GoEmotions: Final Report

Ryan Erickson, Samarth Kumar Samal, Will Han, Raghuram Gowrav, Abhay Shashidhara
eric4758@umn.edu, samal026@umn.edu, han00614@umn.edu, maddu046@umn.edu, shash024@umn.edu

Introduction

Emotion Classification from text is a difficult, and important subsection of Natural Language Processing (NLP), of which has a myriad of high impact applications. Human interaction is intrinsically intertwined with human emotion, and as such emotion and tone is one of the most important factors in understanding other individuals. In the physical world, indicators such as inflection and timbre, as well as many other factors, combined with the social norms of a particular group (as well as more broadly, humanity), clue individuals into what emotion someone is portraying (with relatively high success). However, a majority of these features of spoken word are absent in the age of the internet, when reading online text.

In contrast to novels, which typically will have some language that outright, or highly indicates, what emotion is on display, (such as clearly stating the way a character speaks a specific line) online text is famously fickle with portraying tone and emotion, due to the only indication of emotion being the raw text. This disproportionately affects specific, typically marginalized, groups of people, of whom may have difficulty discerning language from text As online conversation does not naturally flow with the explicit declaration of emotion, there have been adopted alternatives, such as tone indicators (/s, /srs, etc). However, these alternatives are typically only in place by subsects of the internet, and are not widely used by all internet users. As such, the problem remains, how do we as humans accurately discern what tone online text is displaying, algorithmically?

The goal of this project and paper is to explore different classical NLP + Machine Learning (ML) models and transformer architectures that may classify emotions from short segments of text sources from various online forums. We use the GoEmotions Dataset[1], a Google dataset that is one of the largest, and most comprehensive emotion classification datasets available, with over two hundred thousand Reddit comments, annotated with twenty-eight emotion classifications. This dataset contains a large amount of different emotions, ranging from more common emotions such as anger or despair, to sentiments such as approval or gratitude. A full list of the emotions within can be found at the GoEmotions Kaggle/Github page. With thirty-two different unique classifications, this problem has difficulties such as having severe class imbalance, being highly left skewed, notably the neutral, admiration, and approval emotions are overrepresented.

For this paper, we will explore the following model approaches: Logistic Regression, XGBoost and LightGBM (with TFIDF and FastText Embeddings), and DistilBERT/MobileBERT. We aim to create robust models, and analyze the efficacy of results, as well as what improvements could be made.

Methods

EDA and Preprocessing

To begin this project, we a detailed Exploratory Data Analysis (EDA) step, in order to fully understand the dataset being worked with. To do so, we examined the amount of columns as well as data entries, (the number of columns and rows respectively) as well as what sort of data was within the columns. Then, prior to extracting common trends from within the data, we performed our preprocessing. As laid out in our github repository, we checked for the presence of missing data (null attributes) of which none were present. We also checked for and removed all duplicates, as well as all other aspects of the data that would confound our model building process. To do this, we utilized NeatText and Regular Expressions, to remove: all punctuation that was not necessary, all special characters (such as emoji characters), URLs, excessive usage of whitespace, and more. We chose to remove aspects of the data that would cloud the semantic meaning of the text and confuse our model, as well as the aspects of which were not within the scope of our problem. (as we aim to classify text samples, which does not include emojis, as they work similarly to tone indicators)

