ACM Projects Team [TBD]

Colors!

here are the main colors we use at acm!

- binary blue
- big O(range)
- ctf cyan
- prototyping pink
- innovation indigo
- sentient scarlet

Meet the Team!

Vincent Tu
Mentor

Sia Patodia

Aryaman Dayal

Hargen Zheng

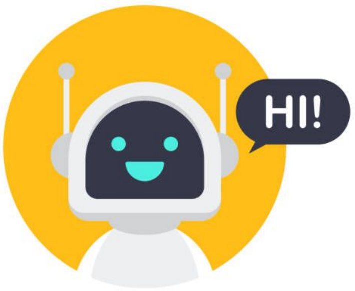
Catherine Zhang

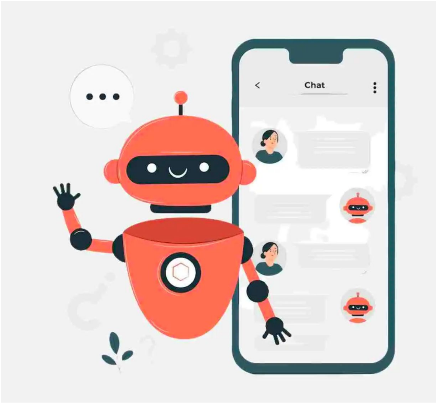
Ryan Wong

Phillip Wu

Inspiration & Background

- AI Chatbot
  - Responds based on user input
- MBTI Personality Test
  - Can influence texting style





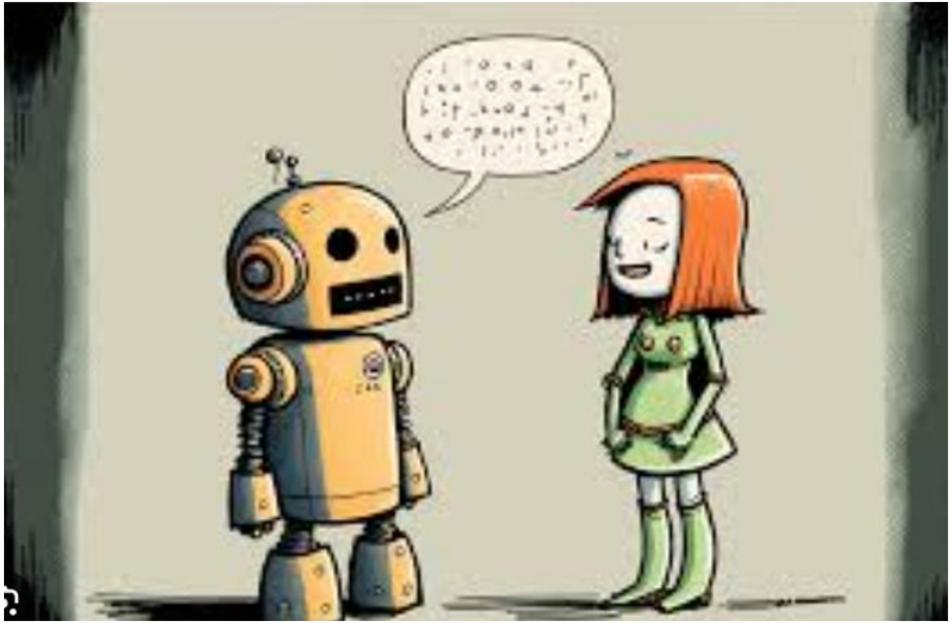
ESTJ - ESFJ - ENFJ - ISTJ -ISFJ -INTJ - INFJ - ESTP - ESFP - ENTP - ENFP - ISTP - ISFP - INTP - INFP-ENTJ
SPORT

Features

✓ Predict MBTI personality with a decent accuracy

○ Have a chatbot talk back to user with that MBTI personality

ISTJ
ISFJ
INFJ
INTJ
ISTP
ISFP
INFP
INTP
ESTP
ESFP
ENFP
ENTP
ESTJ
ESFJ
ENFJ
ENTJ



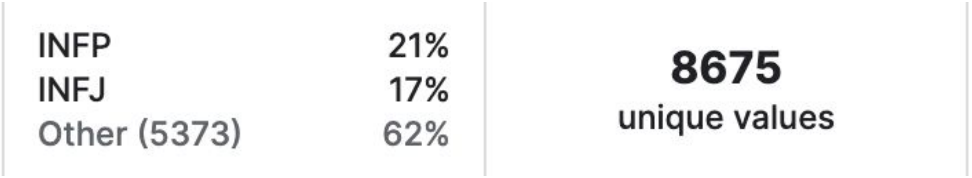
Look at our tech stack !!

