CS 414 Final Project: Executive Report

Casey Ford, Katie Lansing, and Sakhi Naik

Sentiment Analysis on r/AmITheAsshole

✅ Defining Our Problem

The goal of this project is to use Machine Learning to predict how an online community (r/AmITheAsshole) reacts to a story (a post to the subreddit) in text form. While the most direct application of the model is predicting how social media users perceive posts, the model could be applied to predicting how people will react to announcements, emails, and other types of anecdotal internet text communication, which are used by philanthropists and politicians to get funding.

📊 Collecting and Cleaning Data

To be aligned with our problem statement, we had to scrape our data from the subreddit “r/AmITheAsshole” which turned out to not be a simple feat. The data we inputted to our model had to include both the whole text post to determine if the person posting was an “asshole” or not, as well as the ground-truth label we would use as a metric to determine if the model was accurate or not.

This ground-truth label was based upon the top comment underneath the original text post. Our data was therefore supervised and the labels we used were “YTA” = “You’re the Asshole” and “NTA” = “Not the Asshole” so our final output ideally determines one of these two binary labels for each post.

The amount of data needed for training a Bidirectional Neural Network model for sentiment analysis is ideally somewhere from 5,000-10,000 labeled samples, with a functioning model needing a few thousand data points minimum. Our dataset consisted of approximately 3,300 posts due to the limitations of the Reddit API and computational resources we had.

We used a python script to scrape our data from Reddit’s website. Initially, we tried to scrape posts from the “all time” category on Reddit which includes all posts on the site, however, the Reddit API only allows 1000 posts to be scraped at a time. Thus, we had to expand our scraping algorithm to include the most upvoted posts from each of the following categories: “all time”, “year”, “month”, “week”, and “day”. Using this strategy we scraped around 700 posts at a time, resulting in a final batch of around 3,300 data points. The pandas library was used to concatenate all of the data and remove duplicate data points. Duplication removal is important to prevent the model from overfitting the data and resulting in poor generalization performance on unseen data with lower accuracy. Overall, because of timing limitations, our data set was smaller than we wanted it to be, but was sufficient for fine-tuning.

We then cleaned this data as a part of our model's pre-processing to remove formatting issues that could cause input issues for our model. See the "Data Preprocessing" section in the project Google Colab for more detailed information.

We then had to decide the data set distribution to use for training, validation, and testing. With a small dataset like ours, it is recommended to use a larger portion of the data for training so that the model can have enough data to learn from while still having enough data to evaluate the model performance. The final distribution we chose was: 80% training, 10% validation, 10% testing. The data was split randomly to avoid introducing bias but the dataset was imbalanced, with about 88% of ground-truth labels as 'NTA'. See the "Limitations" section of the report for more information about imbalance in our dataset.

🎨 Model Design

We chose to use a pre-trained model call BERT (Bidirectional Encoder Representations from Transformers) that is designed to perform natural language tasks like sentiment analysis using a masked language modeling objective, which is the analysis type our project requires to tell if a person would be the "Asshole" of the post or not. We decided to use a pre-trained model because Natural Language Processing is a very complex task, requiring complex ML models. So we greatly reduced the overhead of the project by using a model that has already been trained to process language, fine-tuning to our specific task. See the "Training" section of the Google Colab for more information about how BERT works.

There are multiple BERT models and the specific model we used was the BERT base uncased model, which works on all uncased English text input and has 110 million parameters. This masked language model works by randomly selecting 15% of the words in the input to learn a bidirectional representation of the sentence. The model uses next sentence prediction which concatenates two masked sentences as inputs to predict if two sentences were following each other or not to understand the chronology of the full text. A constraint of the model is that the result of the two sentences has a combined length of less than 521 tokens, which could lose some key input data. The model is primarily aimed at being fine-tuned for tasks that use a whole sentence to make decisions.

Our final output produces the accuracy of how well our model matches the ground-truth label of "Not the Asshole" or "You're the Asshole" for each post.