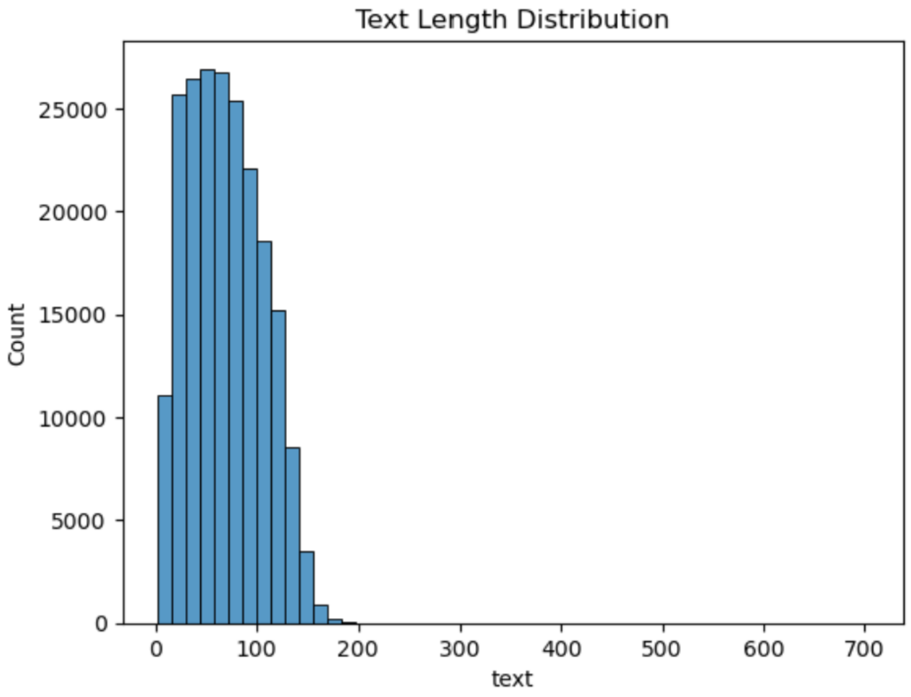


Figure 1: Text content length (characters)

Following this preprocessing and cleaning step, we continued with our analysis of the data. We analyzed the length of the data entries, the frequent subreddit sources, as well as the major emotions, the correlation between emotions as well as subreddits, and even the times that emotions were posted most frequently at. The length of the individual data entries, seen above, was on average very short, structured sections of text, as expected from comment data. The most common subreddit was ”cringe”, followed by appearances of subreddits such as Danganronpa, or the Minnesota Timberwolves, among many more. The nicheness of some of these subreddits (such as a top source of data being a specific NBA team, or specific subsects of a fanbase of a television show, highlights another form of skew from within the dataset on top of being sourced from reddit comments, and that is that the dataset tended to have many samples from the same subreddit (aka from one community) rather than from a large diverse selection of subreddits. A noticeable byproduct of this, on a smaller scale, is some that individual reddit users were slightly more prevalent in the dataset than others, such as user devildriver77, of whom had over 50 entries labeled to their name within the dataset, likely due to being a ”regular” within a community that had a higher sample.

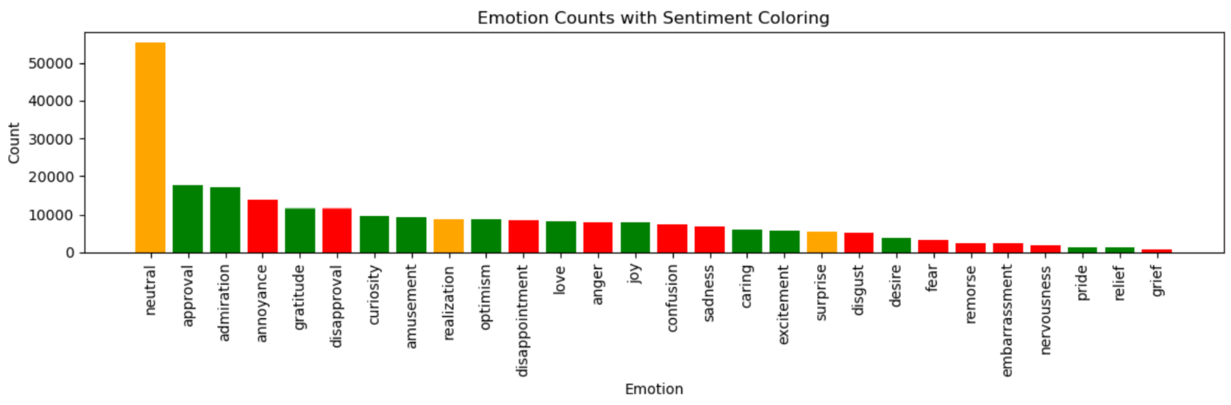


Figure 2: Spread of most common emotions within GoEmotions

The average time (UTC) that comments were posted at was roughly 4:30 PM, and noteworthy is that the least common timeframe to be recorded is that of 6 AM to 2 PM, which is roughly the middle of the night throughout the United States, indicating that a large portion of the data is sourced from US users of reddit, indicating further bias. For the emotions in question, we noted that the neutral classification was by far overrepresented, and was the highest presence. Following that was emotions such as approval or admiration. For the most part, the emotions did not co-occur, however the most prevalent combination was that of fear and nervousness, as well as anger and annoyance.

