, and the lack of non-verbal cues makes Emotion Classification challenging. While pure text is the only information that is available to the agents, it’s far less than the amount and categories of information normally exchanged in a face-to-face conversation. In the physical environment, problems that are exclusive to the language can be clarified with the assistance of non-verbal cues. Problems such as Ambiguity, Sarcasm and Co-reference can be made clear with the presence of body language, tone of speech or simply ask for clarification.

However, text data gathered from different sources also have features that are exclusive to the digital platform which we can leverage to compensate for the loss of other physical mediums. In our project, the studied data are gathered from the social platform Twitter, which means users have additional options to form a post other than pure text. Typical examples are emotion-related HashTags and Emoticons. The former is categorical data that directly gives us clues of possible emotion categories the current sentence belongs to, while the latter can serve as a simulation of facial expression and be associated with a certain class of emotions.

We are aware of the existence of other mediums such as audio files, video files etc but that's beyond the scope of this project. The main focus of this project will be purely textual data.

2.2 Large Language Model

2.2.1 NLP before LLMs

Various Machine Learning Technologies have been used to monitor and uncover the pattern underneath the human language.

Initially, researchers used **Rule-based Systems**, in which the language is studied against a set of rules carefully crafted by the linguistics regarding specific tasks such as Tagging, parsing, or information extraction. For example, a task of Sentiment Analysis, in which researchers try to identify if a sentence's emotion is positive or negative, is done against a collection of words, phrases and emoticons with already assigned scores by linguistics.

Fast forward, **Statistical Models** are used, such as N-gram and Hidden Markov Models etc. These models study the language based on statistical assumptions about sub-sentence independence, which suggests a term is only dependent to terms within a certain distance from it. They use a sliding window fashion, studying the sequence of terms or observations. These methods are good for tasks like Part-of-Speech Tagging, Named Entity Recognition etc. However, some argue the Independence Argument is an unrealistic assumption to make.

Later, Neural Networks were employed to study the Natural Languages. Their capability of capturing the hidden patterns is exceptional at the time and have successfully achieved decent results. However, they also lack flexibility as they normally expect a fixed-size input, which is opposite to the random nature of Natural language.

Right before LLMs, or some might suggest the early stage or the start of the LLMs era, Recurrent Neural Networks and Long Short Term Memory are proposed to monitor the nature of arbitrary length of languages as they are sequential models which can handle different input lengths. This makes them particularly suitable for the NLP tasks. Unfortunately, they still suffer from vanishing gradient problems, which means the long-distance dependency is hard to model due to a series of accumulation of probability which makes the value close to 0, despite researchers coming up with offset mechanisms such as Log Probability Assumptions.

Researchers were looking for alternative models that could overcome all the challenges.

2.2.2 LLM

Large Language Model(LLM) is a type of Machine Learning model, which is specified for processing and generating natural languages. These models are trained on a massive quantity of text data which enables them to generate human-like sentences. They are developed for better studying the natural language by solving some shortcomings of the previously used architectures.

The development of the LLM is closely related to the evolution of the Machine Learning Field. Some helpful definitions will be provided to lay the ground for understanding the basics of LLM in the later section.

Unsupervised Learing
It's a method of training machine learning models with clustered unlabeled data. The model will study the data and uncover the underlying pattern without human intervention. It is fundamental to LLM as the large quantity of text data doesn't come with any predefined target labels. The target labels will change according to the tasks the model is tasked to solve.

As the text data is of extremely massive magnitude, which means the training of a model would take a significant amount of time, such as weeks or even months to finish training. For example, the revolutionary LLM BERT is trained on Wikipedia and Google Books corpus, with the former containing 2.5 billion words and the latter containing 800 million words. The total training time for BERT was roughly 3 days. The later developed LLMs contain even more parameters, which extends the training time. The recent ground-breaking example is GPT-3, which achieved jaw-dropping capability. While the actual training time is not disclosed, researchers have calculated that GPT-3 took roughly 34 days to finish training using 1,024 A100 GPUs, according to this study [13]. We can observe a trend of the increasing amount of parameters in the latest LLMs as shown in the figure below and the training time is not expected to reduce.

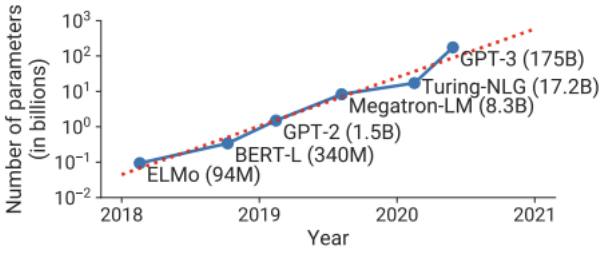

Figure 2.2: Amount of parameters in LLMs

Due to the concern of massive time investment to train an LLM ground up, researchers instead work with pretrained models.:

Pretrained LLM
Since training the LLM ground up for specific tasks is extremely time-inefficient, the workflow adapted now is to base the work on a **pretrained** LLM.

LLMs are pretrained over a massive quantity of text data using unsupervised learning

and are not subject to a single specific task. Hence, they were not given target labels to serve as ground truth, instead they use a technology called **Masked Language Modelling**. The idea is simple, the model randomly masks some part of the sentence and then tries to predict the missing words relying on the surrounding context, then compares it with the actual words to see how precise its prediction is and adjusts accordingly. MLM is not the only task the model is pretrained on, for example, BERT is also pretrained on the task of **Next Sentence Prediction**, in which it predicts whether a sentence should follow its previous sentence according to its knowledge of relationships between sentences. It has access to both correct and incorrect sentence pairs and tries to learn the difference between them. Different LLMs have different pertaining processes, we will describe another LLM that is used in this project in a later section, namely T5.

Having a pretrained model is not useful if LLM is required to solve a specific task, hence LLM needs to go through fine-tuning to make it task-specific.

Fine-tuning
A pretrained model is equipped with enough knowledge to tackle different tasks, it just requires an extra step of task-specific training.

The mechanic works underneath is called **Transfer Learning**. Simply put, the model will take the obtained knowledge from the pre-training process and exploit it to tailor it to the task we are trying to tackle.

We will provide a dataset containing task-specific data to offer guidance and give clues to the model about how well it does. The dataset required for fine-tuning is much smaller, but it is critical if the LLMs were to be transformed into a useful task solver. It will bring a huge impact and significantly improve the model's performance on the specific task.

As the model consumes and processes the new data, it will make predictions with obtained knowledge and compute the difference between them and the actual target. The difference tells the model how far it's from the correct outcome and guides the adjustment for parameters of our pre-trained model. At the end of the fine-tuning stage, the parameters will be altered to the state, enabling our model to perform better on the specific task.

2.2.3 Transformer Architecture

While researchers tend to consider RNN and LSTM as LLMs, the emergence of **Transformer Architecture** revolutionized the Natural Language study and served as a splitting point in NLP research. It was introduced in the study published in 2017, "Attention is all you need" [21].

Due to the limited space and the fact that the Transformer not being the main focus of this project. We will explain the two major components of the Transformer: **Encoder-Decoder Architecture** and **Self-attention Mechanism**. While the details are more sophisticated, these two concepts are backbones of the Transformer architecture.

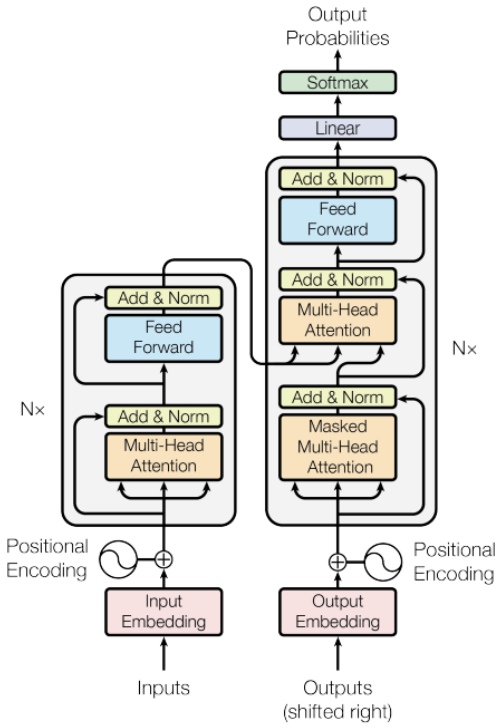

Figure 2.3: Transformer Architecture

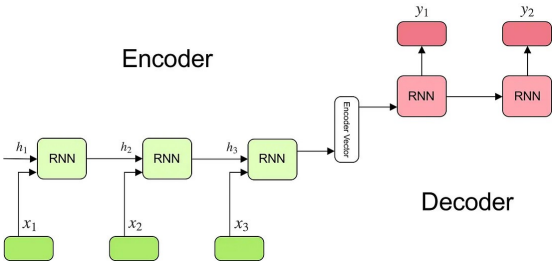

Figure 2.4: Encoder-Decoder Architecture

2.2.4 Encoder-Decoder Architecture

Encoder-Decoder Architecture is being studied by many researchers. While the definitive origin is hard to pinpoint, it was popularized by one of the early studies, ”Sequence to Sequence Learning with Neural Networks” [20] and has since caught the spotlight of Machine Learning research.

The mission of the Encoder is to process the input sequence and transform it into a fixed-dimensional representation. This fixed-dimensional representation encapsulates all the relevant information from the input sequence necessary for generating the output sequence. An encoder is a stack of RNN or LSTM cells, in which each cell accepts one token from the input, collects the information and propagates it to the next cell, the information is captured and stored in the Encoder Hidden State.

The Encoder Vector is the output from the last Encoder cell. It encapsulates all the information collected from the input sequence and serves as the initial hidden state of

the Decoder to help the prediction making.

The Decoder layer is also a stack of recurrent cells, which predicts one output token at each time step. Each decoder cell accepts the hidden state outputted from the previous cell and makes a single token prediction. This process happens iteratively and stops when an End-of-sentence token is generated or the maximum output length is reached.

2.2.5 Self-Attention

In 2014, the study ”Neural Machine Translation by Jointly Learning to Align and Translate”[3] introduced the mechanism **Attention**. It selectively focuses on different parts of the input sequence when generating each word in the output sequence. To do this, the encoder needs to pass **all** the hidden states instead of just the last one to the decoder. It then computes the alignment scores using encoder hidden states, which signify different parts of the input sequence, along with the previous outputted prediction. This score indicates how well the input sequence aligns with the current prediction, and it is done by a feedforward neural network. A softmax function is called against the neural network to obtain the weight of the corresponding encoder hidden state. The weights will amplify relevant parts of input sequences and downsize the rest. Finally, a weighted sum of all the encoder hidden states called context vector is passed to the decoder to aid the prediction that fits the context the most. This essentially instructs the model to pay ”attention” to a specific part of the input sentence relevant to the current prediction.

The Attention mechanism greatly improved the performance of NLP tasks.

However, self-attention, which is introduced along with the transformer architecture from the study [21], is a new approach for tackling Sequence-to-Sequence that relies solely on the **self-attention** mechanism without any recurrence and convolution. The naming is quite self-explanatory, it is a type of attention mechanism where the attention scores are computed based solely on the input sequence itself. In self-attention mechanisms, each position in the input sequence attends to all other positions, including itself, to capture dependencies and relationships between different parts of the sequence, this allows it to capture the global dependency regardless of the distance. The reason is that self-attention is based solely on the input sentence, we are no longer constrained by the alignment we mentioned in the traditional attention mechanism.

The main difference between **attention** and **self-attention** lies in the scope of attention computation. Traditional attention mechanisms typically focus on interactions between the input and output sequences, while self-attention mechanisms focus on interactions within the input sequence.

This mechanism, eliminating all the recurrent and convolution units, allows for parallel training and offers scalability, makes efficient training possible.

2.2.6 Text-to-Text Transfer Transformer - T5

The pretrianed LLM we fine-tuned for our Emotion Classification Task is the T5 model. It was trained and published by Google in 2019, along with the paper ”Exploring

the Limits of Transfer Learning with a Unified Text-to-Text Transformer" [17]. As suggested by the authors, the structure of T5 closely follows the original design of transformers proposed by [21].

T5 is trained on an extremely test-rich corpus, namely **The Colossal Clean Crawled Corpus**, or **C4**, created by crawling and cleaning web pages. It contains a massive amount of text data collected from diverse online sources such as websites, forums, and other publicly accessible online content. It is orders of magnitude larger than most data sets used for pre-training and contains reasonably clean and natural English text.

