Zewail city of science and technology

Natural language processing NLP

Natural Language processing NLP Report

Text classification

Machine Learning – Deep Learning – Transfer Learning

Team members
Ahmed Eid Abdullah 201701264
Ahmed Kamal El melegy 201700295
Mahmoud Ahmed Hamid
201701219

Zewail city of science and technology

Natural language processing NLP

Acknowledgement

Apart from the effort of us, the success of any project depends on the guidelines of many other.

We take this opportunity to express our gratitude to the people who have been always guiding us in the Natural Language Processing course. We would like to show our greatest appreciation to Dr. Nermin Negied. We thank her for her tremendous support and help. We also would like to thank Eng. Ahmed and Eng. Roba for their help and support.

Zewail city of science and technology

Natural language processing NLP

Table of content:

Zewail city of science and technology

Natural language processing NLP

Introduction

Text classification is the process of categorizing a text into a group of words. By using Natural Language Processing, text classification can analyze text or a document and then assign its category based on its context. There is main three text classification approaches which are Rule based system, Machine system and hybrid system. In rule-based approach, texts or documents are separated into an organized groups by using a set of handicraft linguistic rules. Those rules contain users to collect a list of words and characterized them by groups, for example words like Trump, Obama and war would be categorized into politics. Words like Ronaldo and Messi would be categorized into sports. Machine-based classifiers usually use dataset to train a model and then test it. User data is labeled as a train data and test data. The classifier is then learned on train data as these group of documents are categorized into X class, while those are categorized into Y class and so on, and then the classifier is tested by the test data, and we evaluate the accuracy. Machine learning classifier usually use a bag of words for feature extraction of text. In a bag of words, there is a vector represents the frequency of special words in the documents, we can perform NLP using machine learning algorithms like SVM or Naïve Bayer or we can use deep learning for this task. The third approach is hybrid approach. Hybrid approach combines the rule-based and machine-based approaches.

.

Zewait city of science and technology

Natural language processing NLP

Problem definition

We have more than 20,000 newsgroup (documents) and we want to classify them based on their context and classify them into more than 20 categories.

Problem solution

We will use three models to do this task. We want to achieve the highest accuracy, so that will use:

- The Machine Learning Model Support Vector Machine SVM
- CNN for text classification
- Transfer Leaning Technique

We will compare between them and how check their accuracy for testing newgroup and based on the accuracy we will choose the best model. Natural Language Processing techniques play here an important role as we will use NLP as text-preprocessing tool such as tokenization, removing stop words stemming, lemmatization and normalization. We also will use NLP to convert the input text from text into an embedding vector of word (word representation vector) to be used on CNN step.

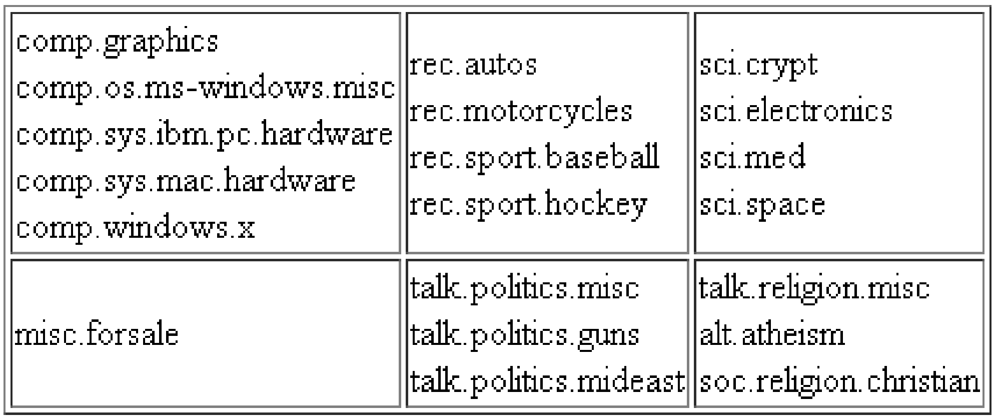
Dataset

Zewail city of science and technology

Natural language processing NLP

Our dataset is called 20-Newsgroup dataset. 20-Newsgroup dataset is a collection of approximately 20,000 newsgroup documents categorized into 20 different newsgroups each newsgroup corresponding to a different topic Some of the newsgroups are very closely related to each other (e.g. **comp.sys.ibm.pc.hardware / comp.sys.mac.hardware**), while others are highly unrelated (e.g **misc.forsale / soc.religion.christian**)

Categories of the 20-Newsgroup dataset:-



comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

The 20 newsgroups dataset has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

Firstly, we loaded the dataset, then we divide it into train data and test data. The dataset contains 20 categories, we worked only on 4 categories. When we worked on all dataset our collab crashed. When we used 10 categories only, It takes more than 2 hours running without any result, so we worked only on 4 categories which are alt.atheism , talk.religion.misc , comp.graphics and sci.space. We number of training samples is 2034, while the number of test samples is 1353.