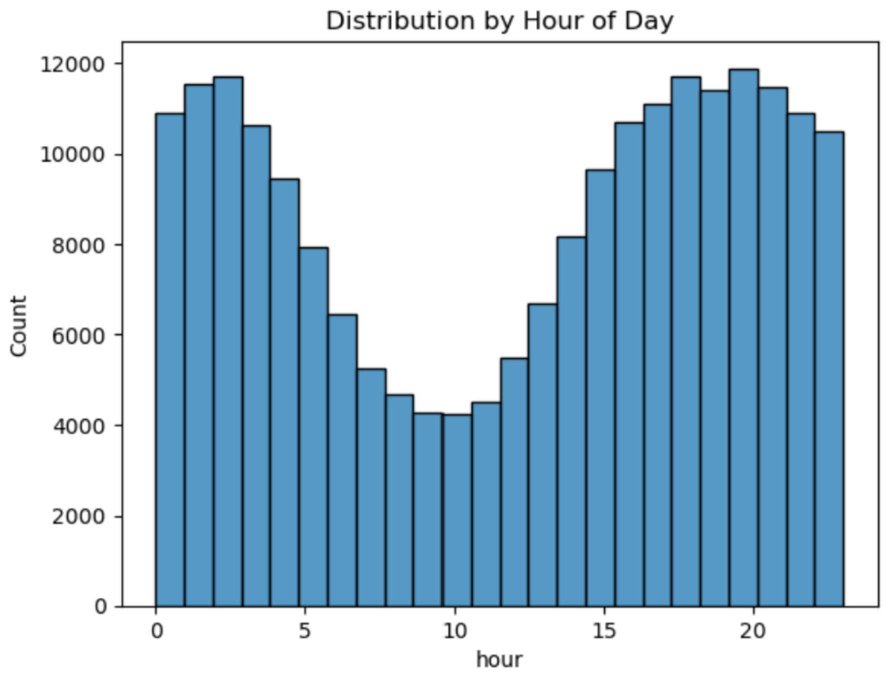


Figure 3: Time Data for Reddit posts (UTC)

Once we had all the data cleaned and processed, we created two subsets of the data, one being the original, unbalanced dataset (with the left skew in place) and the other being the balanced version, of which we did so by initially reclassifying each of the emotion classifications as a Positive, Negative, or Ambiguous emotion. After this, we assigned the dataset entries to each of these categories. Upon doing this, we were met with a left skew of positive emotions designations, making up over 100k of the total entries. As neutral was one of the only ambiguous classifications, neutral only totaled to a little over half that, with negative only slightly ahead of ambiguous.

