Application of Different Transformer Architectures and Models on Twitter Sentiment Analysis

MIS 64061: Final Project Report

Eyob Tadele

05/04/2022

Introduction

Although different types of neural network models have been optimized for different types of datasets/problems, none were without their shortcomings. Convolution neural netwoks (CNN) were widely used in image classification and Recurrent neural networks(RNN) were widely used in natural language processing. In the case of RNNs and natural language processing, where sentiment analysis falls under, it takes input as a sentence, process the words one at a time, and sequentially splits it out. Since order is important, it does it by looking at one word at a time sequentially, but not much further ahead. Most of the limitations of RNNs and CNNs can be attributed to not having long memory, encoding bottleneck, and slow/no parallelization. As a result, concept of Transformers rose to address these issues with emphasis on positional encoding, attention, and self-attention.

The basic idea in positional encoding is to store information about word order (i.e., store it in the data itself, rather than structure of the network). This way, the NN learns the importance of word order from the data. Attention is NN structure that allows a text to look at every single word in the original sentence when making a decision about how to predict the sentiment or translate (in the case of translation). This is learned from the data over time and not a new concept. Self-attention however, is used to get the underlying meaning in language, so as to build a network that can do any number of language tasks. In short, it allows a NN to understand a word in the context of the other words around it, disambiguate words, and recognize parts of speech. Implementation of these concepts is most notable in architectures known as Transformers, where we see them being applied in language processing, biological sequences, and computer vision.

Project Objective

The main objective of this project is to explore sentiment analysis on Twitter dataset using Transformer models. Although sentiment analysis is a well studied problem in machine learning, this specific project attempts to look into how different transformer models affect performance. For this purpose, three models are interchangeably applied consisting of a transformer model built from scratch using tensorflow and keras functionalities, a pre-trained Bidirectional Encoder Representations from Transformers(BERT), and a third generation Generative Pre-Trained Transformer (GPT-3).

Dataset and Data Preparation

The dataset used for this project was originally made available on kaggle (https://www.kaggle.com/datasets/kazanova/sentiment140 (https://www.kaggle.com/datasets/kazanova/sentiment140)). It consists of 1.6 million rows and six fields. The features include ids, date, flag, user, and a target. The most relevant variables for this project are the text, which represents the tweets, and the target representing the sentiments. Due to resource constraints during training, it was decided to reduce the size to 200 thoushand with a 50-50 split of positive and negative sentiments. A series of data preprocessing was done to remove casings, numbers, special characters, and punctuations in order for it to be suitable for use by the different transformer models. The dataset is split into training and test sets in an 80-20 percent share.

```
df.shape =  (200000, 2)
label distribution :
0.0    100000
1.0    100000
Name: label, dtype: int64
   label                                           sentence
0    0.0  @switchfoot http://twitpic.com/2y1zl - Awww, t...
1    0.0  is upset that he can't update his Facebook by ...
2    0.0  @Kenichan I dived many times for the ball. Man...
3    0.0    my whole body feels itchy and like its on fire
4    0.0  @nationwideclass no, it's not behaving at all....
        label                                           sentence
199995    1.0  @jvdouglas  haha, no, the remark on maternity ...
199996    1.0                                @altitis and to you!
199997    1.0  Okie doke!! Time for me to escape for the Nort...
199998    1.0                    finished the lessons, hooray!
199999    1.0  Some ppl are just fucking KP0. Cb ! Stop askin...
```

```
df.shape =  (200000, 2)
label distribution :
0.0    100000
1.0    100000
Name: label, dtype: int64
   label                                           sentence
0    0.0       awww thats a bummer  you shoulda got davi...
1    0.0  is upset that he cant update his facebook by t...
2    0.0    i dived many times for the ball managed to s...
3    0.0    my whole body feels itchy and like its on fire
4    0.0    no its not behaving at all im mad why am i h...
        label                                           sentence
199995    1.0     haha no the remark on maternity leave fired...
199996    1.0                                      and to you
199997    1.0  okie doke time for me to escape for the north ...
199998    1.0                    finished the lessons hooray
199999    1.0  some ppl are just fucking kp cb  stop asking m...
```

Methodology

The approach applied to tackle this project is divided into three stages. Initially, the twitter sentiment dataset is trained on a transformer model built from scratch using tensorflow and keras functionalities. Next, the same twitter dataset is trained on a generic, as well as, a pre-trained Bidirectional Encoder Representations from Transformers(BERT). Finally, a GPT-3 model from OpenAI is used to train and predict the sentiment.

1. Transformer from scratch using tensorflow and keras:

In this case, the transformer only includes the encoder with a classifier layer added on top (i.e., the classifier can be though of as a decoder). The encoder layer consists of a multi-head attention and point wise feed forward networks. Each of these sublayers contains a residual connection around it, followed by a layer of normalization. This is implemented by a TransformerBlock class provided by keras (*source: https://keras.io/examples/nlp/text_classification_with_transformer/ (https://keras.io/examples/nlp/text_classification_with_transformer/)*). Embedding and positional information for the tokens is also implemented by using the TokenAndPositionEmbedding class provided by keras.

```
        self.att = layers.MultiHeadAttention(num_heads=num_heads,
                                             key_dim=embed_dim)
        self.ffn = keras.Sequential([
            layers.Dense(ff_dim, activation="relu"),
            layers.Dense(embed_dim),
        ])
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)

    def call(self, inputs, training):
        attn_output = self.att(inputs, inputs)  # self-attention layer
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(inputs + attn_output)  # layer norm
        ffn_output = self.ffn(out1)  #feed-forward layer
        ffn_output = self.dropout2(ffn_output, training=training)
        return self.layernorm2(out1 + ffn_output)  # layer norm
```

In [6]:
```
# Embedding and Position for the tokens in the embeddings
# Using TokenAndPositionEmbedding provided in keras

class TokenAndPositionEmbedding(layers.Layer):
    def __init__(self, maxlen, vocab_size, embed_dim):
        super(TokenAndPositionEmbedding, self).__init__()
        self.token_emb = layers.Embedding(input_dim=vocab_size,
                                          output_dim=embed_dim)
        self.pos_emb = layers.Embedding(input_dim=maxlen, output_dim=embed_dim)

    def call(self, x):
        maxlen = tf.shape(x)[-1]
        positions = tf.range(start=0, limit=maxlen, delta=1)
        positions = self.pos_emb(positions)
        x = self.token_emb(x)
        return x + positions
```

The encoder is used as an input to a dense layer with a sigmoid activation for the sentiment prediction. A functional API is used to build the model and a code snippet is indicated as follows:

```
embed_dim = 32  # Embedding size for each token
num_heads = 2  # Number of attention heads
ff_dim = 32  # Hidden layer size in feed forward network inside transformer

# Using Functional API
inputs = layers.Input(shape=(maxlen,))
embedding_layer = TokenAndPositionEmbedding(maxlen, vocab_size, embed_dim)
x = embedding_layer(inputs)

transformer_block = TransformerBlock(embed_dim, num_heads, ff_dim)
x = transformer_block(x)
x = layers.GlobalAveragePooling1D()(x)
x = layers.Dropout(0.1)(x)
x = layers.Dense(20, activation="relu")(x)
x = layers.Dropout(0.1)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
```

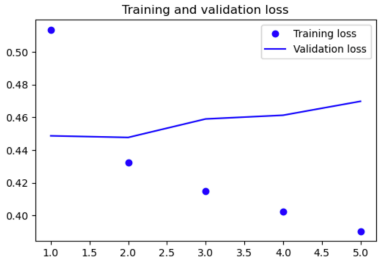
```
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200)]             0

 token_and_position_embeddin  (None, 200, 32)          326400
 g (TokenAndPositionEmbeddin
 g)

 transformer_block (Transfor  (None, 200, 32)          10656
 merBlock)

 global_average_pooling1d (G  (None, 32)               0
 lobalAveragePooling1D)

 dropout_40 (Dropout)        (None, 32)                0

 dense_2 (Dense)             (None, 20)                660

 dropout_41 (Dropout)        (None, 20)                0

 dense_3 (Dense)             (None, 1)                 21

=================================================================
Total params: 337,737
Trainable params: 337,737
Non-trainable params: 0
_________________________________________________________________
```

