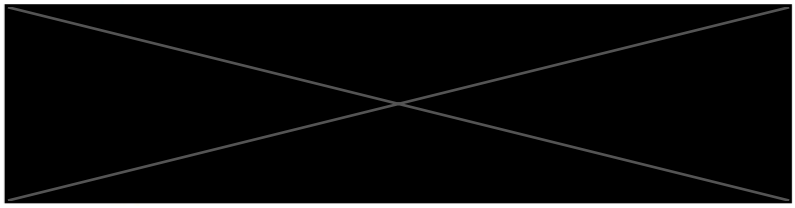

Text Classification Report Using Hugging Face

Name	Pooja Patil
Email Id	[redacted]
Contact Number (Only in WhatsApp)	[redacted]

I. Introduction

Text classification is defined as automatically assigning predefined labels or categories to text, documents, images and so on based on their content. In Natural Language Processing (NLP) this process of text classification is automated for organization and filtering of large amounts of data. Creation of a model for a specific task each time is very tedious, time consuming as well as resource consuming. Here pre-trained models come into picture. As the name suggests, since these models are already trained on large amounts of data, they have learnt a lot about patterns and structures, which makes them efficient as they require less data and compute power which in turn saves time. They often outperform the models trained from scratch and are readily available in open source libraries. In this assignment, Hugging face library is used to classify a dataset of text into one or multiple categories. A pre-trained distilBERT-base-uncased model is used for text classification class.

II. Dataset

The BBC Text dataset is used for the analysis. This dataset is a collection of articles published by British Broadcasting Corporation(BBC) on their website between the time period of 2004 to 2005. The dataset contains 2225 articles from five different categories: business, entertainment, politics, sport, and technology. Each article is labelled with one of these categories. The articles are provided in plain text format, along with their corresponding category labels. The dataset is available for download from various sources, including the UCI Machine Learning Repository and Kaggle.

III. Pre-processing the dataset

To process the dataset for the analysis, “transformers” library is imported. Is the used for NLP task such as text classification, question answering, and language translation. It provides state-of-the-art pre-trained models such as BERT, GPT-2, and RoBERTa, which can be fine-tuned on specific NLP tasks.

Using the “Pandas” library the dataset is loaded into python. Libraries such as tensorflow, numpy, seaborn, matplotlib, pyplot, and nltk is been imported. Prior to using the dataset for analysis, it is important to remove the stopwords and punctuations for it. Thus, applying the "clean_text" function, the text string is first stripped of any punctuations using the string

module. Then, the text is converted to lowercase using the lower() method. Finally, stopwords from the NLTK library are removed from the text using a list comprehension.

The apply method is then used to apply the "clean_text" function to each row of the "text" column in the dataframe. The resulting cleaned text is then saved to a new csv file named "bbc-text-cleaned.csv" using the “to_csv” method of pandas library.

IV. Architecture of the model

The model used for the text classification task is distilBERT-base-uncased pre-trained model. It is a transformers model, smaller and faster than BERT, which was pretrained on the raw texts only. DistilBERT receives as an input X which represents a textual sentence or sentences from the dataset.

The inputted sequences to DistilBERT are converted to a set of embedding vectors where each vector is mapped to each word in a sequence. Each article is represented with a set of tokens (words) of s-length vector. The special token [CLS] is placed as the first token in the sequence whereas the [SEP] token is placed at the end of the sequence. DistilBERT uses the transformer encoder to learn the contextual information for each word. The transformer encoder uses a self-attention mechanism to generate the contextual embeddings. DistilBERT uses knowledge distillation to minimize the BERT base model (bert-base-uncased) parameters by 40%, making the inference 60% faster. The main idea of distillation is to approximate the full output distributions of the BERT model using a smaller model such as DistilBERT
The extracted contextual embeddings for each word are concatenated into a single vector to represent the semantic information presented in the article. Semantic representation is the input of a fully connected layer that outputs a vector of size d where d is the number of neurons. Later, a classification layer is placed at the end of the feature extractor model to fine-tune the pre-trained DistilBERT on the text classification task and predict the corresponding event class for each inputted sequence.

Fine-tuning is the process of adapting a pre-trained neural network to a new task with a smaller labelled dataset. During fine-tuning, the pre-trained model is further trained on the task-specific data, allowing it to learn to make predictions specific to the task. The fine-tuning configuration is set up, including the number of epochs, learning rate, batch size, and optimizer. The pre-trained DistilBERT model is loaded and the classification layer is added on top. Then, the model is trained on the labelled training dataset, with the weights of the pre-trained layers frozen, and only the weights of the newly added classification layer are trained.

