Βαθιά Μάθηση και Ανάλυση Πολυμεσικών Δεδομένων

Ευαγγελία Κυριακοπούλου

ΑΕΜ : 110

Contents:

1. Introduction

Για την εργασία επιλέχθηκε ένα dataset για Fake News Detection Analysis μέσω του Kaggle. Το dataset αυτό περιέχει μια συλλογή 20800 αγγλικών άρθρων. Η δομή του είναι (20800,5), δηλαδή 20800 γραμμών και 5 στηλών-χαρακτηριστικών. Συγκεκριμένα τα χαρακτηριστικά των στηλών περιλαμβάνουν :
id: unique identifier for a news article
title: title of news article
author: author of the news article
text: the text of the article. may be incomplete
tag: a tag that marks the article as potentially untrustworthy
1: unreliable
0: reliable
Τη συγκεκριμένη ανάλυση προσεγγίσαμε με χρήση Βαθιών Αναδρομικών Νευρωνικών Δικτύων (Deep Recurrent Neural Networks). Για την συγγραφή και την εκτέλεση του κώδικα έγινε χρήση του περιβάλλοντος Jupyter Notebook.

2. Importing necessary libraries

Το πρόγραμμα υλοποιήθηκε στη γλώσσα προγραμματισμού Python με χρήση των παρακάτω βιβλιοθηκών :

Import Modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
import re
import nltk
import warnings
%matplotlib inline

from tensorflow import keras
from keras.preprocessing.text import Tokenizer
# from keras_preprocessing.sequence import pad_sequences
from keras.utils import pad_sequences

from sklearn.model_selection import train_test_split
from keras.layers import LSTM, Dropout, Dense, Embedding , SimpleRNN
from keras import Sequential


warnings.filterwarnings('ignore')
```

Εικόνα 1 Φόρτωση Βιβλιοθηκών

3. Loading the data & 4. Data preprocessing

Μετά την εισαγωγή των βιβλιοθηκών ακολούθησε η φόρτωση των δεδομένων εκπαίδευσης. Το dataset αυτό μετέπειτα θα χωριστεί και σε δεδομένα ελέγχου. Έπειτα ξεκίνησε η προ επεξεργασία των δεδομένων έτσι ώστε να έχουμε ένα καθαρό dataset έτοιμο για την διαδικασία της εκπαίδευσης.

Αρχικά αφαιρέθηκαν οι περιττές στήλες, συγκεκριμένα οι στήλες με όνομα ‘ id ’, ‘title’, ‘author’ και οι null τιμές. Στη συνέχεια αφαιρέθηκαν ειδικοί και μη αλφαριθμητικοί χαρακτήρες, χαρακτήρες νέας γραμμής, σημεία στίξης του κειμένου και πολλαπλά κενά αντικαθιστώντας τα σε μεμονωμένα κενά, έγινε μετατροπή των γραμμάτων σε πεζά και τέλος ακολούθησε η αφαίρεση stopwords της αγγλικής γλώσσας (the, is, and,…,) καθώς τα κείμενα είναι γραμμένα σ αυτή. Συνεπώς το dataset που έχουμε να διαχειριστούμε πια είναι της μορφής (20761,3),

δηλαδή 20761 δείγματα με τα χαρακτηριστικά να είναι τα label 0,1 το ανεπεξέργαστο και επεξεργασμένο κείμενο. Παρακάτω βλέπουμε και τον σχετικό κώδικα :

Load The Dataset

```
train = pd.read_csv('/kaggle/input/fake-news/train.csv')
```

```
train.shape
```

```
(20800, 5)
```

```
train.head()
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

Εικόνα 2 Φόρτωση Δεδομένων

Data Preprocesing

```
# drop unnecessary columns
train = train.drop(columns=['id', 'title', 'author'], axis=1)
```

```
# drop null values
train = train.dropna(axis=0)
```

```
train.isnull().sum()
```

```
text     0
label    0
dtype: int64
```

```
len(train)
```

```
20761
```

```
train.shape
```

```
(20761, 2)
```