During the pertaining phase, T5 adopted an unsupervised learning method similar to what has been done in BERT pertaining stage, the **Masked Language Modeling**. T5 drops out 15% of the tokens from the original text, replaces them with numbered sentinel tokens and tries to predict the missing words. T5 masks consecutive spans of tokens and only predicts dropped-out tokens to reduce computation costs. The difference between ground truth and predictions is used to improve the model parameters.

T5 is specialized in **text-to-text tasks** as indicated in the name, it has a collection of downstream tasks which T5 can be fine-tuned to solve such as Text Classification, Question Answering and Text Entailment etc. These tests are all unified in a text-to-text format, meaning the inputs and outputs are all expected to be sentences.

During the Fine-tuning phase, researchers also conducted multiple strategies that can speed up the process by not training the whole set of parameters, instead, they choose to either add "Adaptive Layers", which are additional layers added after each of the existing feedforward layers in the Transformer. Only adaptive layers and the layer normalization parameters are updated, keeping most of the original model untouched. Or use "Gradual Unfreezing", in which the amount of parameters being updated is gradually increasing, starting from only updating parameters in the final layers, and gradually including more parameters. However, the best performances are still observed when all the parameters of the model are fine-tuned.

In short, T5 is a variants of Transformer-based LLM that is capable of solving text-to-text NLP tasks. Our project requires a classifier that consumes tweet text and produces a classification label, which is one of the expertise of T5.

2.3 Explainablility in NLP

Explainable AI, or XAI, has been a topic that is becoming more important due to the introduction of the Deep Learning model and Language Embeddings to NLP tasks which took away the interpretability compared to traditional "white box" NLP approaches, such as Rule-Based, Decision Trees and Hidden Markov Models etc.

XAI is a broad topic that lies across multiple fields and each field has its approaches. We will only focus on the XAI study in NLP and classify our approach accordingly for this project.

According to a coherent survey conducted to study the available XAI techniques in NLP, [6], explanations are classified according to two high-level main aspects.

- **Local** - **Global**

  The local explanation provides the explanations on a specific input.

  The global explanation justifies the predictions by uncovering the underlying reasoning process and models without referring to any specific input.

- **Self-explaining** - **Post-Hoc**

  The self-explaining approach is part of the prediction process, which produces an explanation along with the prediction. Explanation generation and prediction happen at the same time.

  The Post-Hoc approach is conducted after the prediction is made by performing addition operations.

However, the high-level categories don't cover important characteristics. More fine-grained aspects should include the **techniques used for deriving the explanations**. The survey summarised a total of 5 different techniques that have been used across all the NLP XAI studies.

1. **Feature Importance** Explanations are derived based on the feature importance used to output the final prediction.
2. **Surrogate Model** Training a more explainable model as a proxy. This approach can be used to produce both Local and Global explanations and it's widely applicable as it's model agnostic.
3. **Example-Driven** This approach leveraged labelled data as evidence to explain the prediction for new input that has similar structures. They are similar in spirit to the Nearest Neighbour Approach.
4. **Provenance-based** Explanations are presented by showing some or all the reasoning process, normally used when the result is a series of derivation steps.
5. **Declarative Induction** An approach that is based on human-readable representations such as a collection of rules, trees or programs that are induced as explanations.

In this project, the explanation methodology we used belongs to **Local Post-Hoc Feature Importance**, with some similarity to **Provenance-based** approach because we are displaying the complete process of reasoning.

Chapter 3

Project Specific Information

3.1 Dataset

The dataset used in this project is a subset of the dataset composed by the author of [18], focusing on English tweets and their associated emotion. It consists of 436,809 rows of tweets that were already classified into 6 categories of emotions, which are **Joy, Sadness, Anger, Fear, Love, Surprise**. According to the authors, they adapted the preprocessing steps proposed by the study [2].

Most of the tweet data available are not specifically associated with emotions, it is up to the authors to properly label the data according to the needs of the study they are conducting. However, it would require an incredibly huge amount of human effort to annotate enough training data for the Emotion Classification Study, therefore they adopted the **Distant Supervision** technique. In short, Distant Supervision is a method for information extraction, it does so by aligning the Knowledge Base with unannotated text data. The **knowledge base** is a database that stores sentences with annotations that might help computers to understand what these sentences are about. Distant Supervision will look at the new unannotated sentences and examine them with the knowledge stored in the Knowledge Base, then argue whether the new sentence possesses a certain relationship. In our task, the authors decided to leverage Twitter HashTags as an indication of emotion, they extracted all the tweets containing emotion-carry hashtags and used them as an indication of the emotion associated with the text data.

To capture this relationship, authors need collections of hashtags that represent certain emotion categories. To achieve this authors first use the target labels as **seeds**, which means the root of distinct and disjoint emotion categories. They then expand the seeds into a bag of words all associated with the same emotion the seed word conveys. The approach they took is similar to how I collect seeds in the **Emotion Evaluation Metric** chapter, which can be found here 2.

Example hashtags for the ”joy” hashtag are:

> happy, happiness, joy, joyful, joyfully, delighted, jubilant, blithe, beatific, exhilarated, blissful, walkingonair **etc**

Collecting tweets using hashtags that appeared in this collection tells the author that the text content of the tweet is highly likely to be associated with the emotion ”joy”.

Furthermore, various preprocessing techniques were applied due to the noisy nature of human language, especially tweets as it lacks writing standards. The goal is to create a non-overlapping dataset for the emotion classification task, which means a single tweet should only be associated with a single emotion. Therefore, **they removed all the tweets that contained hashtags that appeared in more than one emotion hashtag collection**. Next, **duplicated tweets are removed**. They also considered the duplicated characters that modify the original form of the word are shrunk down to 2 appearances, for example, ”I haaaate waking up early” is transformed into ”I haate waking up early”. In addition, the authors **removed all the tweets that were not written in English**. Lastly, only tweets longer than 5 characters are kept to ensure enough information for the machine-learning model.

After all the filtering and preprocessing, what remains in the dataset only contains unique tweets associated with the most reasonable emotion. Example data for each target label:

Text	**Emotion**
i feel pretty pathetic most of the time	sadness
i already feel like i fucked up though because i dont usually eat at all in the morning	anger
i feel romantic too	love
i would always feel amazed at how impacted these and year olds were by this subject	surprise
i feel very happy and excited since i learned so many things	joy
i had stated to her the reason i feel so fearful is because i feel unsafe	fear

Table 3.1: Example tweets with assigned emotion

To give a clear overview of this dataset, here are some examples and statistics about the dataset. The dataset used for the whole training and testing cycle is a subset which consists of 20,000 tweets, and are split in 80 : 10 : 10. We will be studying the training set as it directly affects how our model performs. The distribution of training data in our dataset is very imbalanced, as shown in the table below.

Emotion	joy	sadness	anger	fear	love	surprise
Count	5362	4666	2159	1937	1304	572

Table 3.2: Amount of tweets associated with each emotion in training set

This would require specific treatment during the training phase to ensure our classifier is not biased towards a certain class.

Further analysing our dataset, we list the most frequent n-grams in all emotion classes to learn the nature of text data that belongs to different emotion classes. Due to the limited space, only the result for the most dominant emotion class, joy, is included.

We notice that almost all the emotion classes share the characteristic of dominant terms being neutral. This indicates the emotion classifier needs to be able to uncover the

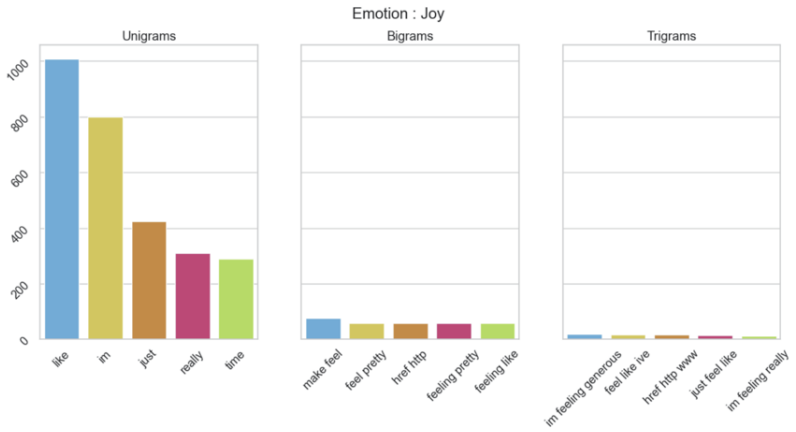

Figure 3.1: Most frequent n-grams of tweets belonging to emotion 'joy'

underlying text patterns between different classes, rather than looking for certain terms.

3.2 Emotion Classifier

The emotion classifier we trained for this specific task is a **T5 model**.

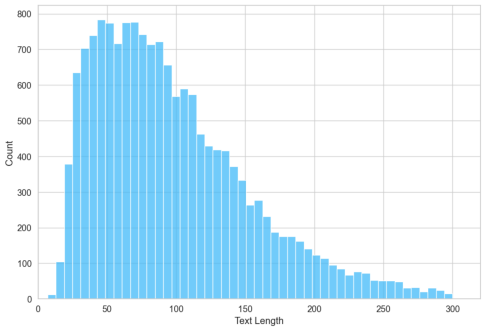

Figure 3.2: The distribution of tweets length in our training set

T5 comes with variants with different amount of parameters, allowing us to choose after deciding a tradeoff between accuracy and performance. Due to the nature of the short length of tweets, we don't need a sophisticated model. For this project, the **T5-base** model which comes with 220 million parameters is enough to serve our purpose.

Since the goal of this project isn't beating the state-of-the-art results, we didn't spend time experimenting and pinpointing the combination of parameters that achieves the best accuracy on the validation set. We fine-tuned the T5 model with a **learning-rate: 1e-4**, **no weight decay**, **train-batch-size: 16**, **eval-batch-size: 8**, **max-grad-norm: 1.0** with an **adam optimizer** for **2 epoches**.

In addition to these basic parameters, we also used **Random Sampling** to counter the extremely imbalanced training data to ensure the classifier is not biased towards dominant classes such as "joy" and "sadness".

This model serves as the baseline of our experiment and achieved **86.02%** accuracy.

Chapter 4

Explanations

Having a working model lays the ground for generating explanations. As mentioned in the previous sections, the methodology I implemented is not specific to the T5 model. In the following sections, I will walk through the steps necessary to obtain the explanations.

Int this chapter, I examined and implemented the algorithm described in the paper[4], which proposed an explanation-generating algorithm that hierarchically breaks down the sentence by detecting the feature interactions. This method resembles the **Shapley Additive Explanations**, which decompose the output of the machine learning model based on the Shapley Values, which is a concept originated from game theory that assigns credits to each subpart of the input and measures the effect they impose on the prediction.[16] [8] [7]

4.1 Methodology

1. Input the original sentence and predicted emotion to the algorithm
2. **Sentence Breakdown** - Find and break at the connection that has the weakest feature interaction score across the whole sentence or among the whole set of sub-sentences. The detail is explained in the subsection 4.2.1 - 4.2.2.
3. **Contribution Computation** - Compute the contribution scores of new sub-sentences that are generated from the previous step. The detail is explained in the subsection 4.2.4.
4. Repeat Steps 2 and 3 until the original sentence is fully broken down into a collection of words.

The algorithm keeps track of all the sub-sentences that are generated throughout along with the contribution scores. The results returned contain both the breaking down of the sentence and the scores. This information is further fed into a visualization class that is implemented by me to generate a visualization.

4.2 Details

4.2.1 Hierarchical Breakdown

The algorithm aims to maintain a collection of bags of subsentences. At every timestep, we aim to find the dividing point of one subsentence that has the lowest feature interaction score overall and split the sentence/subsentence at that dividing point to produce two new sentences, add them back to the collection and repeat the same step until all the subsentences we have just consist of a single word.

A low feature interaction score indicates the two sentences that can be obtained by splitting at the dividing point are mostly present individually compared to all other possible splitting of the sentences. For example, consider the sentence "I went to school in the morning", human would find it normal to split the sentence into "I went to school" and "in the morning" as each subsentence is self-contained and the bonding between these two subsentence are weakest among all the possible dividing points. It is odd to split the sentence into "I went" and "to school in the morning" as the "went" and "to" have a stronger bond. Such an effect is based on the assumption that words with stronger connections in between tend to appear together more often than those with weaker connections.