pandas
GitHub
jupyter
kaggle
git

Technical Process

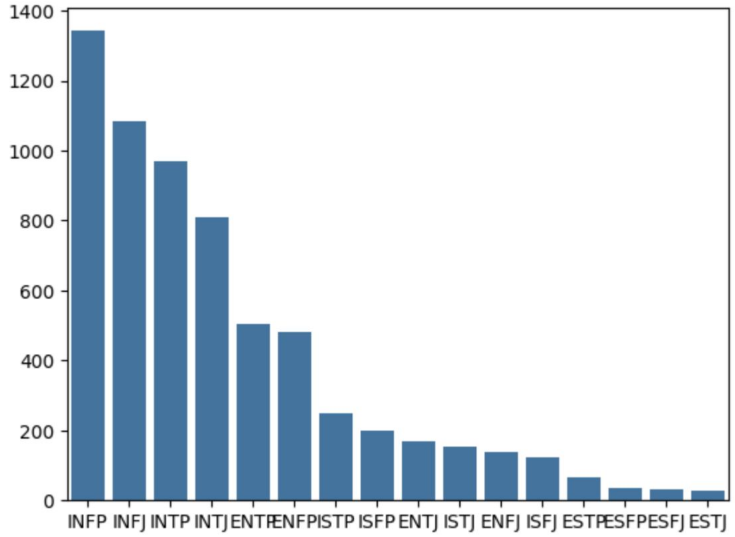
Dataset link: [(MBTI) Myers-Briggs Personality Type Dataset]

- EDA results:
  - Dataset has more introverts (super unbalanced)
  - Each entry has user's MBTI type and words from user's last 50 posted tweets
- Preprocessing:
  - Tokenization using BERT pre-trained tokenizer
  - Encode the MBTI type into integers 0-15
  - Only keep words between 3 and 30 characters



INFP	21%
INFJ	17%
Other (5373)	62%

8675
unique values

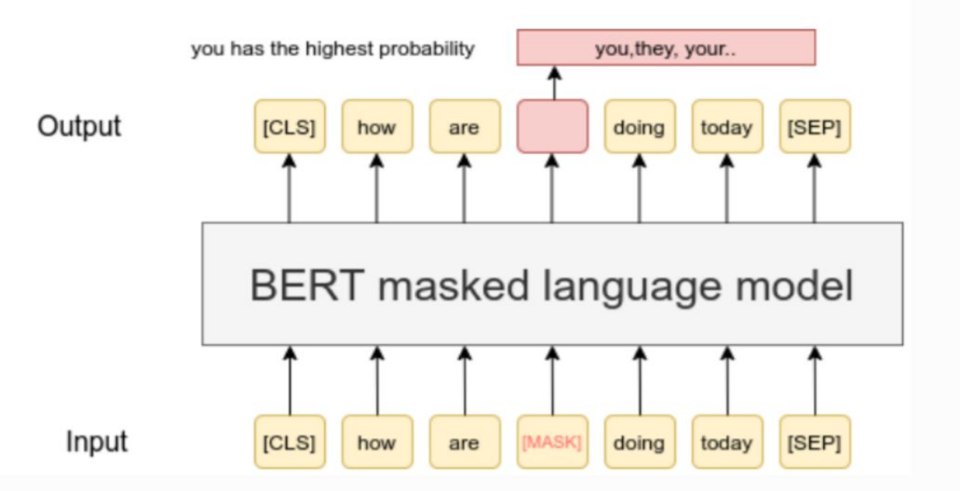


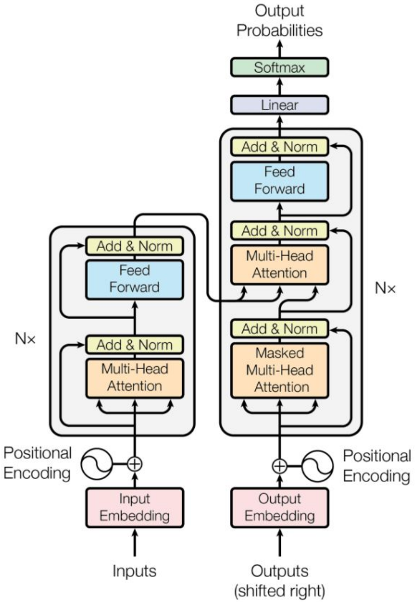
Here are the original notes vincent wrote down

- Talk about dataset (e.g. dataset size, modality, feature engineering, cleaning, EDA, insights, conclusions, motivation behind this dataset)
- Talk about preprocessing (tokenization, stemming, lemmatizing, etc)
- Talk about the model (BERT other neural language models -> talk more about how these models are made and how they work; LSTM -> long-short-term memory loosely inspired by human brain)
- Talk about the training (and finetuning), discuss training configuration and set up; if you CV, then you can also talk about that
- Talk about your experiments (what worked and what didn’t work; possibly hypotheses for why things didn’t work or why things worked)
- Talk inference (how do we run the model)?

