# Detecting Neuroticism from Social Media Posts

Iris Yu, Diane Sun, Lydia Yoder, Anna Lavrentieva

May 17, 2024

**Team Overview**

The authors of this project are Iris Yu, Lydia Yoder, Diane Sun and Anna Lavrentieva. We were able to work collaboratively on this project without publicly posting our data by using a private Github repository. The data processing phase of this project was by far the most thought provoking and the product of the effort from every team member. Every member of our team made valuable and funny contributions to our custom slang text dictionary. For the sake of time, data processing tasks were split up between our team members. Iris applied unsupervised learning methods for identifying the language of the posts and Anna manually verified the results. Lydia and Diane developed the process to remove punctuation, slang, and other unwanted pieces of text. The decision for the last step in our data processing was made with the combined brain power of every member of this team.

In the modeling phase, Lydia took on the main responsibility for training and evaluating baseline models for classification. Iris was in charge of fine-tuning the BERT model. Diane took care of fine-tuning XLNet because she had access to the most powerful computational resources. Anna took the lead in creating visualizations for our model results as well as our presentation slides. Each member of our group contributed to every task in this project and made equally significant contributions to this final writeup.

## 1 Introduction

In psychology, the Big 5 Personality traits used to describe and measure an individual's personality are conscientiousness, agreeableness, extraversion, openness to experience, and neuroticism. These traits are broad representations of personality that help define us as individuals and likely shape the way that we see and interact with each other and the world. Neuroticism stands out from the other Big 5 traits as a trait that is particularly influential on an individual's emotional state and day to day behaviors in real life and online. Neuroticism represents the scale on which an individual is prone to negative emotions and emotional instability [1]. Individuals with high neuroticism are more likely to be anxious, angry, depressed, and pessimistic. Individuals with low neuroticism are more likely to be relaxed and emotionally stable. Therefore, it is likely that an individual's

neuroticism can be reliably detected in their online activity through the language that they choose to use in their social media posts.

Using a corpus of social media posts labeled with the corresponding user's Big 5 Personality traits, we believe that we can use traditional machine learning models and large language models to detect a user's neuroticism through only their posts as a binary classification problem.

## 2 About the Data

We use the 'myPersonality' dataset which contains 9917 fully de-identified status updates from 250 individuals on social media with nominal and numeric measures of their Big 5 personality traits. All identifiable names in the dataset were replaced with '*propname*' for privacy. For the purpose of this project, we only focus on the status updates and the nominal measure for neuroticism in the data. The data is provided as a CSV file where the status posts are stored in the 'STATUS' column and the corresponding nominal label for neuroticism is stored in the 'cNEU' column as either 'y' for neurotic or 'n' for not neurotic. Our data cleaning process is focused on normalizing the text in the 'STATUS' column.

| | STATUS | cNEU |
|---|---|---|
| 0 | likes the sound of thunder. | y |
| 1 | is so sleepy it's not even funny that's she ca... | y |
| 2 | is sore and wants the knot of muscles at the b... | y |
| 3 | likes how the day sounds in this new song. | y |

Fig. 1: Examples of the 'STATUS' and 'cNEU' columns.

## 3 Data Cleaning

The objective of our data cleaning process was to use normalization methods to reduce noise in the corpus without compromising meaning. Each piece of text is a short sentence fragment from social media with significant quantities of misspelled words, text shorthand, slang and emojis. Variations of the same word as typos, shorthand, or slang create noise in our corpus because they are considered separate and unique words in a machine learning context.

The first task in our data cleaning pipeline was to verify that all posts are in English. We applied the SpaCy language detector to our corpus which labeled each piece of text it's detected language. Since SpaCy is not known for state of the art language detection models, we manually reviewed the text with non-English labels to confirm whether those entries were actually not in

English prior to removing them. The remaining English text contained 15,315 unique unigram tokens.

Next, we used JamSpell to spell check the text. JamSpell is a spell-checker library that contains a model trained on 300,000 wikipedia sentences and 300,000 news sentences [2]. We compared the spell-checking performance of JamSpell to other spell-checker libraries such as TextBlob and SymSpell and found that JamSpell was the most suitable for our data cleaning task. This process of spell checking further consolidated our data into 14,461 unique unigram tokens.

