COMP551 Assignment 4

T. Forgione, L. Day, N. Torres-Romero

December 4th, 2024

Abstract

In this assignment, we predicted emotions from text using 3 different models. We used SR, XGBoost, and RF as our baseline and compared them to both Naive Bayes and a pre-trained LLM, distilled GPT-2. All 3 baseline models performed similarly, reporting around 57% accuracy and 60% precision. Our best Naive Bayes model had a relatively poor sub-50% accuracy. Distilled GPT-2 performed best out of all the models with accuracy above 60%. We also finetuned an optimal BERT model, which gave 63% accuracy, the highest of all the models.

Introduction

We were tasked with classifying emotions from the GoEmotions dataset, a dataset of over 50000 word sequences with labeled emotions. This dataset consists of text written by different users and is classified by raters with one or more of 28 emotions. As our baseline model, we used Softmax Regression (SR), a supervised learning model which is generally used in regression tasks. As other baselines, we used Random Forests and Extreme Gradient Boost. Next, we implemented Naive Bayes from scratch. Lastly, we used a pre-trained LLM, distilled GPT-2, from HuggingFace as well as a pre-trained BERT model. We compared all these models in order to see which attention models prove best on this data. Our baseline models performed similar to one another, with 57% accuracy. Our Naive Bayes model reported 47% accuracy at best, and both GPT-2 and BERT had an accuracy above 60%, the best out of all our models. We notice that the LLMs are better at emotion classification tasks (sequence to vector), which is expected given their ability to focus on context.

This was compared to a paper using BERT on the same GoEmotions dataset.[1] Our macro results were quite good by comparison, however it is to note that we used a different test set to the authors. Our optimal BERT model used hyperparameters from a paper about bidirectional transformers[2]

Methods

Regarding the dataset, it was downloaded from google research's website. We removed data that had more than one label (more than one emotion attributed), leaving us with around 85% of the data. We then looked at the distribution of each emotion in the data (figure 1). We notice that class 27, neutral, is by far the most common in the dataset. This makes sense since not every sentence contains some emotion. The dataset was processed differently for each model; for SR, we transformed the data into numerical features, for Naive Bayes, we used a bag-of-words representation to turn the data into numerical features, and for GPT-2, we tokenized the text and turned it into numerical features. For baseline models, we limited the vocabulary to 5000 words to prevent overfitting. To be included in the vocabulary, each token had to appear in at least two Reddit comments, which reduces noise from very rare tokens. It could also appear in at most in 95% of comments, which helps ignore overly common words with no predictive value. Our initial Softmax Regression model was trained on scikit-learn's LogisticRegression class with the 'multinomial' option, 'lbfgs' solver and default parameters, which include a regularization strength of 1.

We implemented a Naive Bayes classifier for multiclass classification which computes the posterior probability of each class for a given text input, selecting the class with the highest probability. This was done by using the training data to calculate the prior probabilities of each class and conditional probabilities of each feature given the class, assuming independence between features. We also tested several different Laplace smoothing values to determine its effects on the training accuracy and test accuracy.

We decided to implement GPT-2 in order to try to classify this data. First, we test the base pre-trained model on the test data.

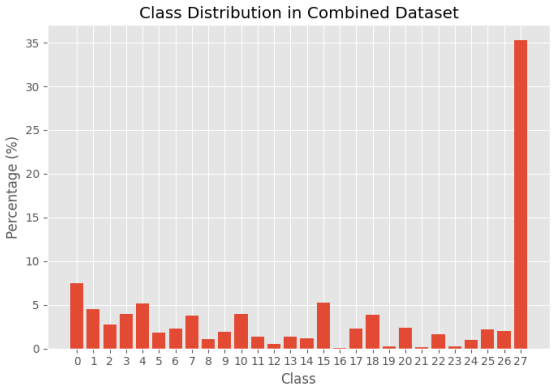


Figure 1: Class Distribution

Next, we finetuned the model, optimizing the hyperparameters along the way. HuggingFace's transformers package has a class which adds a classification head on top of the base model. We trained all of the model's layers on the training set, and optimized certain hyperparameters using the validation set to maximize performance. We optimized the learning rate, batch size, and weight decay, using an epoch limit of 5.