Text Preprocessing

Firstly, we use TfidfVectorizer as a feature extraction. We started by initializing the vectorizer by setting some settings

```
vectorizer = TfidfVectorizer(max_df=0.5, min_df=2, stop_words='english', use_idf=True)
```

TfidfVectorizer used to get words frequency in the documents. By setting max_df = 0.5, the vectorizer will ignore the terms that has frequency higher than 0.5, and also will ignore the terms that has frequency lower than 2, we also do a text preprocessing step which is removing the known English stop words. Then we applied this vectorizer into the train data and test data.

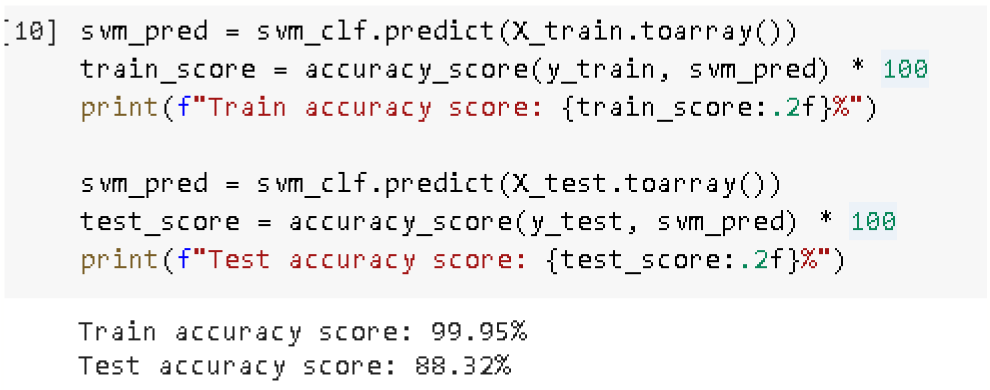
Machine learning model

Support vector machine

By using SVM, we achieved training accuracy 99.90%, but 88.32% test accuracy with C=0.7

SVM evaluation

C=0.9

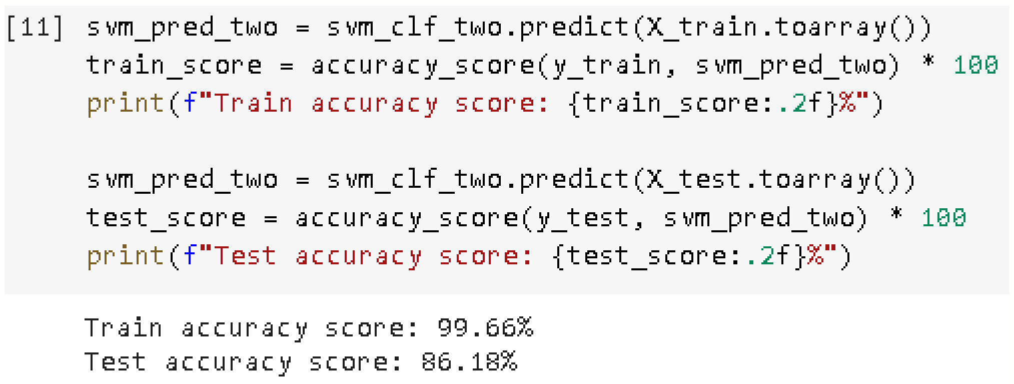


```
[10] svm_pred = svm_clf.predict(X_train.toarray())
     train_score = accuracy_score(y_train, svm_pred) * 100
     print(f"Train accuracy score: {train_score:.2f}%")

     svm_pred = svm_clf.predict(X_test.toarray())
     test_score = accuracy_score(y_test, svm_pred) * 100
     print(f"Test accuracy score: {test_score:.2f}%")

     Train accuracy score: 99.95%
     Test accuracy score: 88.32%
```

Zewail city of science and technology

Natural language processing NLP

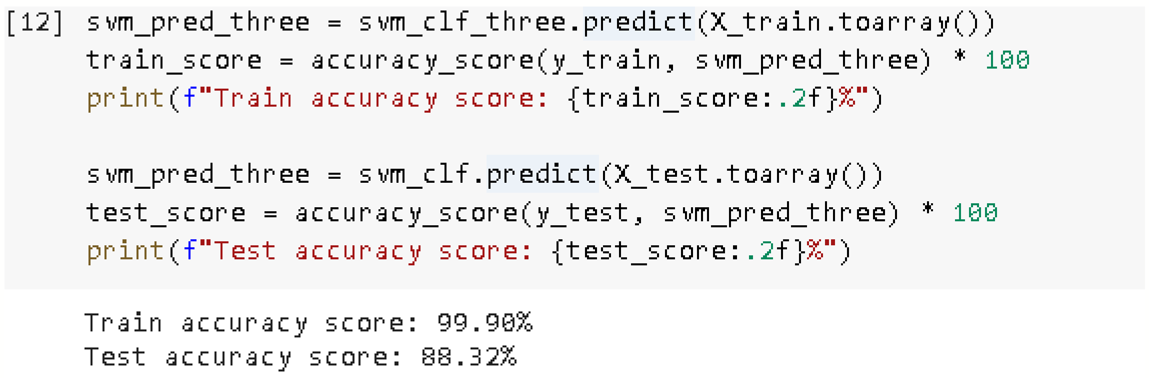
C=0.5



```
[11] svm_pred_two = svm_clf_two.predict(X_train.toarray())
     train_score = accuracy_score(y_train, svm_pred_two) * 100
     print(f"Train accuracy score: {train_score:.2f}%")

     svm_pred_two = svm_clf_two.predict(X_test.toarray())
     test_score = accuracy_score(y_test, svm_pred_two) * 100
     print(f"Test accuracy score: {test_score:.2f}%")
```

```
Train accuracy score: 99.66%
Test accuracy score: 86.18%
```