However, JamSpell did not address all types of typos within our corpus. One frequent occurrence in the text was for a word to be drawn out with many repeating letters such as, "i'm soooooooo lucky". Based on the logic that it is unlikely for multiple users to use the exact same number of repeated letters for a given word and letters can only repeat up to two times consecutively in the English language, we used regular expressions to replace any letter which appeared 3 or more times in a row with that same letter appearing only twice. In this way, "i'm sooooooooo lucky" becomes "i'm soo lucky". Drawn out letters seem to convey more emotional extremes and could potentially be associated with neuroticism, but needed to be normalized to be useful.

| **Before** | **After** |
|---|---|
| mock exams r makin me insomniatic dat is soo not kool! | mock exams r makin me insomniatic that is so not cool |
| is tired. *propname*, let me go to sleep pl0x. | is tired name let me go to sleep please |

Table 1: Data cleaning before and after examples.

| **Before** | **After** |
|---|---|
| is on twitter now. follow me @ www.twitter.com//[**username**]. | is on twitter now follow me website |

Table 2: Example of identifiable features.

We created a custom dictionary to replace common slang terms with their meaning and shorthand word variations with the actual words. For example, the word "tomorrow" could appear as '2moz', '2moro', 'tm', 'tmrw', 'tmr', and '2mo'. This custom dictionary also replaced all instances of the de-identifier, '*propname*' with just 'name'. We also used regular expressions to remove punctuation from the dataset and replace any URLs with the word "website". The final cleaned dataset contained 13,903 unique unigram tokens.

During our data cleaning, we noticed that the data was not fully de-identified due to the presence of at least one social media link. In the example in Table 2, we manually omit the user's real username from the 'Before' but the original link displays the user's full Twitter username. Fortunately, any such links are normalized to 'website' and successfully de-identified during our cleaning process.

We did not remove stopwords from the corpus during the cleaning process but included a step to remove stopwords from the text before fitting the traditional ML models. The traditional machine learning models such as logistic regression and random forest learn the words in each labeled class as representative features of the class. As a result, stopwords that are common under both classification labels contribute to unwanted noise in the corpus. On the other hand, state of the art transformer models such as BERT and XLNET are pre-trained to be aware of the preceding and succeeding context of a word within a sentence, article, and corpus. For this case, we wanted to preserve the contextual meaning of words in our corpus so we did not remove any stopwords.

The final step of our preprocessing was to transform our corpus from long to wide. After cleaning, there were 246 unique users and a total of 9,120 social media status updates in the updated corpus, with some users represented over 200 times and others only being represented once. The length of the cleaned text sequences ranged from 1 token to 130 tokens long, with an average of 19 tokens. While the average length of text is reasonable for social media, it is insufficient to convey the deeper emotional undertones that characterize neuroticism. To combat this, we concatenated rows of text by unique user ID, then sequentially split the concatenated row into new rows of no more than 100 tokens per row. This created a fairly balanced dataset with 1,737 non-neurotic posts and 1,032 neurotic posts with an overall average length of 96 tokens and a median length of 100 tokens. This method effectively creates discernible meaning in the text without exacerbating the sequence length variability. This method is effective for our particular machine learning task and dataset only because nearly all the users in the corpus are featured multiple times. If the corpus consisted of one post per unique user or if most users only had one post in the corpus, this approach would be flawed.

# 4 Classification

The classification objective is to use our models to reliably predict whether a piece of text indicates neuroticism, ‘y’, or does not indicate neuroticism, ‘n’. We reserved 80% of our processed corpus for training and 20% for testing and validation. The training set contains 2,215 rows of text, with corresponding labels ‘y’ or ‘n’.

We build a classification baseline by evaluating the performance of logistic regression and random forest models on our data. We fine-tuned and evaluated two pre-trained language models, BERT and XLNET, for our classification task to compare with our baseline models. BERT and XLNET are neural networks with transformer architecture and were developed to understand word ambiguity by using surrounding words for context.

## 4.1 Traditional Machine Learning Models

We used both logistic regression and random forest models to act as baseline comparisons for how well BERT and XLNet classified the data. Both models are computationally inexpensive and easy to implement. In order to train these models we needed to create features out of our text data

which the models could ingest. For simplicity, we chose to use the CountVectorizer function from the Scikit-Learn module. Count vectorization is a bag-of-words method which counts the number of times each unique token appears within a text and stores this information in a massive sparse matrix wherein each row represents a piece of text and each column represents a different token. We chose to use tokens consisting of both unigrams and bigrams. The drawback of creating features using bag-of-words methodology is that contextual information like the order in which words appear and how close together words are is lost. However, we hypothesized that the meaning of individual unigrams and bigrams might be very relevant for the prediction of neuroticism and so we predicted that this method would work fairly well, especially with logistic regression. We fine-tuned the hyperparameters for both of these models using GridSearchCV. GridSearchCV iterates over every possible combination of a list of possible hyperparameters for a model and at each iteration evaluates the model using cross-validation.

