In [1]:
```
#importing necessary libraries
import keras
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import Embedding,LSTM,Dropout,Dense
from tensorflow.keras .preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
import numpy as np

#Loading the IMDB file
if __name__ == "__main__":
    import urllib.request as req
    import tarfile
    import os

    imdb_url = "http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"

    save_filename = "aclImdb_v1.tar.gz"
    if not os.path.exists(save_filename):
        req.urlretrieve(imdb_url, save_filename)

    imdb_folder = "aclImdb"
    if not os.path.exists(imdb_folder):
        with tarfile.open(save_filename) as tar:
            tar.extractall()
    loaded_model_NLP= keras.models.load_model("20868189_NLP_model")
    import numpy as np
    import re
#loading the test set into a test file alongside its reviews
    def get_test_file(data_folder="/test"):
        reviews = []
        labels = []
        for index,sentiment in enumerate(["/neg/", "/pos/"]):
            path = imdb_folder + data_folder + sentiment
            for filename in sorted(os.listdir(path)):
                with open(path + filename, 'r') as f:
                    review = f.read()
                    review = review.lower()
                    review = review.replace("<br />", " ")
                    review = re.sub(r"[^a-z ]", " ", review)
                    review = re.sub(r" +", " ", review)
                    reviews.append(review)

                    label = [0,0]
                    label[index] = 1
                    labels.append(label)

        return reviews, np.array(labels)
    voc_size=10000 #vocabulary size
    sent_len=100 #sentence length
    test_reviews, test_labels=get_test_file()#loading my test set into test_review and test_label sets
    onehot_reptt=[one_hot(words, voc_size) for words in test_reviews] #vectorization of my train review
s
    test_reviews=pad_sequences(onehot_reptt , maxlen=sent_len) #padding the test reviews to be equal to
 the sentence length
    y_pred=loaded_model_NLP.predict(test_reviews) #Performing prediction with the test review set
    score=loaded_model_NLP.evaluate(test_labels, y_pred,verbose=False)#Evaluating the model with the ac
tual labels vs predicted labels
    print(score)#Printing the evaluation score
```

```
WARNING:tensorflow:Model was constructed with shape (None, 100) for input Tensor("input_1:0", shape=
(None, 100), dtype=float32), but it was called on an input with incompatible shape (None, 2).
WARNING:tensorflow:Model was constructed with shape (None, 100) for input Tensor("input_1:0", shape=
(None, 100), dtype=float32), but it was called on an input with incompatible shape (None, 2).
[0.12165463715791702, 0.9983999729156494]
```

In [ ]: