VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Department of Computer Engineering

Mini Project Report on

MULTI-LABEL TEXT CLASSIFICATION WITH KERAS

Under the subject: Natural Language Processing (NLP)

Submitted by

Manish Shinde D17C 58

Akshay Thite D17C D17C 68

Abhay Tiwari D17C 69

Under the guidance of

Subject Lab Teacher

Sharmila Sengupta, Ashwini Abhijit Gaikwad

(2020-2021)

NLP MINI PROJECT REPORT

TITLE : MULTI-LABEL TEXT CLASSIFICATION WITH KERAS

GROUP MEMBERS :

1. MANISH SHINDE (D17C - 58)
2. AKSHAY THITE (D17C - 68)
3. ABHAY TIWARI (D17C - 69)

1. INTRODUCTION :

We develop a text classification model with multiple outputs. We will be developing a text classification model that analyzes a textual comment and predicts multiple labels associated with the comment. The multi-label classification problem is actually a subset of multiple output models. The approach explained can be extended to perform general multi-label classification. For instance you can solve a classification problem where you have an image as input and you want to predict the image category and image description. At this point, it is important to explain the difference between a multi-class classification problem and a multi-label classification. In multi-class classification problems, an instance or a record can belong to one and only one of the multiple output classes. For instance, in the sentiment analysis problem, a text review could be either "good", "bad", or "average". It could not be both "good" and "average" at the same time. On the other hand in multi-label classification problems, an instance can have multiple outputs at the same time. For instance, in the text classification problem that we are going to solve, a comment can have multiple tags. These tags include "toxic", "obscene", "insulting", etc., at the same time.

2. DESCRIPTION :

MULTI-LABEL CLASSIFICATION MODEL :

The underlying concept is apparent in the name – multi-label classification. Here, an instance/record can have multiple labels and the number of labels per instance is not fixed. Take a look at the below tables,

where ‘X’ represents the input variables and ‘y’ represents the target variables (which we are predicting):

Table 1

X	y
X_1	t_1
X_2	t_2
X_3	t_1
X_4	t_2
X_5	t_1

Binary Classification

Table 2

X	y
X_1	t_2
X_2	t_3
X_3	t_4
X_4	t_1
X_5	t_3

Multi-class Classification

Table 3

X	y
X_1	$[t_2, t_5]$
X_2	$[t_1, t_2, t_3, t_4]$
X_3	$[t_3]$
X_3	$[t_2, t_4]$
X_3	$[t_1, t_3, t_4]$

Multi-label Classification

- ● ‘y’ is a binary target variable in Table 1. Hence, there are only two labels – t1 and t2 ● ‘y’ contains more than two labels in Table 2. But, notice how there is *only one label for every input* in both these tables
- ● You must have guessed why Table 3 stands out. We have multiple tags here, not just across the table, but for individual inputs as well

We cannot apply traditional classification algorithms directly on this kind of dataset. This is because these algorithms expect a single label for every input, when instead we have multiple labels. Multi-label classification is a predictive modeling task that involves predicting zero or more mutually non-exclusive class labels. Neural network models can be configured for multi-label classification tasks. A text classification model is to be developed with multiple outputs. This text classification model will take a statement as an input and will classify this statement. This classification will be done by predicting multiple labels associated with the statement. The approach explained can be extended to perform general multi-label classification. For instance you can solve a classification problem where you have an image as input and you want to predict the image category and image description. At this point, it is important to explain the difference between a multi-class classification problem and a multi-label classification. In multi-class classification problems, an instance or a record can belong to one and only one of the multiple output classes. For instance, in the sentiment analysis problem, a text review could be either "good", "bad", or "average". It could not be both "good" and "average" at the same time. On the other hand in multi-label classification problems, an instance can have multiple outputs at the same time. For instance, in the text classification problem that we are going to solve, a comment can have multiple tags. These tags include "toxic", "obscene", "insulting", etc., at the same time.

THE DATASET :

The dataset contains comments from Wikipedia's talk page edits. There are six output labels for each comment: toxic, severe_toxic, obscene, threat, insult and identity_hate. A comment can belong to all of these categories or a subset of these categories, which makes it a multi-label classification problem.

The dataset can be downloaded from this Kaggle link. We will only use the "train.csv" file that contains 160,000 records. We downloaded the dataset and stored the CSV file in the local directory.

3. METHODOLOGY :

- ANALYSIS OF DATASET :

IMPORT LIBRARIES :

Import the required libraries and load the dataset into our application.

LOAD DATASET :

Load the dataset into the memory.

Remove all the records where any row contains a null value or empty string. Plot the dataset in the form of a 2 dimensional matrix and bar graph.