Εικόνα 3 Αφαίρεση null τιμών

```
# remove special characters and punctuations
train['clean_news'] = train['text'].str.lower()
train['clean_news']
```

```
0        house dem aide: we didn’t even see comey’s let...
1        ever get the feeling your life circles the rou...
2        why the truth might get you fired october 29, ...
3        videos 15 civilians killed in single us airstr...
4        print \nan iranian woman has been sentenced to...
                              ...
20795    rapper t. i. unloaded on black celebrities who...
20796    when the green bay packers lost to the washing...
20797    the macy’s of today grew from the union of sev...
20798    nato, russia to hold parallel exercises in bal...
20799      david swanson is an author, activist, journa...
Name: clean_news, Length: 20761, dtype: object
```

+ Code + Markdown

```
train['clean_news'] = train['clean_news'].str.replace('[^A-Za-z0-9\s]', '')
train['clean_news'] = train['clean_news'].str.replace('\n', '')
train['clean_news'] = train['clean_news'].str.replace('\s+', ' ')
train['clean_news']
```

```
0        house dem aide we didnt even see comeys letter...
1        ever get the feeling your life circles the rou...
2        why the truth might get you fired october 29 2...
3        videos 15 civilians killed in single us airstr...
4        print an iranian woman has been sentenced to s...
                              ...
20795    rapper t i unloaded on black celebrities who m...
20796    when the green bay packers lost to the washing...
20797    the macys of today grew from the union of seve...
20798    nato russia to hold parallel exercises in balk...
20799     david swanson is an author activist journalis...
Name: clean_news, Length: 20761, dtype: object
```

```
# remove stopwords
from nltk.corpus import stopwords
stop = stopwords.words('english')
train['clean_news'] = train['clean_news'].apply(lambda x: " ".join([word for word in x.split() if word not in stop]))
train.head()
```

	text	label	clean_news
0	House Dem Aide: We Didn't Even See Comey's Let...	1	house dem aide didnt even see comeys letter ja...
1	Ever get the feeling your life circles the rou...	0	ever get feeling life circles roundabout rathe...
2	Why the Truth Might Get You Fired October 29, ...	1	truth might get fired october 29 2016 tension ...
3	Videos 15 Civilians Killed In Single US Airstr...	1	videos 15 civilians killed single us airstrike...
4	Print \nAn Iranian woman has been sentenced to...	1	print iranian woman sentenced six years prison...

