# hw-4q2-1

December 12, 2023

**QUESTION 2.**

**1.Create your own dataset for text classification. It should contain at least 1000 words in total and at least two categories with at least 100 examples per category. You can create it by scraping the web or using some of the documents you have on your computer (do not use anything confidential) or ChatGPT**

[1]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.preprocessing import PolynomialFeatures, StandardScaler,␣
 ↪OneHotEncoder
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.utils import shuffle
from sklearn.model_selection import cross_val_score, cross_val_predict,␣
 ↪cross_validate
from sklearn.linear_model import SGDRegressor
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

[11]:
```
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from transformers import GPT2Tokenizer, TFGPT2ForSequenceClassification
from tensorflow.keras import layers, models

# Load your dataset
data = pd.read_csv("/content/fin_state1.txt", sep='\t',
                   names=["category", "text"])

data.head()
```

[11]:

| | category | text |
|---|---|---|
| 0 | positive | With the new production plant the company woul… |
| 1 | positive | According to the company 's updated strategy f… |
| 2 | positive | FINANCING OF ASPOCOMP 'S GROWTH Aspocomp is ag… |
| 3 | positive | For the last quarter of 2010 , Componenta 's n… |
| 4 | positive | In the third quarter of 2010 , net sales incre… |

**2.Split the dataset into training (at least 160examples) and test (at least 40 examples) sets**

[12]:
```
# Split the dataset into training (80%) and test (20%) sets
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
label_mapping = {"negative": 0, "positive": 1}
train_data['category'] = train_data['category'].map(label_mapping)
test_data['category'] = test_data['category'].map(label_mapping)

# Load the GPT-2 model and tokenizer
gpt2_model = TFGPT2ForSequenceClassification.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")

# Set the padding token
tokenizer.pad_token = tokenizer.eos_token

# Encode the text data
max_seq_length = 62  # Set the desired maximum sequence length
train_tokenized = tokenizer(train_data['text'].tolist(), padding=True,␣
 ↪truncation=True, max_length=max_seq_length, return_tensors="tf")
test_tokenized = tokenizer(test_data['text'].tolist(), padding=True,␣
 ↪truncation=True, max_length=max_seq_length, return_tensors="tf")
```

```
All PyTorch model weights were used when initializing
TFGPT2ForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFGPT2ForSequenceClassification were
not initialized from the PyTorch model and are newly initialized:
['score.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

[3]:
```
# Print the shape of tokenized sequences
print("Train input shape:", train_tokenized["input_ids"].shape)
print("Test input shape:", test_tokenized["input_ids"].shape)
```

```
Train input shape: (160, 62)
Test input shape: (40, 62)
```

**3.Fine tune a pretrained language model capable of generating text (e.g., GPT) that you can take from the Hugging Face Transformers library with the dataset your created (this tutorial could be very helpful:**

**https://huggingface.co/docs/transformers/training). Report the test accuracy. Discuss what could be done to improve accuracy.**

[5]:
```
# Convert BatchEncoding to a dictionary of NumPy arrays
train_inputs = {key: train_tokenized[key].numpy() for key in train_tokenized}
test_inputs = {key: test_tokenized[key].numpy() for key in test_tokenized}

# Create an input layer
input_ids = tf.keras.Input(shape=(max_seq_length,), dtype=tf.int32,␣
 ↪name="input_ids")
attention_mask = tf.keras.Input(shape=(max_seq_length,), dtype=tf.int32,␣
 ↪name="attention_mask")

# Connect GPT-2 model
gpt2_output = gpt2_model(input_ids, attention_mask=attention_mask).logits

# Create a classifier head
classifier_output = layers.Dense(1, activation="sigmoid")(gpt2_output)

# Create the model using Functional API
classifier_model = tf.keras.Model(inputs=[input_ids, attention_mask],␣
 ↪outputs=classifier_output)

# Compile the model
classifier_model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

# Assuming you have binary labels (positive and negative)
y_train = (train_data['category'] == 'positive').astype(int)
y_test = (test_data['category'] == 'positive').astype(int)

# Train the model
classifier_model.fit(train_inputs, y_train, epochs=5, batch_size=32)

# Evaluate the model
accuracy = classifier_model.evaluate(test_inputs, y_test)
print(f"Test Accuracy: {accuracy[1]}")
```

```
Epoch 1/5
5/5 [==============================] - 163s 27s/step - loss: 2.8013e-04 -
accuracy: 1.0000
Epoch 2/5
5/5 [==============================] - 114s 23s/step - loss: 2.3506e-08 -
accuracy: 1.0000
Epoch 3/5
5/5 [==============================] - 153s 31s/step - loss: 1.9648e-09 -
accuracy: 1.0000
Epoch 4/5
```

```
5/5 [==============================] - 123s 24s/step - loss: 4.9270e-10 -
accuracy: 1.0000
Epoch 5/5
5/5 [==============================] - 111s 22s/step - loss: 4.8669e-10 -
accuracy: 1.0000
2/2 [==============================] - 12s 2s/step - loss: 0.0000e+00 -
accuracy: 1.0000
Test Accuracy: 1.0
```

To improve accuracy, consider the following steps:

**Fine-Tuning**: Fine-tune the GPT-2 model on a downstream task related to your specific domain. This involves training the model on a task-specific dataset to adapt it to your use case.

**Hyperparameter Tuning**: Experiment with different hyperparameters, such as learning rate, batch size, and model architecture, to find the combination that works best for your data.

**Data Augmentation**: Increase the diversity of your training data through data augmentation techniques, such as adding noise, paraphrasing, or using different sentence structures.

**More Training Data**: If possible, obtain more labeled data for training. A larger and more diverse dataset can often lead to better model performance.

**Model Architecture**: Experiment with different model architectures or try more advanced models that might be better suited for your task.

**Regularization**: Apply regularization techniques, such as dropout, to prevent overfitting on the training data.

**Error Analysis**: Analyze the mistakes made by the model on the test set. Identify patterns in misclassifications and consider incorporating this knowledge into the training process.

**Good dataset** : if possible proper collection of images in data without any distortion,blur, etc.

[19]:
```python
# Extract predictions for the [CLS] token
test_predictions_cls = test_predictions[:, 0, 0]

# Convert predictions to binary labels
predicted_labels = (test_predictions_cls > 0.5).astype(int)

# Compare predicted labels with true labels
results_df = pd.DataFrame({
    'Text': test_data['text'].tolist(),
    'True Label': y_test,
    'Predicted Label': predicted_labels,
    'Predicted Probability': test_predictions_cls
})

# Display the first few rows of the results DataFrame
results_df.head()
```

[19]:

| | Text | True Label | Predicted Label | Predicted Probability |
|---|---|---|---|---|
| 95 | Finnair was able to operate most of its leisur… | 0 | 0 | 2.206591e-11 |
| 15 | Incap Contract Manufacturing Services Pvt Ltd … | 0 | 0 | 4.930414e-11 |
| 30 | Viking Line 's cargo revenue increased by 5.4 … | 0 | 0 | 1.562709e-11 |
| 158 | More than a third of the original participants… | 0 | 0 | 3.189355e-11 |
| 128 | ADP News - Apr 22 , 2009 - Finnish business in… | 0 | 0 | 9.979138e-12 |