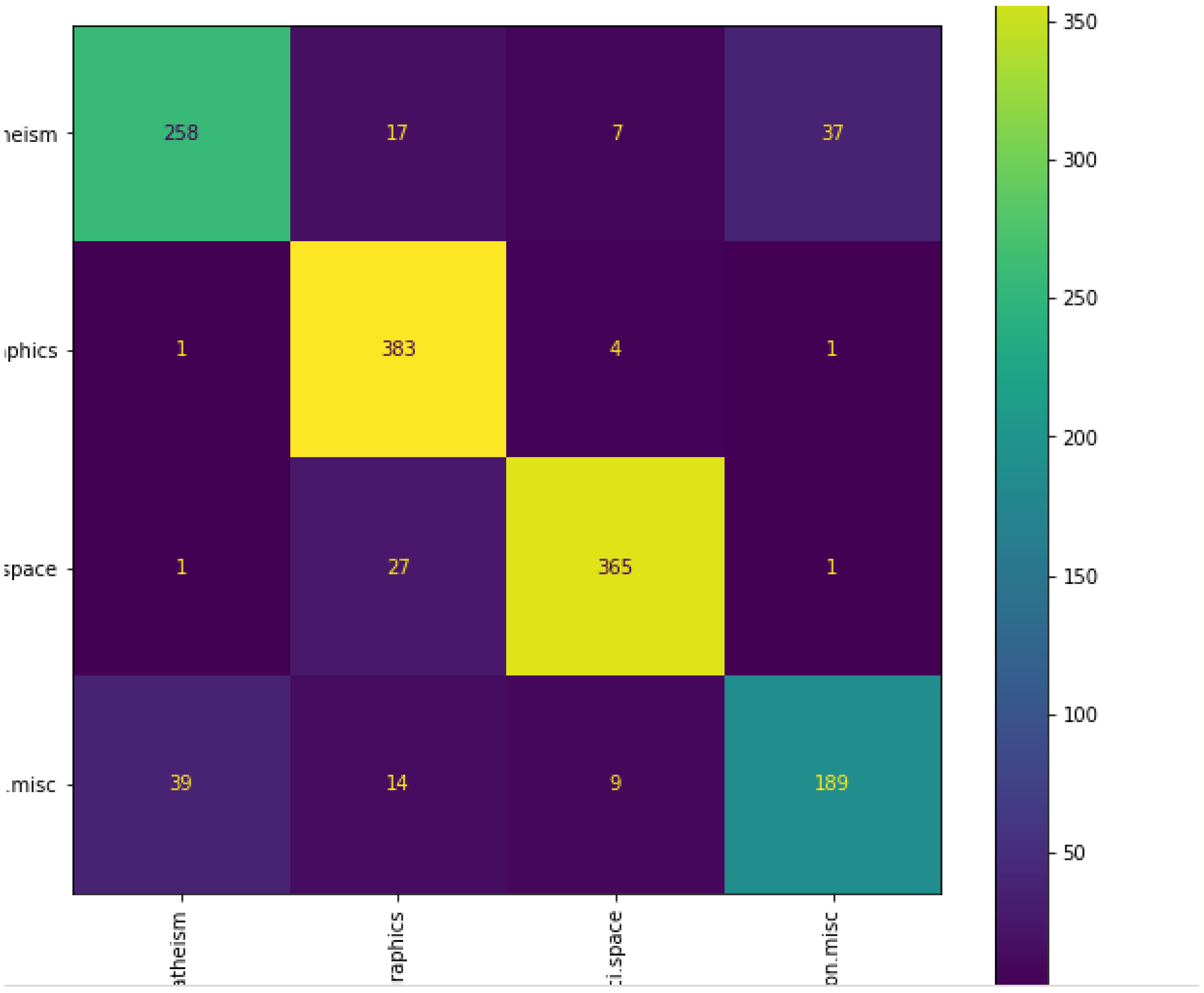
C=0.7



```
[12] svm_pred_three = svm_clf_three.predict(X_train.toarray())
     train_score = accuracy_score(y_train, svm_pred_three) * 100
     print(f"Train accuracy score: {train_score:.2f}%")

     svm_pred_three = svm_clf.predict(X_test.toarray())
     test_score = accuracy_score(y_test, svm_pred_three) * 100
     print(f"Test accuracy score: {test_score:.2f}%")
```

```
Train accuracy score: 99.90%
Test accuracy score: 88.32%
```

Confusion matrix at best C=0.7 last SVM model :



CNN Model

Firstly, we want to do a preprocessing step which is converting each document into vector of words to be ready to be the input to our CNN layers. So, we firstly tokenized the train data.