+ Code + Markdown

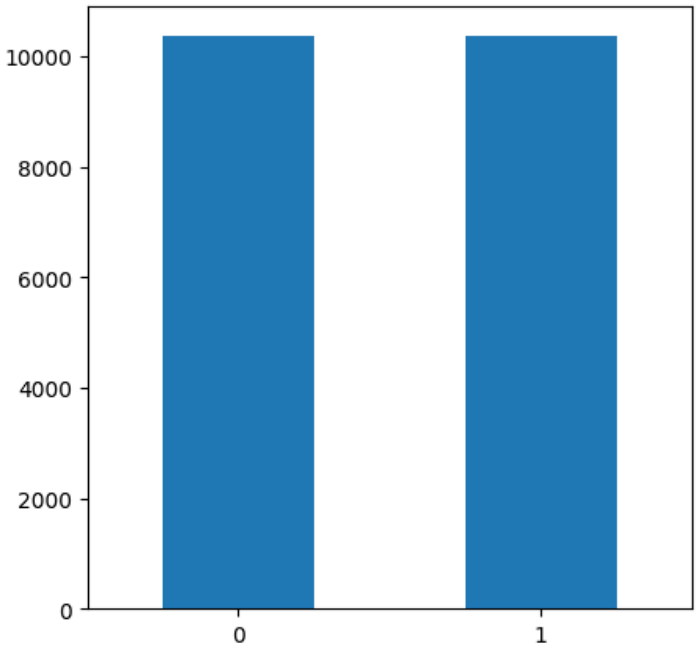
```
train.shape
```

```
(20761, 3)
```

Εικόνα 4 Αφαίρεση χαρακτήρων/Δημιουργία "καθαρού" κειμένου

Έπειτα εκτυπώθηκαν οι τιμές των labels 0 και 1. Όπως φαίνεται και από την επόμενη *Εικόνα 5* οι τιμές των αξιόπιστων και αναξιόπιστων δεδομένων είναι πάνω κάτω οι ίδιες, στην τιμή 0 αντιστοιχούν 10387 αξιόπιστες εγγραφές, ενώ στην τιμή 1 αντιστοιχούν 10374 αναξιόπιστες εγγραφές. Συνεπώς μπορούμε να πούμε ότι έχουμε να κάνουμε μ' ένα ισορροπημένο dataset.

```
plt.figure(figsize=(5,5))
train['label'].value_counts().nlargest(10).plot(kind='bar')
plt.xticks(rotation=360)
plt.show()
```



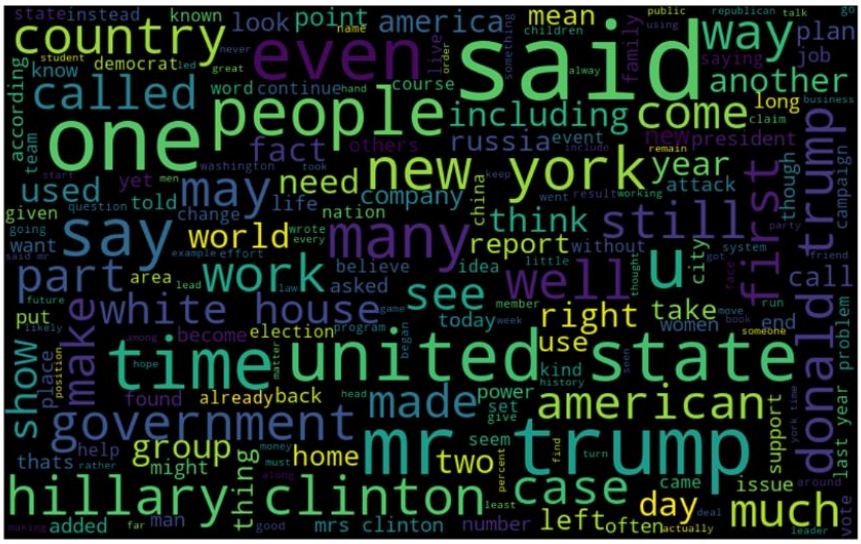
Εικόνα 5 Εκτύπωση Label

Στην συνέχεια ακολουθήσε η εκτύπωση των συχνότερων λέξεων που εμφανίζονται στα καθαρά δεδομένα γενικά και ειδικά στα αναξιόπιστα και αναξιόπιστα δεδομένα. Εδώ όσο μεγαλύτερη είναι η λέξη μέσα στην εικόνα τόσο πιο συχνή είναι και η χρήση της στα άρθρα/κείμενα.

```
# visualize the frequent words
all_words = " ".join([sentence for sentence in train['clean_news']])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15, 9))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

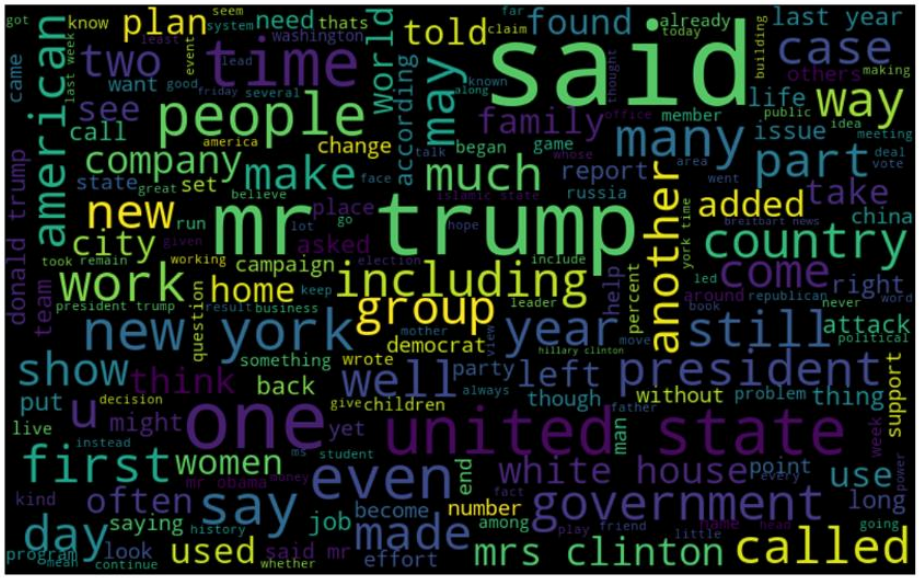


Εικόνα 6 Visualization of frequent words

```
# visualize the frequent words for Real news
all_words = " ".join([sentence for sentence in train['clean_news'][train['label']==0]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15, 9))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

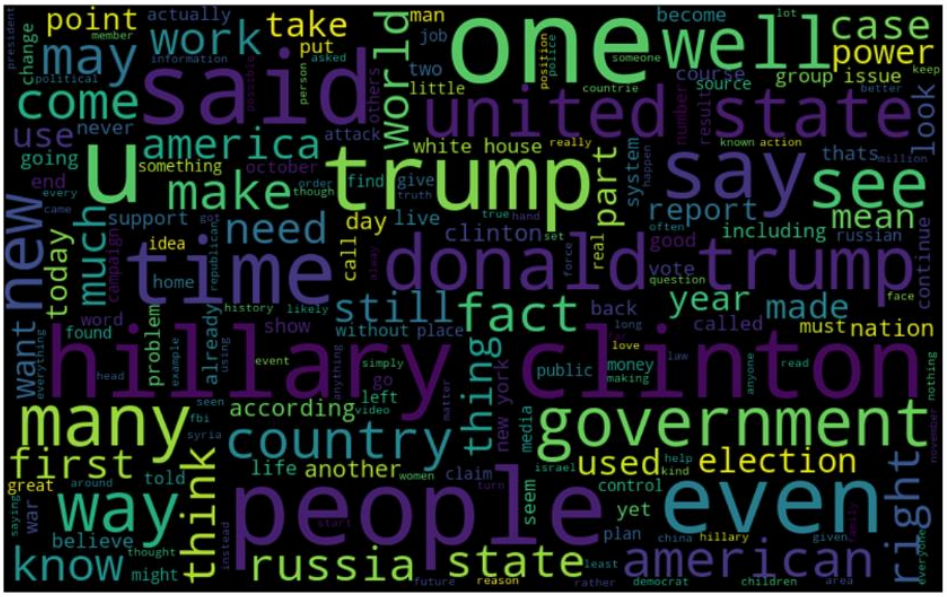


Εικόνα 7 Visualization for reliable news

```
# visualize the frequent words for fake news
all_words = " ".join([sentence for sentence in train['clean_news'][train['label']==1]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15, 9))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



Εικόνα 8 Visualization for unreliable news

Η επόμενη ενέργεια και μια από τις πιο σημαντικές στην επεξεργασία κειμένου είναι η μετατροπή των textual data σε numerical, δηλαδή η αναπαράσταση της λέξης σε αριθμό/διάνυσμα. Έτσι έγινε μετατροπή του κειμένου σε μια ακολουθία λέξεων(tokens). Αφού αναλύθηκε το κείμενο σε λέξεις κατασκευάστηκε ένα λεξικό που αντιστοιχεί κάθε μια λέξη σε έναν μοναδικό αριθμό. Οι μοναδικοί αυτοί αριθμοί αποθηκεύτηκαν στην μεταβλητή vocab_size με το μέγεθος του να είναι 199536.

