Introduction

Ever since high school, I have always had a great interest in psychology. Understanding myself and the people around me was fun and didn't feel like work for me. I took a social psychology course during my undergraduate degree, and it introduced me to many very cool concepts like ingroups and outgroups, social influence, and personalities. This class is where I was introduced to the Myers Briggs Type Indicator (MBTI) test. It's a fairly popular way for people to get to know each other and is a fun thing to ask your friends to see how well you match according to this system. What I'm curious about is- is there a way to look at words from a person and be able to determine their personalities using Neural Networks? Would the NN be able to more accurately predict a person's personality than the Myers-Briggs test?

Actually being able to answer this question is out of the scope for this project but what *is* in scope is- can NLP be used to accurately predict a person's personality based on text? From there, we can look at results and see if it's worth pursuing further.

My work

Since I worked individually, all code was developed by me. For EDA, I took some inspiration from other Kagglers who did work on the same dataset (i.e. the pie chart). My portion of this project is its entirety.

Results

While I was able to get the classical modeling techniques done, I unfortunately ran into complications when trying to run my LSTM, BERT, and zero-shot code. However, I'll cover the classical modeling techniques first. I decided to lay the groundwork for the project with Logistic Regression, Multinomial Naïve Bayes, and XGBoost.

Multinomial Naïve Bayes ended up performing the worst with its validation F1 score coming out to 0.275, which represents its poor precision and recall scores. In second place came the logistic regression model with a validation F1 score of 0.5411, which puts it just barely above the performance of a coin flip. And finally, XGBoost came out on top with a validation F1 score of 0.648. It isn't amazing but it set a pretty fair bar for the neural network models.

	Logistic Regression	Multinomial Naïve Bayes	XGBoost
Precision	0.5785	0.2954	0.6507
Recall	0.5834	0.3607	0.6583
F1	0.5411	0.275	0.6480

Unfortunately, I couldn't get results for LSTM and BERT code I wrote. I figured that having all the code ready to the best of my ability would be the end of the struggle but between library dependencies and undecipherable errors, I feel like I should have given myself more time to sort through them rather than dealing with it all in the last second. Errors ranged from "dnn not implemented" to the new Python version having an issue with tensorflow. Having gone down many rabbit holes to solve these problems and not being able to achieve a result was incredibly frustrating.

However, I was able to get the zero-shot classifier running. The only downside is that I had to use Google Colab and it is running very slowly (53 minutes for 13 predictions). At the time of writing, it is still running and has achieved an accuracy of 23% which is pretty unremarkable. For a zero-shot solution with a task like this, I can't say I expected better but it was cool to try nonetheless.

Summary and Conclusions

From this project, I learned about many of the complexities that go into NLP problem spaces. As Professor Jafari mentioned during one of the presentations, taking out stop words every time may not be a great idea. For instances where classical techniques are used, they should be removed to limit any noise but for neural networks, particular stop words may offer important context that can alter predictions. I also realized that cool, advanced algorithms do not always have to be the answer. While I'm sure the LSTM and BERT models could have performed quite well had I been able to implement them, there doesn't have to be a rush to pursue these more expensive solutions if a classical or simpler solution gets the job done. XGBoost seems like a solid baseline model to try and beat for my next classification problem. However, one major point that was solidified (and maybe it is obvious) is that the performance of the model hinges mostly- if not entirely- on the text processing. As a result, one must be very careful and considerate as to how they go about making the data ingestible to models. This course and project were instrumental to these realizations.

Percentage of Code Found/Repurposed

((20– 12) / (10 + 35) x 100 = 17.7%

References

- Myers-Briggs Test

https://www.themyersbriggs.com/en-US/Campaigns/All-About-the-MBTI-Assessment