Machine Learning Final Project

Doulos Htet

**Introduction**

This machine learning final project focuses on training a chosen dataset from a Google research dataset. The chosen dataset is called “Google Well-formed Query Dataset” and it is a dataset resulted from crowdsourcing well-formedness annotations for 25,100 queries from Paralex corpus. Every query sentence was annotated by five raters with a rating 1/0 of whether or not the query is well-formed, i.e, grammatically correct. There are three datasets given that have been split into train.tsv, test.tsv, and dev.tsv. In the following table, we can see a snippet of the dataset:

|  |  |
| --- | --- |
| Query | Well-formedness rating |
| Which form of government is still in place in greece ? | 1.0 |
| Population of owls just in north america ? | 0.0 |
| Is johnny depp a celtic fan ? | 0.8 |
| Where did Roald Dahl live in his teenaged years ? | 0.6 |

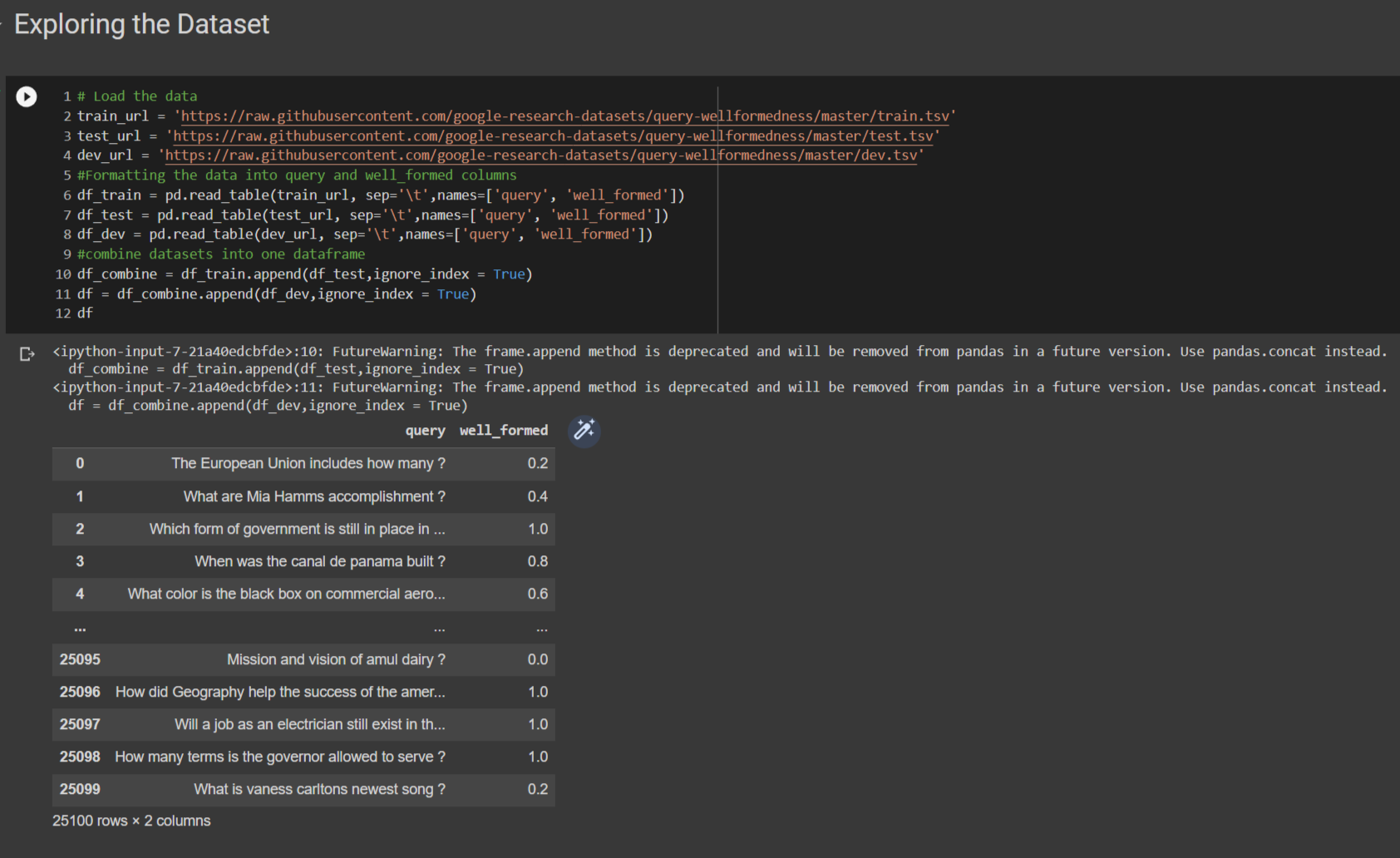
As shown in the snippet, the queries are annotated into well-formed or non-wellformed questions based on three conditions:

1. Query is grammatical.
2. Query is an explicit question.
3. Query does not contain any spelling errors.

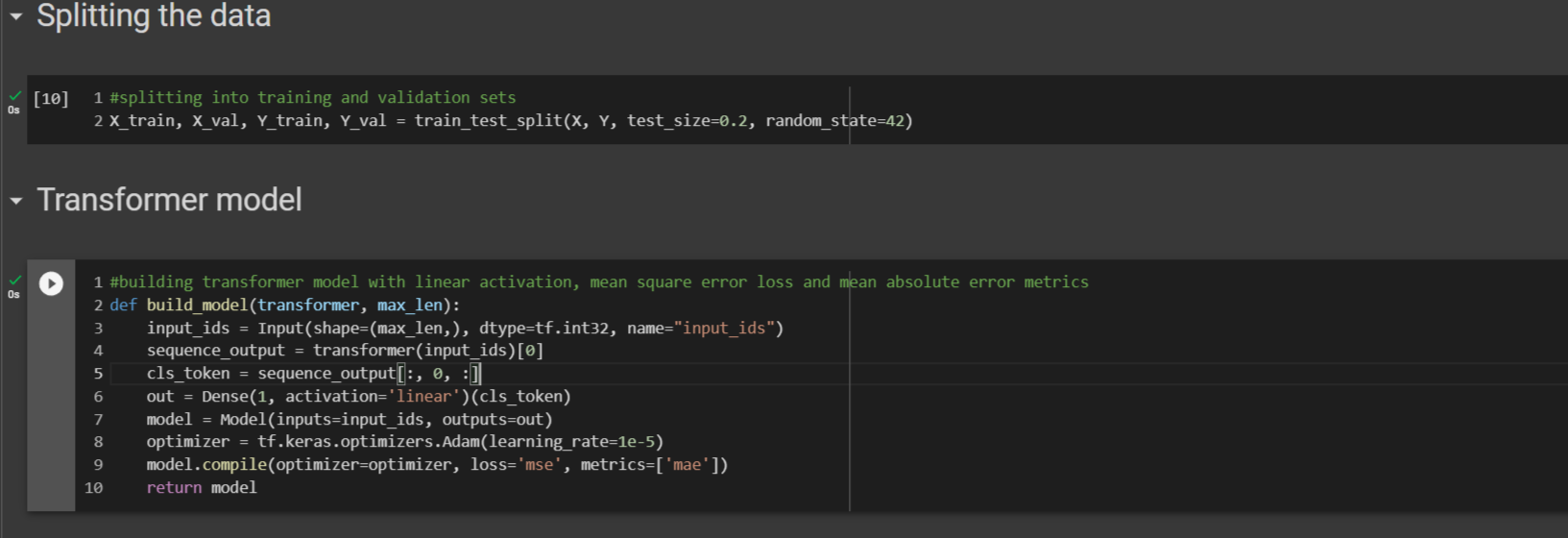
Five annotators reported the probability of whether the query was well-formed or not based on those conditions. To train those datasets, a language model that understands English language, is able to connect the context of the question from the start to end, and produce rating based on how the query is well-formed must be used. Transformer models fit the requirements as they can understand sequence of words in a sentence and a little bit of fine-tuning to the model would yield results with great accuracy.

**Dataset Preparation and Training**

The dataset given was already split into three different sets, however, I combine them all into a pre-processed dataframe and split them again so that I can modify some metrics such as random state and test size. The necessary imports and installation are firstly run which includes pandas, numpy, tensor flow, sciki, and transformers, all of which are important machine learning libraries and tools.



As you can see in the output, the queries rating range from 0.0 to 1.0 with the three conditions mention above to determine the ratings. The total of 25,100 queries are formed in a dataframe of two columns - query and well\_formed. The queries are then tokenized into smaller tokens for our language model to extract them as arrays. Here, I use bert-base-uncased which is a form of transformer trained ona large corpus of English language and is capable of two main functions: Masked language modeling (MLM) and Next sentence prediction (NSP). The first function is capable of predicting masked words in a sentence and the second one could predict the following words in an unfinished sentence. Both functions are very useful in predicting and understanding sentences so bert-base model would output ratings with good accuracy.