Training the model

We used many architectures here , most of them were unsuccessful with us and achieved a very low training accuracy.

Unsuccessful trials

First trail

- We start by using of course the embedding layer (as the input to the layers will be an embedding vector). Then we used 4 conv layers, 3 max pooling layers and one dense layer for the output layer.

By using this architecture, we got 89% training accuracy and 81% test accuracy.

Second trail

- We start by using of course the embedding layer (as the input to the layers will be an embedding vector). Then we used 5 conv layers, 4 max pooling layers and one dense layer for the output layer.

By using this architecture, we got 90% training accuracy and 83% test accuracy.

The accuracy not increased very much.

Third trail

- We start by using of course the embedding layer (as the input to the layers will be an embedding vector). Then we used 3 conv layers, 2 max pooling layers and one dense layer for the output layer.

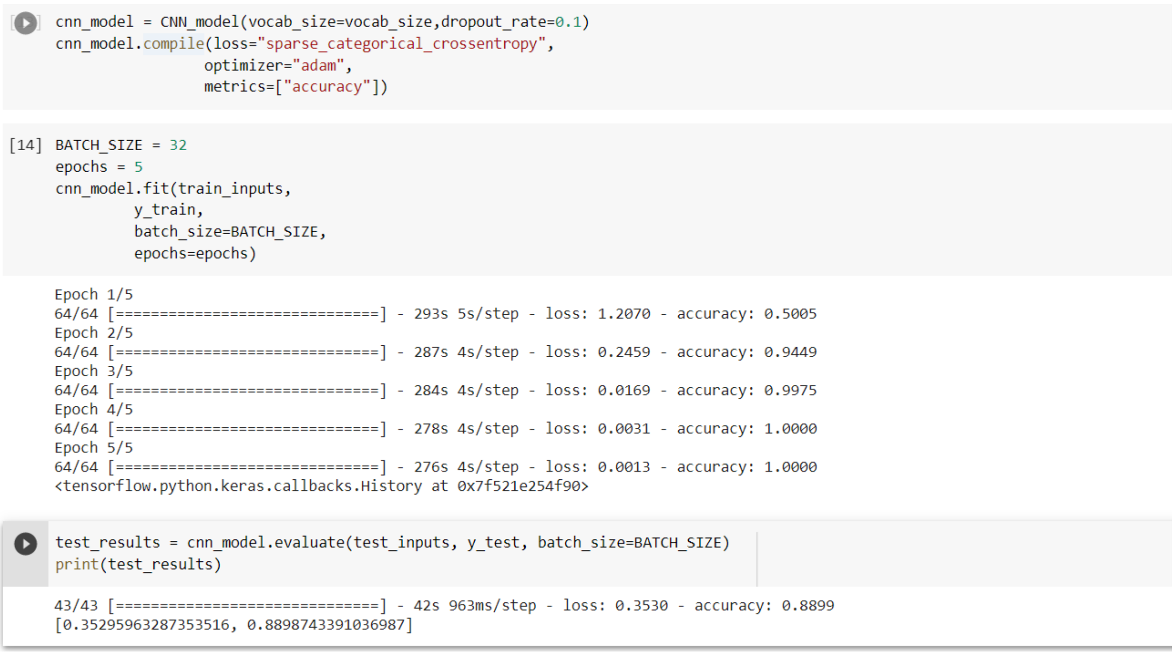
By using this architecture, we got 93% training accuracy and 85% test accuracy.

Fourth trial

- We start by using of course the embedding layer (as the input to the layers will be an embedding vector). Then we used 3 conv layers, 1 max pooling layers and one dense layer for the output layer.

And here we did a regularization step that we add a dropout layer of dropout rate equals to 0.1.