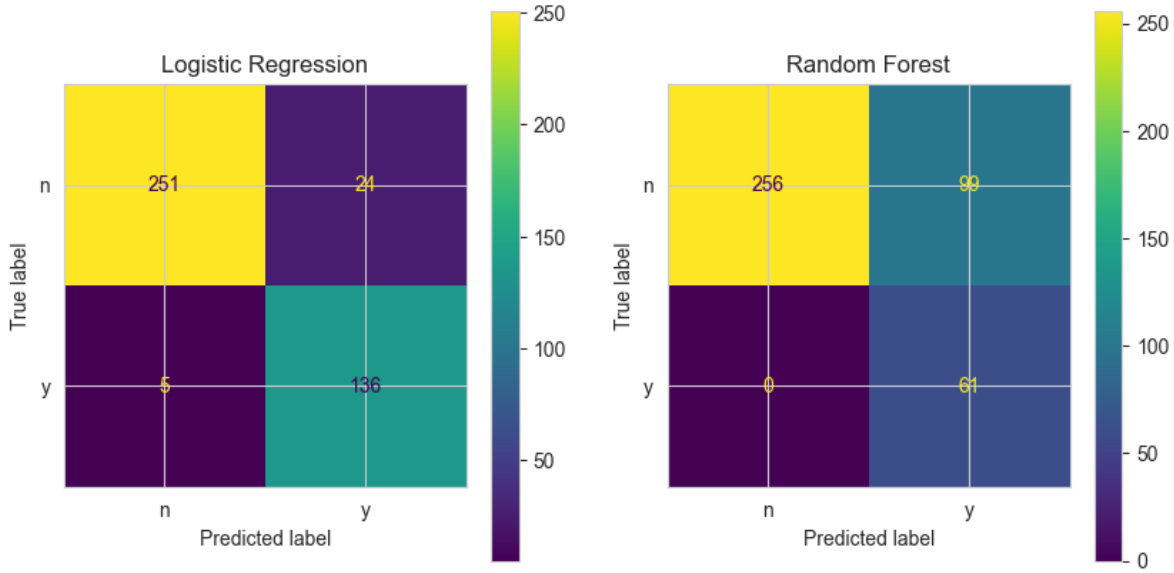


Fig. 2: Confusion matrices generated from the prediction results of the logistic regression and random forest models.

We found that our logistic regression performed quite well with an accuracy of **0.93** and an F1-score of **0.93**. In comparison, random forest was not as successful with an accuracy of **0.76** and F1-score of **0.54**. We observe from Figure 2 that both the logistic regression and random forest models performed well when classifying 'n' labels, but the random forest falls short when classifying 'y' labels. It is likely that random forest did not perform as well because we used Countvectorizer to create features from the text which creates a sparse matrix of feature data. Random forest tends to not be as robust when using sparse data because it tends to create splits on uninformative

variables.

We also examined the most informative features that were calculated by the logistic regression model. Given the logistic regression model's high classification accuracy, we can assume that the model learned the key features that distinguish our two classes from each other.

```
Most informative features
        -0.6178 school                  0.8533  fuck
        -0.5357 eat                     0.7436  xd
        -0.5223 working                 0.6089  fucking
        -0.5094 party                   0.5710  stupid
        -0.5065 place                   0.5143  car
        -0.4948 running                 0.5092  kitty
        -0.4753 feeling                 0.4765  bloody
        -0.4686 week                    0.4694  hell
        -0.4676 let                     0.4633  law
        -0.4663 team                    0.4479  sleep
        -0.4582 bit                     0.4472  perfect
        -0.4473 downtown                0.4448  dance
        -0.4438 enjoy                   0.4426  haha
        -0.4367 touch                   0.4310  frog
        -0.4292 playing                 0.4305  shall
        -0.4243 soon                    0.4240  iphone
        -0.4232 tuesday                 0.4234  fair
        -0.4217 making                  0.4151  cali
        -0.4205 feet                    0.4124  thinking
        -0.4093 birthday wishes         0.4098  okay
```

Fig. 3: Most informative features for 'n' on left and 'y' on right.

