TEXT CLASSIFICATION USING TRANSFORMERS

DONE BY GANTA SRILALITHA

TASK: Build a text classification model using Hugging Face Library to classify dataset of text into one of multiple categories using pre-trained model like BERT and fine-tune for the classification task. Finally obtain the evaluation metrics and the results.

DATASET USED: 20newsgroups(it has 20 classes and 18846 samples)

The dataset has a list of 20 newsgroups

In this dataset, duplicate messages are removed and the original messages only contain "From" and "Subject" headers

Each message in the newsgroup contains the following headers:

Eg:
Newsgroup: alt.newsgroup
Document_id: xxxxxx
From: Cat
Subject: Meow Meow Meow

PREPROCESSING STEPS:

1) Import the following functions from nltk library:
   a) **word_tokenise**-> to split sentences into words(tokens)
   b) **stopwords**-> words that don't add much meaning to sentence like a, the etc.
   c) **PorterStemmer**-> obtain root word of a given word.
2) Removed white spaces and special characters. Removed any rows with empty fields and duplicate records.
3) Cleaned text after removing stop-words, short-words or any link(if present). Stemmer is used to provide near root word.
4) Split the dataset into train & test set(taking test size=0.20)

MODEL USED: DISTILBERT-BASE-UNCASED

Transformers is a deep learning model that uses self-attention mechanism to encode & decode sequences.
Encoder takes input sequences, creates a word embedding of the tokenised words
Decoder takes input from encoder & previously generated words of sentence to predict/generate next word.
Distil BERT is fast and small Transformer Model based on BERT Architecture.

FLOWCHART OF THE ARCHITECTURE OF THE MODEL USED

Pass maximum length of tokenization & target(categories) to model

↓

Preprocess training & testing dataset to have them prepared for the model

↓

Create an instance of distill-bert classifier and use it for learning the model.

↓

Finetune by passing the model, train & test dataset to learner of the model

↓

Find the best learning rate of model by iterating it to a maximum number of epoch

↓

Use best learning rate to train the model using one-cycle method , & obtain its accuracy

↓

Test the model against new input

↓

Obtain accuracy,f1 score,recall and precision of the model

Results obtained when tested against a new input(Obtained 81% Accuracy)

Test against new data

```
[ ] predictor=ktrain.get_predictor(learner.model,preproc=t)
```

```
[ ] #Lets take a sample news
    news_text = dataset['cleaned_text'].iloc[5]
    actual_category = dataset['categories'].iloc[5]
    print(f"Actual category is {actual_category}")

    Actual category is sci_crypt
```

```
[ ] print(f"prdicted category is{predictor.predict(dataset['cleaned_text'].iloc[5])}")

    1/1 [==============================] - 0s 91ms/step
    prdicted category issci_crypt
```

```
learner.validate()
```

	precision	recall	f1-score	support
0	0.79	0.79	0.79	42
1	0.58	0.75	0.65	48
2	0.63	0.80	0.71	50
3	0.60	0.58	0.59	50
4	0.71	0.70	0.71	50
5	0.95	0.69	0.80	51
6	0.79	0.76	0.77	49
7	0.89	0.87	0.88	45
8	0.93	0.85	0.89	46
9	0.96	0.96	0.96	51
10	1.00	0.98	0.99	49
11	0.95	0.95	0.95	44
12	0.79	0.76	0.78	50
13	0.95	0.88	0.91	48
14	0.91	0.98	0.94	52
15	0.85	0.81	0.83	48
16	0.88	0.78	0.82	45
17	0.84	0.96	0.90	49
18	0.81	0.86	0.84	44
19	0.59	0.53	0.56	32
accuracy			0.82	943
macro avg	0.82	0.81	0.81	943
weighted avg	0.82	0.82	0.82	943

The actual category of a sample news is scri_crypt which is also predicted by the model.

Ways to improve accuracy:

1)Increase the number of training samples.(add more data)

2)Increase the number of training epoch while training the dataset

3)Not to overfit or underfit the data

4)Increase the stop-words list(add location, numerals,time stop words.)

5)Use features like- parts of speech.