The model uses "adam" as the optimizer and "binary_crossentropy" as the loss function. The dataset is trained with a batch size of 64 and 4 epochs. The result of the training shows a lowest validation loss around 2 epochs and a validation accuracy of around 78.97%. [The code snippet and results are indicated below.]

```
model.compile(optimizer="adam",
              loss="binary_crossentropy",
              metrics=["accuracy"])

history = model.fit(train_tweets,
                    train_labels,
                    batch_size=64,
                    epochs=5,
                    validation_data=(val_tweets, val_labels))
```

```
Epoch 1/5
2022-05-02 22:18:42.106358: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2500/2500 [==============================] - ETA: 0s - loss: 0.5135 - accuracy: 0.7310
2022-05-02 22:24:33.331488: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
2500/2500 [==============================] - 372s 148ms/step - loss: 0.5135 - accuracy: 0.7310 - val_loss: 0.4487 - val_accuracy: 0.7899
Epoch 2/5
2500/2500 [==============================] - 376s 150ms/step - loss: 0.4322 - accuracy: 0.7993 - val_loss: 0.4477 - val_accuracy: 0.7898
Epoch 3/5
2500/2500 [==============================] - 379s 151ms/step - loss: 0.4149 - accuracy: 0.8074 - val_loss: 0.4590 - val_accuracy: 0.7884
Epoch 4/5
2500/2500 [==============================] - 381s 152ms/step - loss: 0.4025 - accuracy: 0.8132 - val_loss: 0.4613 - val_accuracy: 0.7897
Epoch 5/5
2500/2500 [==============================] - 379s 152ms/step - loss: 0.3904 - accuracy: 0.8172 - val_loss: 0.4698 - val_accuracy: 0.7876
```



2. Using a pre-trained BERT model:

BERT, like the previous "build from scratch" model, does not include a Transformer decoder. It only includes the encoder part with a classifier layer on top. What makes it different is the use of a pre-trained nlp model from huggingface (i.e., "nlptown/bert-base-multilingual-uncased-sentiment"). Applying this model to the twitter dataset results in about a 78.55% accuracy on the training/validation set. [The code snippet and results are indicated below.]

```
model2.summary()
```

```
Model: "tf_bert_for_sequence_classification"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  167356416

 dropout_45 (Dropout)        multiple                  0

 classifier (Dense)          multiple                  3845

=================================================================
Total params: 167,360,261
Trainable params: 167,360,261
Non-trainable params: 0
_________________________________________________________________
```

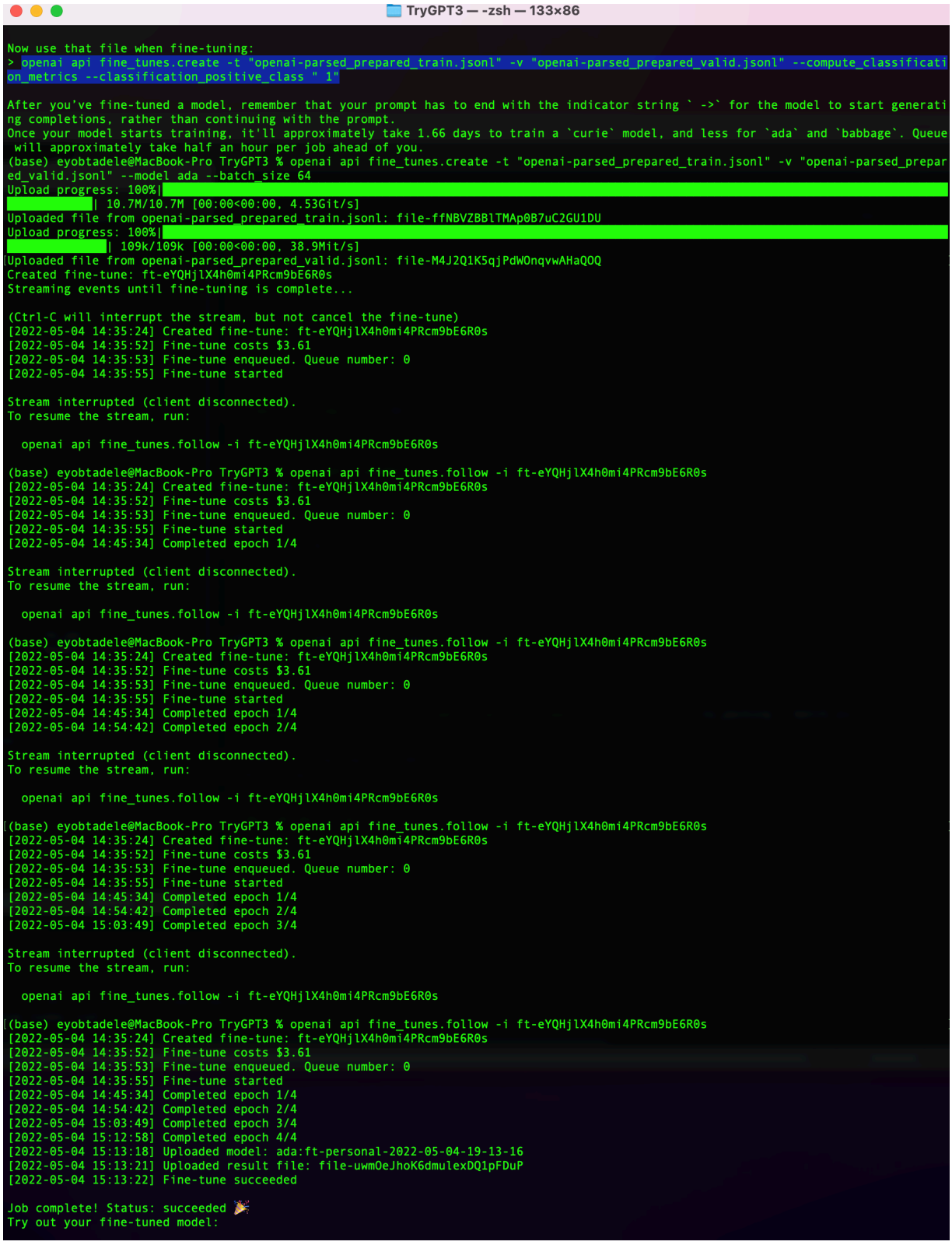
```
model2.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=2e-5, epsilon=1e-08, clipnorm=1.0),
               loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
               metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')])
history2 = model2.fit(train_tweets,
                      train_labels,
                      epochs=2,
                      validation_data=(val_tweets, val_labels))
```

```
Epoch 1/2
2022-05-05 09:27:17.959982: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5000/5000 [==============================] - ETA: 0s - loss: 0.6646 - accuracy: 0.5671
2022-05-05 12:07:43.940173: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is enabled.
5000/5000 [==============================] - 10239s 2s/step - loss: 0.6646 - accuracy: 0.5671 - val_loss: 0.5237 - val_accuracy: 0.7447
Epoch 2/2
5000/5000 [==============================] - 12064s 2s/step - loss: 0.4892 - accuracy: 0.7681 - val_loss: 0.4720 - val_accuracy: 0.7850
```

3. Using third generation Generative Pre-Trained Transformer (GPT-3):

GPT-3 from OpenAI is trained on 175 billion parameters, making it 470 times bigger in size than BERT-Large. Unlike BERT, which requires an elaborated fine-tuning and gathering a large dataset to train, GPT-3 provides a "text-in and text-out" API allowing users to reprogram it using instructions and access it. GPT-3 uses a few-shot learning process on the input token to predict the output result.

In order to make use of GPT-3, the dataset needs to be converted to a JSON format. Additionally, for this specific sentiment analysis problem a specialized model called "Ada" is used. It is the fastest model and can perform tasks like parsing text, address correction and certain kinds of classification tasks that do not require too much nuance. After applying the model to the dataset, an average validation accuracy of 86% is achieved.