By using this architecture, we got 100% training accuracy and 88.9% test accuracy.



```
cnn_model = CNN_model(vocab_size=vocab_size,dropout_rate=0.1)
cnn_model.compile(loss="sparse_categorical_crossentropy",
                  optimizer="adam",
                  metrics=["accuracy"])
```

```
[14] BATCH_SIZE = 32
     epochs = 5
     cnn_model.fit(train_inputs,
               y_train,
               batch_size=BATCH_SIZE,
               epochs=epochs)
```

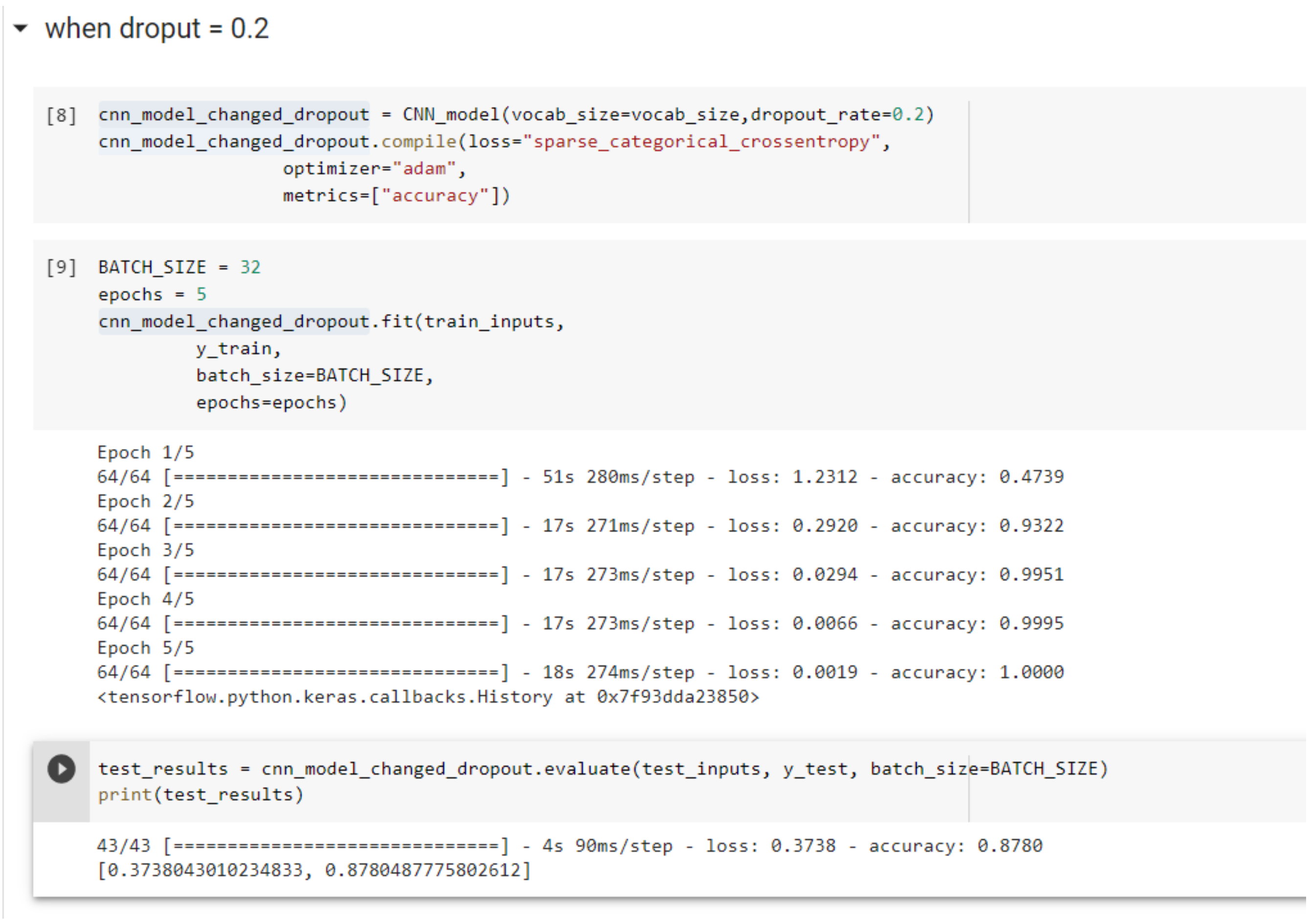
```
Epoch 1/5
64/64 [==============================] - 293s 5s/step - loss: 1.2070 - accuracy: 0.5005
Epoch 2/5
64/64 [==============================] - 287s 4s/step - loss: 0.2459 - accuracy: 0.9449
Epoch 3/5
64/64 [==============================] - 284s 4s/step - loss: 0.0169 - accuracy: 0.9975
Epoch 4/5
64/64 [==============================] - 278s 4s/step - loss: 0.0031 - accuracy: 1.0000
Epoch 5/5
64/64 [==============================] - 276s 4s/step - loss: 0.0013 - accuracy: 1.0000
<tensorflow.python.keras.callbacks.History at 0x7f521e254f90>
```

```
test_results = cnn_model.evaluate(test_inputs, y_test, batch_size=BATCH_SIZE)
print(test_results)
```

```
43/43 [==============================] - 42s 963ms/step - loss: 0.3530 - accuracy: 0.8899
[0.35295963287353516, 0.8898743391036987]
```

Fifth trial

- We start by using of course the embedding layer (as the input to the layers will be an embedding vector). Then we used 3 conv layers, 1 max pooling layers and one dense layer for the output layer. And here we did a regularization step that we add a dropout layer of dropout rate equals to 0.2. By using this architecture, we got 100% training accuracy and 87.8% test accuracy.