The algorithm can be addressed as the following optimization problem:

$$\min_{x_{(s_i,s_{i+1}]} \in \mathcal{P}} \min_{x_{j \in (s_i,s_{i+1})}} \phi(x_{(s_i,j]}, x_{(j,s_{i+1}]} \mid \mathcal{P}) \tag{4.1}$$

where partition $\mathcal{P} = \{x_{(0,s_1]}, x_{(s_1,s_2]}, ..., x_{(s_{P-1},n]}\}$ represents the bag of subsentences of length P that have been collected so far. $x_{(s_i,s_{i+1}]}$ representes a subsentence that contains all the words $(x_{si}+1, ..., x_{si+1})$.

With those terms defined, we can examine the meaning of the optimisation problem. $\phi(x_{(s_i,j]}, x_{(j,s_{i+1}]} \mid \mathcal{P})$ represents the feature interaction score between substences $x_{(s_i,j]}$ and $x_{(j,s_{i+1}]}$ given the context $\mathcal{P}$. The necessity of conditional probability is to include the effect of context surrounding the subsentences, the same subsentences may have different strengths of bonding under different contexts. The inner optimization problem $\min_{x_{j \in (s_i,s_{i+1})}}$ will find the weakest dividing points among all the possible dividing points within the current subsentences we are examining. Lastly, the outer problem $\min_{x_{(s_i,s_{i+1}]} \in \mathcal{P}}$ will find the subsentence that contains the weakest dividing point among all the subsentences that have been collected so far.

Therefore, the goal is to **find one dividing point in one of the subsentences that have the weakest feature interaction score.** at each timestep during the execution of this algorithm.

4.2.2 Feature Interaction Score

In this section, we will clearly define the steps required and the assumptions made to obtain feature interaction scores.

To calculate the feature interaction score of each dividing point of a selected subsentence $x_{(s_i,s_{i+1}]}$ and the potential dividing point j, we adopt the Shapley Values alike approach inspired from Coalition Game Theory, based on words done in this paper[10]. To make the formula cleaner, we note newly obtained subsentences $x_{j\in(s_i,s_{i+1})}\} \cup \{x_{(s_i,j]}, x_{(j,s_{i+1}]}$ as j_1, j_2, the formula for feature interaction score is:

$$\phi(j_1, j_2 \mid \mathcal{P}) = \sum_{S \subseteq P \setminus \{x_{(s_i,s_{i+1}]}\}} \frac{|S|!(|\mathcal{P}| - 1 - |S|)!}{|\mathcal{P}|!} \gamma(j_1, j_2, S) \tag{4.2}$$

S is a subset of current partition P, which contains one of multiple subsentences. The Shapley Values are used to evenly distribute contribution to the final predicted probability for the predicted label $\hat{y}$. A pair of subsentence will contribute less if the fact they are separated doesn't impose a major effect on the final result. We will now examine the terms in the feature interaction score formula. To do this, we will need to break down the term $\gamma(j_1, j_2, S)$, this term is defined as:

$$\begin{aligned}\gamma(j_1, j_2, S) = \mathcal{E}[f(x') \mid S \cup \{j_1, j_2\}] - \mathcal{E}[f(x') \mid S \cup \{j_1\}] \\ - \mathcal{E}[f(x') \mid S \cup \{j_2\}] + \mathcal{E}[f(x') \mid S]\end{aligned} \tag{4.3}$$

$f(\cdot)$ represents the probability our model assigned to the predicted label. x' is obtained by removing words that are not presented in the subset which the current probability is conditioned on from the original sentence x, those terms that are missing should be replaced by placeholder $< pad >$. $\mathcal{E}[f(x') \mid S]$ is the expected probability obtained from $f(\cdot)$ over all the possible x' that can be generated from the subset of subsentences S theoritically. In the actual implementation, the expected value is approximated by the output of $f(x')$ given x' after filtering out the words missing from the subset the current model is conditioned on.

This formula **measures the increase in a coalition's value when both new subsentences generated from the selected subsentence are included in the S, compared to only including either one of them in the** S. The term $\mathcal{E}[f(x') \mid S]$ is added back to compensate for the contribution of subsets that don't include the selected subset since it has been deducted from the correct score twice during the previous computation.

If this formula obtained a high value, it indicates the combination of two new subsentences plays an important role when the model makes the prediction, therefore they are to be broken down later. We need to find the dividing point that contributes the least to the prediction, which means having both subsentences is not significantly better than just having them separately in the original sentence, hence it's better to split them earlier.

After computing the feature interaction score like this for all the dividing points among all the subsentences stored in the current partition P, we will find a dividing point j' with the least feature interaction score. We will now update the partition by actually splitting the subsentence j' belongs to.

For a given subsentence $x_{(s_i,s_{i+1}]} \in \mathcal{P}$ and the dividing point j', we will update the partition by removing the subsentence contained the dividing point and adding two new

subsentences obtained by splitting the removed subsentence at the dividing point back partition $\mathcal{P}$:

$$\begin{aligned}\mathcal{N} &= \mathcal{P} \setminus \{x_{j\in(s_i,s_{i+1})}\} \cup \{x_{(s_i,j']}, x_{(j',s_{i+1}]}\} \\ &= \{x_{(0,s_1]},\, x_{(s_1,s_2]}, \ldots, x_{(s_i,j']}, x_{(j',s_{i+1}]} \ldots,\, x_{(s_{P-1},n]}\}\end{aligned} \tag{4.4}$$

The rest of the term are straight, it's the possibility of a specific coalition appearing. The term $\frac{|S|!(\mathcal{P}-1-|S|)!}{\mathcal{P}!}$ represents the probability of a specific splitting joining a coalition formed by a subset S. This term is straightforward to implement but suffers from scalability as the length of $\mathcal{P}$ grows, i.e. a longer sentence x, therefore in the implementation we need to modify the theoretical formula so that it is scalable.

4.2.3 Implementation Modification

With the number of words contained in the original sentence x increasing, the amount of model evaluation needed to compute the feature interaction score calculation will grow exponentially, making it almost impossible to generate an explanation in an acceptable period for a sentence with more than 8 words in my implementation. This problem would make the algorithm obsolete if it's not addressed and handled properly. Therefore, we assumed that a word usually has strong interactions with its neighbouring words, similar to the work done in this paper [5].

Based on this assumption, we only consider the neighbouring substances that are contained in a window size of m from the selected subsentence:

$$\mathcal{P}_m = \mathcal{P} \setminus \{\{x_{(0,s_1]} \ldots x_{(s_{j-m-1},\, x_{s,j-m}]}\} \cup \{x_{(s_{j+m},\, s_{j+m+1}]} \ldots x_{(s_{\mathcal{P}-1},n]}\}\} \tag{4.5}$$

The updated formula is:

$$\phi(j_1, j_2 \mid \mathcal{P}_m) = \sum_{S \subseteq P \setminus \{x_{(s_i,\, s_{i+1}]}\}} \frac{|S|!(|\mathcal{P}_m| - 2 - |S|)!}{|\mathcal{P}_m|!}\, \gamma(j_1, j_2, S) \tag{4.6}$$

This will bring down the complexity to polynomial.

4.2.4 Contribution Score

To clearly show how each subsentence contributes to the final prediction quantitatively, we need to assign a contribution score during the splitting. The approach taken is easy to interpret.

Here is the formula for calculating the contribution score of subsentence $x_{(s_i,s_{i+1}]}$ with predicted emotion being $\hat{y}$:

$$\psi(x_{(s_i,s_{i+1}]}) = f_{\hat{y}}(x_{(s_i,s_{i+1}]}) - \max_{y' \neq \hat{y},\, y' \in \mathcal{Y}} f_{y']}(x_{(s_i,s_{i+1}]}) \tag{4.7}$$

All the computations are with respect to the current subsentence instead of the original sentence, therefore the predicted label might differ from the result for the whole sentence. $\hat{y}$ is the predicted label for the complete original sentence.

The term $\mathcal{Y}$ is the model target labels collection, $f_{\hat{y}}$ is the probability outputted by our model for a label that is the predicted result for the **complete original sentence** and $f_{y'}$ are the probabilities for the other labels in the target labels collection.

Again, what we are computing here is the subsentence instead of the original sentence, a subsentence can get a higher probability for a label that is different from the $\hat{y}$. If this is true, we will get a negative contribution score for this subsentence, indicating that this subsentence is expressing a different emotion, which can be examined by the human debugger. In contrast, a positive contribution score indicates the subsentence agrees with the $\hat{y}$ and contributed positively during the process of reaching $\hat{y}$. All the scores will then be visualized for human debuggers to examine and evaluate the accuracy of the reasoning process that has been carried out by the model.

4.3 Example

I will briefly walk through two explanations, one for correct and incorrect predictions. I will be showing the explanations visualization, you can examine the example raw explanation output in the appendix. A.1 We need to note that **while the prediction might be correct, the explanation may reveal improper reasoning process.**

- **Correct Prediction** The sentence is ”i feel dirty and ashamed for saying that”,

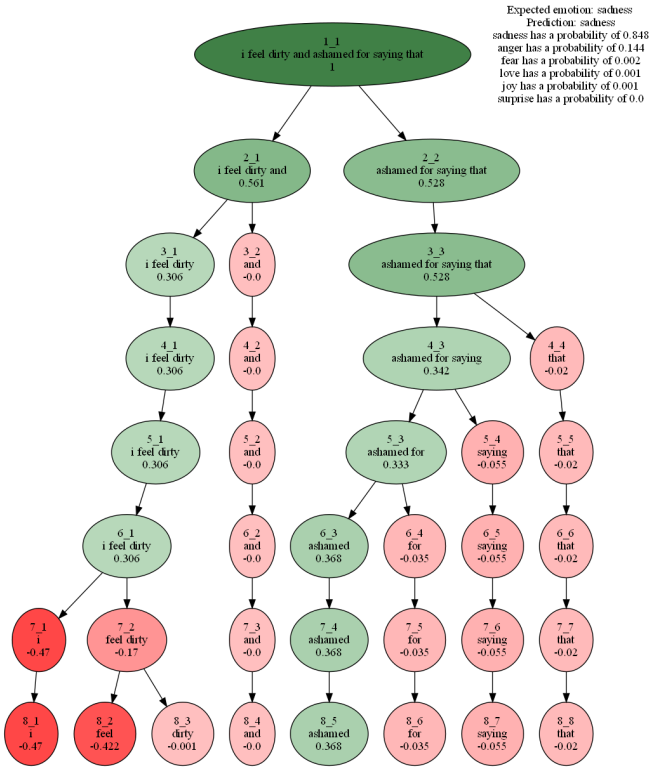

Figure 4.1: Explanation tree for correct prediction. Green means the current subsentence aligns with the **predicted** emotion, while red means the subsentence is against the **predicted** emotion. The intensity of colour indicates the strength of alignment or opposition.

the expected and predicted label is coherent, both being ”sadness”. This aligns with the human interpretation of this sentence if no more context is given.

Upon closely examining the explanation, we notice that both subsentences ”i feel dirty” and ”ashamed for” contribute to the label ”sadness”, while the rest of the sentence contributes negatively. This highlights the subpart of the original sentence that leads to the final prediction. If we closely examine the last layer specifically, we notice that the word ”ashamed” is the only part that contributes to the label ”surprise”, which is a reasonable explanation as ”dirty” as a single word isn't related to the label ”sadness”.

However, as we mentioned at the start of this section, a correct prediction doesn't indicate a perfect explanation is given. If we closely examine the figure, we notice that words like ”i” and ”feel”, which should be neutral terms, actually contribute against the label ”sadness”. This finding indicates that biases exist in the original training data, and neutral terms exist more often in those dominating terms and are treated as an indication of certain labels by our model, which is problematic.