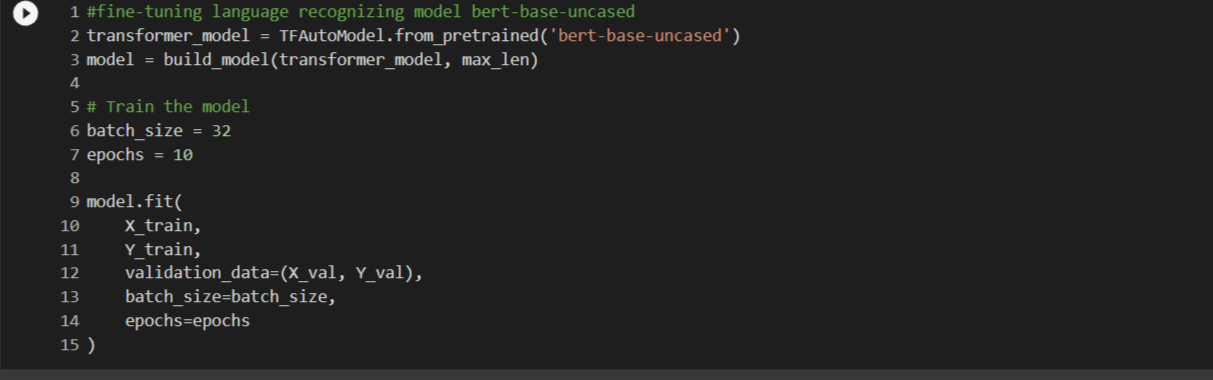


In building the model, important metrics such as optimizers, loss functions and activations, and learning rate are used to train the model with gradual increase in accuracy and decrease in loss. The Adam optimizer with leaning rate of 1e-5 is used combined with mean square error loss and mean absolute error metric. Linear activation function is used to scale inputs in outputs in a linear matter. The batch size used is 32 with 10 epochs to train the dataset. The training took around an hour and with each epoch, the loss is gradually decreasing. With each decrease in loss, the model is becoming accustomed to the training dataset and for the validation set, the validation loss is increasing gradually so I only trained for 10 epochs for the model to not get accustomed to one dataset.

In final epoch, the loss is 0.0168 and validation loss it 0.0723.

loss: 0.0168

val\_loss: 0.0723



**Testing the Model**

In testing new sentences, each one has to be tokenized for the machine to be able to predict the ratings. Here, I tested 4 sentences:

"What were the reasons for everyone to leave the company?",

"tell me way city to the",

"What is the capital of France?",

"what was the reasons for everyone to leave the company"

The first sentence is grammatically correct with coherent and reasonable question. The second sentence does make sense but the order of words and grammatical correctness are disorganized. The third one is also a normal sentence and the final one is the iteration or the first sentence with small letter “what”, “was” and without “?” which is not a well-formed sentence. The scores are as belows:

0.0961

0.0527

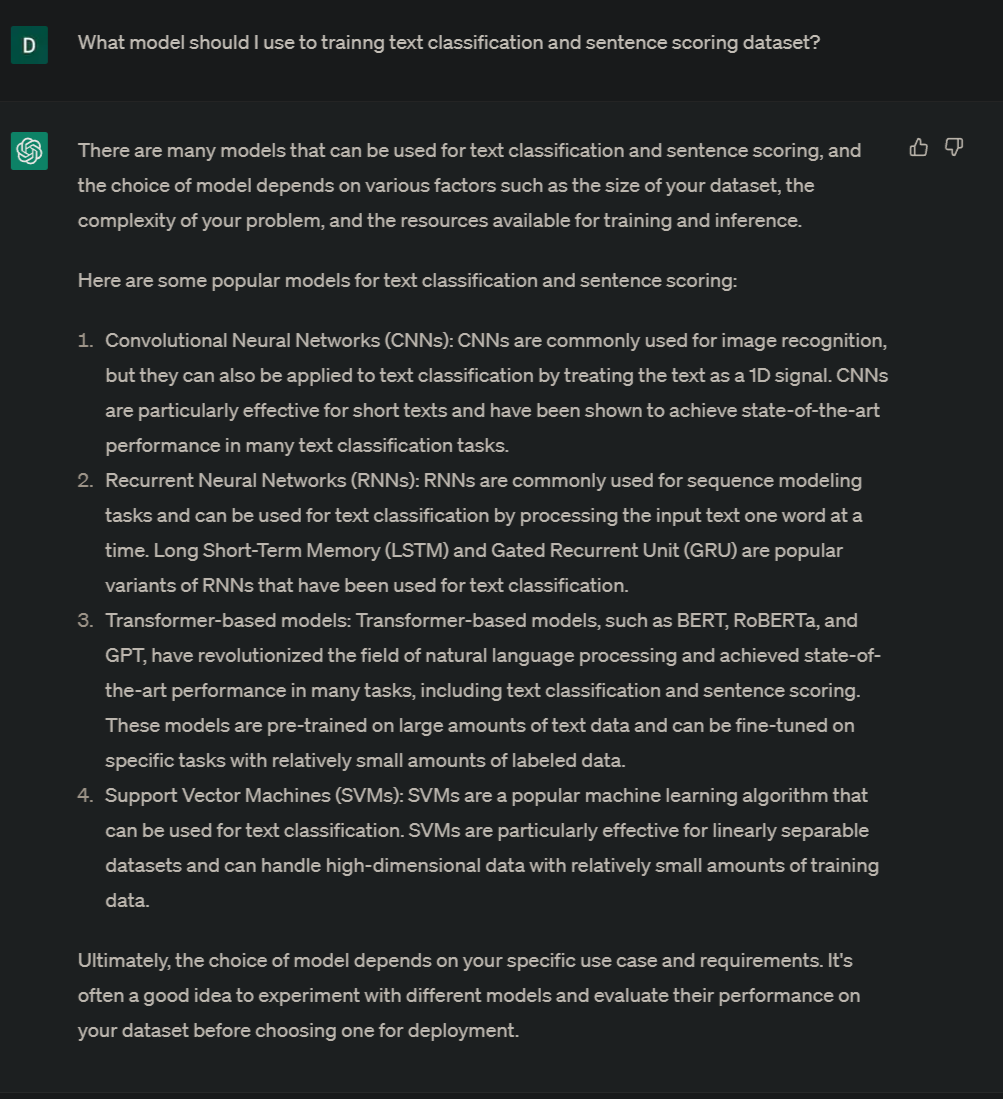
0.9979

0.5177

The first and third sentences are almost perfect and the second one was unique without well-formedness so it gets the lowest score. The fourth sentence is also with errors so it did not receive good ratings as the first and third sentences. Here, the model is able to predict different scores for query results apart from the datasets.

I found another pre-trained model on the same datasets from the hugging face forum and I downloaded the model and tested the same sentences as above.

|  |  |
| --- | --- |
| What were the reasons for everyone to leave the company? | 0.9530 |
| tell me way city to the | 0.0323 |
| What is the capital of France? | 1.0022 |
| what was the reasons for everyone to leave the company | 0.2357 |

Here, the first and third sentences have similar values to our model although the third sentence exceeds the rating of 1. The second sentence has low raings as before but the fourth sentence has even lower rating than our model. The difference is around 0.3 which is not a big of a difference but for a grammatically incorrect sentence, 0.2357 score makes more sense than our model. Although both models have their flaws, I believe they can predict accuracies somewhat close to our five annotators from the dataset. Below is the ChatGPT inquiry 

**Summary**

Sentence classification and prediction models are important functions in Natural Language Processing models. Our fine-tuned model is able to learn from the datasets and predict scores that are reasonable and mostly accurate to our likings. This model building process is accompanied by external helps such as ChatGPT. I asked the ChatGPT about the best way to train a sentence classification and rating score prediction dataset and it told me to use language models, specifically transformers that are capable of understanding contexts in sentences and paragraphs. I also asked the chat bot about which kind of model to use for training and it outputs with several models such as linear regression, binary classification, CNN, RNN. Finally, I also received many debugging results from ChatGPT to fully compile and train the model.