Our distilled GPT-2 used the default settings set by the Transformers package. The default optimizer was AdamW, with a weight decay of 0.01. By default, the GPT-2 model uses dropout, which was set to 0.1.

Results

We began with the implementation of our baseline model, running a grid search on the regularization strengths 0.001, 0.01, 0.1, 0.5, 1, 5, 10, 100, 1000 which coincidentally showed that the default strength of 1 was optimal.

We then moved on to the other two baseline models, starting with the Random Forest. We initially trained it on 200 trees, leaving all other parameters at their default values. The model had a training set accuracy of 0.9934, a validation set accuracy of 0.5638, and a test accuracy of 0.5503. That is, the training and validation accuracies remained similar to the SR, only the training set's accuracy increased significantly. To achieve similar predictive accuracy as the SR on unseen data, the RF fits the training data much more closely, highlighting the RF's comparatively higher expressiveness. We began by optimizing the model's number of decision trees, maximum depth per decision tree, and the minimum number of samples required to split an internal node. We then optimized the minimum number of samples required to be at a leaf node, and the number of features to consider when looking for the best split at each node. Our grid search yielded the following results: max_depth=None, min_samples_split=10, n_estimators=200, max_features=sqrt, min_samples_leaf=1. This improved the model's generalization by a small amount (see table 1).

We then explored the XGBoost model, which gave a training accuracy, validation accuracy, and test accuracy of 0.7259, 0.5778, and 0.5725 respectively. To optimize hyperparameters, we did a similar process as before, using a randomized search in the interest of time. Our search gave depth of trees (max_depth)=9, learning rate=0.069, boosting rounds (n_estimators)=188, fraction of data (subsample)=0.84, features fraction (colsample_bytree)=0.88. We noticed across all three of our baseline models an insensitivity to changes in hyperparameters, as their performances vary little before and after optimizations via grid search. We summarize the performance of baseline models in table 1.

Table 1: Performance Metrics of Different Models

Model:	**Softmax Regression**		**Random Forest**		**XGBoost**	
Set	**Accuracy**	**Weighted Precision**	**Accuracy**	**Weighted Precision**	**Accuracy**	**Weighted Precision**
Training set	0.64	0.68	0.98	0.98	0.73	0.80
Validation set	0.56	0.57	0.57	0.61	0.58	0.59
Test set	0.56	0.56	0.55	0.60	0.57	0.58

Next, we trained our multi-class Naïve Bayes model on the training data using various values for the Laplace constant 2. We found that a Laplace smoothing value of 0.27 gave the best test accuracy of **46.19%** with a training accuracy of **74.22%**. After further investigation we noticed that the model had a higher accuracy predicting classes with a higher representation in the training set. This correlation was present for both the training accuracies 3 and test accuracies 4. This observation is not unexpected, a higher representation of a class provides more robust estimates, whereas the naive Bayes model will struggle to predict underrepresented classes.