We can understand from Figure 3 that swear words appear to be highly associated with neuroticism. We can assume that neurotypical, emotionally stable individuals are less likely to swear or use vulgar language in public forums. The use of offensive language could be indicative of strong negative emotions that are commonly associated with neuroticism, such as anger and frustration. Conversely, the most informative features that are not associated with neuroticism are what we typically associate with stable and well-rounded individuals. Terms such as "school", "eat", "working", "running" and "party" could imply that these users have a healthy balanced lifestyle. Lighthearted features like "playing" and "enjoy" strongly contrast with neurotic features like "hell" and "stupid".

### 4.2 BERT

Bidirectional Encoder Representation from Transformers, known as BERT, is an innovation in language modeling released by a team of Google researchers in 2019 [3]. Unlike unidirectional language models at the time, BERT's pre-training objectives featured masked language modeling and next sentence prediction. The masked language modeling objective allows representations to be learned with both preceding and succeeding context, thus creating a bidirectional representation of language. This differs from traditional recurrent neural networks that see words one after another and autoregressive transformers models like OpenAI GPT that only learns from left to right. This bidirectional strategy frees BERT from the confines of unidirectional task-specific architecture and makes BERT user-friendly and easily fine-tuned for a wide range of downstream tasks.

The pre-trained base BERT model for our fine-tuning task is loaded from the PyTorch transformers library. The BERT base model consists of 12 transformer layers, 768 hidden layers, 12 attention heads, and 110M trainable parameters. We decided to use Adam optimization for training because it is more computationally efficient than SGD (Stochastic Gradient Descent) and compatible with tasks with noisy or sparse gradients [4]. We set a relatively low learning rate of 2e-5 with a training batch size of 32 for fine-tuning. While low learning rates have a higher computational cost and could require more training iterations to converge, setting the rate too high could lead to accidentally skipping over the optimal solution. We limited our training iterations to 4 epochs to minimize the risk of BERT overfitting on our relatively small amount of training samples.

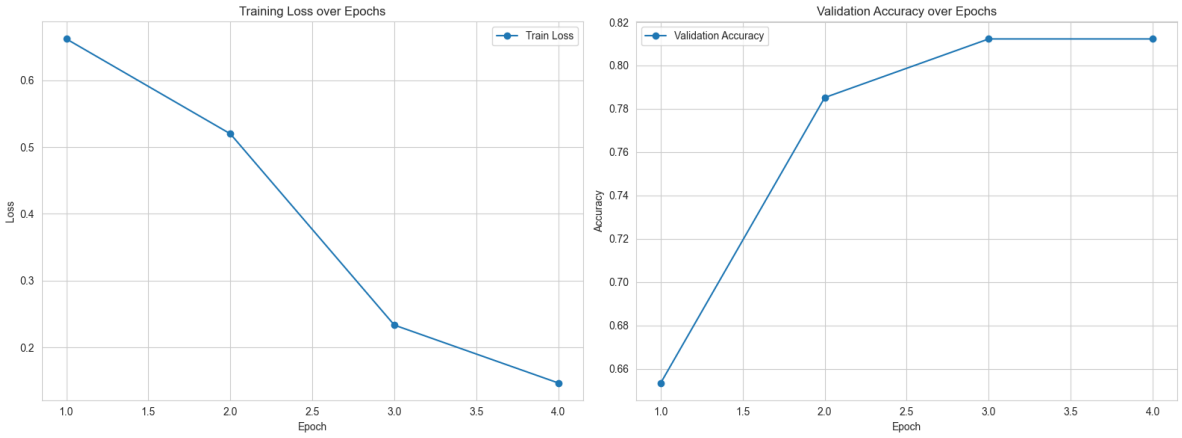


Fig. 4: Training loss and validation over 4 epochs.