- **Wrong Prediction** The sentence is ”I was feeling extremely whiney and lonely

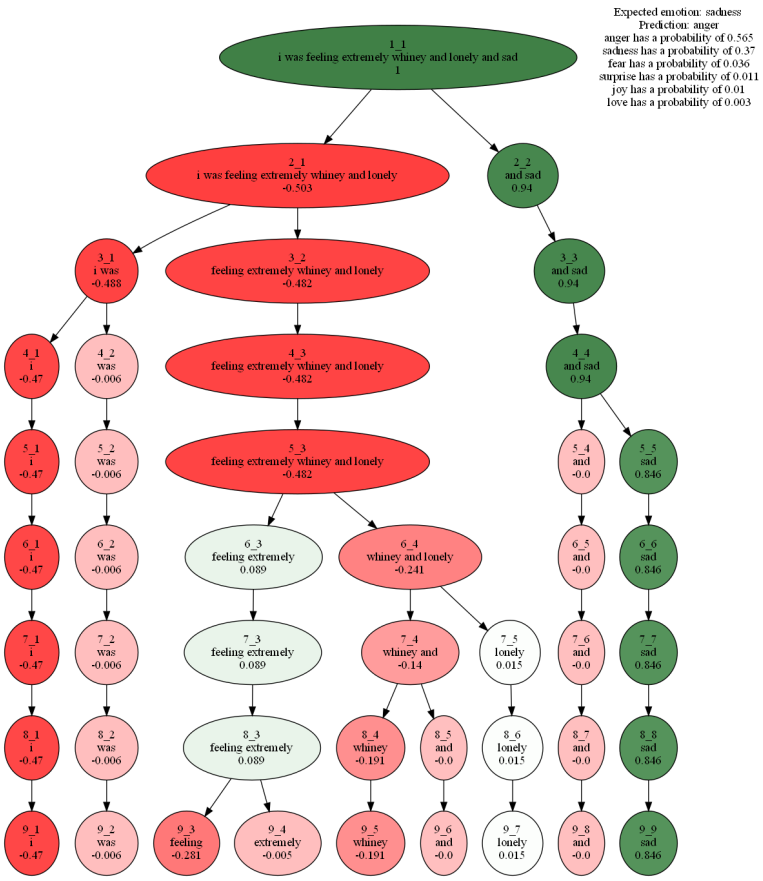

Figure 4.2: Explanation tree for wrong prediction. Green means the current subsentence aligns with the **predicted** emotion, while red means the subsentence is against the **predicted** emotion. The intensity of colour indicates the strength of alignment or opposition.

and sad". In addition to the improper contribution score for the neutral term problem, the explanation also points out that the word "sad" is likely to be associated with the label "anger" due to data imbalance. It would be ideal if specific training data are generated to tackle this problem.

4.4 Limitation of Explanation

While the explanation sounds like the solution to the "black box" nature of machine learning models, it suffers from certain problems in this particular project.

1. Target label being incorrect During the examination of generated explanations, I noticed certain validation cases expect an unreasonable target label, which means the "supposed" correct prediction is less sensible than the predicted label. "Correcting" the model behaviour in this case will make the model worse.

   Example:

Text	Expected	Prediction
i feel that this is important in itself the fact that we all have our own individual way of grieving	joy	sadness

Table 4.1: Example in which the prediction is more aligned with human perception comparing to expected emotion label. The sentence has a negative tone and the prediction of "sadness" is more accurate than the expected emotion "joy"

2. Target label and predicted label both make sense Emotion is a complex topic to study because there is no firm and square answer for the sentence that is given to the model. For humans, the same sentences can express different emotions, based on different contexts or perspectives. Therefore, it's not unusual to find an explanation that makes perfect sense even when the predicted label differs from the target.

   While the introduction of Sentiment Similarity corrects this issue to a certain degree, it still exists and affects the process of generating new training data using explanations, preventing us from making use of the explanation to its full extent.

   Example:

Text	Expected	Prediction
i have been feeling lied to and abused by lenders	sadness	anger

Table 4.2: Example in which the prediction and expectation both make sense. In this example, the sentence can either express "sadness" or "anger" under different contexts, both emotions make sense in the proper context.

3. Explanation not easily comprehensible While revealing the reasoning process of ML models drastically increases the understandability and decreases the difficulties in understanding the reasoning process behind a prediction, the nature of the

ML models still makes most explanations not human-friendly. It is not always straightforward for humans to identify the underlying issue within our model that caused the wrong prediction, especially with the growth in the input length.

Example of an explanation tree that is not human-friendly:

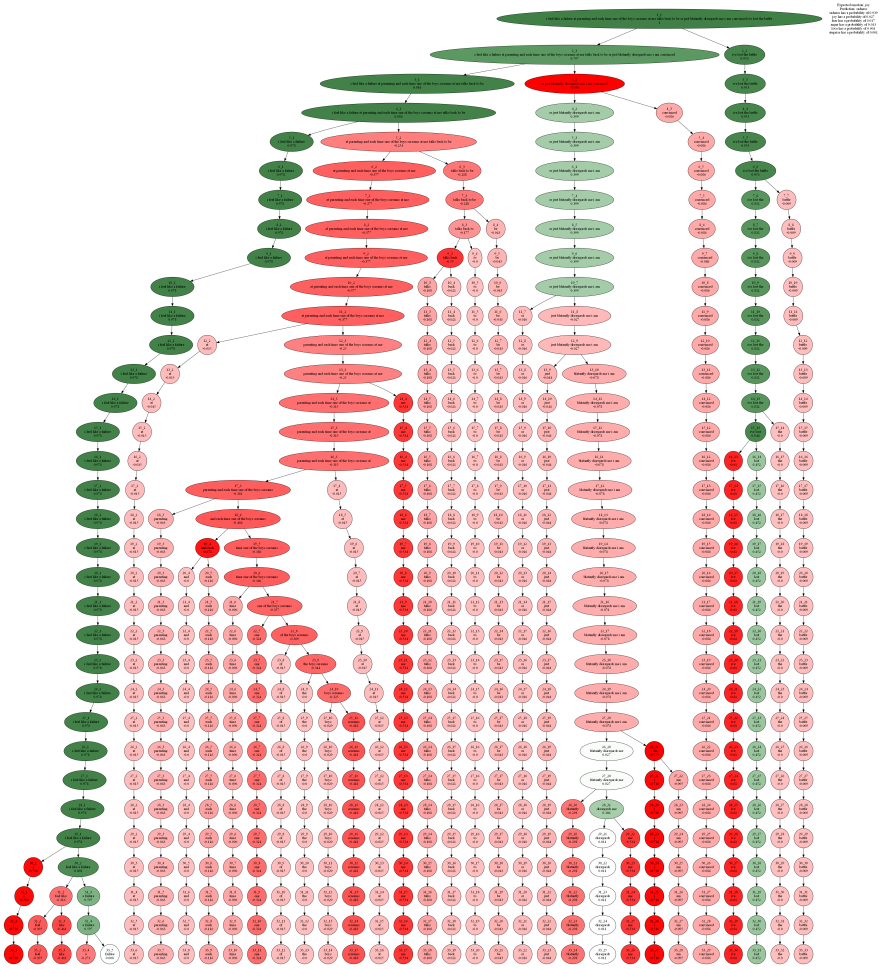

Figure 4.3: Example which is hard for humans to analyze due to the vast amount of subsentences and scoring. This is also an example of expected labels being less aligned with human perception compared to prediction, but it's hard to study due to the explosion of subsentences.
The text is ”i feel like a failure at parenting and each time one of the boys screams at me talks back to be or just blatantly disregards me i am convinced ive lost the battle”, while the clear indication of negative emotion from the subsentence ”i feel like a failure”, the expected label is set to ”joy” while the prediction is ”sadness”, which is obviously more accurate.

Chapter 5

Human Debugging

Explainable AI not only contributes to increasing user trust in the model outputs but also gives a better insight into the model itself. It reveals all the paths it has taken to reach the final prediction. While the prediction may be desired, it could have been reached using an unreasonable path. Failure to identify such issues within the model will be detrimental in the future as it will eventually produce predictions which are believed by users to be true due to good performance during the testing stage by simply comparing the results but are in fact false.

Therefore, it would be ideal to leverage the explanation I obtained in the previous experiments, as it contains new information that might benefit the model training. I searched for a way to integrate the information with the training process to effectively improve the model's performance by pointing it to the right direction.

To achieve this, I experimented with the Human Debugging Cycle. In this paper [9], a framework called Explanation-Based Human Debugging (EBHD) is examined across many NLP experiences, which constructed a cycle that makes use of the explanation to improve the model.

5.1 Framework

To better understand the experiments I conducted based on the EBHD framework, I will walk through the framework and present multiple variants and then justify the variants I used in my project.

1. The model provides prediction along with the explanation that leads up to the prediction and feeds the pairing to a human tester to gather feedback.
2. Human testers provide feedback given the input and task the model is solving.
3. Update the model by using feedback.

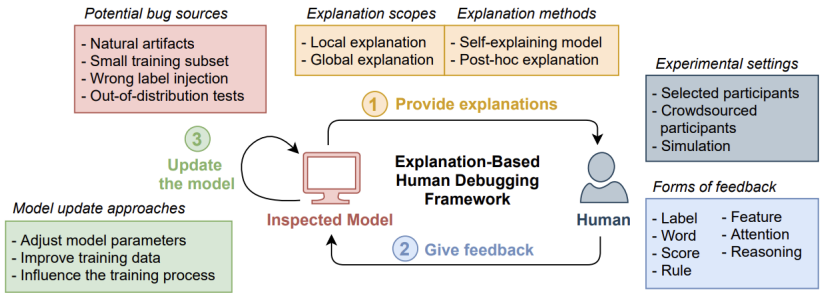

Figure 5.1: Explanation-Based Human Debugging(EBHD) framework. It consists of the model that ought to be improved, the humans providing feedback, and a three-step workflow. Boxes list available options for each component and stage of the framework. [9]

5.2 Experiment Setup

The explanation generation has been done in the previous chapter, here we investigate the rest two steps.

5.2.1 Feedback Gathering

While it would be ideal to get human testers involved in the process, the sheer amount of work required to gather enough information for the model is impossible to reach during the period of the dissertation. The more efficient alternative approach is to leverage GPT API and feed it our generated explanations to get human-alike feedback.

This approach however does impose limitations because GPT is also a LLM. While GPT's performance has been shocking and achieved many state-of-the-art results, the model itself is still not enough to completely replace the human testers as it can't simulate a human's thinking process. Feeding the explanation to GPT and asking it to provide feedback is not the same as asking the real person to study and evaluate.

5.2.2 Model Update

As presented in the figure 5.1, there are multiple ways to update the model based on the gathered feedback. We can adjust the training parameter directly, modify the training data or even directly influence the training process such as user-co-training, which makes humans an extra layer of classifier.

In the Emotion Classification task, considering both the time and resources, I decided to **modify and update the training data after gathering the feedback**. This option is the most intuitive and straightforward as adding more sentences to the training data doesn't require us to directly modify the model training parameters, allowing us to see a direct effect of the introduced data that are generated based on the explanation.

The exact cycle of how we obtain feedback, generate additional data accordingly and feed them back to the model in this project is described in the later chapter.

Chapter 6

Emotion Evaluation Metric

Before entering the experiment and evaluation, I decided to implement a metric that is more suitable for emotion label comparison.

6.1 Motivation

6.1.1 Standrad Metric

The standard metrics used in the text classification tasks are **Precision**, **Recall** and **F1-scores**. The definitions for them are:

- **Precision** Precision measures the number of positive class predictions that indeed belong to the positive class. The higher the number is, the higher the accuracy the model has when predicting the positive class.
- **Recall** Recall measures how many positive class instances are indeed classified as positive class. The higher the number is, the higher the completeness of the model is when predicting the positive class.
- **F1-score** It is not good enough if our model achieved a high precision but low recall, same vice versa. We want our model to both predict accurately and be able to pick out all the instances that belong to each class. Therefore we need a metric that captures both the accuracy and the completeness of our model. F1-score is a harmonic mean of precision and recall that embeds both previous metrics.

The more detailed explanation and formula for these metrics can be found in A.4 and A.6.

6.1.2 Drawbacks

While all the metrics mentioned in the previous are widely used in machine learning research, I would argue it doesn't capture the performance of the Emotion Classifier well enough.

Considering the example of the sentence, ”I’m so frustrated about the way they treat me”. Such a sentence may embed different emotions based on the context, it is perfectly reasonable for someone to say the sentence is expressing the emotion of Sadness, while the others may agree with the emotion of Anger.

None of the judgment is wrong, the reason behind this scenario is the lack of context. Most sentences are vague without the context they are used in, even when the sentence itself may express an emotion that seems to dominate, adding context can flip the meaning of it completely. Even extremely positive sentences such as ”I feel so happy for you” may also embed a sense of Sarcasm, under certain contexts. This is itself a field to study, but the point is, when dealing with emotion classification tasks, the lack of context naturally makes multiple predictions reasonable. Therefore, **in the setting of measuring model performance using standard metrics such as Precision, Recall and F1-score, we are treating the prediction of ”Anger” for a sentence that has a target emotion label ”Sadness” as wrong as a prediction of ”Joy”.** This is not the most ideal way of handling misprediction and can’t identify the actual performance difference between two models. A model which makes completely wrong predictions may be measured and considered to have similar performance as a model that predicts emotions that are reasonable but different from target labels.