Technical Process (cont.)

- Model: **BERT (Bidirectional Encoder Representations from Transformers)**
    - Unique since it is considered a bidirectional model (captures context well)
    - Uses a transformer architecture
    - Combined predicting words along with predicting if sentences belonged to each other to create the BERT model





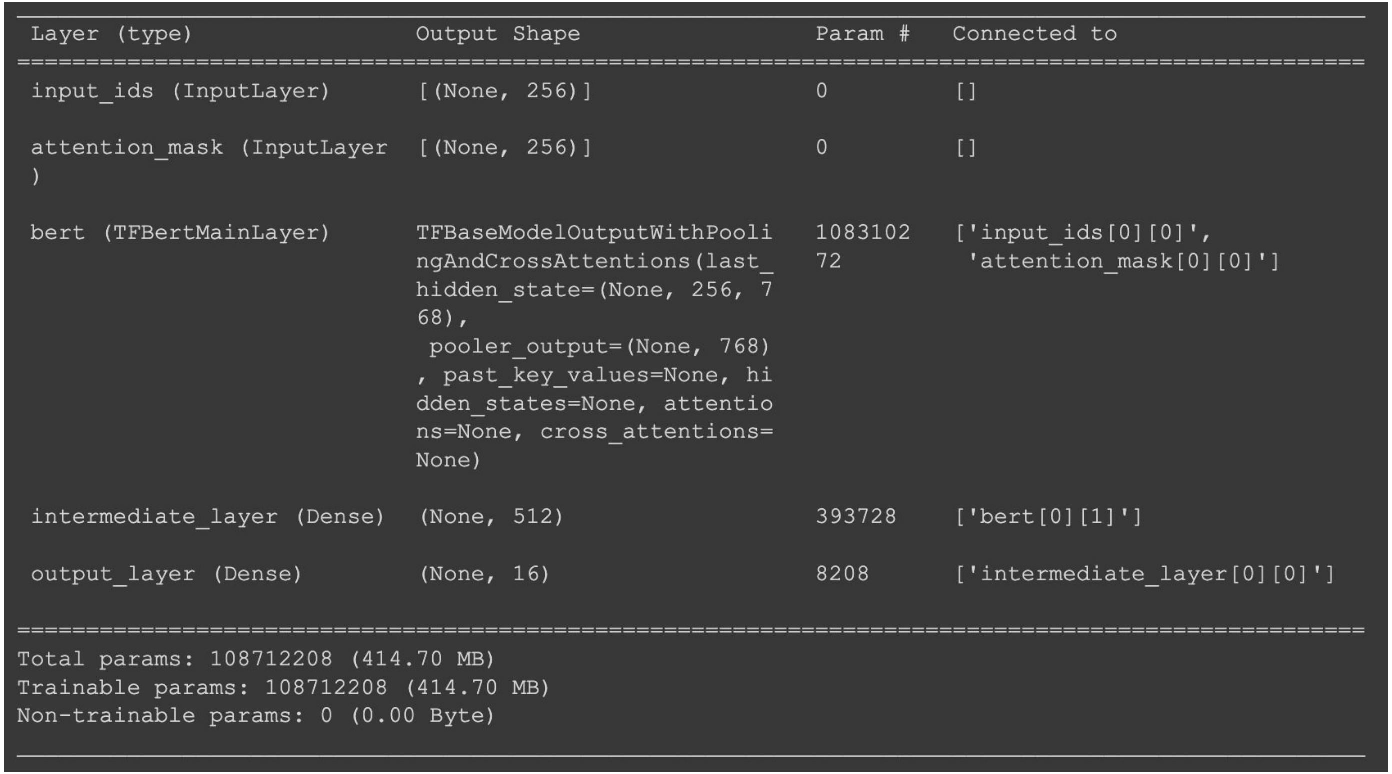
Technical Process (cont.)

Training the BERT Model

- Transfer learning (cold start issue due to size of parameters, limited computation power). We used Colab T4 GPU to train the model.
- Embed each text input into a vector with maximum length of 256 (padding).
- Shuffle the data and use 80/20 train-validation split.
- Batch size of 16 (drop the remainder). This results in 433 batches in each epoch.
- Add a fully connected layer with 512 hidden units, with ReLU activation function.
- Use a softmax layer with 16 output units to represent the probabilities of each MBTI type, given the input text corpus.
- Used the built-in Adam Optimizer with decay and Cross Entropy Loss function.

Technical Process (cont.)