```
# converting the textual data to numerical data
```

```
# tokenize text
tokenizer = Tokenizer()
tokenizer.fit_on_texts(train['clean_news'])
word_index = tokenizer.word_index
vocab_size = len(word_index)
vocab_size
```

```
199536
```

Εικόνα 9 Ttokenize Text

Ακολούθησε η τεχνική padding (γέμισμα) με την επιλογή γεμίσματος να είναι ο αριθμός μηδέν, εξασφαλίζοντας έτσι ότι όλες οι ακολουθίες θα έχουν το ίδιο μήκος, δηλαδή 500 όπως ορίσαμε. Η προσθήκη των μηδενικών γίνεται μετά την ακολουθία των λέξεων. Αν η ακολουθία υπερβαίνει το μέγιστο μήκος των 500, τότε γίνεται αποκοπή (truncation) των επιπλέον λέξεων.

```
# padding data
sequences = tokenizer.texts_to_sequences(train['clean_news'])
padded_seq = pad_sequences(sequences, maxlen=500, padding='post', truncating='post')
```

```
padded_seq.shape
```

```
(20761, 500)
```

Εικόνα 10 Μέθοδος Padding

Στη συνέχεια ακολουθεί η διανυσματική αναπαράσταση των λέξεων του κειμένου. Αρχικά δημιουργείται ο embedding index, που περιέχει τις διανυσματικές αναπαραστάσεις λέξεων από το προ εκπαιδευμένο μοντέλο glove.6b.100d.text. Το μοντέλο αυτό περιέχει έξι δισεκατομμύρια λέξεις (tokens) στις 100 διαστάσεις. Για κάθε γραμμή του αρχείου glove διαχωρίζονται οι τιμές με βάση το κενό διάστημα και αποθηκεύονται στην μεταβλητή word, ενώ τα αριθμητικά δεδομένα (τα embeddings) στη μεταβλητή coefs. Στο λεξικό αυτό embedding index, δημιουργούνται ζεύγη κλειδιών- τιμών, όπου το κλειδί αντιστοιχεί στη λέξη και η τιμή στα αριθμητικά δεδομένα (δηλ. τα embeddings). Τα αριθμητικά δεδομένα βρίσκονται στην μορφή NumPy array.

```
# create embedding index
embedding_index = {}
with open('/kaggle/input/glove6b/glove.6B.100d.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embedding_index[word] = coefs
```

+ Code + Markdown