TryGPT3 — -zsh — 133×86
Now use that file when fine-tuning:
> openai api fine_tunes.create -t "openai-parsed_prepared_train.jsonl" -v "openai-parsed_prepared_valid.jsonl" --compute_classificati
on_metrics --classification_positive_class " 1"
After you've fine-tuned a model, remember that your prompt has to end with the indicator string ` ->` for the model to start generati
ng completions, rather than continuing with the prompt.
Once your model starts training, it'll approximately take 1.66 days to train a `curie` model, and less for `ada` and `babbage`. Queue
will approximately take half an hour per job ahead of you.
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.create -t "openai-parsed_prepared_train.jsonl" -v "openai-parsed_prepar
ed_valid.jsonl" --model ada --batch_size 64
Upload progress: 100%| 10.7M/10.7M [00:00<00:00, 4.53Git/s]
Uploaded file from openai-parsed_prepared_train.jsonl: file-ffNBVZBBlTMAp0B7uC2GU1DU
Upload progress: 100%| 109k/109k [00:00<00:00, 38.9Mit/s]
Uploaded file from openai-parsed_prepared_valid.jsonl: file-M4J2Q1K5qjPdWOnqvwAHaQOQ
Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
Streaming events until fine-tuning is complete...
(Ctrl-C will interrupt the stream, but not cancel the fine-tune)
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
[2022-05-04 15:03:49] Completed epoch 3/4
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
[2022-05-04 15:03:49] Completed epoch 3/4
[2022-05-04 15:12:58] Completed epoch 4/4
[2022-05-04 15:13:18] Uploaded model: ada:ft-personal-2022-05-04-19-13-16
[2022-05-04 15:13:21] Uploaded result file: file-uwmOeJhoK6dmulexDQ1pFDuP
[2022-05-04 15:13:22] Fine-tune succeeded
Job complete! Status: succeeded 🎉
Try out your fine-tuned model:

Discussion of Results

As observed form the runs of the models in the previous section, each one gives a different accuracy result on the test/validation set. The transformer model built from scratch using tensorflow and keras functionalities reaches a test/validation accuracy of about 78.97%. Similarly, the pre-trained BERT model also achieves a test/validation accuracy of 78.55%. This is somewhat expected as the architectures of the two are pretty similar (i.e., consisting of only an encoder). However, since the BERT model is already pre-trained, we would have expected a slightly better accuracy score. This might indicate further combinations of hyperparameter tuning needs to be done to fully utilize the pre-trained capabilities.

When it comes to OpenAI's GPT-3, we see a clear improvement in accuracy, even with its fastest model called "Ada". Average accuracy score on test/validation set is about 86%. The downside with GPT-3 is the fact that it is not open source and gives little to no flexibility to modify or tune the hyperparameters. It is therefore important to discuss the differences between BERT and GPT-3 in terms of architecture, size, and learning approach to understand the difference in performance.

Architecture: In terms of transformer building blocks, BERT is built on encoder blocks; while GPT-3 is on decoder block. BERT is trained on masked language model and next sentence prediction, while GPT-3 predicts the next word. BERT uses self attention and GPT-3 uses masked self attention.

Size: Looking at number of parameters and training data, BERT has 340 million and 2.5 billion tokens from Wikipedia respectively. As for GPT-3, its largest model has 175 billion parameters and trained on 499 billion tokens.

Learning Approach: GPT-3 has a single model for all downstream tasks and does not require fine-tuning. It also learns from examples through zero shot, one shot, or few shot approach. In the case of BERT, it has pre-trained models for different downstream NLP tasks which are further fine-tuned on custom data.

It is fair to assume here that the performance of a pre-trained BERT model may perform a little better if a series of hyper-parameter tuning steps are taken. Similarly, the use of bigger GPT-3 models like "Curie" might also yield an even better accuracy, but on the same token, no hyper-parameter tuning can be done on it.

Conclusion

Sentiment analysis is a relatively well studied problem with different types of datasets, even Twitter. However, with the advent of different neural network architectures, approaches to problems and performance of models has changed. This specific project attempted to look into three approaches, namely, build from scratch Transformer, pre-trained BERT, and GPT-3.

The performance similarities in terms of accuracy between pre-trained BERT and built-from-scratch Transformer can be attributed to the fact that both have similar architecture (i.e. using Encoder only). However, it is fair to assume that a pre-trained BERT model should have a slight edge over the other one. This might be improved by adjusting the number of epochs, changing the value of the learning rate, as well as freezing/unfreezing the weights. This could potentially address the level of accuracy, but similar adjustments have to be made with the other model in order for it to be directly comparable.

With respect to pre-trained BERT and GPT-3, there are significant differences in architecture, size, and learning approach as discussed earlier in this paper. Hence, the difference in accuracy. However, in order for these two models to be truly comparable, GPT-3 needs to be open source where hyper-parameter tuning is made possible. Unfortunately, this is not possible. It still does not hide the fact that GPT-3 is a very powerful architecture even with its fastest "Ada" model. The sheer size of its trainable parameters is almost 10 times more than any of the previous models out there. With GPT-3 many of the NLP tasks can be done without any fine-tuning or parameter updates. It can perform tasks with little, to no examples (i.e., referred to as shots). Although its performance is incredible in various NLP tasks, it is not suitable if we are looking to understand its basic mechanics under the hood.

The different architectures explored here attempted to shed some light into how they differ in performance. Although not directly comparable, some level of inferences can be made. The choice of a model can depend on the nature of the problem, what a specific model is good at, or whether transparency is important in explaining the model. In academic settings, it would be ideal to have an open source model where tuning can be done in various experimental situations. However, real-world problems might dictate using an off-the-shelf API based model that is also high on accuracy. The field of deep learning is constantly changing and there are, and will be, new architectures and models that improve on current advances. Further exploration into these new trends is an interesting area of discovery.

Reference

A detailed run of all the code is indicated below as reference for more information.

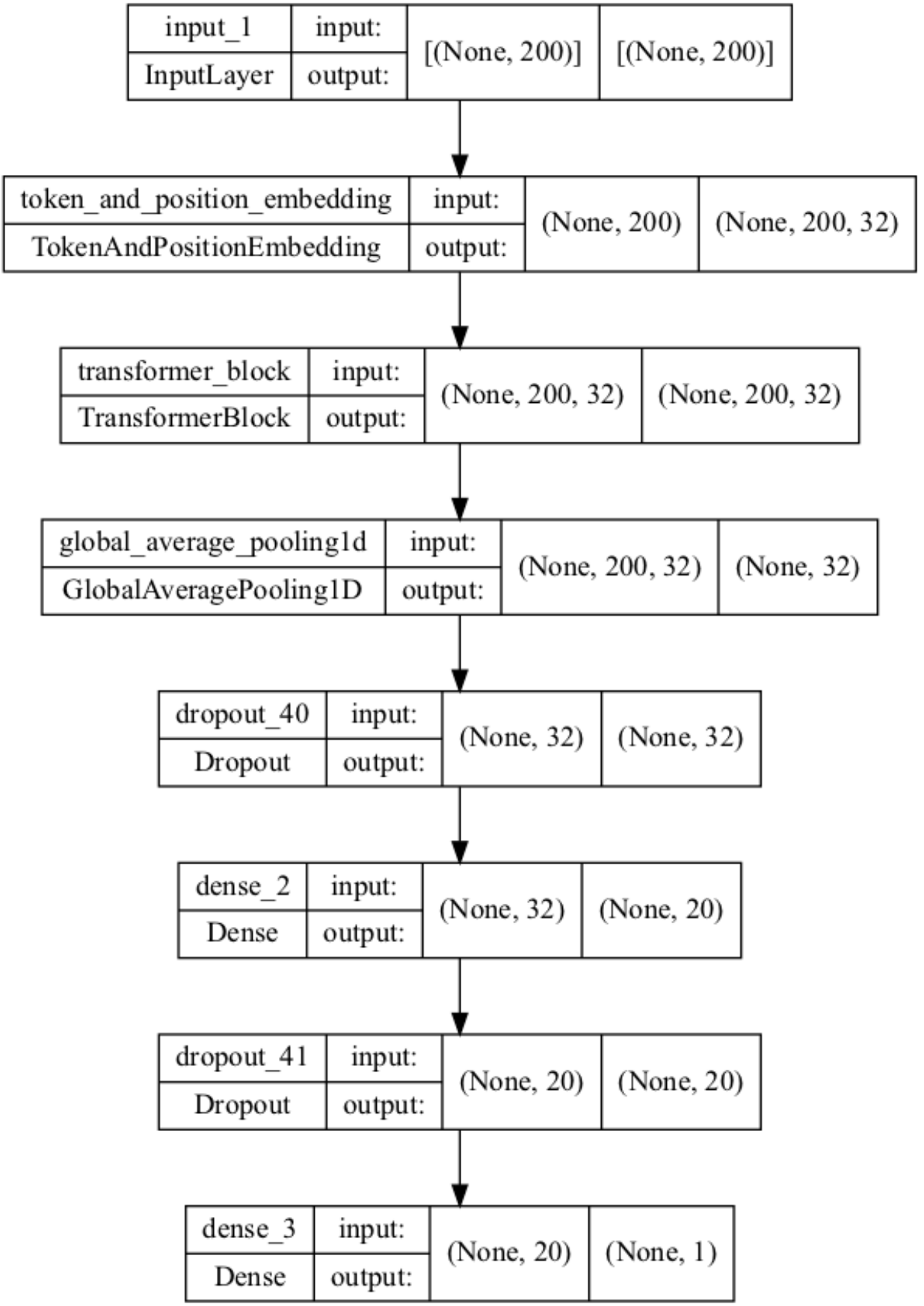
```
df.shape =  (200000, 2)
label distribution :
0.0    100000
1.0    100000
Name: label, dtype: int64
   label                                           sentence
0    0.0  @switchfoot http://twitpic.com/2y1zl
(http://twitpic.com/2y1zl) - Awww, t...
1    0.0  is upset that he can't update his Facebook by ...
2    0.0  @Kenichan I dived many times for the ball. Man...
3    0.0    my whole body feels itchy and like its on fire
4    0.0  @nationwideclass no, it's not behaving at all....
        label                                           sentence
199995    1.0  @jvdouglas  haha, no, the remark on maternity ...
199996    1.0                                @altitis and to you!
199997    1.0  Okie doke!! Time for me to escape for the Nort...
199998    1.0                     finished the lessons, hooray!
199999    1.0  Some ppl are just fucking KP0. Cb ! Stop askin...
```