BERT Model Summary



```
_____________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #    Connected to
=============================================================================================
 input_ids (InputLayer)      [(None, 256)]                0          []

 attention_mask (InputLayer  [(None, 256)]                0          []
 )

 bert (TFBertMainLayer)      TFBaseModelOutputWithPooli   1083102    ['input_ids[0][0]',
                             ngAndCrossAttentions(last_   72          'attention_mask[0][0]']
                             hidden_state=(None, 256, 7
                             68),
                              pooler_output=(None, 768)
                             , past_key_values=None, hi
                             dden_states=None, attentio
                             ns=None, cross_attentions=
                             None)

 intermediate_layer (Dense)  (None, 512)                  393728     ['bert[0][1]']

 output_layer (Dense)        (None, 16)                   8208       ['intermediate_layer[0][0]']

=============================================================================================
Total params: 108712208 (414.70 MB)
Trainable params: 108712208 (414.70 MB)
Non-trainable params: 0 (0.00 Byte)
_____________________________________________________________________________________________
```

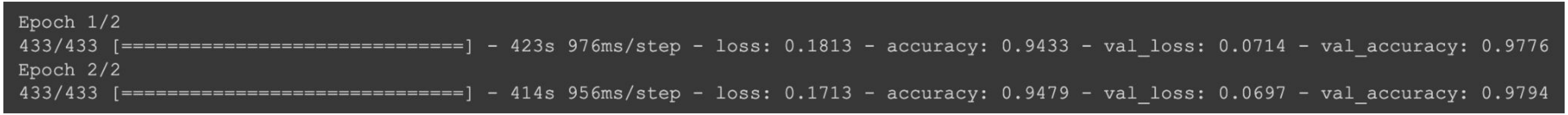
Technical Process (cont.)

Experiments

- Though 1e-5 is the recommended learning rate for most of the transformer models, it was super slow to train for us. As a rookie I changed to 1e-4 and we were overshooting – after 1 epoch, the accuracy went from 35% down to 18% and it kept decreasing.
- Used 5e-5 as learning rate and decay of 1e-6 – accuracy keeps increasing all the way to ~95% on the training set. This is way better than the original Softmax regression model, which obtained a 84% on the training set.
- Could've tune the maximum size of word embeddings and the size of hidden layers

Technical Process (cont.)

Eventual Model Performance



```
Epoch 1/2
433/433 [==============================] - 423s 976ms/step - loss: 0.1813 - accuracy: 0.9433 - val_loss: 0.0714 - val_accuracy: 0.9776
Epoch 2/2
433/433 [==============================] - 414s 956ms/step - loss: 0.1713 - accuracy: 0.9479 - val_loss: 0.0697 - val_accuracy: 0.9794
```

Technical Process (cont.)

Inference

- We clean the text corpus in the same way as we did for the training and validation examples.
- Then, we used the BERT tokenizer to embed the input text corpus.
- The feature vector of the input text is then fed into our transformer model, which gives us 16 probabilities corresponding to the likelihood of each personality type.
- We then extract the top three most likely personalities and display to the user.

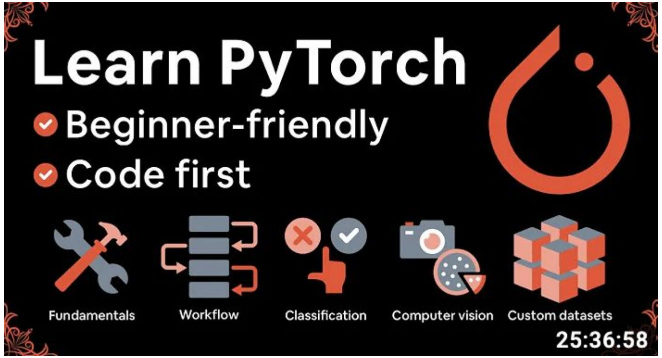
Technical Process (cont.)

Streamlit App

- We created a Streamlit application that shows the distribution of MBTI personality types in our dataset.
- User could input paragraphs of text and the app will predict their top three likely MBTI personalities using our model.
- WHY TOP 3, not just 1???
- After the prediction result pops up, there will be additional MBTI information of the three likely personalities, so the user can learn more about their potential MBTI type.

Challenges

- Meeting scheduling conflicts
- Balancing classwork and project
- Lack of experience (PyTorch, Model Choice, Training)
- Model overfitting



Overfitting

Reflection

- Smaller group sizes may make it easier to meet
- Self-guided learning is difficult
- Talk about how RoBERTa could potentially work better (as it was trained on a larger text corpus)

Where do we go from here?

- Create mobile application for interacting with chatbot
- Learn more about NLP and deep learning

Demo

Questions?

Thank you!

- Mentor: Vincent Tu LinkedIn | GitHub
- Catherine Zhang LinkedIn | GitHub
- Aryaman Dayal LinkedIn | GitHub
- Hargen Zheng LinkedIn | GitHub
- Sia Patodia LinkedIn | GitHub
- Phillip Wu GitHub
- Ryan Wong LinkedIn | GitHub