Lab6-Assignment: Topic Classification

Use the same training, development, and test partitions of the the 20 newsgroups text dataset as in Lab6.4-Topic-classification-BERT.ipynb

- Fine-tune and examine the performance of another transformer-based pretrained language models, e.g., RoBERTa, XLNet

- Compare the performance of this model to the results achieved in Lab6.4-Topic-classification-BERT.ipynb and to a conventional machine learning approach (e.g., SVM, Naive Bayes) using bag-of-words or other engineered features of your choice. Describe the differences in performance in terms of Precision, Recall, and F1-score evaluation metrics.

```
import pandas as pd
from nltk.corpus import stopwords
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report
from simpletransformers.classification import ClassificationModel,
ClassificationArgs

STOPWORDS = stopwords.words('english')
```

Loading dataset

```
categories = ['alt.atheism', 'comp.graphics', 'sci.med', 'sci.space']

newsgroups_train = fetch_20newsgroups(subset='train',
remove=('headers', 'footers', 'quotes'), categories=categories,
random_state=420)
newsgroups_test = fetch_20newsgroups(subset='test', remove=('headers',
'footers', 'quotes'), categories=categories, random_state=420)

train = pd.DataFrame({'text': newsgroups_train.data, 'labels':
newsgroups_train.target})
train, val = train_test_split(train, test_size=0.1, random_state=420,
stratify=train[['labels']])
test = pd.DataFrame({'text': newsgroups_test.data, 'labels':
newsgroups_test.target})
```

Pretrained language model

```
model_args = ClassificationArgs()

model_args.overwrite_output_dir=True
model_args.evaluate_during_training=True

model_args.num_train_epochs=10
```

```python
model_args.train_batch_size=64
model_args.learning_rate=4e-6
model_args.max_seq_length=256

model_args.use_early_stopping=True
model_args.early_stopping_delta=0.01 # "The improvement over best_eval_loss necessary to count as a better checkpoint"
model_args.early_stopping_metric='eval_loss'
model_args.early_stopping_metric_minimize=True
model_args.early_stopping_patience=2
model_args.evaluate_during_training_steps=32 # how often you want to run validation in terms of training steps (or batches)

model = ClassificationModel('roberta', 'roberta-large', num_labels=4, args=model_args, use_cuda=True)

model.train_model(train, eval_df=val)

print(classification_report(test.labels, model.predict(test.text.to_list())[0]))
```

{"model_id":"125dba1db9c74b75b11af966d77ed53a","version_major":2,"version_minor":0}

{"model_id":"308678b2a1254bdebc3003dcc06673e9","version_major":2,"version_minor":0}

```
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       319
           1       0.88      0.93      0.90       389
           2       0.95      0.85      0.90       396
           3       0.77      0.85      0.81       394

    accuracy                           0.86      1498
   macro avg       0.86      0.85      0.85      1498
weighted avg       0.86      0.86      0.86      1498
```

Output from the BERT model

```
          precision    recall  f1-score   support

       0       0.82      0.83      0.82       319
       1       0.90      0.91      0.91       389
       2       0.87      0.91      0.89       396
       3       0.88      0.81      0.84       394

accuracy                           0.87      1498
macro avg       0.87      0.87      0.87      1498
weighted avg   0.87      0.87      0.87      1498
```

Comparing ROBERTA with BERT, it seems BERT has a slightly higher accuracy with 0.87 compared to ROBERTA's 0.86. Both therefore have a good performance. This score comes from both the precision and recall, the macro avg and weighted avg are all higher for BERT. Thus, with regards to the F1-score, BERT is also performing better when comparing the macro avg and weighted avg. Overall BERT's performance in all categories is slightly better.

Conventional ML model

```
vectorizer =
CountVectorizer(stop_words=STOPWORDS).fit(newsgroups_train.data +
newsgroups_test.data)

train = vectorizer.transform(newsgroups_train.data)
test = vectorizer.transform(newsgroups_test.data)

clf = LinearSVC(C=0.01, max_iter=int(1e6), random_state=420)
clf.fit(train, newsgroups_train.target)
print(classification_report(newsgroups_test.target, clf.predict(test),
target_names=newsgroups_test.target_names))
```

```
               precision    recall  f1-score   support

  alt.atheism       0.83      0.76      0.80       319
comp.graphics       0.75      0.90      0.82       389
      sci.med       0.87      0.73      0.80       396
    sci.space       0.78      0.79      0.79       394

     accuracy                           0.80      1498
    macro avg       0.81      0.80      0.80      1498
 weighted avg       0.81      0.80      0.80      1498
```

Comparison of the SVM to roberta and BERT

Overall the f1-score of the SVM is 0.80. The recall is also 0.80 with a higher precision of 0.81. Comparing the conventional ML model to the previous 2 models (roberta and BERT) we see that the SVM model is worse with an accuracy of 0.80 compared to the roberta (0.86) and BERT (0.87).

Overall there is no class imbalance in the dataset. In the SVM the topics comp.graphics and sci.space have a lower precision than recall meaning the model returns a lot of topics but does not assign the correct class to those topics. For sci.space this difference is only 0.1 For alt.atheism and sci.med the recall is low while the percision is high meaning the model is not finding a lot of those instances but it is able to return the correct topic of these instances.