6.2 Sentiment Similarity

To address this issue, I propose to use Sentiment Similarity as an additional performance metric, which can capture the scenario described above.

Sentiment Similarity is a metric that is used to measure the similarity between two words according to the emotion they normally convey or embed.

6.2.1 Similarity Metrics

There are many word similarity measures which are available to use to answer questions like ”How similar two words/sentences/articles are to each other?”. This question is studied in the field of Text Similarity, which can be answered considering many different aspects of similarities. We answer these questions by computing how **close** these documents are, and the closeness can be lexical or in terms of the meaning.

6.2.2 Lexical Similarity

Lexical Similarity measures how similar the word set/vocabulary of two documents are, giving a degree of similarity between 0 to 1. The higher this metric is, the larger the overlap between the vocabularies.

For example, ”A man got bit by a dog” is lexically identical to ”A dog got bit by a man” as the vocabularies are identical for both documents. This, while it is straightforward, doesn’t capture the ”similarity” our humans are used to when comparing documents, as the **meaning** of these two sentences are completely different, despite having the same vocabularies. Therefore, we want to use measures to compare **Semantic Similarity.**

6.2.3 Semantic Similarity

6.2.3.1 Existing Algorithms

Many measures can be used to compute Semantic Similarity, such as Point-wise Mutual Information, Latent Semantic Analysis(LSA) etc. These methods can be considered as transforming the terms into vectors and then computing the distance between the resulting vectors. The smaller the distance is, the more similar the documents are. Since Semantic Similarity is not the main focus of the project, I will briefly walk through one of the techniques that is later used.

Latent Semantic Analysis(LSA) is a technique in NLP that is used to study the relationship between a set of documents and the terms contained within them. LSA is based on the assumption that **words that are close in meaning will occur in similar pieces of text based on distributional hypothesis, which suggests linguistic items with similar distributions have similar meanings.** [22]. LSA is explained in more detail in the appendix A.2.

6.2.3.2 Drawback in Emotion Classification

The catchline is the assumption we made when computing the semantic similarity, **words that are close in meaning will occur in similar pieces of text based on distributional hypothesis**. This assumption works fairly well when it's applied to find semantically similar terms, such as synonyms. However, it fails to capture the **sentiment similarity** between words that express similar sentiments but are not essentially similar in terms of semantics.

For example, LSA will assign a high score for terms like "car" and "vehicle", as they both mean automobiles and can be used interchangeably, therefore it would frequently appear in a similar context. As a consequence, "car" and "vehicle" normally can score high LSA scores, indicating the similarity shared between their semantics. However, consider two pairs of words:

1. **"Good"** and **"Superior"**
2. **"Good"** and **"Bad"**

While pair 1 share a more similar sentiment compared to pair 2, the LSA assigned a higher score for pair 2 than to pair 1.

$$LSA(\text{"}GOOD\text{"},\text{"}BAD\text{"}) > LSA(\text{"}GOOD\text{"},\text{"}SUPERIOR\text{"})$$

One explanation is that not only do synonyms appear frequently in the same context, but so are the antonyms. The semantic similarity depends on the context that surrounds the terms we are comparing, antonyms of words tend to appear in similar contexts to convey opposite meanings, which can't be said for the sentiment of similar words. While they may share similar sentiments, the nature of the words may result in different sets of context. In our example, "Good" is an adjective that is normally associated with an attribute of an entity, while "Superior" embeds a sense of comparison, hence they may be used in different contexts more often than in the same context.

6.3 Sentiment Similarity

To capture the sentiment similarity, we will build based on the semantic similarity.

1. **Collect central emotions**

   We need a collection of emotions to serve as the base of each central category for all the other emotions. While debates are going on about what emotions should count as the central emotions, we will use the one defined by [14]. In the paper, the central emotions are anger, disgust, fear, guilt, sadness, shame, interest, joy, and surprise. However, in the dataset this project used, there are only five emotions in the target label set. Therefore we only maintain anger, fear, sadness, joy, and surprise. And we will also include the last target label in our data, which is love. This is similar to the work done by [15].

   Therefore we have a collection of central emotions, with joy, surprise and love being positive emotions and the rest being negative, achieving a balanced split in terms of the general sentiment direction.

2. **Expand each central emotion category**

   We need to extend each central emotion into a category, which contains words that contain a similar sentiment direction as the base emotion. All the words that embed similar sentiment direction as the base emotion are called **seeds** . To collect such seeds, we will add synonyms of the base emotion hierarchically by iterative computing and collecting them.

   In order to balance the collection of seeds in each category, we need to take care of two factors:

   (a) Relevance of seed

   Seeds should be collected and ranked according to their relevance, we want the most relevant seeds to be prioritized in each emotion category. We will use the semantic similarity scores to rank the synonyms, since seeds are mostly synonyms, we compare how close they are to the base emotion in terms of semantic.

   (b) Balanced Category

   We need to ensure the sum of occurrences of all the seeds in each emotion category is similar to ensure the balance between different central emotions.

3. **Construct the emotion vector**

   We will construct an emotion vector that stores the intensity of each emotion in the base emotions collection. The emotion vector is defined as: $I = (I_1, I_2 ....., I_6)$, in which I_k represents the intensity of k_{th} emotion in the word. In our project, the emotion vector is in the following order: 'joy', 'sadness', 'surprise', 'anger', 'fear', 'love'.

   In order to construct the emotion vector of the word w, we will calculate the

intensity of each emotion in the word separately, the calculation is done by:

$$I_k = Intensity(w, cat_k) = \sum_{seed_j \in cat_k} co\_occur(w, seed_j) \tag{6.1}$$

The intensity of the k_{th} emotion of word w, is the total occurrence of the word and all the seeds belong to that emotion category. This is the reason of the necessity of having balanced emotion categories as one category with a significantly larger occurrence will naturally be biased as its seed will have a larger opportunity to co-occur with the word w.

However, the word w itself might be a rare term in itself and only occurs a handful amount of time in the corpus. This might leads to extremely sparse occurrences, causing difficulties to calculate the intensity. Therefore, to capture the sentiment direction of the word w more accurately, we also compute the sentiment direction of some synonyms of the word w with a similar sense and combine them. This is done by:

$$I_k = \sum_{syn_i \in synset(w, sense(w))} Intensity(syn_i, cat_k) \tag{6.2}$$

We will calculate the intensity of the word w and its synonyms. In the actual implementation, due to the occurrences of multiple senses of the same word and the lack of quality synonyms computed by synset. I decided to:

(a) Include all the synonyms regardless of the sense.

(b) Manually add quality synonyms for words that is important for my project.

4. Compute the Sentiment Similarity between words

We first compute the correlation between the emotion vector of word X and Y. This is defined as:

$$corr(X, Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)S_X S_Y} \tag{6.3}$$

where $n = 6$ is the total amount of emotion categories. $\bar{X}, \bar{Y}$ are the means and S_X, S_Y are the standard deviation of emotion vectors. The formula computes the strength of the linear relationship between two words in terms of their sentiment direction. The correlation ranges from -1 to 1, indicating a complete dissimilarity to a strong similarity.

However, the positivity only indicates the general direction of the sentiment similarity, we still need to know the magnitude of similarity. To do this, we adopt an approach similar to [12]. As explained in the work, antonyms are in the same scalar group but have opposite sentiment directions. Therefore, for two words to have similar sentiment direction, they must satisfy the two conditions:

(a) $corr(w_1, w_2) > corr(\tilde{w}_1, w_2)$

(b) $corr(w_1, w_2) > corr(w_1, \tilde{w}_2)$

where $\tilde{w}_i$ means the antonym of w_i. This essentially suggests that for two words to be sentimentally similar, they must pose a stronger linear relationship compared to the relationship between their antonyms.

Lastly, to properly measure the Sentiment Similarity, we will subtract the strength of the relationship between antonyms from the relationship between the original two words.

$$Sentiment\_Similarity(w_1, w_2) = corr(w_1, w_2) - \max\{corr(\tilde{w}_1, w_2), corr(\tilde{w}_2, w_1)\} \tag{6.4}$$

6.4 Example

Here we present an example:

Word	Example Synonyms	Emotion Vector
joy	'happy', 'excitement', 'delight', 'elation', 'exhilaration'	[243, 54, 20, 98, 28, 110]
anger	'irritation', 'rage', 'fury', 'indignation', 'angry', 'pissed'	[119, 72, 4, 140, 7, 61]

Table 6.1: Example emotion vector for word "joy" and "anger"

As we can see from the table, word "joy" have a much higher value on the fisrt index which resembles the emotion "joy" than word "anger", while a lower value on the fourth index which resembles the emotion "anger". This aligns with human understanding of word "joy" and "anger", (in this case they are literally the same as two of the target emotions). We can also obtain useful insights by examine other values in the emotion vector. For example, the last term in the emotion vector resembles emotion "love" and "joy" achieved a higher score on it compared to "anger". This is telling us the word "joy" often appears around words that express the emotion of "love", which makes sense from human percetion of emotion.

We can obtain the emotion vectors for the collection of synonyms and antonyms following the same procedure and use them to calculate various correlation we need in order to calculate the Sentiment Similarity.

Chapter 7

Experiment and Evaluation

In this chapter, I will walk through the actual setup of the experiment and evaluate the results obtained.

7.1 Baseline Performance

The metrics in the following table show the averaged performance achieved by our baseline model on all the random sampled test sets. To learn more about randomly sampled test sets, check the later section. 7.2.5

Models	Weighted Precision	Weighted Recall/Accuracy	Weighted F1-score	Similalrity Score
Baseline	0.8618	0.8602	0.8598	0.8301

Table 7.1: Baseline Performance averaged over 400 test sets

7.2 Setup

7.2.1 Test Sets

In addition to the split 20,000 emotion tweets that are already split into train, validation and test sets, the author of the emotion database also provided 417,000 additional unsplit data.

Since the performance of each model on a single test set is not a strong indication of how well a model performs compared to another model, creating multiple test sets by randomly sampling from these raw data is a sound strategy to gather more metrics for comparison. We decided to generate 400 randomly sampled test sets to benchmark our models against.

Author-provided split data employed a $80:10:10$ split between train-validation-tests, the same test set size is followed when forming additional test sets. I also decided to generate balanced test sets of size 1998, which contain 333 sentences for each of the six

emotion classes. This would help us pick out models that are biased towards specific classes.

7.2.2 Explanation Gathering

While it would be beneficial to gather the explanations for both correct and incorrect predictions, the time required to compute an explanation for a sentence ranges from 4 seconds to 2 minutes, based on the length of the sentence, when the neighbouring window size is set to 2 subsentences. I could've achieved better performance by reducing window size, but this will lead to less context taken into consideration, therefore I kept the value unchanged. The bottleneck is the string modification operations as I have to mask subsentences, and the amount of possible combinations of masked and unmasked subsentences grows exponentially when the sentence size grows.

Due to the time constraints, I've only gathered the explanation for all the mispredictions in the validation set as it is more time efficient.

7.2.3 Data Generation and Model Retraining

To verify the effect of introducing explanations back into the model update and check if the performance is improved effectively. We came up and experimented with 4 data augmentation methods. All the prompts sent to GPT can be found in the appendix. A.5

1. **Human Random Generation** For every misprediction, we randomly sample 20 tweets with the expected emotion from the extra data available from the dataset.
2. **GPT Random Generation** For every misprediction, we pass **only** the expected emotion to the GPT and ask for 20 sentences with proper lengths that fit the nature of our dataset.
3. **Random Sampling** For every misprediction, we pass the expected emotion and one random sentence selected from the training data with the same expected emotion. We request 20 sentences of a similar writing style as the referenced tweet and embedding the expected emotion.
4. **Explanation-Based Generation** For every misprediction, we pass in the explanations generated for the tweets, along with predicted and expected emotions. We explain the structure of explanations and instruct GPT to detect problems and generate sentences that counter the biases and mistakes.

We will generate the same amount of new training sentences for all 4 different ways, which will allow us to check if leveraging the explanations can help researchers generate data that can improve the model performance and correct the inner bias and mistakes a model imposes more effectively. If training on new training data that are generated by leveraging explanations fixes the model's bias better than training on random new data, **we may conclude that explanations can help generate new training data that can fix the model's issue in a more effective way when the same amount of resources are given.**

7.2.4 Sentiment Similarity

As explained in the dedicated chapter, sentiment similarity depends heavily on the co-occurrence of words contained in the seeds for each emotion label. The ideal scenario for us is to use the extra data provided by the authors to compute the similarities. However, tweet data are naturally short, which results in a short count of co-occurrence if we consider every single tweet as a stand-alone corpus. Initially, I attempted to concatenate the tweets with the same emotion label and treat it as a single corpus, but the co-occurrence across tweet data doesn't appear to be a reasonable design decision. Therefore, I need to find a replacement corpus that contains sentences with greater length but with a similar nature to the tweets.