```
len(coefs)
```

```
100
```

Εικόνα 11 Δημιουργία embedding_intex

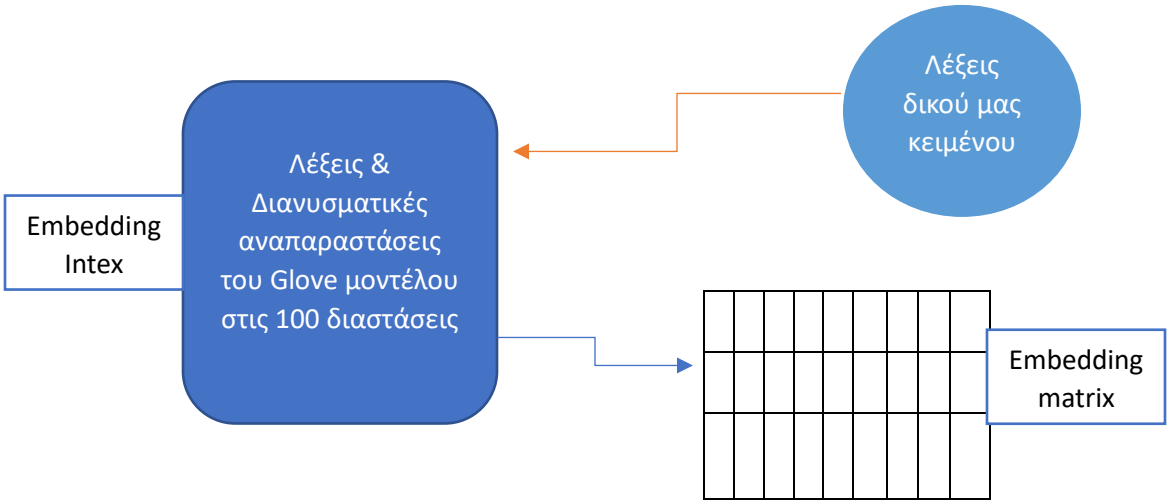
Ο embedding index τώρα θα χρησιμοποιηθεί για την δημιουργία του embedding matrix. Ο embedding matrix αρχικοποιείται με μηδενικές τιμές στον οποίον θα αποθηκευτούν οι διανυσματικές αναπαραστάσεις των λέξεων από το δικό μας κείμενο. Δηλαδή ο embedding matrix θα ελέγξει εάν οι λέξεις του κειμένου μας βρίσκονται εντός του embedding index. Εάν λοιπόν μια λέξη υπάρχει στον embedding index, τότε το αντίστοιχο διάνυσμα αναπαράστασης αντιγράφεται στην αντίστοιχη γραμμή του embedding matrix. Αν μια λέξη δεν υπάρχει τότε η αντίστοιχη γραμμή του embedding matrix παραμένει μηδενική και την λέξη αυτή την αναγνωρίζει ως άγνωστη. Το +1 προσθέτει μια επιπλέον γραμμή στον πίνακα για τις άγνωστες λέξεις, δηλαδή τις λέξεις που δεν υπάρχουν στο λεξικό του προ εκπαιδευμένου μοντέλου. Τέλος ο embedding matrix έχει διαστάσεις (199537,100). Κάθε γραμμή αντιστοιχεί σε μια μοναδική λέξη vocab_size= (199536 + 1) = 199537 και κάθε στήλη αντιστοιχεί στην 100-διανυσματική αναπαράσταση της μοναδικής λέξης.

```
# create embedding matrix
embedding_matrix = np.zeros((vocab_size+1, 100))
for word, i in word_index.items():
    embedding_vector = embedding_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
embedding_matrix.shape
```