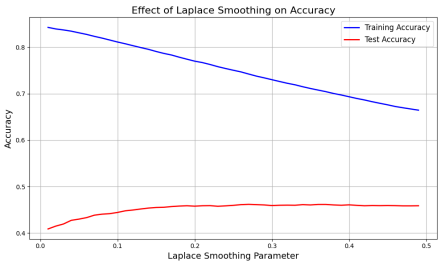


Figure 2: Effects of Laplace smoothing on multi class naive Bayes model accuracy

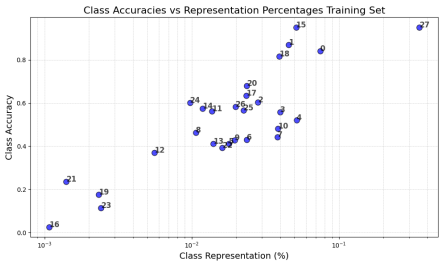


Figure 3: Model accuracy per class versus class representation in training set

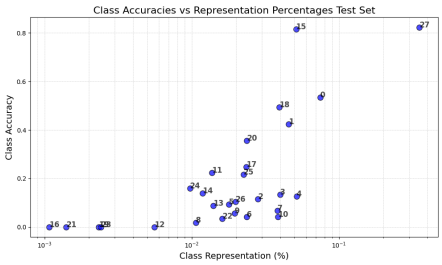


Figure 4: Model accuracy per class versus class representation in test set

Our first test for GPT-2 was to simply evaluate the pre-trained model on the data, which resulted in an accuracy of 4.25%, only slightly above chance level. This seems quite low, however this model was not trained using emotion classification data. Thus, we cannot expect the model to be able to make accurate predictions.

With regards to GPT-2, we used the HuggingFace Transformers package to implement a trainer, which takes care of training and evaluation given a dictionary of parameters. We first optimized the learning rate using a batch size of 16 and leaving everything else at their default. The results of this can be seen in figure 5, where we notice that a learning rate of 0.00002 provides the best accuracy. The larger learning rates overfit and the smaller one did not perform well.

Next, we decided to optimize the batch size. Since we already had a run with the optimal learning rate with 16 batches, we compared that with batch size 32 and 64. These seemed to be a good midground between time-to-run and memory requirements. The results, seen in figure 6, show that batch size 16 was optimal.

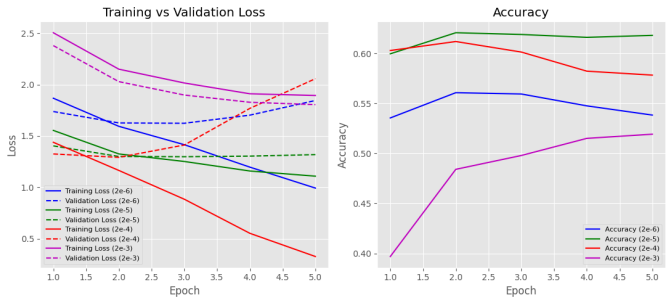


Figure 5: Loss and Accuracy for Different LearnRate over 5 epochs

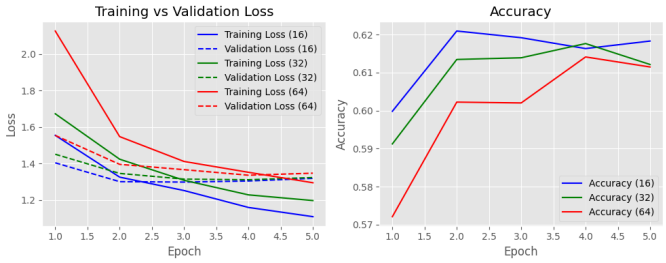


Figure 6: Loss and Accuracy for Different Batch Sizes over 5 epochs

Our final optimization was weight decay. Once again, we had a run with optimal learning rate and batch size (with weight decay 0.01) already, so we compared that with decay of 0.3, 0.6, and 0.95. We notice in figure 7, that a weight decay of 0.3 was optimal.

Figure 7: Loss and Accuracy for Different Weight Decay over 5 epochs

With the optimal model set, we notice that it overfits after only 4 epochs, so we ran it for 4 epochs and computed the accuracy, precision, recall, and F1-score. The latter 3 will be weighted, as there is a large imbalance in the data. We see the results for each epoch in table 2.

We then experimented with BERT. Our initial set of hyperparameters came from both the Devlin et al. (2019) and Demszky et al. (2020) papers, which include a learning rate of 0.00002, batch size of 16, weight decay of 0.95, and a dropout of 0.1. From there, we finetuned all model layers using similar tuning methods as for GPT-2 to get our finetuned BERT. We ran each model for two epochs and the results, in table 3, show a relatively strong model which is similar to our finetuned GPT-2 model.