After some research, I decided to compute sentiment similarity based on the IMDB corpus, constructed by the author of this paper. [11]. Movie reviews are emotion-rich corpus just like tweets but come with greater average lengths, which makes them a better candidate for the seeds words co-occurrence computation.

To compute the **Sentiment Score**, we will compute the average of sentiment similarity over the whole test set. We will punish for misprediction by deducting the scores our models achieved on the test set, and the strength of punishment will be adjusted according to how distant the predicted and expected emotions are. More distant emotions will be punished harder. For example, a misprediction of "sadness" for expected "joy" will result in a **-0.81** penalty, which is much stronger than the penalty if the misprediction was "love" instead, which only results in a penalty of **-0.04**.

After carrying out the steps described in the Sentiment Similarity section, we obtained the scores for the 6 distinct emotion target labels as shown in figure 7.1.

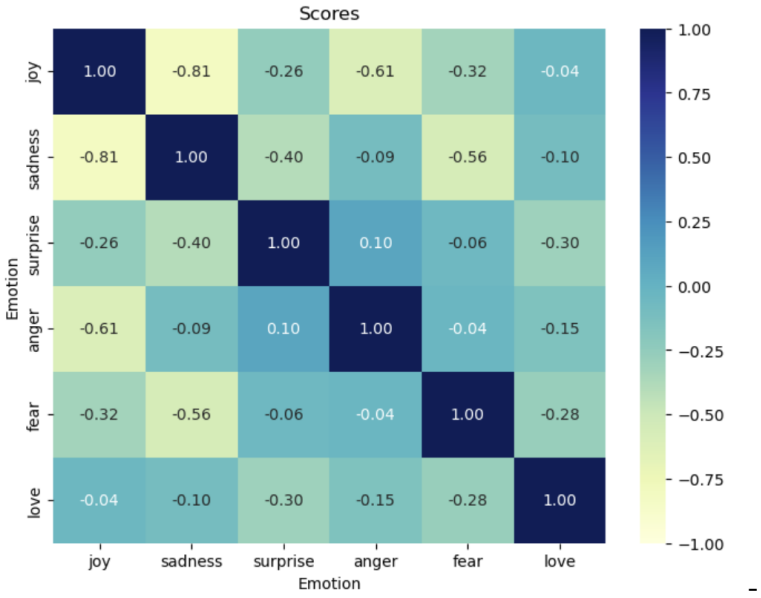

Figure 7.1: Sentiment Similarity for 6 target emotions, showing how dissimilar our emotions are and punishes harder for less related emotion pairs. A misprediction of "sadness" for expected "joy" will be punished harder than a misprediction of "love".

7.2.5 Test Sets

In addition to the split 20,000 emotion tweets that are already split into train, validation and test sets, the author of the emotion database also provided 417,000 additional unsplit data.

Since the performance of each model on a single test set is not a strong indication of how well a model performs compared to another model, creating multiple test sets by randomly sampling from these raw data is a sound strategy to gather more metrics for comparison. We decided to generate **400** randomly sampled test sets to benchmark our models against.

Author-provided split data employed a 80 : 10 : 10 split between train-validation-tests, the same test set size is followed when forming additional test sets. I also decided to **generate balanced test sets of size 1998, which contain 333 sentences for each of the six emotion classes.** This would help us pick out models that are biased towards specific classes.

7.2.6 One-tailed Dependant Paired Sample T-test

To statistically compare the performance of different models on the collection of generated test sets, I decided to employ the One-tailed Paired Sample T-test. To gain more insight into the type of T-test conducted in this project, please refer to this section A.3 in the appendix.

In our case, all the performance metrics achieved by all the models form normal distributions. An example figure plots the metric **Sentiment Score** of all the models is displayed below.

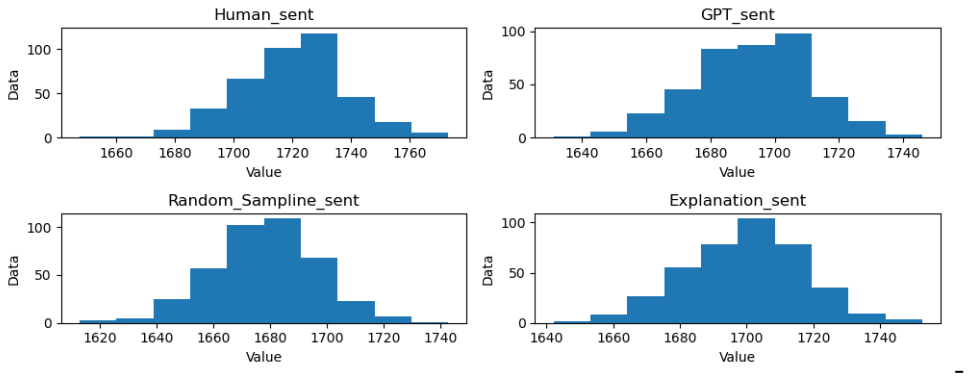

Figure 7.2: Histogram of performance metric ”Sentiment Score” of all models

The variance ratio between the performance metrics achieved by different models is approximately 1 hence we are safe to consider them as having the same variance.

Before conducting a T-test, we need to ensure the prerequisite for it is satisfied. In our project-specific t-test, the null hypothesis is that the two models have no significant performance difference, and the alternative hypothesis is the model with a lower metric

mean performs worse than the model with a higher metric mean. In this case, the p-value is calculated as the $P(T < t - value)$ under the t-distribution.

7.3 Evaluation

7.3.1 Standard Metric and Sentiment Score

The performance of all the models is displayed in the following table, note that precision, recall and F1-score are all weighted.

Models	Precision	Recall/Accuracy	F1-score	Sentiment Score
Baseline	0.8618	0.8602	0.8598	0.8301
		Human-Based Models		
Human Random Generation	**0.8847**	**0.8838**	**0.8832**	**0.8601**
		GPT-Based Models		
GPT Random Generation	0.8732	0.8721	0.8713	0.8467
Random Sampling	0.8679	0.8670	0.8664	0.8390
Explanation-Based Generation	**0.8770**	**0.8755**	**0.8744**	**0.8496**

Table 7.2: Averaged Metrics for the models over all the randomly sampled test set

All the comparison done in the following evaluation is conducted through **T-test** rather than mere number comparison, however, we will refer to the numbers in the table as an indication of overall model performance to make the analysis less cluttered.

We first compare the performance among all the models, it is clear that the model trained on human-generated data achieved the best results among all the metrics and surpasses the second-best model, which is the explanation-based model by a significant amount. It is also worth emphasising that the improvement of the Sentiment Score achieved by the Human Random Generation model over the Baseline is the most significant, indicating that more correct predictions were made and mispredictions are less distant from the expected emotions.

It might appear weird that Human Random Generation is better than other methodologies. However, this behaviour is **expected** since the human-generated data comes from the same dataset as the tweet data used in the test sets, hence resembling that pattern underneath better than all the GPT-generated data. Due to the inherent difference between the human-generated data and GPT-generated data, it would be fair to compare the sub-group of models trained with the GPT-generated data and determine which model within this subset achieved the best performance. Among all the GPT-based models, the **Explanation-Based Generation** is ruling the rest, with GPT Random Generation being the second and the Random Sampling achieving the least ideal performance.

Considering the inner difference residing within the additional training data, we will hence still consider the following conclusion valid:

Explanation Based $>$ **Randomly Generated** $>$ **Random Sampling**

Therefore, the Explanation-driven data augmentation is proven to be **beneficial** in terms of model performance improvement according to the results. We would also argue that the better performance achieved by Human Random Generation compared to the Explanation-Based model is still debatable without additional experiments using properly formatted training data that better resembles the original dataset.

7.3.2 Explanation Examination

While the newly trained models are compared according to the performance, we still want to examine if the bias we discovered in the Explanation Generation section is corrected. Ideally, the neutral terms should have smaller contribution scores comparing to the baseline model as our goals are to fix any existing bias. It would also be desirable if the wrong prediction is now corrected.

We will examine the same example sentence as the ones we presented in the Explanation Generation Example section.

Correct Prediction By the Baseline Model: *"i feel dirty and ashamed for saying that"*: Examining all the explanation output by the models, we noticed that despite

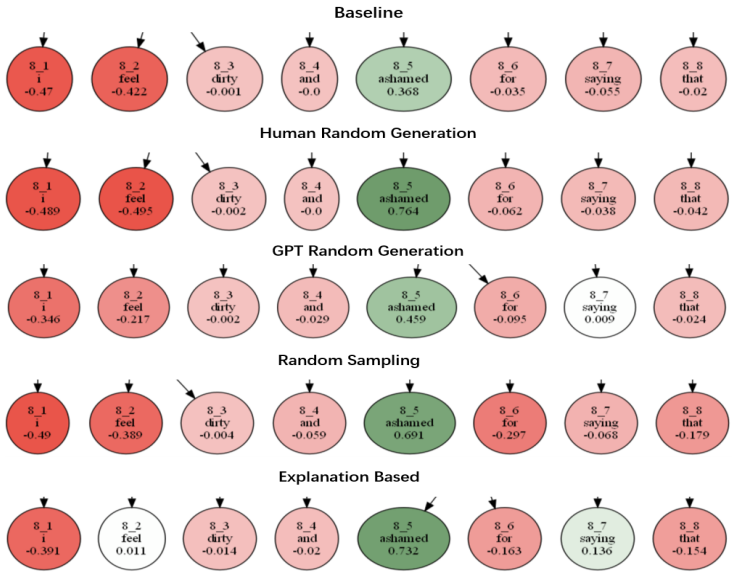

Figure 7.3: Explanation for correct prediction example using all models

of the fact that the human random-generated model achieved the best performance in all the metrics, it didn't eliminate the bias associated with the neutral term. The contribution score of neutral terms such as "i" and "feel" indicates a worse bias in our model. Among the 3 GPT-based models, the random sampling model is the worst when it comes to eliminating biased contribution scores. These observations make sense since the training data are generated not based on the original sentences hence they don't put any focus on shifting the contribution scores of specified neutral terms.

To my surprise, the GPT-based random-generated model surprisingly reduces the contribution scores of the neutral terms to a quite satisfactory degree. It is not expected

since the new data generated are solely based on the emotion label and has no access to the original sentence. I would argue this scenario happens due to coincidence.

Finally, the explanation-based model successfully reduced the contribution score of the neutral term ”feel” to a very satisfactory magnitude while other neutral terms were also brought down to a reasonable range. It is also worth noting that when compared to the GPT-based random-generated model, in addition to the better scoring for neutral terms, the core term ”ashamed” which contributes heavily to the label ”sadness” is given a higher contribution score in the explanation-based model, which is more desirable.

Please note that the effect not only appears at the token level, but the explanation-based model also assigned more reasonable contribution scores for the subsentences. Due to the space restriction, the complete explanation tree will be attached to the appendix.

Wrong Prediction By the Baseline Model: *”i was feeling extremely whiney and lonely and sad”*:
Before going into the examination of the resulting explanations, we need to note that

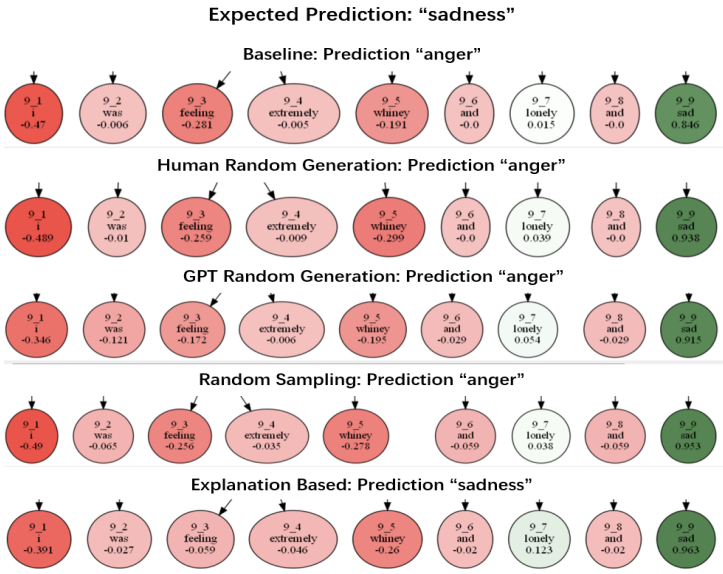

Figure 7.4: Explanation for wrong prediction example using all models

despite the similarity between results, **the explanation-based model actually makes the correct prediction while the rest of the models still make the same wrong prediction as the baseline model**. The explanation tree might appear to be similar but we need to be aware that the explanations for all the models except for the explanation-based model are respect to wrong label ”anger” while the one for explanation-based model is respect to the correct label ”sadness”.