2.2 Class Frequency after categorization

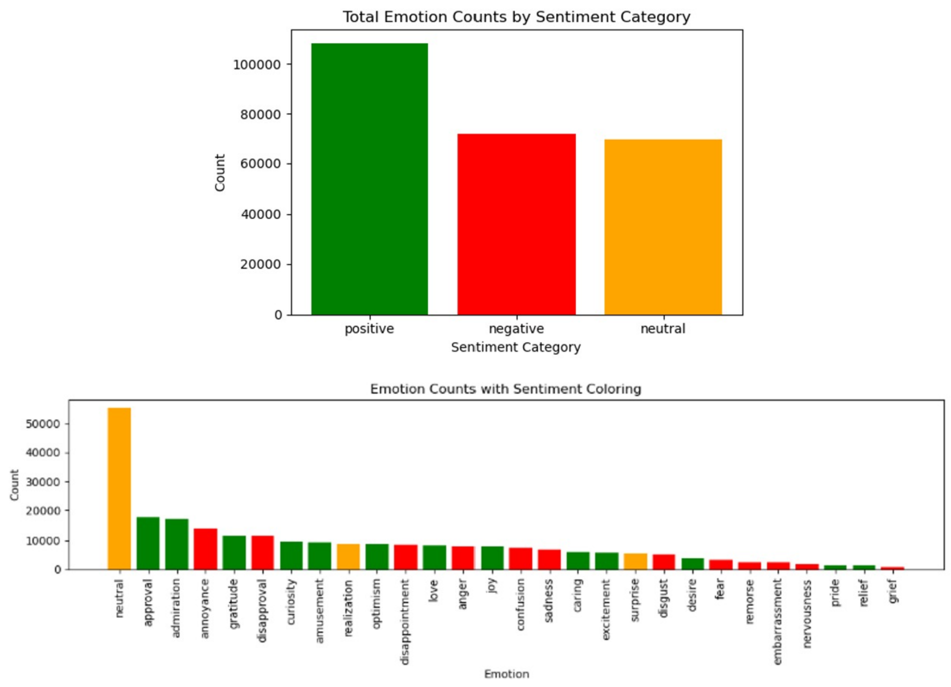
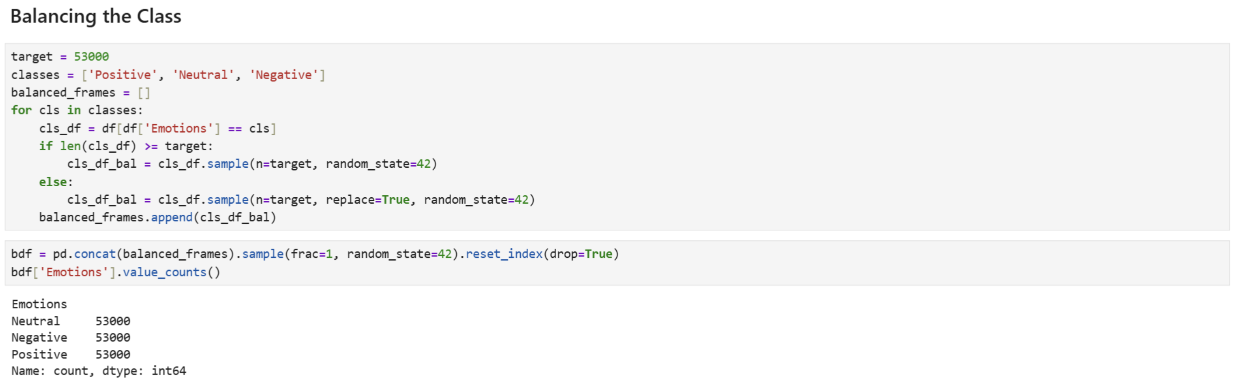


Figure 4: Three-way reclassification split within the dataset

Upon reaching this three-way split of the data, we then created the balanced dataset, by sampling a target value of 53,000 entries from each of the emotion designations. We chose not to sample on an emotion by emotion basis due to the severe disparity between the most prevalent classifications, and the least. After this step, we had our unbalanced and balanced datasets ready to be used in model training and comparison.

Models

After analyzing and understanding the dataset itself, we began to train our models. As discussed earlier, we trained several different models to compare and contrast our results. These models were either classical models such as a Logistic Regression model, XGBoost, and LightGBM, or pretrained transformer models being MobileBERT and DistilBERT. We chose these models due to their ability to handle interconnected relationships between the features. Our classical models utilized TF-IDF embeddings to assist in pattern identification, and FastText to capture subword information. The pretrained models were levied due to their existing understanding of language, allowing powerful comparison between speed and accuracy.


Balancing the Class

```
target = 53000
classes = ['Positive', 'Neutral', 'Negative']
balanced_frames = []
for cls in classes:
    cls_df = df[df['Emotions'] == cls]
    if len(cls_df) >= target:
        cls_df_bal = cls_df.sample(n=target, random_state=42)
    else:
        cls_df_bal = cls_df.sample(n=target, replace=True, random_state=42)
    balanced_frames.append(cls_df_bal)
```

```
bdf = pd.concat(balanced_frames).sample(frac=1, random_state=42).reset_index(drop=True)
bdf['Emotions'].value_counts()
```

```
Emotions
Neutral     53000
Negative    53000
Positive    53000
Name: count, dtype: int64
```

Figure 5: Creation of Balanced Dataset

Results

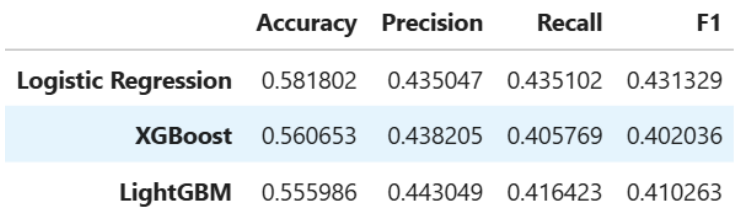
In evaluating the performance of our models, we compared the Accuracy, Precision, Recall, and F1-Scores of each model upon 20% of the dataset. Additionally, we tested and trained four different models, per approach, on two separate subsects of the data, being balanced and unbalanced datasets, split by TF-IDF and FastText embeddings, so as to fine-tune and mitigate the effect of the left-skew. The classical models, as expected, performed poorer than the transformer models, however they performed reasonably well given the skew within the dataset, and the existing work.