V. Working

The programming is done using google collab and python. Firstly, the transformers library with the other libraries used for coding are imported and the dataset (.csv file) is uploaded. The dataset is pre-processed by cleaning it, removing stopwords, punctuations and other irrelevant characters. A new column named 'encoded_text' to the DataFrame 'df', containing the encoded category values is created in the form of a list. This dataset is split into the subsets - training, test and validation datasets. Each article and the respective encodings are represented with a set of tokens (words) of s-length vector. The special token [CLS] is placed as the first token in the sequence whereas the [SEP] token is placed at the end of the sequence. The original pre-trained distilBERT-base-uncased model is then loaded and fine-tuned.

Keras sequential model for text classification using an embedding layer is created, followed by a Flatten layer and a dense output layer with softmax activation. The model using Adam optimizer and categorical cross-entropy loss function is compiled, the input shape (max_len,), and the maximum length of input sequences is set to 148.The training data is prepared by encoding the text using the tokenizer's encode method, padding the sequences to the max_len, and one-hot encoding the labels. The model is then trained using the fit method on the padded input sequences and one-hot encoded labels. The batch size is set to 32, and the model is trained for 50 epochs.

VI. Performance Evaluation

The fine tuned model created for the task classification task performs with an accuracy directly proportional to the number of epochs. The overall accuracy of the model is 0.889, which means that it correctly classified 89% of the samples in the dataset.

The classification report provides additional information on the performance of the model for each class. It shows precision, recall, and F1-score for each class. Precision is the proportion of correctly classified samples out of all the samples predicted as belonging to that class. Recall is the proportion of correctly classified samples out of all the samples that actually belong to that class. F1-score is a weighted harmonic mean of precision and recall, with the best value being 1.0 and the worst being 0.0.

The table shows the results for each class, labelled 0 through 4, along with the number of samples (support) in each class. For example, for class 0, the precision is 1.000, which means that all of the samples predicted to belong to class 0 were correctly classified. The recall is 0.750, which means that 75% of the samples that actually belong to class 0 were correctly classified. The F1-score is 0.857, which is the harmonic mean of precision and recall for class 0.

The macro average and weighted average are also provided in the table. The macro average is the unweighted mean of precision, recall, and F1-score across all classes. The weighted average is the weighted mean of precision, recall, and F1-score, where the weight is the number of samples in each class.

```
Accuracy: 0.889
Classification report:
              precision    recall  f1-score   support

           0      1.000     0.750     0.857         4
           1      1.000     0.667     0.800         3
           2      0.750     1.000     0.857         3
           3      1.000     1.000     1.000         4
           4      0.800     1.000     0.889         4

    accuracy                          0.889        18
   macro avg      0.910     0.883     0.881        18
weighted avg      0.914     0.889     0.886        18
```

VII. Analysis

To evaluate the performance of the model few examples were categorized using predict function. The example given below shows the functioning of fine tuned distilBERT-base-uncased model. The article belonged to the category of technology and was encoded accurately. This shows that the model is correctly fitted and gives the significant outcome.

```
test_text = test_texts[7]
test_text
```

```
'software watching work software monitor every keystroke action performed pc also used legally binding evidence wrongdoing unveiled worries cybercrime s
abotage prompted many employers consider monitoring employees developers behind system claim breakthrough way data monitored stored privacy advocates co
ncerned invasive nature software system joint venture security firm 3ami storage specialists bridgehead software joined forces create system monitor com
puter activity store retrieve disputed files within minutes firms finding deep water result data misuse sabotage data theft commonly committed within or
ganisation according national hitech crime unit nhtcu survey conducted behalf nop found evidence 80 medium large companies victims form cybercrime bridg
ehead software come techniques prove legal standard stored file pc tampered ironically impetus developing system came result freedom information act req
uires companies store data certain amount time storage system incorporated application de...'
```

```
predict_input = [tokenizer.encode(text, max_length=max_len, truncation=True) for text in [test_text]]
predict_input_padded = tf.keras.preprocessing.sequence.pad_sequences(predict_input, maxlen=max_len, padding='post')
output = model(predict_input_padded)[0]
prediction_value = tf.argmax(output, axis=-1).numpy()
prediction_value
```

```
4
```

VIII. Future Scope

The DistilBERT model has proven to be quite efficient in natural language processing. It can be applied to different areas like translating tests from one language to other, speech recognition, speech to text translations, to analyze large datasets and build predictive models for finance and healthcare sectors.