- CREATING MULTI-LABEL TEXT CLASSIFICATION MODELS :

There are two ways to create multi-label classification models: Using single dense output layer and using multiple dense output layers. In the first approach, we can use a single dense layer with six outputs with sigmoid activation functions and binary cross entropy loss functions. Each neuron in the output dense layer will represent one of the six output labels. The sigmoid activation function will return a value between 0 and 1 for each neuron. If any neuron's output value is greater than 0.5, it is assumed that the comment belongs to the class represented by that particular neuron. In the second approach we will create one dense output layer for each label. We will have a total of 6 dense layers in the output. Each layer will have its own sigmoid function.

MULTI-LABEL TEXT CLASSIFICATION MODEL WITH SINGLE OUTPUT LAYER :

In this section, we will create a multi-label text classification model with a single output layer. As always, the first step in the text classification model is to create a function responsible for cleaning the text. In the next step we will create our input and output set. Here we do not need to perform any one-hot encoding because our output labels are already in the form of one-hot encoded vectors. In the next step, we will divide our data into training and test sets. We need to convert text inputs into embedded vectors. We will be using GloVe word embeddings to convert text inputs to their numeric counterparts. The following script creates the model. Our model will have one input layer, one embedding layer, one LSTM layer with 128 neurons and one output layer with 6 neurons since we have 6 labels in the output. Now, the training of the model is done. We will train our model for 5 epochs. Now evaluate the model on the test set. Our model achieves an accuracy of around 98%. Finally, we will plot the loss and accuracy values for training and test sets to see if our model is overfitting.

MULTI-LABEL TEXT CLASSIFICATION MODEL WITH MULTIPLE OUTPUT LAYERS :
In this section we will create a multi-label text classification model where each output label will have a dedicated output dense layer. The second step is to create inputs and output for the model. The input to the model will be the text comments, whereas the output will be six labels. Then divide the data into

training and testing sets. The y variable contains the combined output from 6 labels. However, we want to create an individual output layer for each label. We will create 6 variables that store individual labels from the training data and 6 variables that store individual label values for the test data. The next step is to convert textual inputs to embedded vectors. We will use the GloVe word embeddings. Now creation of the model is performed. Our model will have one input layer, one embedding layer followed by one LSTM layer with 128 neurons. The output from the LSTM layer will be used as the input to the 6 dense output layers. Each output layer will have 1 neuron with sigmoid activation function. Each output will predict integer value between 1 and 0 for the corresponding label. Then, training of the model is done. An accuracy of only 31% is achieved on the test set via multiple output layers. Finally, plot the graph to check the accuracy of the model. The model starts to overfit after the first epochs and hence we get a poor performance on unseen test sets.

4. CONCLUSION :

Multi-label text classification is one of the most common text classification problems. We studied two deep learning approaches for multi-label text classification. In the first approach we used a single dense output layer with multiple neurons where each neuron represented one label. In the second approach, we created separate dense layers for each label with one neuron. Results show that in our case, a single output layer with multiple neurons works better than multiple output layers.

5. CODE AND OUTPUT :

```
from numpy import array
from keras.preprocessing.text import one_hot
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers.core import Activation, Dropout, Dense
from keras.layers import Flatten, LSTM
from keras.layers import GlobalMaxPooling1D
from keras.models import Model
from keras.layers.embeddings import Embedding
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.layers import Input
from keras.layers.merge import Concatenate
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
toxic_comments = pd.read_csv("/content/drive/My Drive/Colab Datasets/toxic_comments.csv")

toxic_comments = pd.read_csv("/content/drive/My Drive/Colab Datasets/toxic_comments.csv")
```

Output:

```
(159571,8)
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

```
filter = toxic_comments["comment_text"] != ""
toxic_comments = toxic_comments[filter]
toxic_comments = toxic_comments.dropna()
print(toxic_comments["comment_text"][168])
```

Output:

You should be fired, you're a moronic wimp who is too lazy to do research. It makes me sick that people like you exist in this world.