Link to Code: https://github.com/gantasrilaitha/text-classification/blob/main/text_classification_.ipynb

Code Snippets

Install ktrain, which is a light-weight python wrapper library for keras and tensorflow to buld & deploy neural network models

```
!pip install ktrain
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ktrain
  Downloading ktrain-0.33.2.tar.gz (25.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 25.3/25.3 MB 55.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dist-packages (from ktrain) (1.2.1)
Requirement already satisfied: matplotlib>=3.0.0 in /usr/local/lib/python3.8/dist-packages (from ktrain) (3.5.3)
Requirement already satisfied: pandas>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from ktrain) (1.3.5)
Requirement already satisfied: fastprogress>=0.1.21 in /usr/local/lib/python3.8/dist-packages (from ktrain) (1.0.3)
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from ktrain) (2.25.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.8/dist-packages (from ktrain) (1.2.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from ktrain) (23.0)
```

Set runtime to GPU & make neceaasry settings

```
%reload_ext autoreload
%autoreload 2
%matplotlib inline
import os
os.environ["CUDA_DEVICE_FOLDER"]="PCI_BUS_ID";
os.environ["CUDA_VISIBLE_DEVICES"]="0";
```

Fetch the dataset

```
[ ] import ktrain
    from ktrain import text
    from sklearn.datasets import fetch_20newsgroups
```

Import libraries for preprocessing

```
[ ] import warnings
    warnings.filterwarnings("ignore")

    import numpy as np
    import re
    import pandas as pd
    import nltk
    nltk.download('words')
    nltk.download('stopwords')
    from nltk.tokenize import word_tokenize
    from nltk.corpus import stopwords
    from nltk.stem import PorterStemmer
    from nltk.tokenize import word_tokenize
    ps=PorterStemmer()
```

```
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Load Dataset

```
[ ] newsgroups=fetch_20newsgroups(subset='all')
    df=pd.DataFrame(newsgroups.data,columns=['text'])
    df['categories']=[newsgroups.target_names[index] for index in newsgroups.target]
    df.head(8)
```

	text	categories
0	From: Mamatha Devineni Ratnam <mr47+@andrew.cm...	rec.sport.hockey
1	From: mblawson@midway.ecn.uoknor.edu (Matthew ...	comp.sys.ibm.pc.hardware
2	From: hilmi-er@dsv.su.se (Hilmi Eren)\nSubject...	talk.politics.mideast
3	From: guyd@austin.ibm.com (Guy Dawson)\nSubjec...	comp.sys.ibm.pc.hardware
4	From: Alexander Samuel McDiarmid <am2o+@andrew...	comp.sys.mac.hardware
5	From: tell@cs.unc.edu (Stephen Tell)\nSubject:...	sci.electronics
6	From: lpa8921@tamuts.tamu.edu (Louis Paul Adam...	comp.sys.mac.hardware
7	From: dchhabra@stpl.ists.ca (Deepak Chhabra)\n...	rec.sport.hockey

Read dataset

```
[ ] df=pd.read_csv("/content/20-newsgroups-dataset.csv")
    df.head()
```

	text	categories
0	From: jim.zisfein@factory.com (Jim Zisfein) Su...	sci_med
1	From: G.R.Price@cm.cf.ac.uk (and thats a fact)...	rec_sport_baseball
2	From: egreen@east.sun.com (Ed Green - Pixel Cr...	rec_motorcycles
3	From: andy@ie.utoronto.ca (Andy Sun) Subject: ...	comp_sys_mac_hardware
4	From: cfaehl@vesta.unm.edu (Chris Faehl) Subje...	talk_religion_misc

Remove stop words, special characters,links,white spaces or duplicates

Also obtain near root word using stemmer

```
[ ] def dataset_cleaning(df_data):
        """This function helps to remove row with missing value or if there is any dupicate records"""
        df = df_data.dropna()
        df_data = df_data.drop_duplicates()
        df_data = df_data.reset_index(drop=True)
        return df_data

    def text_cleaning(text):
        """This function helps to clean a text after removing stop words, short words, special character,
        any link present and use stemmer to provide near to root word"""
        stop = set(stopwords.words('english'))
        text = text.lower()
        text = re.sub('[^abcdefghijklmnopqrstuvwxyz]',' ', text)
        text = re.sub(r'http\S+', ' ', text)
        text = " ".join([ps.stem(word) for word in text.split() if (word not in stop) and len(word)>1])
        return text
```

Split into train & test dataset

```
[ ] from sklearn.model_selection import train_test_split
    X_train,X_test,y_train,y_test=train_test_split(np.array(dataset['cleaned_text']),np.array(dataset['categories']),
                                                   test_size = 0.20, random_state=42, stratify = dataset['categories'])
```

Create an instance of classifier & use it for learning the model