💪 Training Our Model

Our training was not done from scratch because the BERT model architecture was already created and pretrained for us. The Bert model was pre-trained for one million steps with a batch size of 258 and a sequence length of 128 tokens for 90% of the steps and a length of 512 tokens for the remaining 10% of steps. The texts are tokenized using WordPiece and a vocabulary size of 30,000.

The hyperparameters we chose are

- Batch size: 16
- Epoch number: 3

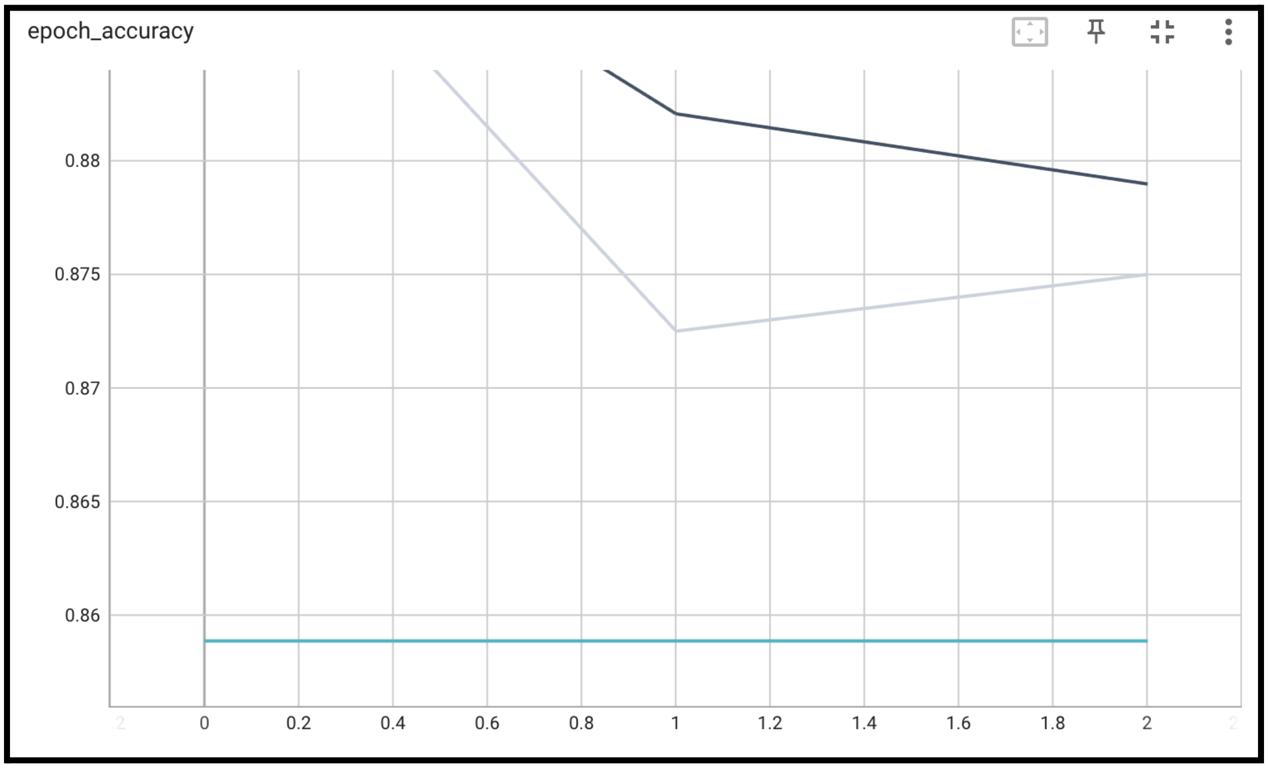
- Learning rate: 2e-5
- Epsilon: 1e-8

All these fell within the range recommended for fine-tuning a BERT model. Ideally, we would have liked to use a batch size of 32 for 4 epochs but we chose smaller values to reduce the computational load of the model.

