0wwzypajc

December 11, 2023

For the second part of the Assignment, We have used chatGPT and webscraping to create data corpus of two categories. Namely: Artificial intelligence and Politics.

The training set has 160 examples, and testing set has 40 examples. Split in the ratio of 80:20. We read them as pandas dataframe from excel file.

Class 0 belongs to texts that are "AI" related, and Class 1 belongs to "Politics"

[1]:
```
import pandas as pd
```

Read Train and Test Files:

[20]:
```
train = pd.read_excel("/content/train.xlsx")
test = pd.read_excel("/content/test.xlsx")
```

[21]:
```
train[train["class"]==0].shape
```

[21]:
```
(80, 2)
```

[22]:
```
train[train["class"]==1].shape
```

[22]:
```
(80, 2)
```

[23]:
```
test[test["class"]==0].shape
```

[23]:
```
(20, 2)
```

[25]:
```
test[test["class"]==1].shape
```

[25]:
```
(20, 2)
```

[26]:
```
#train shape
train.shape
```

[26]:
```
(160, 2)
```

[28]:
```
#test shape
test.shape
```

```
[28]: (40, 2)
```

```
[60]: #AI category texts
      train["text"][6]
```

```
[60]: "One of the big problems we're already seeing is that algorithms using machine-
      learning can come up with decisions that we have no way to understand or
      challenge. We know the human assumptions on which the algorithms are based, but
      there's no easy way of seeing why a neural network trained on billions of
      disparate items of data will, for example, wrongly flag a world-famous war photo
      as pornography or scandalously miscategorize a photo of three black teenagers.
      In the second case, after Google's algorithms tagged people as gorillas, Google
      didn't even bother trying to fix the problem: it simply stopped its algorithms
      labeling any photo as a gorilla, chimpanzee, or monkey. The AI problem was too
      hard to solve, so they solved an easier problem instead."
```

```
[61]: #politics category texts
      train["text"][90]
```

```
[61]: "Voter registration procedures can vary significantly across states, reflecting
      the decentralized nature of U.S. elections. Some states offer online
      registration, while others require mail-in forms or in-person visits.
      Registration deadlines typically fall several weeks before the election day.
      Familiarizing oneself with these specific rules and procedures is paramount to
      ensuring one's eligibility to vote and making one's voice heard in the election
      process."
```

Let's import transformers library, including BERT (Bidirectional Encoder Representations from Transformers).

```
[29]: from transformers import AutoTokenizer
      tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

```
tokenizer_config.json:   0%|          | 0.00/29.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/213k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/436k [00:00<?, ?B/s]
```

```
[30]: import numpy as np
      tokenized_data = tokenizer(train["text"].tolist(), return_tensors="np",␣
       ↪padding=True)
      # Tokenizer returns a BatchEncoding, but we convert that to a dict for Keras
      tokenized_data = dict(tokenized_data)
      labels = np.array(train["class"].tolist())  # Label is already an array of 0␣
       ↪and 1
```

```
[31]: labels
```

```
[31]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 1, 1, 1, 1, 1])
```

Import the necessary packages for fine tuning the model:

TFAutoModelForSequenceClassification is part of the Transformers library and is designed for fine-tuning pre-trained transformer models (such as BERT) for sequence classification tasks.

It automatically loads the appropriate pre-trained model architecture for sequence classification tasks.

```
[32]: from transformers import TFAutoModelForSequenceClassification
      from tensorflow.keras.optimizers import Adam
```

```
[33]: # Load and compile our model
      model = TFAutoModelForSequenceClassification.from_pretrained("bert-base-cased")
      # Lower learning rates are often better for fine-tuning transformers
      model.compile(optimizer=Adam(3e-5))  # No loss argument!
      model.fit(tokenized_data, labels)
```

```
model.safetensors:   0%|          | 0.00/436M [00:00<?, ?B/s]

All PyTorch model weights were used when initializing
TFBertForSequenceClassification.

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

5/5 [==============================] - 375s 67s/step - loss: 0.5995
```

```
[33]: <keras.src.callbacks.History at 0x7bd04849fa00>
```

Loss on training data looks to be 0.6 Lets tokenize our testing data

Inference on the trained model

```
[54]: tokenized = tokenizer(test["text"].tolist(), return_tensors="np",␣
        ↪padding="longest")
      outputs = model(tokenized).logits
```

```
classifications_test = np.argmax(outputs, axis=1)
print(classifications_test)
```

```
[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1]
```

```
[55]: test_labels = np.array(test["class"].tolist())
```

predictions of newly generated text

```
[35]: ai_text = "When artificial intelligence (AI) is mentioned Microsoft, Nvidia,␣
       ↪and Alphabet are often the first stocks that come to mind. And for good␣
       ↪reason. They're all excellent companies, and each one, in its own way, is␣
       ↪responsible for exciting AI innovations.\n\nThere is, however, another stock␣
       ↪worth mentioning. Indeed, in some respects, this company's future is riding␣
       ↪on its ability to develop a specific form of AI. If it fails, it may never␣
       ↪reach its full potential. But if it succeeds -- watch out. What's more,␣
       ↪according to Cathie Wood, its share price might soar to over $2,000 by 2027"
```

```
[40]: ai_text = "Artificial Intelligence, a cutting-edge field of computer science,␣
       ↪empowers machines to mimic human intelligence, enabling them to learn,␣
       ↪reason, and make informed decisions, revolutionizing the way we solve␣
       ↪complex problems and interact with technology"
```

```
[56]: tokenized_sentence = tokenizer(ai_text, return_tensors="np", padding="longest")
      outputs = model(tokenized_sentence).logits
      classifications = np.argmax(outputs, axis=1)
      if classifications == 0:
        print("Predicted Class = AI")
      else:
        print("Politics")
```

```
Predicted Class = AI
```

```
[57]: politics_text = "Politics, the art and science of governance, involves the␣
       ↪distribution and exercise of power, as well as the formulation and␣
       ↪implementation of policies that shape societies. It encompasses diverse␣
       ↪ideologies, political systems, and the dynamics of decision-making,␣
       ↪influencing the direction and character of nations and communities."
```

```
[58]: tokenized_sentence = tokenizer(politics_text, return_tensors="np",␣
       ↪padding="longest")
      outputs = model(tokenized_sentence).logits
      classifications = np.argmax(outputs, axis=1)
      if classifications == 0:
        print("Predicted Class = AI")
      else:
```

```
  print("Predicted Class = Politics")
```

```
Predicted Class = Politics
```

[59]:
```
# Accuracy of the model
from sklearn.metrics import accuracy_score

# Calculate accuracy
accuracy = accuracy_score(test_labels, classifications_test )
print("Accuracy:", accuracy)
```

```
Accuracy: 0.975
```

Even with only around 200 samples, an accuracy of 97.5 is really good. But on the other hand, we do have very less testing data. In Real world scenarios, the data may not always be of good quality. It is prone to a lot of noise. Good pre-processing techniques is essential. And moreover, the texts may not always differ so much like in our case [AI and politics] are very different categories. But when we are trying to classify something such as AI and Machine Learning, the task will only be even more difficult. Perhaps, training with better pre-trained models and parameters will do the trick.