Classical Models

Logistic Regression had relatively stable performance between unbalanced and balanced datasets, performing the best on TF-IDF model, but the worst on FastText model. While the regression model did perform better on the TF-IDF models, notably the precision was lower, both balanced and unbalanced. The regression model achieved a peak of 45% F1 on TF-IDF Balanced dataset, and performed with a 53% F1 on the FastText Balanced dataset, the smallest growth between TF-IDF and FastText of the three classical models. The smaller growth can likely be attributed to logistic regression works well with TF-IDF, and does not benefit as much from the density that FastText provides. Thus is likely why FastText only marginally improves performance compared to the other classical models.

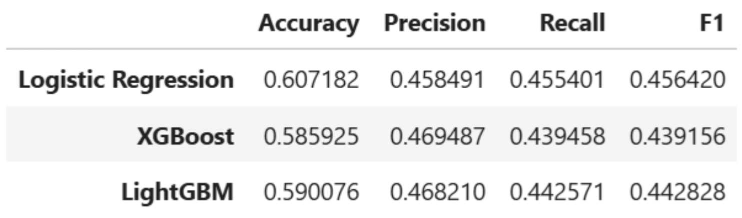
XGBoost showed a marked improvement between TF-IDF and FastText models. XGBoost had a much more complex performance, with strictly (by our metrics) the best performance out of the classical models with FastText, both on the unbalanced and balanced, achieving a 60% F1 score on the unbalanced and balanced datasets. This could be for various reasons, but most likely is that the density that FastText provides complements the tree structure that XGBoost follows, by providing dense data for tree splitting to follow, resulting in a much better performance than the tokenization and thus higher feature count, but much weaker individual vectors due to less interconnectivity between each word feature.

Finally, the third of the classical models, LightGBM was the middling approach for both TF-IDF and FastText, performing relatively closely to the better performing model for both approaches. Similar to XGBoost, LightGBM saw a marked improvement between TF-IDF and FastText, jumping from a 44% F1 score for Balanced TF-IDF, to a 55% F1 score for Balanced FastText. Much like the other models, its accuracy score overperformed on Accuracy compared to other metrics on the TF-IDF testing. This makes sense as LightGBM is a lighter, more efficient model than XGBoost, and thus performs slightly worse on FastText than XGBoost, and is also much less linear than Logistic Regression, and thus performs better with FastText, benefiting more from the density provided. The inverse is also true, as being nonlinear, it does not outperform Logistic Regression with TF-IDF. As such, LightGBM effectively sits as a comfortable middle ground between Logistic Regression and XGBoost in practice.



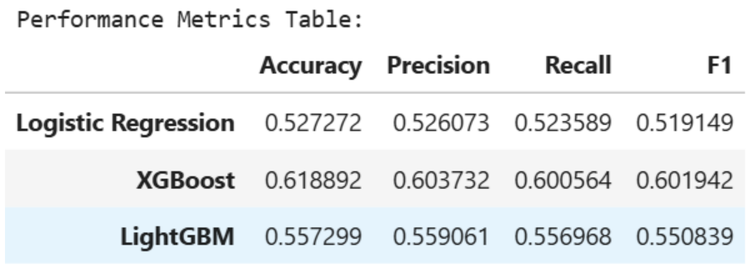
	Accuracy	Precision	Recall	F1
Logistic Regression	0.581802	0.435047	0.435102	0.431329
XGBoost	0.560653	0.438205	0.405769	0.402036
LightGBM	0.555986	0.443049	0.416423	0.410263

Figure 6: TF-IDF Unbalanced Performance



	Accuracy	Precision	Recall	F1
Logistic Regression	0.607182	0.458491	0.455401	0.456420
XGBoost	0.585925	0.469487	0.439458	0.439156
LightGBM	0.590076	0.468210	0.442571	0.442828