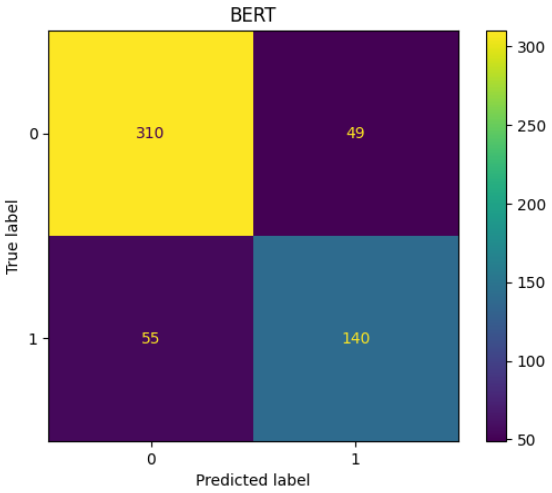


Fig. 5: Confusion Matrix generated from predictions made on the validation set after epoch 4. Labels have been encoded as 0 for 'n' and 1 for 'y'.

We evaluate the model on our validation set at the end of each epoch to validate our training process. With the given hyper-parameter combination, we observe from Fig. 4 that our fine-tuned BERT's validation classification accuracy plateaus at **0.81** by the third training iteration. While we did not reserve additional data for test evaluation after training, the confusion matrix 5 created from BERT's predictions for the validation set after 4 epochs indicates that BERT was able to learn the representative features of the text. Though there is signficant room for improvement and more testing, these preliminary results indicate BERTs potential for this classification task.

### 4.3 XLNet

XLNet, which is short for "eXtreme Language understanding NETwork" was also introduced in 2019 by researchers from Carnegie Mellon University and Google [5]. XLNet was designed to address BERT's limitations in modeling dependency between masked tokens by building off of BERT's bidirectional concepts and autoencoding pre-training. XLNet integrates autoregressive methods to learn bidirectional contexts via permutation based training as well as concepts from another autoregressive model, Transformer-XL into its pre-training framework. As a result, XLNet is able to learn more dependency pairs with more dense training representations than BERT, given the same data. Given the same training recipe, datasets and hyper-parameters, the XLNet-Large model also significantly outperforms BERT-Large across GLUE (General Language Understanding Evaluation) tasks. However, XLNet's enhanced performance potential comes with a higher computational cost and takes approximately 1.5x longer to complete the same task as BERT on a modern processing unit [6].

The pre-trained base XLNet model for our fine-tuning task is loaded from the PyTorch transformers library. XLNet-Base has 12 transformer layers, 12 attention heads, 768 hidden layers and 110M trainable parameters. We reuse the same hyper-parameter and optimizer combinations used to fine-tune BERT for comparative purposes. We use Adam optimization with a learning rate of 2e-5 and a training batch size of 32. We trained the XLNet model for 4 epochs with validation at the end of each epoch.

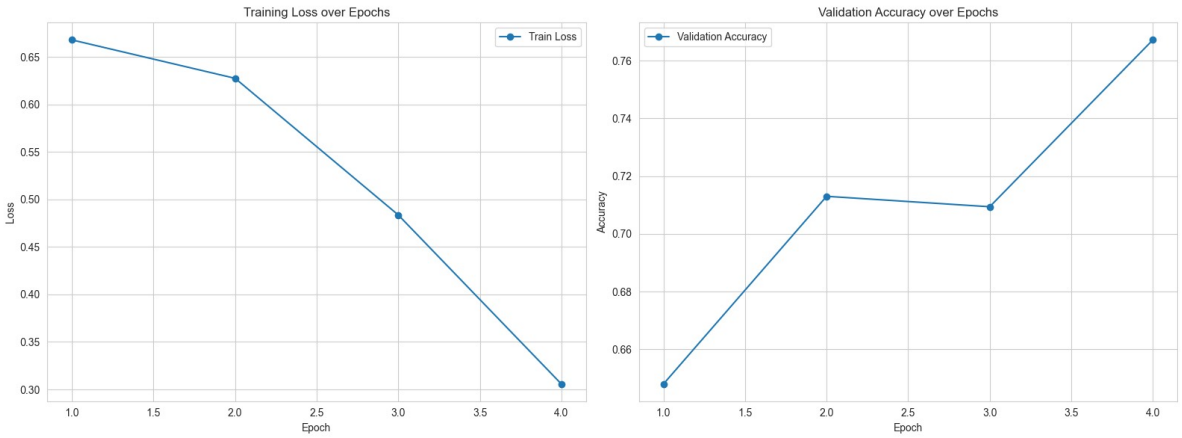


Fig. 6: Training loss and validation over 4 epochs.

We observe from Figure 6 that our XLNet model exhibits some training instability as the validation accuracy dips down between epoch 2 and 3 before improving to **0.77** in the final epoch.