Epoch	Accuracy	Precision	Recall	F1
1	0.577996	0.582792	0.577996	0.558619
2	0.605882	0.598753	0.605882	0.586667
3	0.613508	0.600865	0.613508	0.596189
4	0.614161	0.599831	0.614161	0.598944

Table 2: Train and Test Metrics Across Epochs for Optimal GPT2 (Weighted)

As a last look, we retrieved the macro (non-weighted) averages for precision, recall, and f1 for both GPT-2 and Bert and compared our results to the Demszky paper. We can see this comparison in table 4, although it is to note that the paper used a different test set than we did, so we will not emphasize this result.

Epoch	Accuracy	Precision	Recall	F1
1	0.626797	0.601949	0.626797	0.600372
2	0.629630	0.617634	0.629630	0.614473

Table 3: Bert Performance (Weighted)

	Precision	Recall	F1
Finetuned Distilled GPT-2	0.53	0.48	0.49
Finetuned Bert	0.49	0.46	0.47
Demszky Bert	0.40	0.63	0.46

Table 4: Precision, Recall, and F1 Scores (Macro)

The final thing we did was looking at the attention matrices for some properly and improperly classified documents. The first, seen in figure 8, shows the attention matrix for all the heads in the first layer (just after the input) for the correctly classified document "It's wonderful because it's awful. A not with." We see that in most of the heads, the class token [CLS] is mostly focused on "It's", "wonderful", and "awful". This makes sense, as these are the words that show the most emotion. However, some heads attend to the word "because" quite a lot. The reason this occurs is because the words "wonderful" and "awful" are tied together with "because". This may actually aid the model in classifying the sentence correctly as admiration.

Next, we look at a misclassified document such as "Kings fan here, good luck to you guys! Will be an interesting game to watch!" This document's attention matrix for all heads at the first layer can be seen in figure 9. Here, we see that the class token is focusing a lot on the words "Kings" and "fan", which offer little in terms of emotion. This document should be classified as excitement, which makes sense, however our model predicted this as being optimism. While the model did incorrectly classify this document, it was not terribly off in its assessment, as optimism and excitement are somewhat similar.