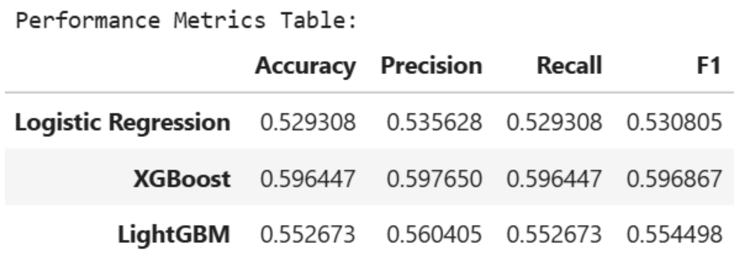
Figure 7: TF-IDF Balanced Performance



```
Performance Metrics Table:
```

	Accuracy	Precision	Recall	F1
Logistic Regression	0.527272	0.526073	0.523589	0.519149
XGBoost	0.618892	0.603732	0.600564	0.601942
LightGBM	0.557299	0.559061	0.556968	0.550839

Figure 8: FastText Unbalanced Performance



```
Performance Metrics Table:
```

	Accuracy	Precision	Recall	F1
Logistic Regression	0.529308	0.535628	0.529308	0.530805
XGBoost	0.596447	0.597650	0.596447	0.596867
LightGBM	0.552673	0.560405	0.552673	0.554498

Figure 9: FastText Balanced Performance

We finetuned our FastText and TF-IDF models to get to the best performance, detailed above, by changing the parameters on said models. For FastText, we arrived at a minimum word count of 2, and 10 epochs. For TF-IDF, we landed on a minimum document frequency of 5, and an ngram range of (1,2). We utilized the same parameters for each of the 3 models, however in the future (See: Conclusions and Future Work) we may independently test changes in these parameters, such as a lower minimum document frequency for TF-IDF Logistic Regression, however we do not expect to see these parameters change the broader results.

Transformer Models

The other subsection of models we worked with was that of the pretrained transformer models. For this section, we used two different versions of the BERT model, being DistilBERT[2], and MobileBERT[3], once again tested on both the unbalanced, and balanced datasets.

For a brief explanation on these models, DistilBERT is a ”Distilled version of the original BERT architecture, as a ”student” of the original model. It works as an uncased model trained on the same data of which was used to train BERT. The reason for BERT’s existence was to have a more computationally ”inexpensive” (relatively) version of BERT, wherein DistilBERT makes the tradeoff of a few percentile values of accuracy, in favor of a much higher efficiency and processing speed. This was achieved by stragically removing roughly 40% of the parameters that BERT possessed (still having roughly 67 million parameters). This makes it run around 60% faster, in exchange for that bump in accuracy. Due to our processing restraints, being limited mostly to personal level machines for the duration of this project, we chose to make that small sacrifice on accuracy to be able to reasonably gather results, as otherwise the resource intensity of the teacher BERT would nigh prevent us from achieving our results realistically.

Then, we also used MobileBERT as a comparison model within the BERT and Transformer class of models, as we could not learn from simply including one model. As such, we included MobileBERT in our evaluation, as it is the ”Mobile Friendly” version of BERT. We chose it as it is quite similar to DistilBERT, being uncased (which suits our dataset, due to the nature of online communication) as well, and being similarly less resource intensive than the original BERT algorithm. MobileBERT, being designed for less resourceful machines, was a good pick for the same reasons as DistilBERT, as it also makes a sacrifice on accuracy in favor of efficiency. Most importantly for the comparison, MobileBERT allows us to make two major, related comparisons, being the impact of higher efficiency and speed on the performance of the model, as well as learning how a severe drop in the amount of parameters (down to 25 million) impacts the performance. This allows us to mock the impact of BERT to DistilBERT to a certain extent (although not entirely due to the difference between the models).

Prior to collecting our final results, much like with the classical models, we finetuned to discover what parameters for the Distil and Mobile BERT models would work best upon training for our dataset, and as such we tested different epoch values, eventually landing on 2. We updated the models end-to-end, allowing the internal attention mechanisms and contextual embeddings to adapt to the sentiment cues within the data.