# 5 Comparing Results

| | **Accuracy** |
|---|---|
| Logistic Regression | **0.93** |
| Random Forest | 0.76 |
| BERT | 0.81 |
| XLNet | 0.77 |

Table 3: Comparing accuracy of tested models.

Our best performing model overall was logistic regression. Our logistic regression model achieved a test accuracy of **0.93** and required minimal computational cost to train. Unsurprisingly, the random forest model performed relatively poorly by comparison. The random forest model or any tree-based model is likely not the ideal type of model to use for the purposes of our classification task. As a note, we did not reserve a test set for a final evaluation round for BERT

and XLNet so we are comparing validation accuracy from BERT and XLNet with the test accuracy of our baseline models

It is interesting to observe in Table 3 that our fine-tuned pre-trained models performed worse than the logistic regression model. This could be because BERT and XLNet are both designed to learn highly complex representations of language that could not be encapsulated by our limited number of training data points. In this case, the logistic regression model is able to adequately learn and represent all the distinguishing features between neurotic and non-neurotic text.

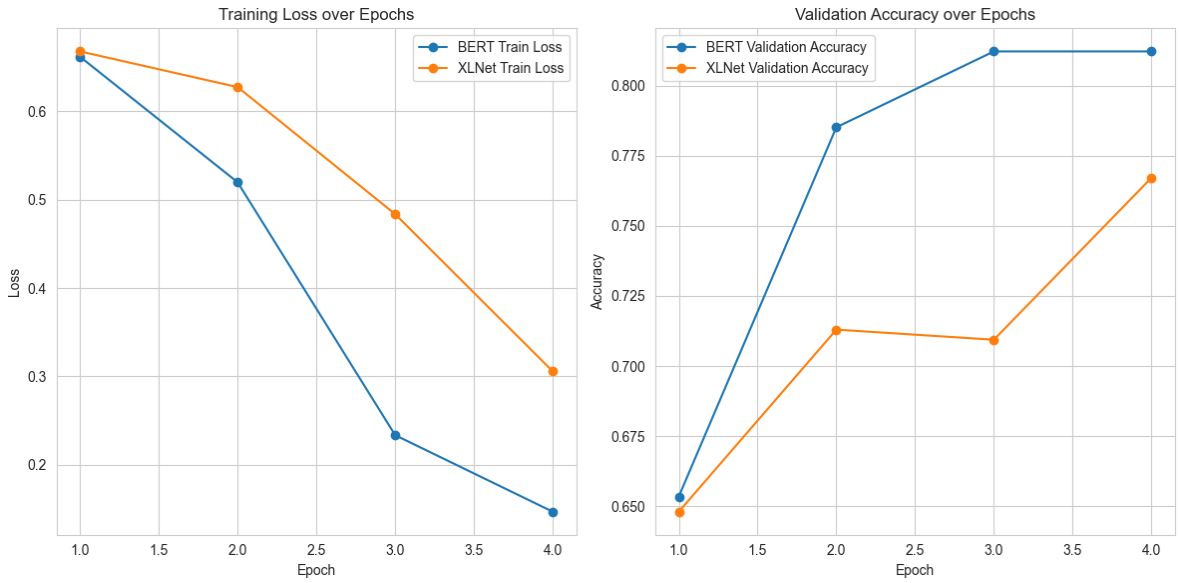


Fig. 7: Training loss and Validation Accuracy of BERT and XLNet

It is not surprising that BERT outperformed XLNet on the same task given the same hyper-parameter combinations. BERT is a smaller, less complex model than XLNet that requires less computational power to fine-tune. As a result, BERT was able to achieve better results with a shorter training time than XLNet on the same data. BERT was also designed with ease of use and easy fine-tuning in mind, which seems to have been reflected in its performance in Figure 7. XLNet was designed to have even improved ability to learn pairwise dependency from BERT and therefore model even more complex aspects of language. XLNet is likely far too complex to be fine-tuned to achieve excellent results for our classification task with our limited data.

## 6 Limitations and Future Directions

The authors of the 2020 paper presenting a novel 'mixout' regularization method eloquently summarize the main issue that plagued our project– "... fine-tuning a large pre-trained language model

on a downstream task is prone to degenerate performance when there are only a small number of training instances available" [7]. It is well known that large models typically require around 10,000 data points to achieve superior downstream performance after fine tuning. If the quantity of data is limited, then we can assume that the quality of the data available becomes exceedingly important. Fortunately, there are a number of methods that we can continue to pursue to refine our models and deepen our understanding of neuroticism as a binary classification problem.