```
[ ] MODEL_NAME ='distilbert-base-uncased'
    t=text.Transformer(MODEL_NAME,maxlen=512,classes=dataset['categories'].unique()) #maxlen-> max len of tokenisation
    train=t.preprocess_train(X_train,y_train) #prepare train& test dataset for transformer
    val=t.preprocess_test(X_test,y_test)
    model=t.get_classifier() #get distill bert classifier
    learner=ktrain.get_learner(model,train_data=train,val_data=val,batch_size=20)
```

```
preprocessing train...
language: en
train sequence lengths:
        mean : 175
        95percentile : 400
        99percentile : 1315
Is Multi-Label? False
preprocessing test...
language: en
test sequence lengths:
        mean : 191
        95percentile : 485
        99percentile : 2264
```

Find best learning rate

```
[ ] learner.lr_find(show_plot=True,max_epochs=5)
```

```
simulating training for different learning rates... this may take a few moments...
Epoch 1/5
188/188 [==============================] - 222s 1s/step - loss: 2.9819 - accuracy: 0.0822
Epoch 2/5
188/188 [==============================] - 210s 1s/step - loss: 1.9094 - accuracy: 0.4817
Epoch 3/5
188/188 [==============================] - 208s 1s/step - loss: 2.7534 - accuracy: 0.1288
Epoch 4/5
188/188 [==============================] - 202s 1s/step - loss: 3.0586 - accuracy: 0.0515
Epoch 5/5
188/188 [==============================] - 200s 1s/step - loss: 5.0745 - accuracy: 0.0501
```

```
[ ] learner.fit_onecycle(2e-5, 5)
```

```
begin training using onecycle policy with max lr of 2e-05...
Epoch 1/5
189/189 [==============================] - 245s 1s/step - loss: 2.4983 - accuracy: 0.3181 - val_loss: 1.7261 - val_accuracy: 0.5504
Epoch 2/5
189/189 [==============================] - 229s 1s/step - loss: 1.3059 - accuracy: 0.6532 - val_loss: 1.0170 - val_accuracy: 0.6957
Epoch 3/5
189/189 [==============================] - 229s 1s/step - loss: 0.7665 - accuracy: 0.7734 - val_loss: 0.7698 - val_accuracy: 0.7773
Epoch 4/5
189/189 [==============================] - 229s 1s/step - loss: 0.4585 - accuracy: 0.8700 - val_loss: 0.6714 - val_accuracy: 0.8017
Epoch 5/5
189/189 [==============================] - 230s 1s/step - loss: 0.2926 - accuracy: 0.9353 - val_loss: 0.6355 - val_accuracy: 0.8155
<keras.callbacks.History at 0x7f1b0f6dd550>
```

```
learner.validate()
```

```
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        42
           1       0.58      0.75      0.65        48
           2       0.63      0.80      0.71        50
           3       0.60      0.58      0.59        50
           4       0.71      0.70      0.71        50
           5       0.95      0.69      0.80        51
           6       0.79      0.76      0.77        49
           7       0.89      0.87      0.88        45
           8       0.93      0.85      0.89        46
           9       0.96      0.96      0.96        51
          10       1.00      0.98      0.99        49
          11       0.95      0.95      0.95        44
          12       0.79      0.76      0.78        50
          13       0.95      0.88      0.91        48
          14       0.91      0.98      0.94        52
          15       0.85      0.81      0.83        48
          16       0.88      0.78      0.82        45
          17       0.84      0.96      0.90        49
          18       0.81      0.86      0.84        44
          19       0.59      0.53      0.56        32

    accuracy                           0.82       943
   macro avg       0.82      0.81      0.81       943
weighted avg       0.82      0.82      0.82       943
```

Test against new data

```
predictor=ktrain.get_predictor(learner.model,preproc=t)
```

```
#Lets take a sample news
news_text = dataset['cleaned_text'].iloc[5]
actual_category = dataset['categories'].iloc[5]
print(f"Actual category is {actual_category}")
```

```
Actual category is sci_crypt
```

```
print(f"prdicted category is{predictor.predict(dataset['cleaned_text'].iloc[5])}")
```

```
1/1 [==============================] - 0s 91ms/step
prdicted category issci_crypt
```

Obtain confidence score of predicted category

```
reloaded_predictor = ktrain.load_predictor('/content/distilbert')
reloaded_predictor.predict(dataset['cleaned_text'].iloc[5])
print(f"confidence score: {np.max(reloaded_predictor.predict_proba(dataset['cleaned_text'].iloc[5]))}")
```

```
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 87ms/step
confidence score: 0.957402229309082
```