These two models, as expected, performed better than our classical models, which makes sense, as these models surpass our capabilities with non-pretrained models, which is reflected by their better performance. As the small BERT models are pretrained on a drastically larger amount of data, they possess a much larger degree of knowledge on word relationships than the relatively limited knowledge than the models strictly trained on the dataset, and as such they perform better. Of note, is that the BERT models do not have nearly as sizable of a gap between their unbalanced and balanced performances, once again due to the training that the BERT architecture receive, causing them to be far more robust, and thus able to handle class imbalances far better, likely the most notable performance difference between the models.

The DistilBERT and MobileBERT models performed relatively similarly to one another, with MobileBERT actually performing the slightest bit better. When referencing the performance perks of these models, however, the performance between them is far closer than that of the classical models. DistilBERT resulted in a 63.1% unbalanced performance,

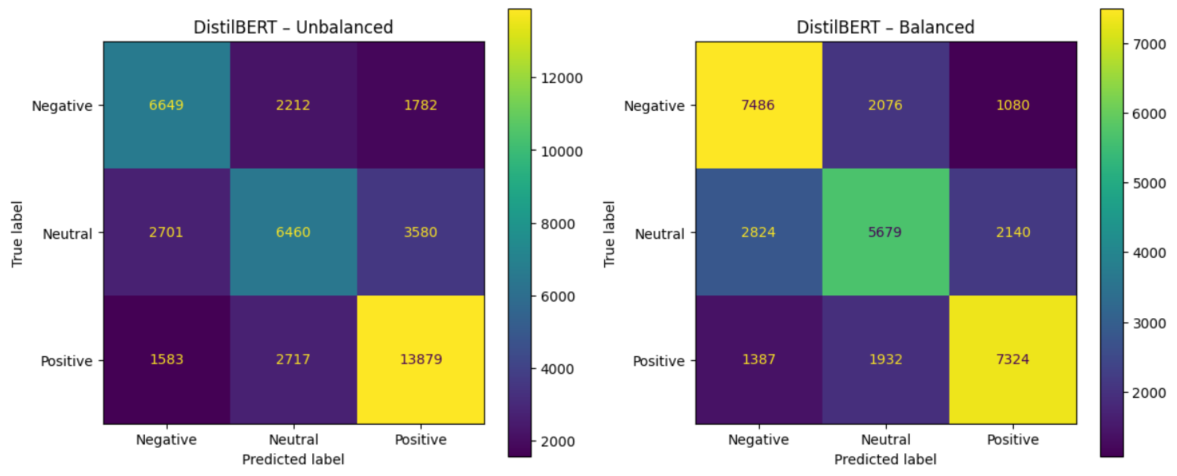


Figure 10: DistilBERT Confusion Matrix

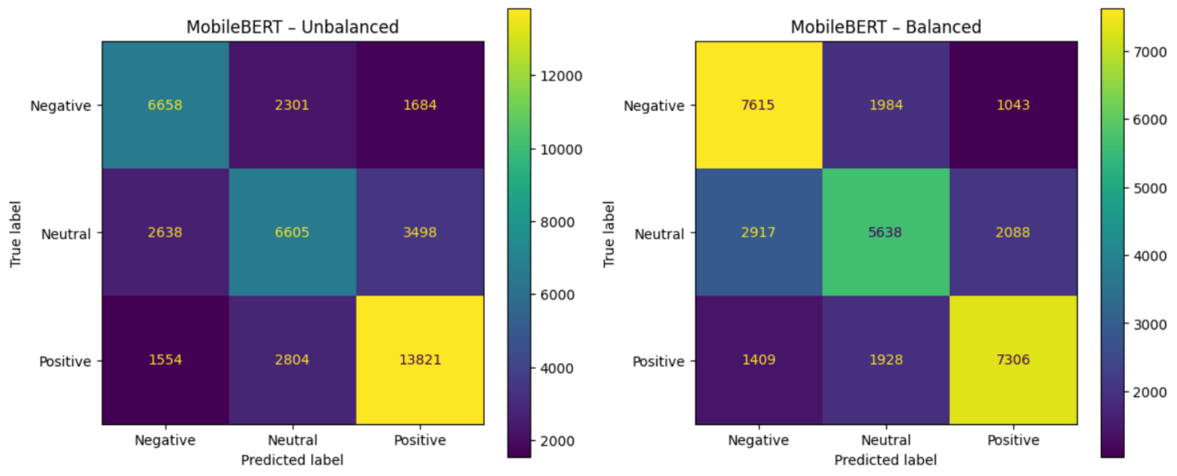
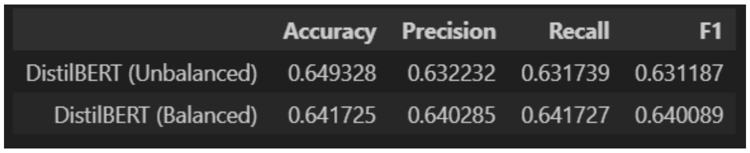