when droput = 0.2

```
[8] cnn_model_changed_dropout = CNN_model(vocab_size=vocab_size,dropout_rate=0.2)
    cnn_model_changed_dropout.compile(loss="sparse_categorical_crossentropy",
                     optimizer="adam",
                     metrics=["accuracy"])
```

```
[9] BATCH_SIZE = 32
    epochs = 5
    cnn_model_changed_dropout.fit(train_inputs,
              y_train,
              batch_size=BATCH_SIZE,
              epochs=epochs)
```

```
Epoch 1/5
64/64 [==============================] - 51s 280ms/step - loss: 1.2312 - accuracy: 0.4739
Epoch 2/5
64/64 [==============================] - 17s 271ms/step - loss: 0.2920 - accuracy: 0.9322
Epoch 3/5
64/64 [==============================] - 17s 273ms/step - loss: 0.0294 - accuracy: 0.9951
Epoch 4/5
64/64 [==============================] - 17s 273ms/step - loss: 0.0066 - accuracy: 0.9995
Epoch 5/5
64/64 [==============================] - 18s 274ms/step - loss: 0.0019 - accuracy: 1.0000
<tensorflow.python.keras.callbacks.History at 0x7f93dda23850>
```

```
test_results = cnn_model_changed_dropout.evaluate(test_inputs, y_test, batch_size=BATCH_SIZE)
print(test_results)
```

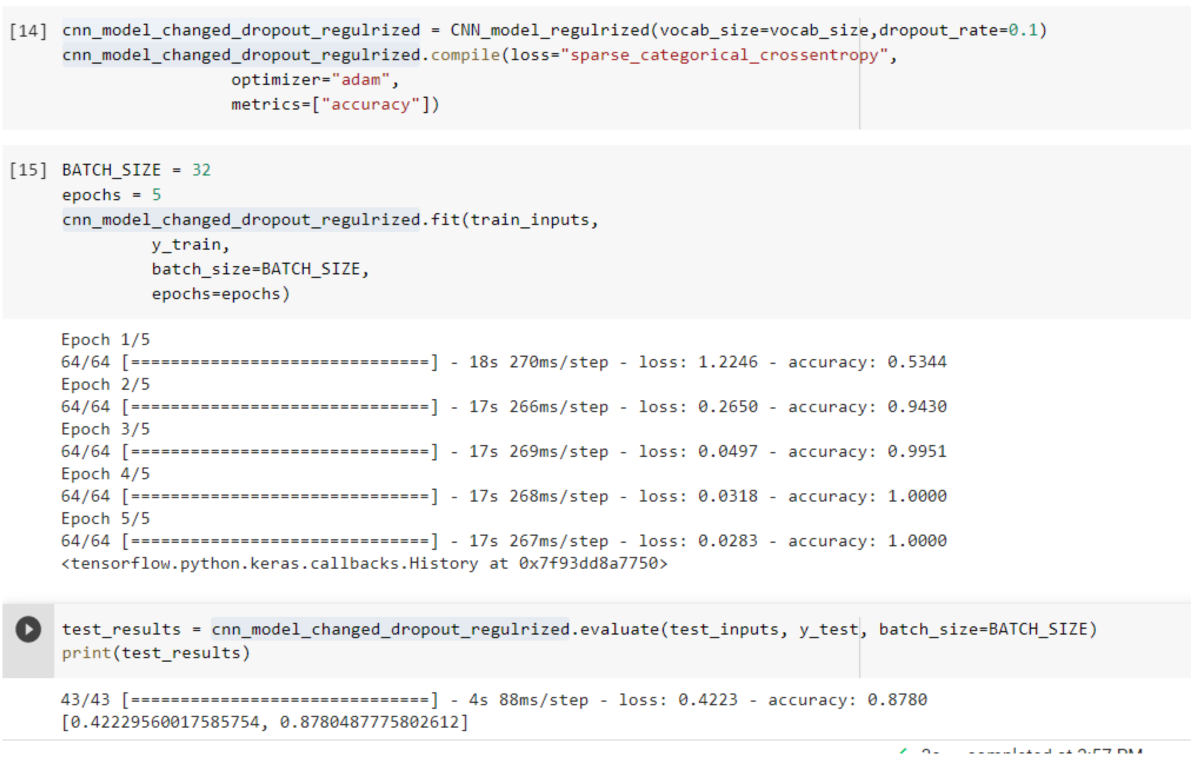
```
43/43 [==============================] - 4s 90ms/step - loss: 0.3738 - accuracy: 0.8780
[0.3738043010234833, 0.8780487775802612]
```