Out[7]: (200000, 2)

```
   label                                           sentence
0    0.0       awww thats a bummer  you shoulda got davi...
1    0.0  is upset that he cant update his facebook by t...
2    0.0    i dived many times for the ball managed to s...
3    0.0    my whole body feels itchy and like its on fire
4    0.0    no its not behaving at all im mad why am i h...
        label                                           sentence
199995    1.0     haha no the remark on maternity leave fired...
199996    1.0                                      and to you
199997    1.0  okie doke time for me to escape for the north ...
199998    1.0                        finished the lessons hooray
199999    1.0  some ppl are just fucking kp cb  stop asking m...
```

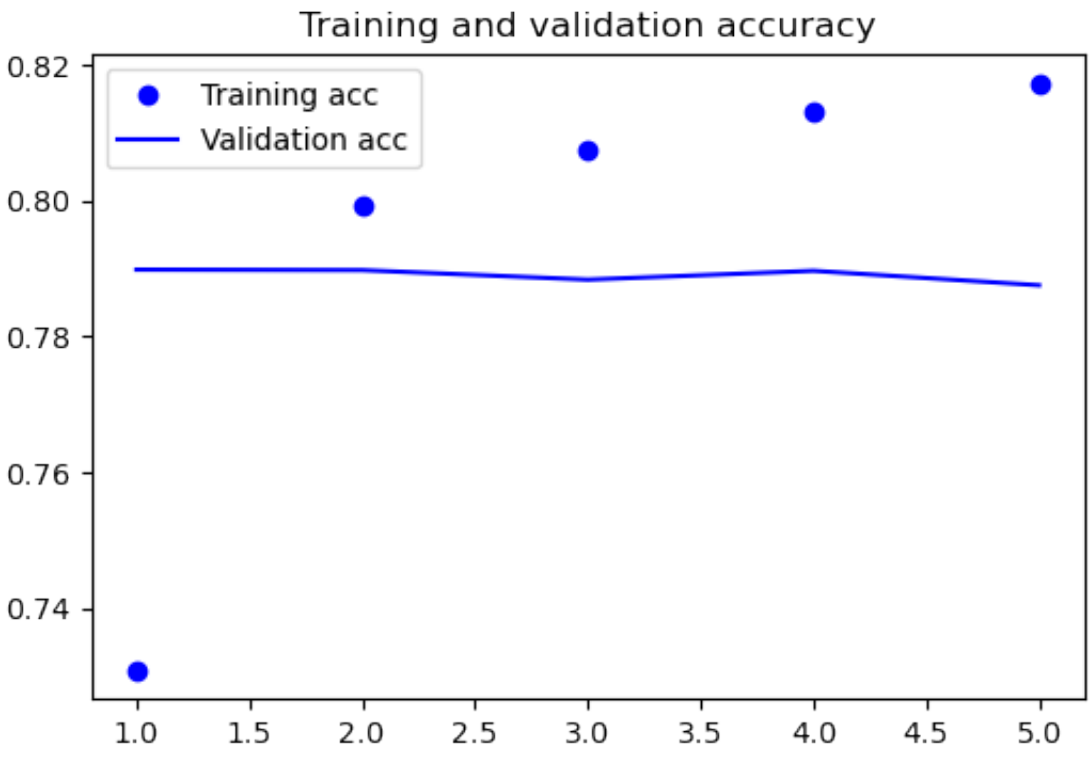
```
(160000, 200)
(160000,)
(40000, 200)
(40000,)

Total Vocabulary Size (Untrimmed): 86341
Vocabulary Size (trimmed): 10000
```

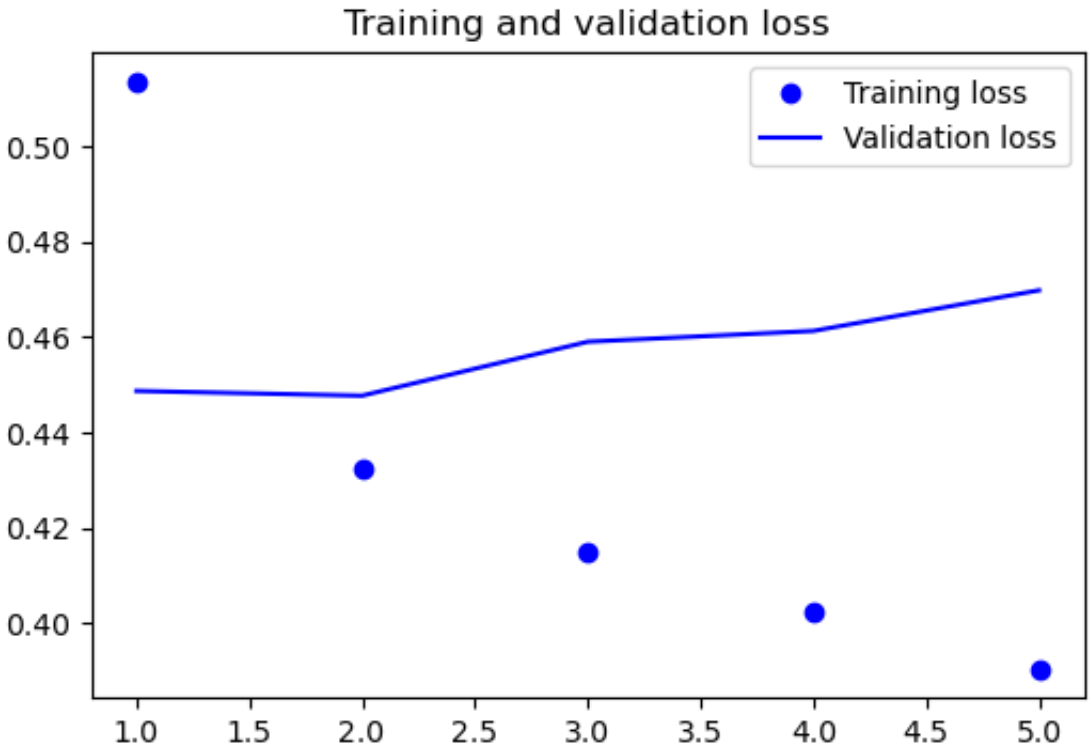
Out[17]:


input_1
InputLayer
input:
output:
[(None, 200)]
[(None, 200)]
token_and_position_embedding
TokenAndPositionEmbedding
input:
output:
(None, 200)
(None, 200, 32)
transformer_block
TransformerBlock
input:
output:
(None, 200, 32)
(None, 200, 32)
global_average_pooling1d
GlobalAveragePooling1D
input:
output:
(None, 200, 32)
(None, 32)
dropout_40
Dropout
input:
output:
(None, 32)
(None, 32)
dense_2
Dense
input:
output:
(None, 32)
(None, 20)
dropout_41
Dropout
input:
output:
(None, 20)
(None, 20)
dense_3
Dense
input:
output:
(None, 20)
(None, 1)

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200)]             0

 token_and_position_embeddin  (None, 200, 32)          326400
 g (TokenAndPositionEmbeddin
 g)

 transformer_block (Transfor  (None, 200, 32)          10656
 merBlock)

 global_average_pooling1d (G  (None, 32)               0
 lobalAveragePooling1D)

 dropout_40 (Dropout)        (None, 32)                0

 dense_2 (Dense)             (None, 20)                660

 dropout_41 (Dropout)        (None, 20)                0

 dense_3 (Dense)             (None, 1)                 21

=================================================================
Total params: 337,737
Trainable params: 337,737
Non-trainable params: 0
_________________________________________________________________
```

```
Epoch 1/5

2022-05-02 22:18:42.106358: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.