Figure 11: MobileBERT Confusion Matrix

and 64% balanced performance, whilst MobileBERT performed with a 63.4% unbalanced performance, and 64.2% balanced performance. This minor difference may come from many reasons, with the hypothesis being due to the inherent structure of Reddit commenting, being shorter, lower than 100 characters in length. This general lower size of entries may account for the slight boost that MobileBERT displayed, meaning that due to the strengths of which MobileBERT possessed over DistilBERT, the major reduction in cost did not cause a major decrease in performance, at least not one that was not offset by the other characteristics of MobileBERT. Looking beyond the F1 and accuracy scores, and at the confusion matrices included above, the story is slightly different, wherein MobileBERT is more skewed towards majority classes within the dataset. This explains the slight decrease in Accuracy between the unbalanced, and balanced splits for MobileBERT. The same patterns, albeit less severe, may be recorded with the DistilBERT model. Thus, DistilBERT and MobileBERT have their benefits and drawbacks respectively, the former being more reliable, and the latter experiencing stronger gains from balancing.

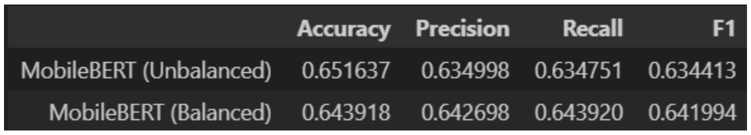
Transformation vs Classical Models

The BERT models saw a considerable spike in both robustness, and performance compared to our classical models, for a myriad of different reasons. Chief among them, the fact that the BERT algorithms are pretrained by a massive



	Accuracy	Precision	Recall	F1
DistilBERT (Unbalanced)	0.649328	0.632232	0.631739	0.631187
DistilBERT (Balanced)	0.641725	0.640285	0.641727	0.640089

Figure 12: DistilBERT Performance Metrics



	Accuracy	Precision	Recall	F1
MobileBERT (Unbalanced)	0.651637	0.634998	0.634751	0.634413
MobileBERT (Balanced)	0.643918	0.642698	0.643920	0.641994

Figure 13: MobileBERT Performance Metrics

amount of data, and thus are much more powerful on analyzing connections between words compared to the much smaller training data available to our classical models. This pretraining is the major cause of the model differences displayed, however it is notable that the XGBoost did see a closer F1 score to the BERT models, showing promise despite the limited training size. As it stands, the BERT architecture displays the best results, however, due to their ability to see larger trends within the data than the smaller dataset-trained models, with specifically the balanced BERT models, depending on the balance of needs of the problem at hand (performance necessities, etc), being the most effective.

Conclusions and Future Work

The comparison between different models, from the BERT architecture, to classical logistic regression, XGBoost, and LightGBM, led us to many great insights on what would work best for the task of classifying emotions from short samples of textual data online. Our results, improving on existing work by providing a useful comparison and classification on a much larger total number of emotion classifications, proved to be fruitful, gleaning a lot of information about the interaction of these models with the problem of emotion classification.

In the future, steps that we have considered consist of other ways of rebalancing the dataset at hand, such as re-classifying what consists of ambiguous, positive, or negative emotions. Additionally, with a sufficiently large dataset, division and balancing based on individual emotions may be possible. However, a larger dataset in general would also be useful, if such means became available. Finally, we may explore changes between the models in the parameters, rather than testing on streamlined parameter values. Regardless, the results we have obtained proved a meaningful solution, and exploration, for the problem at hand.

Github Repository

References

[1] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. Goemotions: A dataset of fine-grained emotions, 2020.

[2] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[3] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. Mobilebert: a compact task-agnostic bert for resource-limited devices, 2020.