Comment: when dropout rate = 0.1 is better than dropout rate = 0.2

Sixth trial

In this trial we tried to put regularizer L1-L2 with dropout rate = 0.1

By this change, we got 100% training accuracy and 87.8% test accuracy



```
[14] cnn_model_changed_dropout_regulrized = CNN_model_regulrized(vocab_size=vocab_size,dropout_rate=0.1)
     cnn_model_changed_dropout_regulrized.compile(loss="sparse_categorical_crossentropy",
                     optimizer="adam",
                     metrics=["accuracy"])
```

```
[15] BATCH_SIZE = 32
     epochs = 5
     cnn_model_changed_dropout_regulrized.fit(train_inputs,
              y_train,
              batch_size=BATCH_SIZE,
              epochs=epochs)
```

```
Epoch 1/5
64/64 [==============================] - 18s 270ms/step - loss: 1.2246 - accuracy: 0.5344
Epoch 2/5
64/64 [==============================] - 17s 266ms/step - loss: 0.2650 - accuracy: 0.9430
Epoch 3/5
64/64 [==============================] - 17s 269ms/step - loss: 0.0497 - accuracy: 0.9951
Epoch 4/5
64/64 [==============================] - 17s 268ms/step - loss: 0.0318 - accuracy: 1.0000
Epoch 5/5
64/64 [==============================] - 17s 267ms/step - loss: 0.0283 - accuracy: 1.0000
<tensorflow.python.keras.callbacks.History at 0x7f93dd8a7750>
```

```
test_results = cnn_model_changed_dropout_regulrized.evaluate(test_inputs, y_test, batch_size=BATCH_SIZE)
print(test_results)
```

```
43/43 [==============================] - 4s 88ms/step - loss: 0.4223 - accuracy: 0.8780
[0.42229560017585754, 0.8780487775802612]
```

FUN FACT: Using simple architecture achieves the highest accuracy

Transfer learning

Until now we did not get a high-test accuracy, so we think of what about using transfer learning

What model we will use ?

For the model, we will use BERT through a library called HuggingFace. HuggingFace is an open-source provider of natural language processing (NLP) technologies. Their library offers ways to easily build, train and deploy state-of-the-art NLP models.

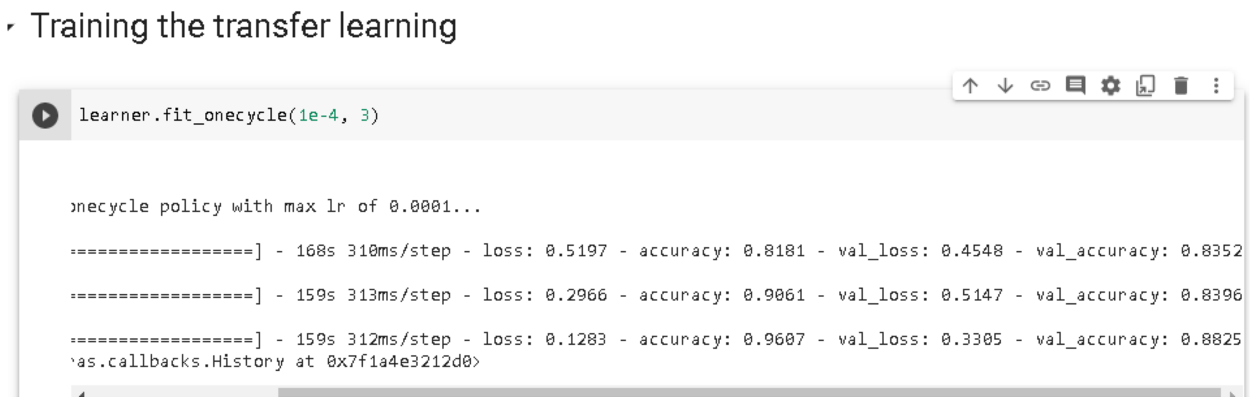
-HuggingFace provides a large variety of pre-trained models, which can be found here.

One of hugging face library model that we will use "distilbert-base-uncased"