2500/2500 [==============================] - ETA: 0s - loss: 0.5135 -
accuracy: 0.7310

2022-05-02 22:24:33.331488: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.

2500/2500 [==============================] - 372s 148ms/step - loss:
0.5135 - accuracy: 0.7310 - val_loss: 0.4487 - val_accuracy: 0.7899
Epoch 2/5
2500/2500 [==============================] - 376s 150ms/step - loss:
0.4322 - accuracy: 0.7993 - val_loss: 0.4477 - val_accuracy: 0.7898
Epoch 3/5
2500/2500 [==============================] - 379s 151ms/step - loss:
0.4149 - accuracy: 0.8074 - val_loss: 0.4590 - val_accuracy: 0.7884
Epoch 4/5
2500/2500 [==============================] - 381s 152ms/step - loss:
0.4025 - accuracy: 0.8132 - val_loss: 0.4613 - val_accuracy: 0.7897
Epoch 5/5
2500/2500 [==============================] - 379s 152ms/step - loss:
0.3904 - accuracy: 0.8172 - val_loss: 0.4698 - val_accuracy: 0.7876
```


Training and validation accuracy
Training acc
Validation acc
0.82
0.80
0.78
0.76
0.74
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Training and validation loss
Training loss
Validation loss
0.50
0.48
0.46
0.44
0.42
0.40
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

```
2022-05-02 22:57:30.694332: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.
```

Out[31]:
```
array([[0.92811018, 0.0718898 ]])
```

Out[32]:
```
0.0
```

```
Requirement already satisfied: transformers in /Users/eyobtadele/Docu
ments/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (4.18.0)
Requirement already satisfied: numpy>=1.17 in /Users/eyobtadele/Docum
ents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transformer
s) (1.21.2)
Requirement already satisfied: tqdm>=4.27 in /Users/eyobtadele/Docume
nts/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transformers
) (4.62.3)
Requirement already satisfied: pyyaml>=5.1 in /Users/eyobtadele/Docum
ents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transformer
s) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /Users/eyobtadele
/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from trans
formers) (2022.3.15)
Requirement already satisfied: sacremoses in /Users/eyobtadele/Docume
nts/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transformers
) (0.0.49)
Requirement already satisfied: filelock in /Users/eyobtadele/Document
s/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transformers)
(3.6.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /Users/
eyobtadele/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (
from transformers) (0.5.1)
Requirement already satisfied: packaging>=20.0 in /Users/eyobtadele/D
ocuments/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transfo
rmers) (21.3)
Requirement already satisfied: requests in /Users/eyobtadele/Document
s/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from transformers)
(2.27.1)
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in /
Users/eyobtadele/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-pack
ages (from transformers) (0.12.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /Users/e
yobtadele/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (f
rom huggingface-hub<1.0,>=0.1.0->transformers) (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/eyo
btadele/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (fro
m packaging>=20.0->transformers) (3.0.4)
```

```
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/eyobta
dele/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from r
equests->transformers) (1.26.8)
Requirement already satisfied: certifi>=2017.4.17 in /Users/eyobtadel
e/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from requ
ests->transformers) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /Users/eyobtadele/Docu
ments/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from requests->
transformers) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /Users/ey
obtadele/Documents/KSU-MSBA/AdvML/env/lib/python3.8/site-packages (fr
om requests->transformers) (2.0.12)
Requirement already satisfied: six in /Users/eyobtadele/Documents/KSU
-MSBA/AdvML/env/lib/python3.8/site-packages (from sacremoses->transfo
rmers) (1.15.0)
Requirement already satisfied: click in /Users/eyobtadele/Documents/K
SU-MSBA/AdvML/env/lib/python3.8/site-packages (from sacremoses->trans
formers) (8.1.2)
Requirement already satisfied: joblib in /Users/eyobtadele/Documents/
KSU-MSBA/AdvML/env/lib/python3.8/site-packages (from sacremoses->tran
sformers) (1.1.0)
Note: you may need to restart the kernel to use updated packages.


All model checkpoint layers were used when initializing TFBertForSequ
enceClassification.

Some layers of TFBertForSequenceClassification were not initialized f
rom the model checkpoint at bert-base-uncased and are newly initializ
ed: ['classifier']
You should probably TRAIN this model on a down-stream task to be able
to use it for predictions and inference.


Model: "tf_bert_for_sequence_classification_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

 dropout_79 (Dropout)        multiple                  0

 classifier (Dense)          multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
```

```
2022-05-02 23:26:34.122057: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.

5000/5000 [==============================] - ETA: 0s - loss: 0.6954 -
accuracy: 0.5002

2022-05-03 02:03:00.192030: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.

5000/5000 [==============================] - 9999s 2s/step - loss: 0.
6954 - accuracy: 0.5002 - val_loss: 0.6940 - val_accuracy: 0.4998


Downloading:   0%|          | 0.00/639M [00:00<?, ?B/s]

Some layers from the model checkpoint at nlptown/bert-base-multilingu
al-uncased-sentiment were not used when initializing TFBertForSequenc
eClassification: ['dropout_37']
- This IS expected if you are initializing TFBertForSequenceClassific
ation from the checkpoint of a model trained on another task or with
another architecture (e.g. initializing a BertForSequenceClassificati
on model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertForSequenceClass
ification from the checkpoint of a model that you expect to be exactl
y identical (initializing a BertForSequenceClassification model from
a BertForSequenceClassification model).
All the layers of TFBertForSequenceClassification were initialized fr
om the model checkpoint at nlptown/bert-base-multilingual-uncased-sen
timent.
If your task is similar to the task the model of the checkpoint was t
rained on, you can already use TFBertForSequenceClassification for pr
edictions without further training.
```

```
Model: "tf_bert_for_sequence_classification_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  167356416