**Code Snippets**

#transformer models from hugging face open source website

!pip install transformers

#import libraries

import pandas as pd

import numpy as np

import tensorflow as tf

from tensorflow.keras.layers import Input, Dense

from tensorflow.keras.models import Model

from transformers import TFAutoModel, AutoTokenizer

from sklearn.model\_selection import train\_test\_split

# Load the data

train\_url = 'https://raw.githubusercontent.com/google-research-datasets/query-wellformedness/master/train.tsv'

test\_url = 'https://raw.githubusercontent.com/google-research-datasets/query-wellformedness/master/test.tsv'

dev\_url = 'https://raw.githubusercontent.com/google-research-datasets/query-wellformedness/master/dev.tsv'

#Formatting the data into query and well\_formed columns

df\_train = pd.read\_table(train\_url, sep='\t',names=['query', 'well\_formed'])

df\_test = pd.read\_table(test\_url, sep='\t',names=['query', 'well\_formed'])

df\_dev = pd.read\_table(dev\_url, sep='\t',names=['query', 'well\_formed'])

#combine datasets into one dataframe

df\_combine = df\_train.append(df\_test,ignore\_index = True)

df = df\_combine.append(df\_dev,ignore\_index = True)

df

# Preprocess the data

tokenizer = AutoTokenizer.from\_pretrained('bert-base-uncased')

max\_len = 128

X = []

for query in df['query']:

    encoded = tokenizer.encode\_plus(

        query,

        add\_special\_tokens=True,

        max\_length=max\_len,

        padding='max\_length',

        truncation=True,

        return\_tensors='tf'

    )

    X.append(encoded['input\_ids'][0])

X = np.array(X)

Y = df['well\_formed'].values

#splitting into training and validation sets

X\_train, X\_val, Y\_train, Y\_val = train\_test\_split(X, Y, test\_size=0.2, random\_state=42)

#building transformer model with linear activation, mean square error loss and mean absolute error metrics

def build\_model(transformer, max\_len):

    input\_ids = Input(shape=(max\_len,), dtype=tf.int32, name="input\_ids")

    sequence\_output = transformer(input\_ids)[0]

    cls\_token = sequence\_output[:, 0, :]

    out = Dense(1, activation='linear')(cls\_token)

    model = Model(inputs=input\_ids, outputs=out)

    optimizer = tf.keras.optimizers.Adam(learning\_rate=1e-5)

    model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

    return model

#fine-tuning language recognizing model bert-base-uncased

transformer\_model = TFAutoModel.from\_pretrained('bert-base-uncased')

model = build\_model(transformer\_model, max\_len)

# Train the model

batch\_size = 32

epochs = 10

model.fit(

    X\_train,

    Y\_train,

    validation\_data=(X\_val, Y\_val),

    batch\_size=batch\_size,

    epochs=epochs

)

queries = ["What were the reasons for everyone to leave the company?", "tell me way city to the", "What is the capital of France?", "what was the reasons for everyone to leave the company"]

results = []

for query in queries:

    encoded = tokenizer.encode\_plus(

        query,

        add\_special\_tokens=True,

        max\_length=max\_len,

        padding='max\_length',

        truncation=True,

        return\_tensors='tf'

    )

    X\_test = encoded['input\_ids']

    prediction = model.predict(X\_test)[0][0]

    results.append((query, prediction))

for query, prediction in results:

    print(f"The predicted well-formedness score for query '{query}' is: {prediction}")

#testing another trained model from the forums https://huggingface.co/salesken/query\_wellformedness\_score

import torch

from transformers import AutoTokenizer, AutoModelForSequenceClassification

tokenizer = AutoTokenizer.from\_pretrained("salesken/query\_wellformedness\_score")

model = AutoModelForSequenceClassification.from\_pretrained("salesken/query\_wellformedness\_score")

sentences = ["What were the reasons for everyone to leave the company?", "tell me way city to the", "What is the capital of France?", "what was the reasons for everyone to leave the company"]

features = tokenizer(sentences,  padding=True, truncation=True, return\_tensors="pt")

model.eval()

with torch.no\_grad():

    scores = model(\*\*features).logits

print(scores)

References

Google-Research-Datasets. (n.d.). *Google-Research-datasets/query-wellformedness: 25,100 queries from the Paralex Corpus (fader et al., 2013) annotated with human ratings of whether they are well-formed natural language questions.* GitHub. Retrieved April 12, 2023, from <https://github.com/google-research-datasets/query-wellformedness>

*Bert-base-uncased · hugging face*. bert-base-uncased · Hugging Face. (n.d.). Retrieved April 12, 2023, from <https://huggingface.co/bert-base-uncased?text=Paris%2Bis%2Bthe%2B%5BMASK%5D%2Bof%2BFrance>

*Salesken/query\_wellformedness\_score · hugging face*. salesken/query\_wellformedness\_score · Hugging Face. (n.d.). Retrieved April 12, 2023, from https://huggingface.co/salesken/query\_wellformedness\_score