Zewail city of science and technology

Natural language processing NLP

Training

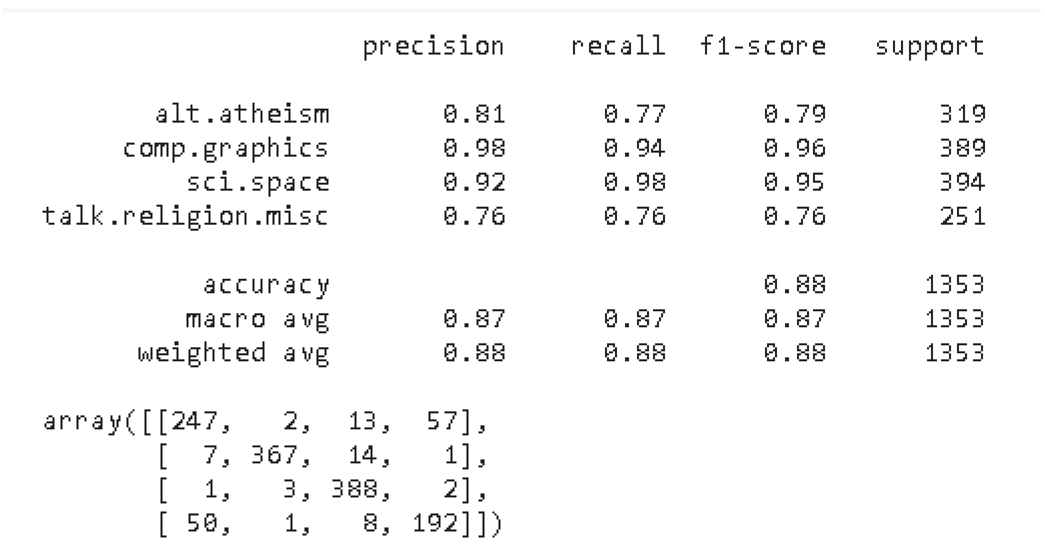


Training the transfer learning

```
learner.fit_onecycle(1e-4, 3)


onecycle policy with max lr of 0.0001...

=================] - 168s 310ms/step - loss: 0.5197 - accuracy: 0.8181 - val_loss: 0.4548 - val_accuracy: 0.8352

=================] - 159s 313ms/step - loss: 0.2966 - accuracy: 0.9061 - val_loss: 0.5147 - val_accuracy: 0.8396

=================] - 159s 312ms/step - loss: 0.1283 - accuracy: 0.9607 - val_loss: 0.3305 - val_accuracy: 0.8825
as.callbacks.History at 0x7f1a4e3212d0>
```

Accuracy is 96 which is lower than our SVM model and CNN model

Model evaluation



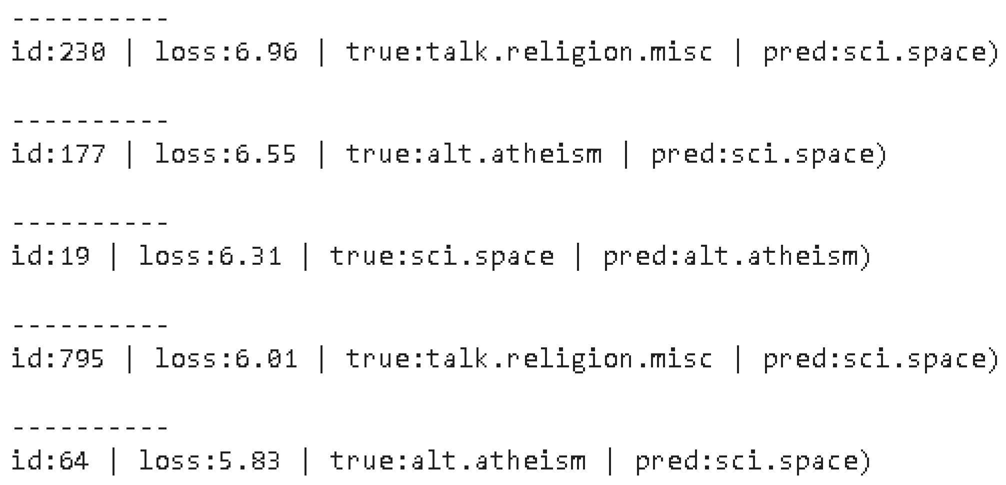
```
                    precision    recall  f1-score   support

       alt.atheism       0.81      0.77      0.79       319
     comp.graphics       0.98      0.94      0.96       389
         sci.space       0.92      0.98      0.95       394
talk.religion.misc       0.76      0.76      0.76       251

          accuracy                           0.88      1353
         macro avg       0.87      0.87      0.87      1353
      weighted avg       0.88      0.88      0.88      1353

array([[247,   2,  13,  57],
       [  7, 367,  14,   1],
       [  1,   3, 388,   2],
       [ 50,   1,   8, 192]])
```

-The precision is the fraction of relevant instances among the retrieved instances higher value at comp.graphics lower at Talk.religion.misc

-The recall is s the fraction of relevant instances that were retrieved higher at sci.space and lower at talk.religion.misc

-On average the precision and recall and f1-score is 87.

What is the most difficult categories to classify ?



```
----------
id:230 | loss:6.96 | true:talk.religion.misc | pred:sci.space)

----------
id:177 | loss:6.55 | true:alt.atheism | pred:sci.space)

----------
id:19 | loss:6.31 | true:sci.space | pred:alt.atheism)

----------
id:795 | loss:6.01 | true:talk.religion.misc | pred:sci.space)

----------
id:64 | loss:5.83 | true:alt.atheism | pred:sci.space)
```

Conclusion

In conclusion we used the NLP techniques to be able to represent the documents of the dataset properly to be able to train different models on the dataset such as machine learning model SVM , deep learning model such as CNN , and Transfer Learning.

And the result showed that the first model SVM with regulation parameter C=0.7 and CNN with the defined layers their test accuracy was close almost 88% to each other , but in case of transfer learning it result in training accuracy 88% which is very close to SVM and CNN model.