Since our final corpus contained less than 3,000 task-specific data points, we can perform data augmentation to create additional synthetic data points. We can apply word swapping by swapping words with their synonyms, randomly insert synonyms of non-stop words into the data, and randomly delete certain words. We can further apply sentence level swaps by swapping the order in which we concatenated posts during the cleaning phase. We can also re-incorporate all the non-English language text omitted from our training corpus by performing translation tasks and create additional synthetic data by performing back-translation tasks. These processes can help to create a larger training corpus and hopefully models that are more robust to unseen data.

Due to limitations to our computational resources, we were unable to employ hyper-parameter tuning techniques for both BERT and XLNet within the timeframe of this project. Hyper-parameter tuning is an art for large complex models and the ideal hyper-parameter combinations are rarely intuitive. We can potentially employ techniques to test different hyper-parameter combinations such as learning rate, batch size, and number of training iterations by using methods such as Grid Search, population based training and bayesian optimization [8]. We can leverage cloud computing platforms to provide the computational resources needed to find the ideal hyper-parameter combinations for our task for each model.

We can incorporate fine-tuning techniques to accommodate the models' high capacities to absorb complex patterns. Stochastic weight averaging (SWA) and re-initializing pre-trained layer weights are methods that we can apply to evaluate how we can adjust layer weights in our models for potential performance improvements. We can apply gradual unfreezing during the training process by gradually unfreezing the layers of our models starting from the last layer which essentially trains layers separately. This method is computationally efficient because the model does not train every single layer during each training epoch and reduces the risk of overfitting by restricting the overall number of parameters that can be updated in each epoch. We can incorporate the novel 'mixout' regularization method, a method inspired by the existing dropout regularization techniques for the purposes of fine-tuning pre-trained language models for downstream tasks. This method stochastically mixes parameters from two models to minimize the deviation from the target parameters from one of the models [7]. As demonstrated by the authors of this technique on applications with BERT for downstream GLUE tasks, we can potentially improve the training stability and overall classification accuracy of our models.

# 7 Applications

The accurate detection of broad personality traits from public social media posts has a number of applications across industries beyond the field of psychology. Being able to detect personality traits in social media posts can help to build detailed customer, or even regional profiles that are invaluable for customer identification and segmentation objectives in the marketing and ads industries. Understanding personality traits can potentially provide insight into different voter profiles and political sentiments. These approaches could even be used to potentially characterize the 'culture' of colleges and universities to help students decide where they want to go to pursue higher education!

# 8 Conclusion

We have determined that it is possible to detect the presence of neuroticism in our corpus of posts from 246 users by using both traditional machine learning models and large language models. It is important to state that our sample size of social media users is small and not necessarily representative of the general population. Our models cannot be ethically or reliably used to diagnose the traits of any other social media users by their posts. The use of social media data itself raises ethical privacy questions as we have observed in our data that it is difficult to guarantee all data is fully de-identified. We have also observed how the ideal solution depends on the problem posed and is limited by resource availability. We demonstrated that traditional machine learning methods are often adequate for simple downstream tasks such as binary classification. Our logistic regression model was able to achieve an accuracy of 0.93 on our test corpus and required few computational resources to optimize hyper-parameters, train, and test. Conversely, large pre-trained models like BERT and XLNet have a higher capacity to learn complex representations, but are computationally expensive and require large quantities of data to achieve acceptable results. Such models would be ideal for complicated representations of language in conjunction with larger datasets.

# Acknowledgments

We thank our professor, Dr. Shabnam Tafreshi, for providing us with the necessary data and skills to complete this project.

## References

[1] "Neuroticism | definition, psychology, big five, & examples | britannica," Apr. 2024.

[2] "Jamspell: spell checker."

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019. arXiv:1810.04805 [cs].

[4] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017. arXiv:1412.6980 [cs].

[5] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," Jan. 2020. arXiv:1906.08237 [cs].

[6] H. Li, J. Choi, S. Lee, and J. H. Ahn, "Comparing BERT and XLNet from the Perspective of Computational Characteristics," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–4, Jan. 2020.

[7] C. Lee, K. Cho, and W. Kang, "Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models," Jan. 2020. arXiv:1909.11299 [cs, stat].

[8] "Weights & Biases."