 dropout_117 (Dropout)       multiple                  0

 classifier (Dense)          multiple                  3845

=================================================================
Total params: 167,360,261
Trainable params: 167,360,261
Non-trainable params: 0
_________________________________________________________________
```

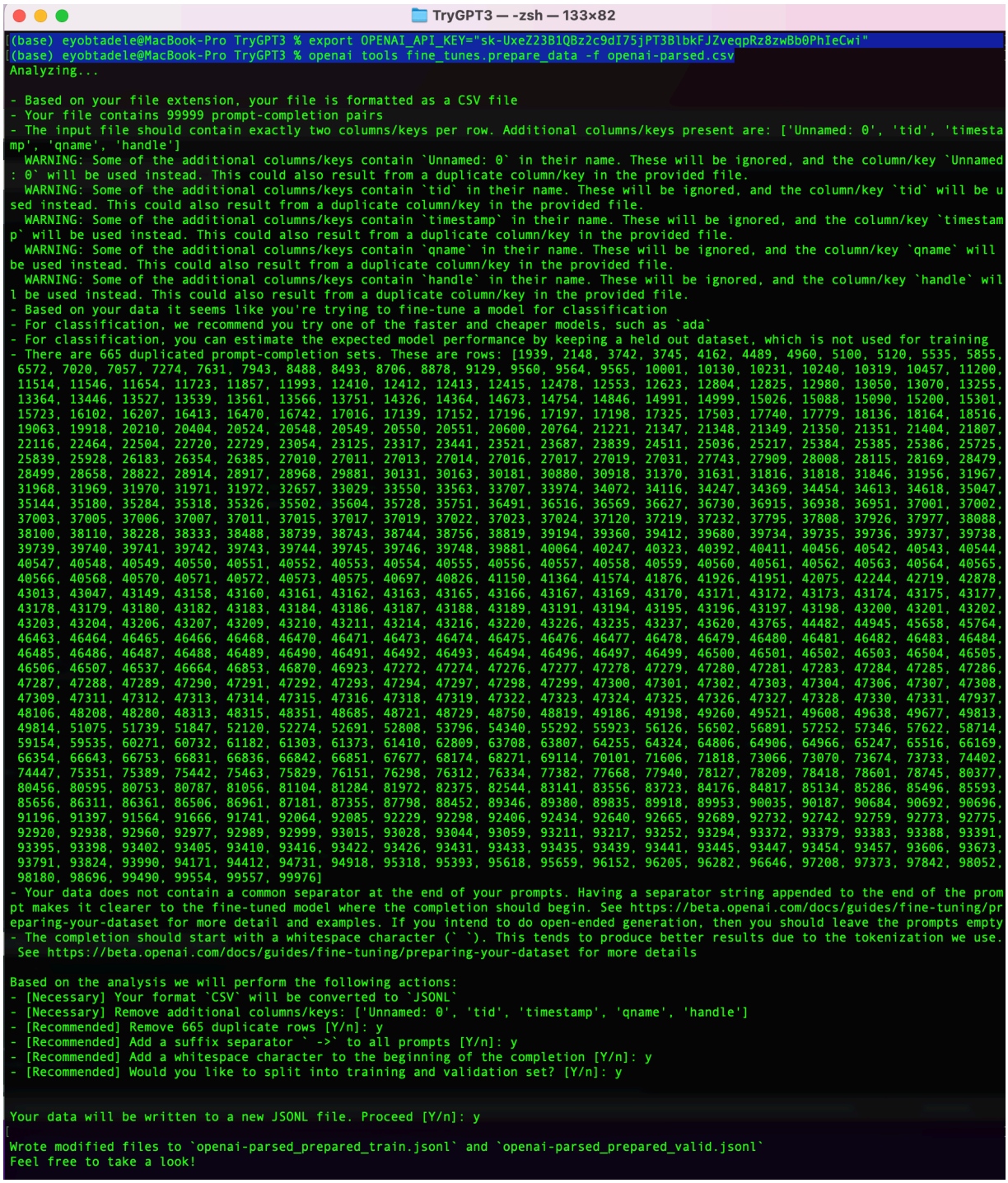
```
Epoch 1/2
```

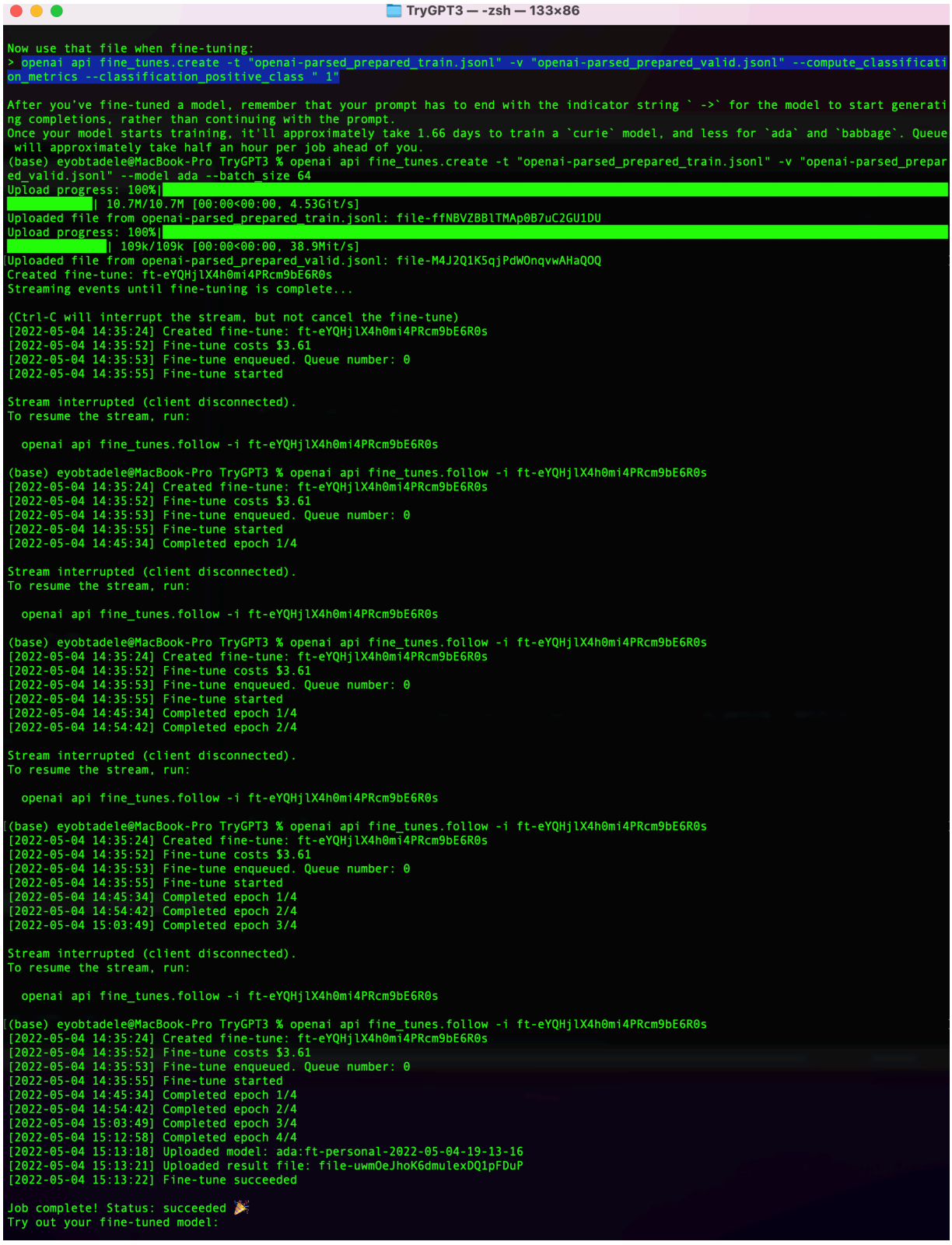
```
2022-05-03 10:29:14.532077: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.
```

```
5000/5000 [==============================] - ETA: 0s - loss: 0.4634 -
accuracy: 0.7844
```

```
2022-05-03 13:27:28.304846: I tensorflow/core/grappler/optimizers/cus
tom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type
GPU is enabled.
```