```
print("Toxic:" + str(toxic_comments["toxic"][168]))
print("Severe_toxic:" + str(toxic_comments["severe_toxic"][168]))
print("Obscene:" + str(toxic_comments["obscene"][168]))
print("Threat:" + str(toxic_comments["threat"][168]))
print("Insult:" + str(toxic_comments["insult"][168]))
print("Identity_hate:" + str(toxic_comments["identity_hate"][168]))
```

Output:

Toxic:1

Severe_toxic:0

Obscene:0

Threat:0

Insult:1

Identity_hate:0

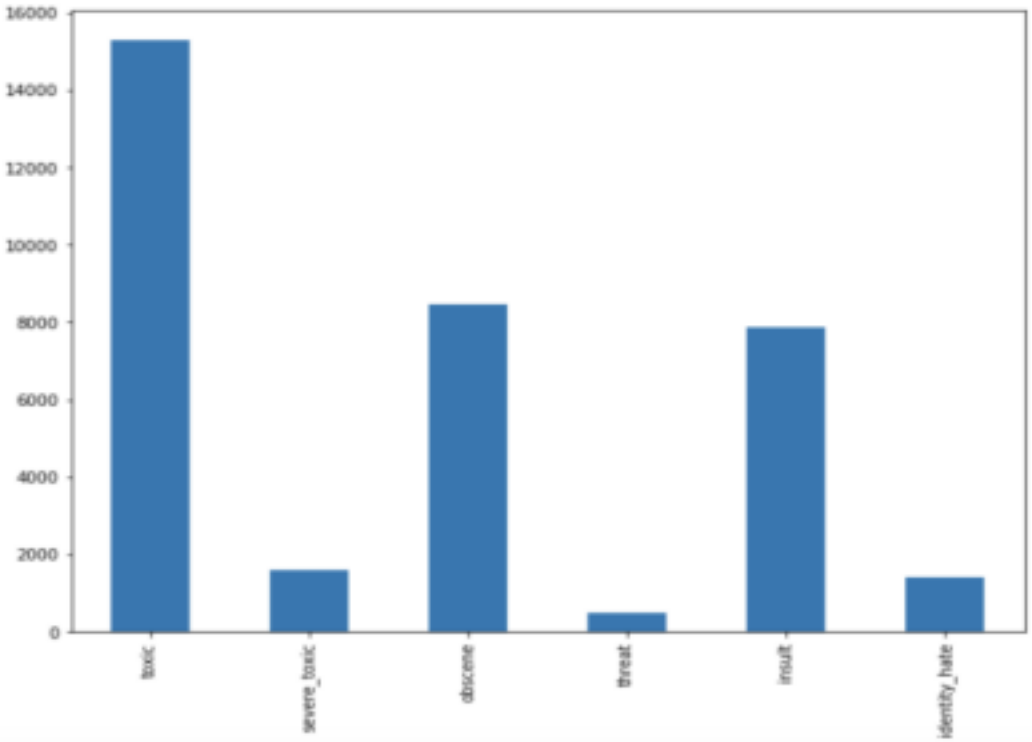
```
toxic_comments_labels = toxic_comments[["toxic", "severe_toxic", "obscene", "threat", "insult",
"identity_hate"]]
toxic_comments_labels.head()
```

Output:

	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0

```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 10
fig_size[1] = 8
plt.rcParams["figure.figsize"] = fig_size
toxic_comments_labels.sum(axis=0).plot.bar()
```

Output:



Multi-label Text Classification Model with Single Output Layer :