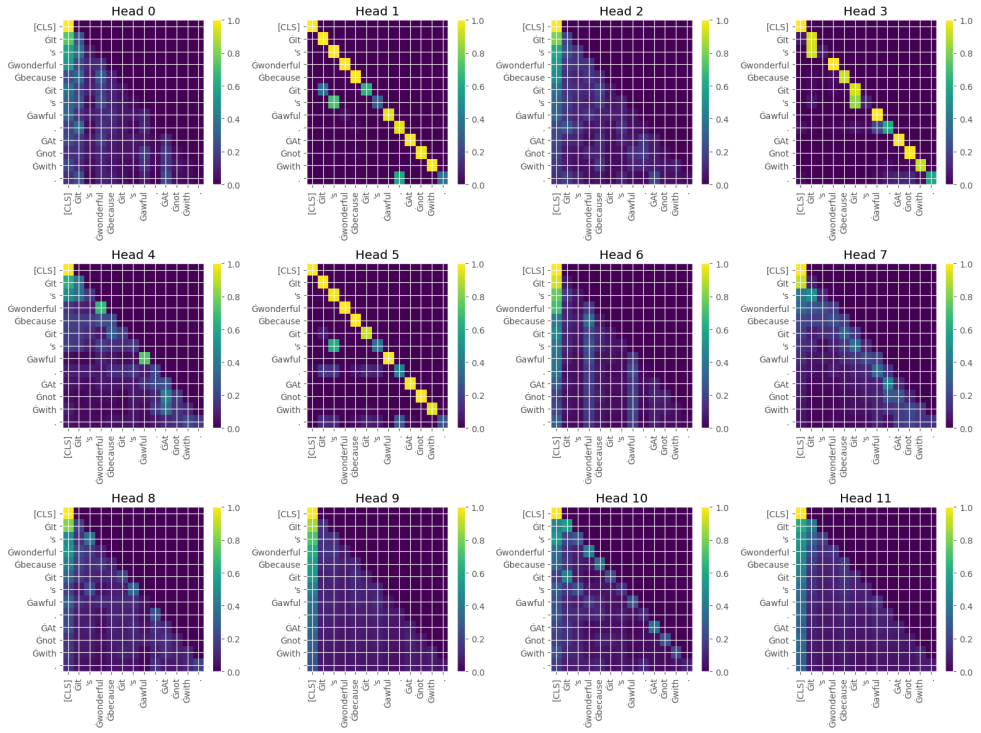


Figure 8: Properly classified document

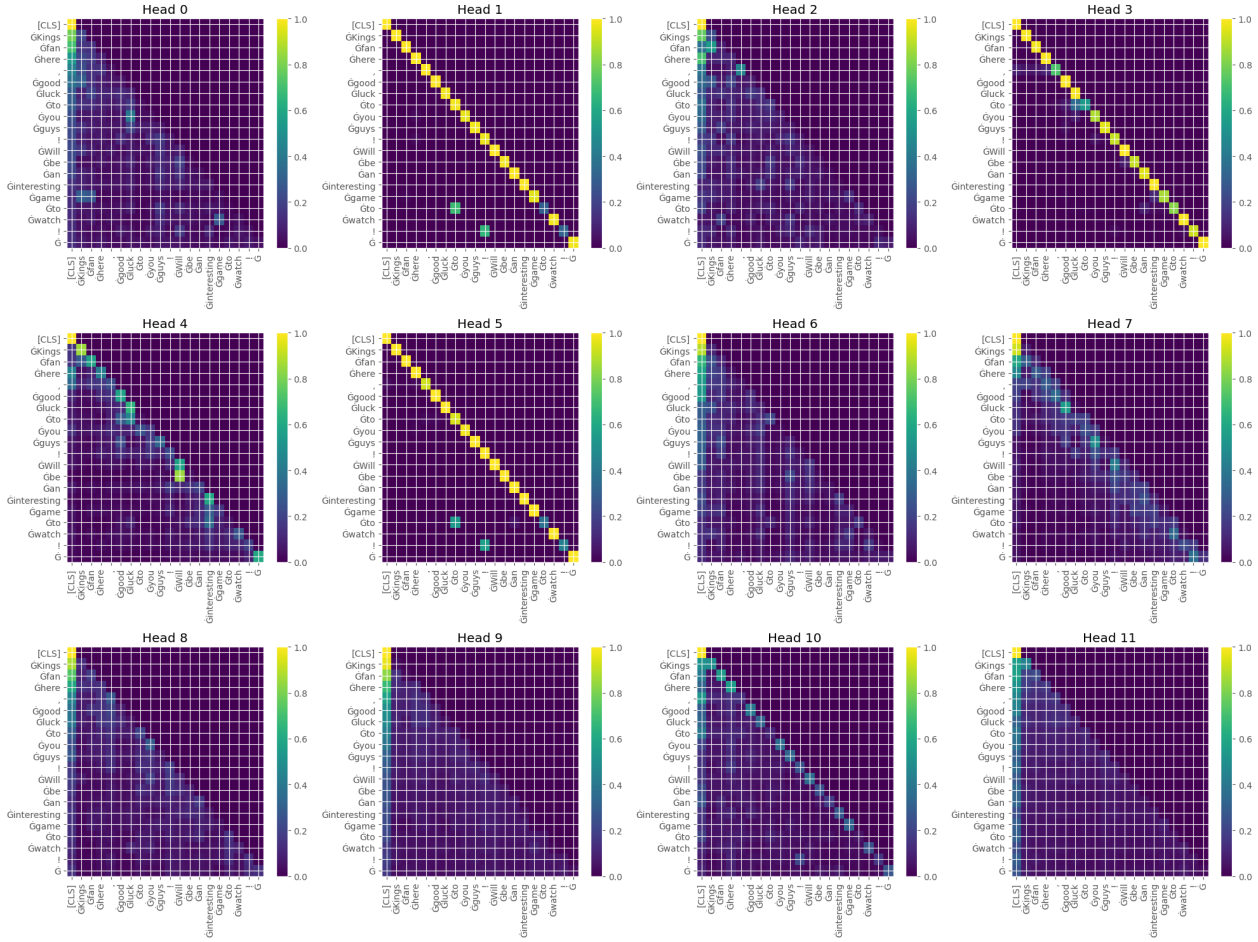
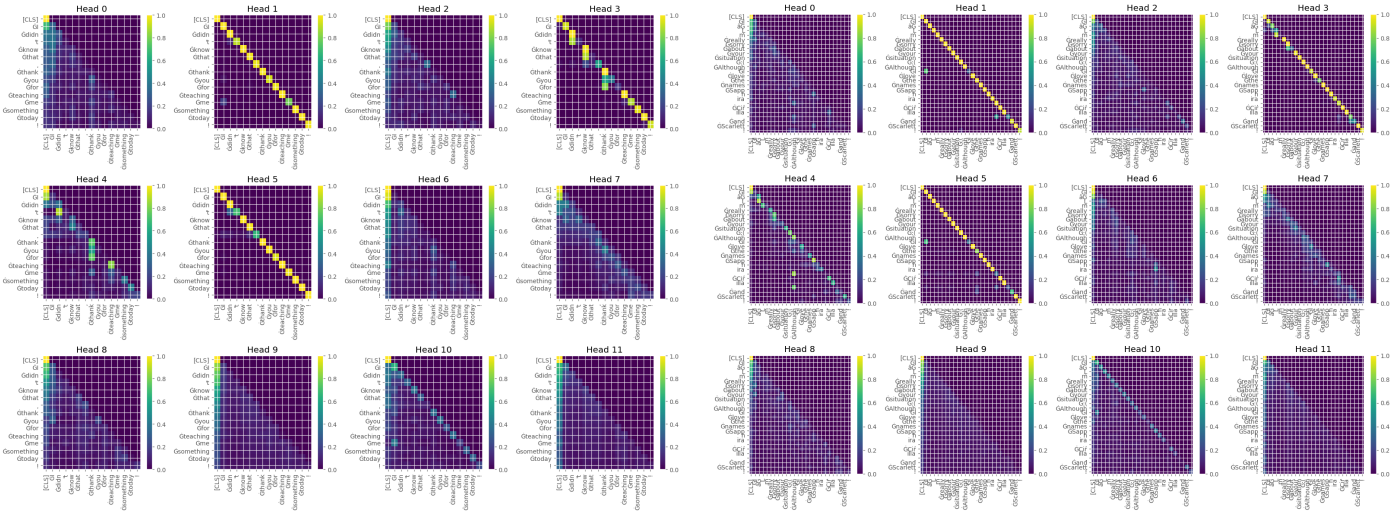


Figure 9: Improperly classified document

Lastly, we show another example of properly classified and misclassified documents (figure 10a, 10b respectively). The properly classified document is "I didn't know that, thank you for teaching me something today!" which was correctly classified as gratitude. The misclassified document "I'm really sorry about your situation :(Although I love the names Sapphira, Cirilla, and Scarlett!" was classified as love instead of sadness.