The model uses an Adam optimizer specifically created for BERT models called the BertAdam optimizer.

🧪 Testing and Evaluation

The model was tested using 10% of the dataset and had 87% accuracy with 3.8% loss.



Evaluation of the model using TensorFlow's TensorBoards showed that the model performed with ~88% of accuracy on the training dataset and with ~86% accuracy on the validation dataset. So overall, the model is performing pretty well, although it could be improved. There is a gap between the training and validation accuracy so the model could be overfitting. With more time and computational resources, we would address this by training the model on more data, reducing its complexity, and using regularization techniques like dropout.

Given more time, we would have introduced more hyperparameter tuning and architecture changing of our model to try and achieve a better performance. See the

“Hyperparameter Tuning” section in the project’s Google Colab for more information about future tuning work.

⛔ Project Limitations

Scraping: Having to scrape Reddit was not an easy task for developing our data set. We had a hard time scraping a lot of posts off of the Reddit website because of the time-period filters that Reddit uses.
Imbalance Dataset: Because we scraped our own data, we did find that most of the ground-truth labeled data we collected was imbalanced since most of the ground-truth labels ended up being “NTA” instead of “YTA”. Considering 88% of the data was “NTA”, this was a largely unbalanced dataset and likely introduced a lot of bias into our model.
Fine-tuning: Using BERT meant that our model was computationally intensive. The fine-tuning step had to be run on a Desktop PC with 32GB RAM and it still crashed multiple times while running. In order to meet the timeline of the project, we were not able to do hyperparameter tuning to improve the model. Additionally, we had to reduce the batch size from 32 to 16 as well as the epoch size to 3 in order to reduce crashing.

⏭️ Future Work

Hyperparameter Tuning: The model could be improved by tuning hyperparameters. Although we did not get to this part due to our early limitations, with time we would like to look into optimizing inputs such as the model type, learning rate, and epsilon values of weight decay.

Performance Metrics: Evaluating the model performance further would be useful for indicating how it can be improved. Particularly, because the input dataset was heavily imbalanced, it would be interesting to compute a confusion matrix and calculate the precision of the model performance to see if the 87% accuracy is attributed to the model’s accuracy, or a high tendency for the model to label posts as ‘NTA’.

Expanding the Dataset: Scraping more data would allow our model to train better and not introduce as much bias due to the small dataset size. However, this would likely be even computationally intensive than our smaller dataset.
Balanced data set: Hand-in-hand with scraping more data, we could further balance our dataset to reduce bias introduced by lots of the data points having the same ground-truth label (explained earlier)
Leveraging other labels: Many top comments on Reddit posts will not explicitly label “YTA” or “NTA”. Instead, they will have a negative or positive sentiment towards the OP. Currently, we are not able to use these data points because they lack the label of “YTA” or “NTA” that we used as our ground-truth. Performing a separate sentiment analysis on these top comments would create additional labeled data to use. There are labels on Reddit besides

“NTA” and “YTA” that our model could be expanded to include such as “ESH” meaning “Everyone Sucks Here” or “NAH” meaning “No Assholes Here” in future iterations of this project.

🗨 ChatGPT Usage

We used ChatGPT throughout our project, for both debugging and guidance on next steps. Some questions we asked ChatGPT include:

- What is the best Machine Learning model for a Sentiment Analysis problem to determine whether someone “sucks” or not
-
- What proportions should I use to split my data into training, validation, and test sets for fine-tuning BERT model for sentiment analysis?
- Why is this code not working? //Followed by buggy code
- Are there any pre-trained large language models for prediction on social media?
    - Which of your above suggestions would be best for sentiment analysis?
- Can you give me code for using bert?
    - It gave us code with bugs so we did not really use it
- Can I upload a file greater than 10MB to github?
- Can you give me a basic outline for an HTML website with CSS