```
def preprocess_text(sen):
    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sen)
    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)
    # Removing multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)
    return sentence
X = []
sentences = list(toxic_comments["comment_text"])
for sen in sentences:
    X.append(preprocess_text(sen))
y = toxic_comments_labels.valuesX_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=42)
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
vocab_size = len(tokenizer.word_index) + 1
maxlen = 200
X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
from numpy import array
from numpy import asarray
from numpy import zeros
embeddings_dictionary = dict()
glove_file = open('/content/drive/My Drive/Colab Datasets/glove.6B.100d.txt', encoding="utf8")
for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary[word] = vector_dimensions
glove_file.close()
embedding_matrix = zeros((vocab_size, 100))
for word, index in tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
deep_inputs = Input(shape=(maxlen,))
```

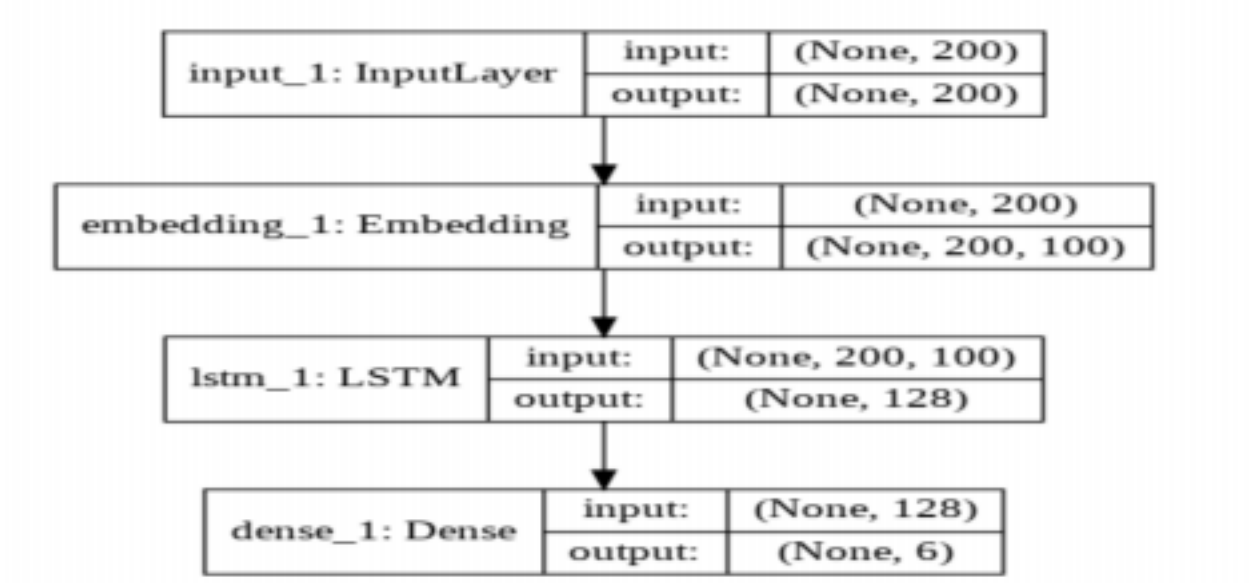
```
embedding_layer = Embedding(vocab_size, 100, weights=[embedding_matrix],
trainable=False)(deep_inputs)
LSTM_Layer_1 = LSTM(128)(embedding_layer)
dense_layer_1 = Dense(6, activation='sigmoid')(LSTM_Layer_1)
model = Model(inputs=deep_inputs, outputs=dense_layer_1)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['acc'])
print(model.summary())
```

Output:

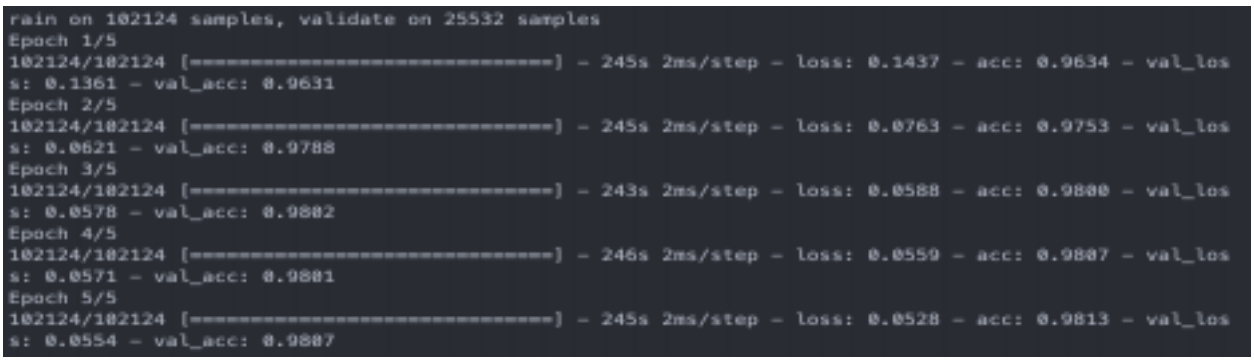
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 200)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 200, 100)          14824300
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               117248
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 14,942,322
Trainable params: 118,022
Non-trainable params: 14,824,300
```

```
from keras.utils import plot_model
plot_model(model, to_file='model_plot4a.png', show_shapes=True, show_layer_names=True)
```

Output:




```
history = model.fit(X_train, y_train, batch_size=128, epochs=5, verbose=1, validation_split=0.2)
```




```
score = model.evaluate(X_test, y_test, verbose=1)
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

Output:

31915/31915 [==========] - 108s 3ms/step

Test Score: 0.054090796736467786

Test Accuracy: 0.9810642735274182

```
import matplotlib.pyplot as pl
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```

Output:

```
test_predictions=model.predict(X_test)
test_predictions_int=test_predictions.astype(int)
test_predictions[39]
```

Output :

```
array([0.9960841 , 0.5188308 , 0.98905927, 0.06231129, 0.9261966 ,
       0.21072385], dtype=float32)
y_test[39]
```

Output :

array([1, 0, 1, 0, 1, 0])

text_test[39]

Output :

You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk You re jerk JACKASS

APPLICATIONS :

- Security purposes
- Feedback from the users

REFERENCES :

- https://towardsdatascience.com/multi-label-classification-using-bag-of-words-bow-and-tf-idf-4f95858740e5
- https://stackabuse.com/python-for-nlp-multi-label-text-classification-with-keras/