```
5000/5000 [==============================] - 12292s 2s/step - loss: 0
.4634 - accuracy: 0.7844 - val_loss: 0.4645 - val_accuracy: 0.7830
Epoch 2/2
5000/5000 [==============================] - 21831s 4s/step - loss: 0
.4420 - accuracy: 0.8009 - val_loss: 0.4661 - val_accuracy: 0.7855
```


TryGPT3 — -zsh — 133×82
(base) eyobtadele@MacBook-Pro TryGPT3 % export OPENAI_API_KEY="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"
(base) eyobtadele@MacBook-Pro TryGPT3 % openai tools fine_tunes.prepare_data -f openai-parsed.csv
Analyzing...
- Based on your file extension, your file is formatted as a CSV file
- Your file contains 99999 prompt-completion pairs
- The input file should contain exactly two columns/keys per row. Additional columns/keys present are: ['Unnamed: 0', 'tid', 'timestamp', 'qname', 'handle']
WARNING: Some of the additional columns/keys contain `Unnamed: 0` in their name. These will be ignored, and the column/key `Unnamed: 0` will be used instead. This could also result from a duplicate column/key in the provided file.
WARNING: Some of the additional columns/keys contain `tid` in their name. These will be ignored, and the column/key `tid` will be used instead. This could also result from a duplicate column/key in the provided file.
WARNING: Some of the additional columns/keys contain `timestamp` in their name. These will be ignored, and the column/key `timestamp` will be used instead. This could also result from a duplicate column/key in the provided file.
WARNING: Some of the additional columns/keys contain `qname` in their name. These will be ignored, and the column/key `qname` will be used instead. This could also result from a duplicate column/key in the provided file.
WARNING: Some of the additional columns/keys contain `handle` in their name. These will be ignored, and the column/key `handle` will be used instead. This could also result from a duplicate column/key in the provided file.
- Based on your data it seems like you're trying to fine-tune a model for classification
- For classification, we recommend you try one of the faster and cheaper models, such as `ada`
- For classification, you can estimate the expected model performance by keeping a held out dataset, which is not used for training
- There are 665 duplicated prompt-completion sets. These are rows: [1939, 2148, 3742, 3745, 4162, 4489, 4960, 5100, 5120, 5535, 5855, 6572, 7020, 7057, 7274, 7631, 7943, 8488, 8493, 8706, 8878, 9129, 9560, 9564, 9565, 10001, 10130, 10231, 10240, 10319, 10457, 11200, 11514, 11546, 11654, 11723, 11857, 11993, 12410, 12412, 12413, 12415, 12478, 12553, 12623, 12804, 12825, 12980, 13050, 13070, 13255, 13364, 13446, 13527, 13539, 13561, 13566, 13751, 14326, 14364, 14673, 14754, 14846, 14991, 14999, 15026, 15088, 15090, 15200, 15301, 15723, 16102, 16207, 16413, 16470, 16742, 17016, 17139, 17152, 17196, 17197, 17198, 17325, 17503, 17740, 17779, 18136, 18164, 18516, 19063, 19918, 20210, 20404, 20524, 20548, 20549, 20550, 20551, 20600, 20764, 21221, 21347, 21348, 21349, 21350, 21351, 21404, 21807, 22116, 22464, 22504, 22720, 22729, 23054, 23125, 23317, 23441, 23521, 23687, 23839, 24511, 25036, 25217, 25384, 25385, 25386, 25725, 25839, 25928, 26183, 26354, 26385, 27010, 27011, 27013, 27014, 27016, 27017, 27019, 27031, 27743, 27909, 28008, 28115, 28169, 28479, 28499, 28658, 28822, 28914, 28917, 28968, 29881, 30131, 30163, 30181, 30880, 30918, 31370, 31631, 31816, 31818, 31846, 31956, 31967, 31968, 31969, 31970, 31971, 31972, 32657, 33029, 33550, 33563, 33707, 33974, 34072, 34116, 34247, 34369, 34454, 34613, 34618, 35047, 35144, 35180, 35284, 35318, 35326, 35502, 35604, 35728, 35751, 36491, 36516, 36569, 36627, 36730, 36915, 36938, 36951, 37001, 37002, 37003, 37005, 37006, 37007, 37011, 37015, 37017, 37019, 37022, 37023, 37024, 37120, 37219, 37232, 37795, 37808, 37926, 37977, 38088, 38100, 38110, 38228, 38333, 38488, 38739, 38743, 38744, 38756, 38819, 39194, 39360, 39412, 39680, 39734, 39735, 39736, 39737, 39738, 39739, 39740, 39741, 39742, 39743, 39744, 39745, 39746, 39748, 39881, 40064, 40247, 40323, 40392, 40411, 40456, 40542, 40543, 40544, 40547, 40548, 40549, 40550, 40551, 40552, 40553, 40554, 40555, 40556, 40557, 40558, 40559, 40560, 40561, 40562, 40563, 40564, 40565, 40566, 40568, 40570, 40571, 40572, 40573, 40575, 40697, 40826, 41150, 41364, 41574, 41876, 41926, 41951, 42075, 42244, 42719, 42878, 43013, 43047, 43149, 43158, 43160, 43161, 43162, 43163, 43165, 43166, 43167, 43169, 43170, 43171, 43172, 43173, 43174, 43175, 43177, 43178, 43179, 43180, 43182, 43183, 43184, 43186, 43187, 43188, 43189, 43191, 43194, 43195, 43196, 43197, 43198, 43200, 43201, 43202, 43203, 43204, 43206, 43207, 43209, 43210, 43211, 43214, 43216, 43220, 43226, 43235, 43237, 43620, 43765, 44482, 44945, 45658, 45764, 46463, 46464, 46465, 46466, 46468, 46470, 46471, 46473, 46474, 46475, 46476, 46477, 46478, 46479, 46480, 46481, 46482, 46483, 46484, 46485, 46486, 46487, 46488, 46489, 46490, 46491, 46492, 46493, 46494, 46496, 46497, 46499, 46500, 46501, 46502, 46503, 46504, 46505, 46506, 46507, 46537, 46664, 46853, 46870, 46923, 47272, 47274, 47276, 47277, 47278, 47279, 47280, 47281, 47283, 47284, 47285, 47286, 47287, 47288, 47289, 47290, 47291, 47292, 47293, 47294, 47297, 47298, 47299, 47300, 47301, 47302, 47303, 47304, 47306, 47307, 47308, 47309, 47311, 47312, 47313, 47314, 47315, 47316, 47318, 47319, 47322, 47323, 47324, 47325, 47326, 47327, 47328, 47330, 47331, 47937, 48106, 48208, 48280, 48313, 48315, 48351, 48685, 48721, 48729, 48750, 48819, 49186, 49198, 49260, 49521, 49608, 49638, 49677, 49813, 49814, 51075, 51739, 51847, 52120, 52274, 52691, 52808, 53796, 54340, 55292, 55923, 56126, 56502, 56891, 57252, 57346, 57622, 58714, 59154, 59535, 60271, 60732, 61182, 61303, 61373, 61410, 62809, 63708, 63807, 64255, 64324, 64806, 64906, 64966, 65247, 65516, 66169, 66354, 66643, 66753, 66831, 66836, 66842, 66851, 67677, 68174, 68271, 69114, 70101, 71606, 71818, 73066, 73070, 73674, 73733, 74402, 74447, 75351, 75389, 75442, 75463, 75829, 76151, 76298, 76312, 76334, 77382, 77668, 77940, 78127, 78209, 78418, 78601, 78745, 80377, 80456, 80595, 80753, 80787, 81056, 81104, 81284, 81972, 82375, 82544, 83141, 83556, 83723, 84176, 84817, 85134, 85286, 85496, 85593, 85656, 86311, 86361, 86506, 86961, 87181, 87355, 87798, 88452, 89346, 89380, 89835, 89918, 89953, 90035, 90187, 90684, 90692, 90696, 91196, 91397, 91564, 91666, 91741, 92064, 92085, 92229, 92298, 92406, 92434, 92640, 92665, 92689, 92732, 92742, 92759, 92773, 92775, 92920, 92938, 92960, 92977, 92989, 92999, 93015, 93028, 93044, 93059, 93211, 93217, 93252, 93294, 93372, 93379, 93383, 93388, 93391, 93395, 93398, 93402, 93405, 93410, 93416, 93422, 93426, 93431, 93433, 93435, 93439, 93441, 93445, 93447, 93454, 93457, 93606, 93673, 93791, 93824, 93990, 94171, 94412, 94731, 94918, 95318, 95393, 95618, 95659, 96152, 96205, 96282, 96646, 97208, 97373, 97842, 98052, 98180, 98696, 99490, 99554, 99557, 99976]
- Your data does not contain a common separator at the end of your prompts. Having a separator string appended to the end of the prompt makes it clearer to the fine-tuned model where the completion should begin. See https://beta.openai.com/docs/guides/fine-tuning/preparing-your-dataset for more detail and examples. If you intend to do open-ended generation, then you should leave the prompts empty
- The completion should start with a whitespace character (` `). This tends to produce better results due to the tokenization we use. See https://beta.openai.com/docs/guides/fine-tuning/preparing-your-dataset for more details
Based on the analysis we will perform the following actions:
- [Necessary] Your format `CSV` will be converted to `JSONL`
- [Necessary] Remove additional columns/keys: ['Unnamed: 0', 'tid', 'timestamp', 'qname', 'handle']
- [Recommended] Remove 665 duplicate rows [Y/n]: y
- [Recommended] Add a suffix separator ` ->` to all prompts [Y/n]: y
- [Recommended] Add a whitespace character to the beginning of the completion [Y/n]: y
- [Recommended] Would you like to split into training and validation set? [Y/n]: y
Your data will be written to a new JSONL file. Proceed [Y/n]: y
Wrote modified files to `openai-parsed_prepared_train.jsonl` and `openai-parsed_prepared_valid.jsonl`
Feel free to take a look!


TryGPT3 — -zsh — 133×86
Now use that file when fine-tuning:
> openai api fine_tunes.create -t "openai-parsed_prepared_train.jsonl" -v "openai-parsed_prepared_valid.jsonl" --compute_classificati
on_metrics --classification_positive_class " 1"
After you've fine-tuned a model, remember that your prompt has to end with the indicator string ` ->` for the model to start generati
ng completions, rather than continuing with the prompt.
Once your model starts training, it'll approximately take 1.66 days to train a `curie` model, and less for `ada` and `babbage`. Queue
will approximately take half an hour per job ahead of you.
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.create -t "openai-parsed_prepared_train.jsonl" -v "openai-parsed_prepar
ed_valid.jsonl" --model ada --batch_size 64
Upload progress: 100%| 10.7M/10.7M [00:00<00:00, 4.53Git/s]
Uploaded file from openai-parsed_prepared_train.jsonl: file-ffNBVZBBlTMAp0B7uC2GU1DU
Upload progress: 100%| 109k/109k [00:00<00:00, 38.9Mit/s]
Uploaded file from openai-parsed_prepared_valid.jsonl: file-M4J2Q1K5qjPdWOnqvwAHaQOQ
Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
Streaming events until fine-tuning is complete...
(Ctrl-C will interrupt the stream, but not cancel the fine-tune)
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
[2022-05-04 15:03:49] Completed epoch 3/4
Stream interrupted (client disconnected).
To resume the stream, run:
openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
[2022-05-04 15:03:49] Completed epoch 3/4
[2022-05-04 15:12:58] Completed epoch 4/4
[2022-05-04 15:13:18] Uploaded model: ada:ft-personal-2022-05-04-19-13-16
[2022-05-04 15:13:21] Uploaded result file: file-uwmOeJhoK6dmulexDQ1pFDuP
[2022-05-04 15:13:22] Fine-tune succeeded
Job complete! Status: succeeded 🎉
Try out your fine-tuned model:

```
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.follow -i ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:24] Created fine-tune: ft-eYQHjlX4h0mi4PRcm9bE6R0s
[2022-05-04 14:35:52] Fine-tune costs $3.61
[2022-05-04 14:35:53] Fine-tune enqueued. Queue number: 0
[2022-05-04 14:35:55] Fine-tune started
[2022-05-04 14:45:34] Completed epoch 1/4
[2022-05-04 14:54:42] Completed epoch 2/4
[2022-05-04 15:03:49] Completed epoch 3/4
[2022-05-04 15:12:58] Completed epoch 4/4
[2022-05-04 15:13:18] Uploaded model: ada:ft-personal-2022-05-04-19-13-16
[2022-05-04 15:13:21] Uploaded result file: file-uwmOeJhoK6dmulexDQ1pFDuP
[2022-05-04 15:13:22] Fine-tune succeeded

Job complete! Status: succeeded 🎉
Try out your fine-tuned model:

openai api completions.create -m ada:ft-personal-2022-05-04-19-13-16 -p <YOUR_PROMPT>
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api completions.create -m ada:ft-personal-2022-05-04-19-13-16 -p "the people here were
 very rude.. ->" --max-tokens 1
the people here were very rude.. -> 0
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api completions.create -m ada:ft-personal-2022-05-04-19-13-16 -p "i am encouraged by t
he progress. ->" --max-tokens 1
i am encouraged by the progress. -> 1
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.results -i "ft-eYQHjlX4h0mi4PRcm9bE6R0s" > "results.csv"
(base) eyobtadele@MacBook-Pro TryGPT3 % openai api fine_tunes.results --help
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	step	elapsed_tokens	elapsed_examples	training_loss	training_sequence_accuracy	training_token_accuracy	validation_loss	validation_sequence_accuracy	validation_token_accuracy				
2	4009	13048384	256576	0.18773298	0.890625	0.890625	0.170350317	0.953125	0.953125				
3	311	1009088	19904	0.17963863	0.828125	0.828125	0.197630998	0.921875	0.921875				
4	773	2519872	49472	0.19112359	0.828125	0.828125	0.156970202	0.921875	0.921875				
5	1543	5026752	98752	0.13867548	0.796875	0.796875	0.177795416	0.921875	0.921875				
6	2161	7042112	138304	0.16300491	0.875	0.875	0.184751646	0.921875	0.921875				
7	3393	11045952	217152	0.16966161	0.921875	0.921875	0.134814194	0.921875	0.921875				
8	4163	13548736	266432	0.18121659	0.921875	0.921875	0.145397164	0.921875	0.921875				
9	5705	18562112	365120	0.15394555	0.9375	0.9375	0.145342164	0.921875	0.921875				
10	6012	19560704	384768	0.17950227	0.921875	0.921875	0.145881837	0.921875	0.921875				
11	3083	10040000	197312	0.14415187	0.875	0.875	0.154127984	0.90625	0.90625				
12	4470	14553984	286080	0.16624171	0.953125	0.953125	0.186830918	0.90625	0.90625				
13	4779	15556288	305856	0.12355198	0.984375	0.984375	0.172627158	0.90625	0.90625				
14	4934	16051072	315776	0.13968054	0.9375	0.9375	0.205241793	0.90625	0.90625				
15	927	3017152	59328	0.15712677	0.8125	0.8125	0.165019241	0.890625	0.890625				
16	2469	8041792	158016	0.15010291	0.828125	0.828125	0.219244616	0.890625	0.890625				
17	2623	8542144	167872	0.18492304	0.96875	0.96875	0.178773089	0.890625	0.890625				
18	2776	9041408	177664	0.14753309	0.828125	0.828125	0.177019104	0.890625	0.890625				
19	4316	14049536	276224	0.17661818	0.890625	0.890625	0.114538389	0.890625	0.890625				
20	5088	16552448	325632	0.13681377	0.921875	0.921875	0.170675958	0.890625	0.890625				
21	5397	17560384	345408	0.15561994	1	1	0.195330037	0.890625	0.890625				
22	5550	18058112	355200	0.17161597	0.921875	0.921875	0.201254487	0.890625	0.890625				
23	465	1510976	29760	0.17969442	0.9375	0.9375	0.140386241	0.875	0.875				
24	1698	5535872	108672	0.17754977	0.796875	0.796875	0.160083193	0.875	0.875				
25	2930	9542272	187520	0.20178122	0.84375	0.84375	0.197016475	0.875	0.875				
26	4625	15056448	296000	0.18414139	0.875	0.875	0.206577934	0.875	0.875				
27	1388	4525312	88832	0.20140479	0.71875	0.71875	0.149715895	0.859375	0.859375				
28	2315	7541952	148160	0.17207596	0.875	0.875	0.154399038	0.859375	0.859375				
29	3239	10546112	207296	0.14194805	0.921875	0.921875	0.17838075	0.859375	0.859375				
30	3702	12053376	236928	0.16734674	0.875	0.875	0.188269943	0.859375	0.859375				
31	5243	17062592	335552	0.13756787	0.921875	0.921875	0.212106378	0.859375	0.859375				
32	5859	19060928	374976	0.14757052	0.921875	0.921875	0.196457053	0.859375	0.859375				
33	156	505088	9984	0.19539554	0.828125	0.828125	0.187039034	0.84375	0.84375				
34	1235	4020928	79040	0.18732718	0.875	0.875	0.129672618	0.84375	0.84375				
35	2007	6540736	128448	0.15369158	0.828125	0.828125	0.171517081	0.84375	0.84375				
36	3548	11550464	227072	0.16016183	0.890625	0.890625	0.158268607	0.84375	0.84375				
37	1081	3519552	69184	0.18608901	0.8125	0.8125	0.177338958	0.828125	0.828125				
38	3856	12553728	246784	0.1561371	0.890625	0.890625	0.187031695	0.828125	0.828125				
39	1853	6039872	118592	0.19247578	0.859375	0.859375	0.159445875	0.8125	0.8125				
40	619	2014912	39616	0.18361247	0.828125	0.828125	0.226646239	0.796875	0.796875				
41	1	3136	64	0.35218158	0	0	0.312115641	0.15625	0.15625				