(a) Properly classified document

(b) Improperly classified document

Figure 10: Another comparison of properly and improperly classified documents

Discussion/Conclusion

We've seen that all three baseline models deliver close accuracy on unseen data: all between 55% and 58%. Despite their apparent similarity on unseen data, the models fit the training data differently. The Softmax Regression model fits the training data less closely due to it assuming a linear relationship between the input features and the class probabilities. Thus, it fails to capture non-linear patterns in the data, leading to a

lower training accuracy of 64%. The Random Forest on the other hand is much more expressive with its 98% training accuracy. In the random forest, each decision tree grows until it overfits its portion of the data. The forest averages the predictions of the trees, which slightly reduces variance without fully eliminating overfitting. Its generalization capacity is thus low which may be why it does not outperform SR. XGBoost on the other hand does generalize slightly better. With boosting, it builds the trees sequentially and optimizes for bias and variance based on the previous tree. It also adds regularization parameters like min_samples_leaf and max_depth which we optimized using a randomizes grid search. The result is that it generalizes slightly better with a lower training accuracy of 73% and a slightly higher accuracy on unseen data of 57.5% on average compared to 56% for the two other baseline models.

Of all our models, Naïve Bayes had the lowest test accuracy of 46.19%. This may be explained by two things. First, with the Bag of Words representation, the model fails to capture any semantic and contextual information coming from ordering of words within comments, which is ignored by the BoW representation. However, baseline models also rely on the BoW representation by using the TfidfVectorizer so this explains why Naïve Bayes is less accurate than fine-tuned LLM's, but not why it performs worse than baseline models. The reason why Naïve Bayes performs worse than the baseline models is that the main assumption behind the Naïve Bayes model—conditional independence between the features given class label—is unsuitable for our problem: given a label, certain features influence the probability of others. For example, given a comment whose label is "gratitude", the presence of the word "thank" makes the presence of the word "you" more probable. This sets Naïve Bayes at a particular disadvantage.

Lastly, our results show that Distilled GPT-2 and BERT give the best weighted accuracy on this data (61.5% and 63% respectively) by a considerable margin. These models, using attention, gain context that Naive Bayes and baseline models crucially miss. BERT also gets aided by the fact that it is a bidirectional model, which gives it access to words both before and after what it is attending to at the current moment. GPT-2, on the other hand, is unidirectional, giving it access only to words prior to the given word. Although its accuracy was not much worse, it is possible that with a different dataset, BERT would significantly outperform it.

Because this is an extremely unbalanced dataset, F1 score may be a better representation of performance than accuracy. Looking at the F1 scores for both BERT and GPT-2, we see that BERT continues to outperform GPT-2.

In terms of the macro averages, GPT-2 outperforms BERT in all statistics, but, as mentioned previously, this dataset is extremely unbalanced, so the macro averages may be skewed.

A fundamental problem with this dataset is the use of raters in classifying the emotions. Above, we mentioned a document that was misclassified as sadness even thought its true label was love. However, reading the document again, we notice that there are two parts: the first part "I'm really sorry about your situation :(" evokes more sadness, while the second part "Although I love the names Sapphira, Cirilla, and Scarlett!" is not at all sad. This document could have had two labels and the label it was given may not be ideal. This could be a large reason as to why the results are so poor for this dataset.

In the future, it would be an interesting experiment to compare GPT-2 and BERT by only finetuning their classification layers. This would likely give much worse accuracy than finetuning all layers, but showing the amount of accuracy lost could be important in understanding the role of all the layers in these LLMs. Another important exploration would be to remove a number of the neutral documents from the data. Because such a large number of documents were classified as neutral, all of our models were especially biased towards that label (see our Naïve Bayes implementation).

Statement of Contributions

Nicolas did the baseline models, Luca implemented Naive Bayes, and Tyler did the LLM. We all did data preprocessing separately and came to the same results.

References

[1] Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., & Ravi, S. (2020). GoEmotions: A Dataset of Fine-Grained Emotions. arXiv:2005.00547. Retrieved from `https://arxiv.org/pdf/2005.00547`.

[2] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. Retrieved from https://arxiv.org/pdf/1810.04805.