In terms of the neutral terms, we do observe some of them being brought down to a proper magnitude such as the term ”feeling”, but overall the neutral terms assigned better contribution scores compared to the other models. This might be related to the amount of sentences we generated for each misprediction, the newly generated data itself can biases, which should be fixed in the next cycle.

Chapter 8

Conclusion and Limitation

8.1 Discussion

The project aims to verify the feasibility of explanation-driven training data generation. According to our experiment, although the Human-based random-generated model outperforms all the other models, the explanation-based model is still proven to be valuable as it dominates all the models that are trained on GPT-generated data that are collected using different methodologies.

In addition, when we further examine the explanations generated by the model, we notice that despite not showing the best performance, the explanation-based model can reduce the biased contribution score assigned to neutral terms to an acceptable magnitude. While it is not the only model that achieves this purpose, it certainly poses the greatest potential in removing the bias in the long run if more cycles or training is introduced.

Therefore, we can conclude that **explanation-based model is indeed beneficial when the same data augmentation methods are applied compared to random sampling and random data generation**.

In addition, we were also curious about the feasibility of replacing human feedback with GPT-generated text data. Judging from the model performance, I would argue that **while GPT is efficient in terms of generating massive amount of data, it still can't offer text data of the same quality as human-generated data**. A more detailed discussion is in the **Limitation** section. 8.2.2

8.2 Limitation and Future Work

While our assumption is proven to a certain degree, the experiments are still far from perfect, and more aspects of the explanation-based model need to be examined. In the following section, I will list all the limitations and future works that I believe are beneficial for offering stronger evidence or proving the conclusion incorrect.

8.2.1 Explnations are only generated for misprediction

Despite limiting the neighbouring window size as described in the Explanation chapter which reduces the computation complexity to polynomial, the time required to generate explanations for longer sentences is still not ideal. This is the main obstacle that prevented me from generating explanations for the whole validation set, as it will take days to generate explanations. The bottleneck is the computation time required for iterating all the possible combinations of the neighbouring subsentences by using repetitive Python string slicing and heavy optimisation is required if the whole explanation generation pipeline were to be applied to a larger dataset.

While it may be justifiable to make such a time investment for a validation set with 2,000 sentences, it would be useless when dealing with sentences of greater length or a larger validation set. Therefore, in terms of potential future work, maybe we can replace the heavy use of string slicing by introducing pre-sliced sentences and changing the code to adapt a more dynamic programming style.

If we can obtain the complete set of explanations for the whole validation set, then studying biased neutral terms even if the prediction is correct will be possible. This will most likely make a positive contribution to the experiment.

8.2.2 GPT-generated data doesn't resemble data in original dataset

We suspect that the reason behind the poor performance achieved by models trained on GPT-generated data is the subtle difference between GPT-generated data and the original human-written tweets. Introducing too much synthetic data will pollute the training dataset and eventually affect the model's ability to classify real human text.

However, it's a tradeoff must be made as it would be extremely time and cost-inefficient if only human-generated data is acceptable. In our experiment, all the models accepted an average of 4,000 new sentences. Requesting a human feedback provider to come up with this quantity of tweet data while following the guidance of explanation is simply impossible to achieve within a reasonable timeframe. This project has to accept this problem and only compare models trained with additional synthetic data, and the resulting performance already indicates a deterioration due to the introduction of non-human tweets despite the small quantity.

To fully confirm the effectiveness of explanation-driven training data generation, future work should ideally collect new data that are generated by human. Only when such data is available, then we can verify the project's assumption more rigorously.

It's also worth noting that while engineering the prompt passed to GPT is possible, we can't quantify how well GPT leverages the explanations. It is impossible to rely on the GPT and assume it considers solely the explanation instead of simply generating sentences with a similar writing style as the original mispredicted sentence. In my project, despite my explicitly prompt engineering to prevent GPT from simply returning sentences with the same format as much as possible, the final quality of generated sentences based on explanation is still questionable.

8.2.3 More fine-tuning of parameters throughout the experiment

Many parameters used throughout the experiment were not fine-tuned enough to ensure the best outcome of explanation-driven training data generation, not limited to the training parameters of the model.

In the explanation generation algorithm, a window size of 4 is used, with equal size on both sides of the currently examined subsentences. Future work may set imbalanced side sizes according to the nature of the language they are examining or increase the window size to include more context for a better explanation generation.

Furthermore, we are generating 20 sentences for every misprediction, this may not be the most optimal amount. Too little new data may not be enough to counter the biases existing in the current model while too much new data may introduce new biases. Lastly, we believe a single cycle of introducing new data and retraining is not enough, a better outcome may be achieved if the model is retrained multiple times using the explanation-driven data augmentation.

8.2.4 Explanation still not human-friendly

Despite the effort of trying to make the model's reasoning process more humanly comprehensible, the resulting explanations are still puzzling due to the nature of the Machine Learning and the Algorithm we implemented. In some cases, the root of misprediction is obvious, while in others it's less so. Even though the illustration is more straightforward than studying the numbers, it can still get confusing when the input is too long. Future work should investigate either more sophisticated ways of generating explanations or constructing a framework that helps humans understand the explanations better.

8.2.5 Additional preprocessing required for efficient bias cancelling

The original training data, while preprocessed already, is not in the best form to be learnt and improved upon. The same terms presented in different forms due to capitalization not being altered to be coherent leads to separate optimization. For example, new training data are generated separately to cancel the bias presented in the terms "i" and "I" leading to the generation of new training data that can't be leveraged to cancel the bias in both terms while it was supposed to be. Such preprocessing steps are not hard to implement but due to the time required to walk through the complete process and retrain for all the new models, we sadly have to choose to leave it to future work.

8.2.6 Misalignment of ground truth label and human expectation

The original dataset contains ground true labels that doesn't align with he human expectation. Some of them can even be considered mislabeled. This leads to unnecessary correction when our model's prediction is more reasonable but forced to be "corrected" into a wrong prediction. Or it might overlook a wrong prediction due to the incorrect target label. Future work can spend some time verifying the obtained dataset to ensure its correctness or following better-constructed data-gathering steps.

Bibliography

[1] How to interpret a confusion matrix for a machine learning model.

[2] Muhammad Abdul-Mageed and Lyle Ungar. EmoNet: Fine-grained emotion detection with gated recurrent neural networks. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[4] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection, 2020.

[5] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. L-shapley and c-shapley: Efficient model interpretation for structured data, 2018.

[6] Marina Danilevsky, Kun Qian, Ranit Aharonov, Yannis Katsis, Ban Kawas, and Prithviraj Sen. A survey of the state of explainable AI for natural language processing. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 447–459, Suzhou, China, December 2020. Association for Computational Linguistics.

[7] Katsushige Fujimoto, Ivan Kojadinovic, and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55(1):72–99, 2006.

[8] Michel Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2):167–189, 1997. Fuzzy Measures and Integrals.

[9] Piyawat Lertvittayakumjorn and Francesca Toni. Explanation-based human debugging of nlp models: A survey, 2021.

[10] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational*

Linguistics: Human Language Technologies, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[12] Mitra Mohtarami, Hadi Amiri, Man Lan, and Chew Lim Tan. Predicting the uncertainty of sentiment adjectives in indirect answers. In *International Conference on Information and Knowledge Management*, 2011.

[13] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021.

[14] Alena Neviarouskaya, Helmut Prendinger, and Mitsuru Ishizuka. Compositionality principle in recognition of fine-grained emotions from text. *Proceedings of the International AAAI Conference on Web and Social Media*, 2009.

[15] Andrew Ortony and Terence J. Turner. What's basic about basic emotions? *Psychological review*, 97 3:315–31, 1990.

[16] Guillermo Owen. Multilinear extensions of games. *Manage. Sci.*, 18(5-part-2):64–79, jan 1972.

[17] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[18] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[19] Sabrina Simmons and Zachary Estes. Using latent semantic analysis to estimate similarity. 01 2006.

[20] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[22] wikipedia. Distributional semantic.

Appendix A

First appendix

A.1 Raw Explanation outputs from algorithm

Input: i feel accepted by the boys
Expected Emoiton: joy
Predicted Emotion: love

([['i feel accepted by the boys'], ['i feel accepted', 'by the boys'], ['i', 'feel accepted', 'by the boys'], ['i', 'feel accepted', 'by the', 'boys'], ['i', 'feel accepted', 'by', 'the', 'boys'], ['i', 'feel', 'accepted', 'by', 'the', 'boys']], [(['i feel accepted'], tensor(0.0227, device='cuda:0')), ([' by the boys'], tensor(-0.1250, device='cuda:0')), (['i'], tensor(-0.3338, device='cuda:0')), ([' feel accepted'], tensor(0.1998, device='cuda:0')), (['by the'], tensor(-0.0625, device='cuda:0')), ([' boys'], tensor(-0.0017, device='cuda:0')), (['by'], tensor(-0.0009, device='cuda:0')), ([' the'], tensor(1.4480e-06, device='cuda:0')), (['feel'], tensor(-0.3163, device='cuda:0')), ([' accepted'], tensor(0.0043, device='cuda:0'))])

Input: i feel hated there but had to remind my selfish self that none of this was about me
Expected Emotion: sadness **Predicted Emotion**: anger

([['i feel hated there but had to remind my selfish self that none of this was about me'], ['i feel hated there but', 'had to remind my selfish self that none of this was about me'], ['i feel hated there but', 'had to remind my', 'selfish self that none of this was about me'], ['i feel hated', 'there but', 'had to remind my', 'selfish self that none of this was about me'], ['i feel hated', 'there but', 'had to remind my', 'selfish self that none of this was', 'about me'], ['i feel hated', 'there but', 'had to remind my', 'selfish self that none of this was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had to remind my',

'selfish self that none of this was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had', 'to remind my', 'selfish self that none of this was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had', 'to', 'remind my', 'selfish self that none of this was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self that none of this was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self that none', 'of this was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self that none', 'of this', 'was', 'about', 'me'], ['i feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self that none', 'of', 'this', 'was', 'about', 'me'], ['i', 'feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self that none', 'of', 'this', 'was', 'about', 'me'], ['i', 'feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self that', 'none', 'of', 'this', 'was', 'about', 'me'], ['i', 'feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish self', 'that', 'none', 'of', 'this', 'was', 'about', 'me'], ['i', 'feel hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish', 'self', 'that', 'none', 'of', 'this', 'was', 'about', 'me'], ['i', 'feel', 'hated', 'there', 'but', 'had', 'to', 'remind', 'my', 'selfish', 'self', 'that', 'none', 'of', 'this', 'was', 'about', 'me']], [(['i feel hated there but'], tensor(0.6048, device='cuda:0')), ([' had to remind my selfish self that none of this was about me'], tensor(0.4523, device='cuda:0')), (['had to remind my'], tensor(-0.4600, device='cuda:0')), ([' selfish self that none of this was about me'], tensor(0.3932, device='cuda:0')), (['i feel hated'], tensor(0.7835, device='cuda:0')), ([' there but'], tensor(-0.0512, device='cuda:0')), (['selfish self that none of this was'], tensor(0.4147, device='cuda:0')), ([' about me'], tensor(-0.2404, device='cuda:0')), (['about'], tensor(-0.0006, device='cuda:0')), ([' me'], tensor(-0.1833, device='cuda:0')), (['there'], tensor(-0.0237, device='cuda:0')), ([' but'], tensor(-0.0057, device='cuda:0')), (['had'], tensor(-0.0151, device='cuda:0')), ([' to remind my'], tensor(-0.6445, device='cuda:0')), (['to'], tensor(-1.8749e-07, device='cuda:0')), ([' remind my'], tensor(-0.5509, device='cuda:0')), (['remind'], tensor(-0.0097, device='cuda:0')), ([' my'], tensor(-0.2514, device='cuda:0')), (['selfish self that none'], tensor(0.0299, device='cuda:0')), ([' of this was'], tensor(-0.0722, device='cuda:0')), (['of this'], tensor(-0.0514, device='cuda:0')), ([' was'], tensor(-0.0073, device='cuda:0')), (['of'], tensor(-1.8943e-06, device='cuda:0')), ([' this'], tensor(-0.0136, device='cuda:0')), (['i'], tensor(-0.5130, device='cuda:0')), ([' feel hated'], tensor(0.6622, device='cuda:0')), (['selfish self that'], tensor(0.0894, device='cuda:0')), ([' none'], tensor(-0.0048, device='cuda:0')), (['selfish self'], tensor(0.0216, device='cuda:0')), ([' that'], tensor(-0.0214, device='cuda:0')), (['selfish'], tensor(0.0091, device='cuda:0')), ([' self'], tensor(-0.0003, device='cuda:0')), (['feel'], tensor(-0.4481, device='cuda:0')), ([' hated'], tensor(0.0701, device='cuda:0'))])

Input: i hide what i am truly feeling thinking for fear that it will lead to something far more dangerous
Expected Emotion: anger

Predicted Emotion: fear

([['i hide what i am truly feeling thinking for fear that it will lead to something far more dangerous'], ['i hide what i am truly feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide what i am truly feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide what i am truly', 'feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide what i am', 'truly', 'feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide what', 'i am', 'truly', 'feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide', 'what', 'i am', 'truly', 'feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for fear that it will lead to something far more dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for fear that it will lead to', 'something far more dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for fear that it will lead to', 'something', 'far more dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for fear that it will lead to', 'something', 'far', 'more dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for fear that it will lead to', 'something', 'far', 'more', 'dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for fear that it will lead', 'to', 'something', 'far', 'more', 'dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for', 'fear that it will lead', 'to', 'something', 'far', 'more', 'dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for', 'fear that it will', 'lead', 'to', 'something', 'far', 'more', 'dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for', 'fear', 'that it will', 'lead', 'to', 'something', 'far', 'more', 'dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for', 'fear', 'that', 'it will', 'lead', 'to', 'something', 'far', 'more', 'dangerous'], ['i', 'hide', 'what', 'i', 'am', 'truly', 'feeling', 'thinking', 'for', 'fear', 'that', 'it', 'will', 'lead', 'to', 'something', 'far', 'more', 'dangerous']], [(['i hide what i am truly feeling thinking'], tensor(-0.3614, device='cuda:0')), ([' for fear that it will lead to something far more dangerous'], tensor(0.9882, device='cuda:0')), (['i'], tensor(-0.5151, device='cuda:0')), ([' hide what i am truly feeling thinking'], tensor(-0.3712, device='cuda:0')), (['hide what i am truly'], tensor(-0.6109, device='cuda:0')), ([' feeling thinking'], tensor(0.1671, device='cuda:0')), (['hide what i am'], tensor(0.0131, device='cuda:0')), ([' truly'], tensor(-0.2278, device='cuda:0')), (['hide what'], tensor(0.4861, device='cuda:0')), ([' i am'], tensor(-0.7614, device='cuda:0')), (['hide'], tensor(0.0054, device='cuda:0')), ([' what'], tensor(-0.0113, device='cuda:0')), (['i'], tensor(-0.5151, device='cuda:0')), ([' am'], tensor(-0.0048, device='cuda:0')), (['feeling'], tensor(-0.3216, device='cuda:0')), ([' thinking'], tensor(0.0029, device='cuda:0')), (['for fear that it will lead to'], tensor(0.9722, device='cuda:0')),

([' something far more dangerous'], tensor(0.5996, device='cuda:0')), (['something'], tensor(-0.0018, device='cuda:0')), ([' far more dangerous'], tensor(0.1739, device='cuda:0')), (['far'], tensor(-0.0004, device='cuda:0')), ([' more dangerous'], tensor(0.2102, device='cuda:0')), (['more'], tensor(-0.0323, device='cuda:0')), ([' dangerous'], tensor(0.0454, device='cuda:0')), (['for fear that it will lead'], tensor(0.9818, device='cuda:0')), ([' to'], tensor(-1.8040e-07, device='cuda:0')), (['for'], tensor(-0.0312, device='cuda:0')), ([' fear that it will lead'], tensor(0.9774, device='cuda:0')), (['fear that it will'], tensor(0.9706, device='cuda:0')), ([' lead'], tensor(-0.0085, device='cuda:0')), (['fear'], tensor(0.9593, device='cuda:0')), ([' that it will'], tensor(0.0634, device='cuda:0')), (['that'], tensor(-0.0173, device='cuda:0')), ([' it will'], tensor(-0.0431, device='cuda:0')), (['it'], tensor(-0.0201, device='cuda:0')), ([' will'], tensor(-0.1454, device='cuda:0'))]) Click to add a cell.

A.2 Latent Semantic Analysis

LSA consists of 3 steps in order to compute the semantic similarity between word pairs [19]:

1. Construct a matrix with row being words and columns being context, context can be any types of documents. Essentially we are trying to construct a Document-Term Matrix, which has a dimension of $\#vocabularies * \#documents$. Such Matrix will store the terms appearance in all the documents.

2. In order to uncover the higher order associations between words, we need to apply SVD decomposition and dimension reduction.

   SVD decomposition and dimension reduction allows us to extract and untangle information. After, the words occur in the same or similar context will become more similar. The words representation now are a vector in semantic space which summarize the information of context in which the term is found.

3. Calculate the distance between words' vector representation, we have many options that can be used to calculate distance. In LSA, we use Cosine Similarity, which has a formula of:

$$cosine\_similarity = \frac{v1 \cdot v2}{||v1|| * ||v2||} = \frac{\sum_{i=1}^{d}(v1_i * v2_i)}{\sqrt{\sum_{i=1}^{d}(v1_i)^2} * \sqrt{\sum_{i=1}^{d}(v2_i)^2}}$$

A.3 One-tailed Dependant Paired Sample T-test

T-test is a Hypothesis Testing Procedure used to compare the mean of two different variables, and the goal is to examine if the mean difference is statistically different from 0.

Dependent Paired Sample T-test measures the same entity twice using different measures which we are comparing, resulting in a paired observation. In our case, the entity is the test set that our models all evaluated on and the observations are the performance metrics.

With the observations ready, we want to define two competing hypotheses, the **Null Hypothesis** and the **Alternative Hypothesis**. The former assumes the mean difference between two observations is 0 and any observed difference is merely caused by random variation. In contrast, the latter assumes that the mean difference is different from 0, indicating the performance between our models is statistically significantly different. We denote **Null Hypothesis** and **Alternative Hypothesis** as H_0 and H_1 respectively.

When conducting Hypothesis Testing, two options are available for us to choose from, namely two-tailed and one-tailed Hypothesis Testing. Two-tailed Hypothesis Testing is used when we are unsure of the direction of the mean difference, i.e. we are not certain which model performs better, therefore we aim to check the assumption that **"There are statistically significant mean differences between two samples"**. Conversely, the one-tailed test is better suited when we are aware of the direction of the mean difference. In this case, the assumption we are checking is **"The mean of one sample is more/less than the mean of the other sample"**.

Since the performance difference is easy to examine and the direction of the performance difference, i.e. which one of the two models we are comparing performs better, is easy to observe via examining the mean, I would argue that **one-tailed** would be more suitable.

The next step is to compute the T-value, which measures the size of difference relative to the variance of our sample data. Assume we are comparing sample **A** and **B**, the formula for two samples t-value calculation is:

$$t - value = \frac{\overline{X_A} - \overline{X_B}}{\sqrt{\frac{std_A^2}{N_A} + \frac{std_B^2}{N_B}}}$$

in which N_A and N_B represent the size and $\overline{X_A}$ and $\overline{(X_B)}$ represent the sample mean of samples A and B.

The final step is to compute the P-value from the T-value and use it to decide whether to reject or accept the Null Hypothesis. The P-value is the indication of how likely the current observations happen if the null hypothesis is indeed true. We can compare the p-value we obtained with a cut-off value. The critical value at a certain significance level can be thought of as a cut-off point, which indicates how sure we want to be about our assumption. A significance level is the probability of an event occur by chance, normally set to 5%. If the p-value falls lower than the significance level, we argue that the current observation is unlikely to happen if the null hypothesis is true, therefore we reject it. p-value is calculated differently according to the type of t-test we are conducting, the formula used to calculate p-value in our specific case will be presented below.

Finally, we need to check the observation data we collected satisfies certain conditions to conduct an ideal T-test:

1. The data needs to form a normal distribution.
2. The data must have the same variance.
3. There can't be outliers since they will affect the result and may incorrectly reject or accept the Null Hypothesis.

Variance are approximately the same since the ratio between observations variance are within the range of 1.006 - 1.08, it's safe for us to assume that the variance is approximately the same. If we want a more rigorous verification, we can conduct **F-test** on the variances and determine if the variances are indeed statistically the same. **F-test** is a one-tailed Hypothesis Testing problem and follows similar steps described above. Due to the insignificant difference between the variance ratio and 1, we are not discussing **F-test** here. The observations also don't contain extreme outliers that need to be removed. Hence we can conduct the One-tailed Dependent Paired T-test.

A.4 Confucion Matrix and its content

There are four categories of results that a machine-learning model can predict: Considering one emotion table as Positive Class during the testing phase, and all the classes as negative classes. By doing so we are measuring how good our model performs when it encounters an input that is supposed to be classified as the positive class.

With Positive Class defined, we further define the True and False as predicted the result correctly and incorrectly.

1. True Positive (TP): The cases where the model correctly predicts the positive class.
2. True Negative (TN): The cases where the model correctly predicts the negative class.
3. False Positive (FP): The cases where the model incorrectly predicts the positive class (Type I error).
4. False Negative (FN): The cases where the model incorrectly predicts the negative class (Type II error).

Having those 4 terms defined, we can understand the **Confusion Matrix**. A confusion matrix is a table used to evaluate the performance of a classification model. The definition of terms contained within are in appendix :

A.5 Data augmentation prompts

1. **Random Sampling**

   Context

   Dealing with an emotion classification task.

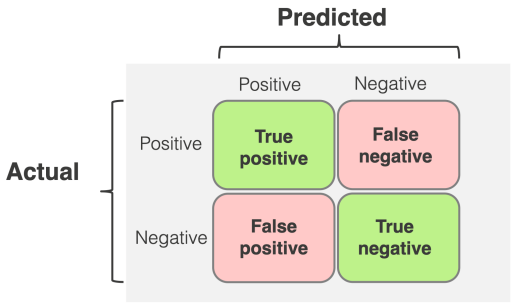

Figure A.1: Confusion Matrix. Image taken from [1]

> I'm passing you a group of sentences, and their expected emotion, separated by line break. The 'Expected emotion' is not part of the sentence. Generate one sentence for each sentence in the group, and they should be sentimentally similar to the corresponding sentence I sent. Use the expected emotion as reference. The length of generated sentences should be similar the corresponding sentences.
>
> The only output you should give me are the generated sentences, in the format of number indexed list, nothing else should be returned.

Content Group of 10 reference sentences and the expected emotions.

2. **Explanation-Based Generation**

> Dealing with an emotion classification task.
>
> I'm passing you a tuple of two lists, followed by a predicted emotion and an expected emotion, which are different. Predicted emotion and expected emotion are not part of the sentence. The first list contains the breakdown of the sentence, the second list contains the contribution scores of each subpart of the sentence to the predicted emotion. Predicted emotion is considered wrong, study the two lists to identify subparts that heavily contribute to the wrong prediction. Generate 20 sentences based on the information learnt from tuple that will help fix the wrong contribution scores, which means the generated sentences should help model assign subparts which indicate expected emotion higher contribution scores and the rest subparts low or negative contribution score. The only output you should give me are the generated sentences, in the format of number indexed list, nothing else should be returned.

Context Explanation, predicted and expected emotion

3. **GPT Random Generation Context**

> I'm passing you an emotion, generate 20 sentences that express this emotion. You don't have to include emotion in every generated sentence.
>
> The only output you should give me are the generated sentences, in the format of number indexed list, nothing else should be returned.

Content Group of expected emotions of mispredicted tweets

A.6 Machine Learning Metrics Formula

The formula for Precision calculation is:

$$Precision = \frac{TP}{TP + FP}$$

The formula for Recall calculation is:

$$Recall = \frac{TP}{TP + FN}$$

The formula for F1-scores is:

$$F1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * Precision * Recall}{Precision + Recall}$$