```
(199537, 100)
```

Εικόνα 12 Δημιουργία embedding_matrix



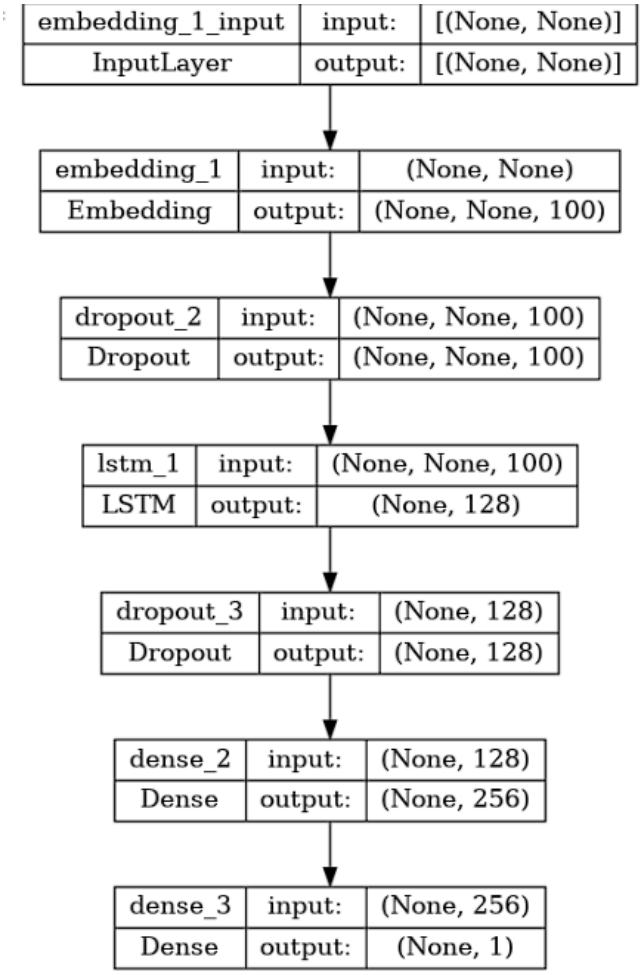
Τέλος ακολουθεί ο διαχωρισμός των δεδομένων εκπαίδευσης και σε δεδομένα ελέγχου μέσω της εντολής train_test_split με τα δεδομένα ελέγχου(test) και εκπαίδευσης(train) να είναι 50%.

5. Design models

Το πρώτο απλό αναδρομικό μοντέλο που δημιουργήθηκε και εκπαιδεύτηκε παρουσιάζεται παρακάτω μαζί με την αρχιτεκτονική και τα αποτελέσματα του. Στο μοντέλο χρησιμοποιήθηκε ο βελτιστοποίητής Adam, συνεπώς η εκπαίδευση του πραγματοποιήθηκε με την default τιμή του learning_rate , δηλαδή learning_rate=0.001.

```
model = Sequential([
    Embedding(vocab_size+1, 100, weights=[embedding_matrix], trainable=False),
    Dropout(0.2),
    LSTM(128),
    Dropout(0.2),
    Dense(256),
    Dense(1, activation='sigmoid')
])
```

Εικόνα 13 Αρχιτεκτονική Πρώτου Μοντέλου




```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 100)         19953700

 dropout_2 (Dropout)         (None, None, 100)         0

 lstm_1 (LSTM)               (None, 128)               117248

 dropout_3 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 256)               33024

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 20,104,229
Trainable params: 150,529
Non-trainable params: 19,953,700
_________________________________________________________________
```

Με την μέθοδο summary έχουμε την σύνοψη του μοντέλου χωρίς το στρώμα εισόδου. Το πρώτο στρώμα που βάζουμε στο μοντέλο είναι ένα στρώμα ενσωμάτωσης - embedding layer. Το στρώμα αυτό έχει μορφή (None, None, 100) και αναφέρεται στη μορφή των δεδομένων που εισέρχονται στο επίπεδο. Το πρώτο "None" αντιστοιχεί στο batch size , το δεύτερο "None" αντιστοιχεί στο μήκος της εισόδου (sequence length), και το 100 αντιστοιχεί στις διαστάσεις του ενσωματωμένου διανύσματος. Ακολουθεί Dropout Layer, θερίζοντας από το δίκτυο κόμβους με πιθανότητα p=0.2. Το LSTM layer δέχεται την είσοδο από το προηγούμενο επίπεδο και δημιουργεί μια κρυφή κατάσταση με μέγεθος 128. Ακολουθεί ξανά Dropout Layer, θερίζοντας από το δίκτυο κόμβους πάλι με πιθανότητα p=0.2. Τέλος έχουμε δύο Dense Layers με το πρώτο να είναι πλήρως συνδεδεμένο επίπεδο με 256 νευρώνες και το τελευταίο να είναι πάλι ένα πλήρως συνδεδεμένο επίπεδο με έναν νευρώνα που αντιπροσωπεύει την έξοδο του μοντέλου.

Μπορούμε επίσης να επιβεβαιώσουμε τον αριθμό των παραμέτρων :

Για το πρώτο embedding στρώμα έχουμε (vocab_size+1)*100 = (199536+1)*100 = 19953700

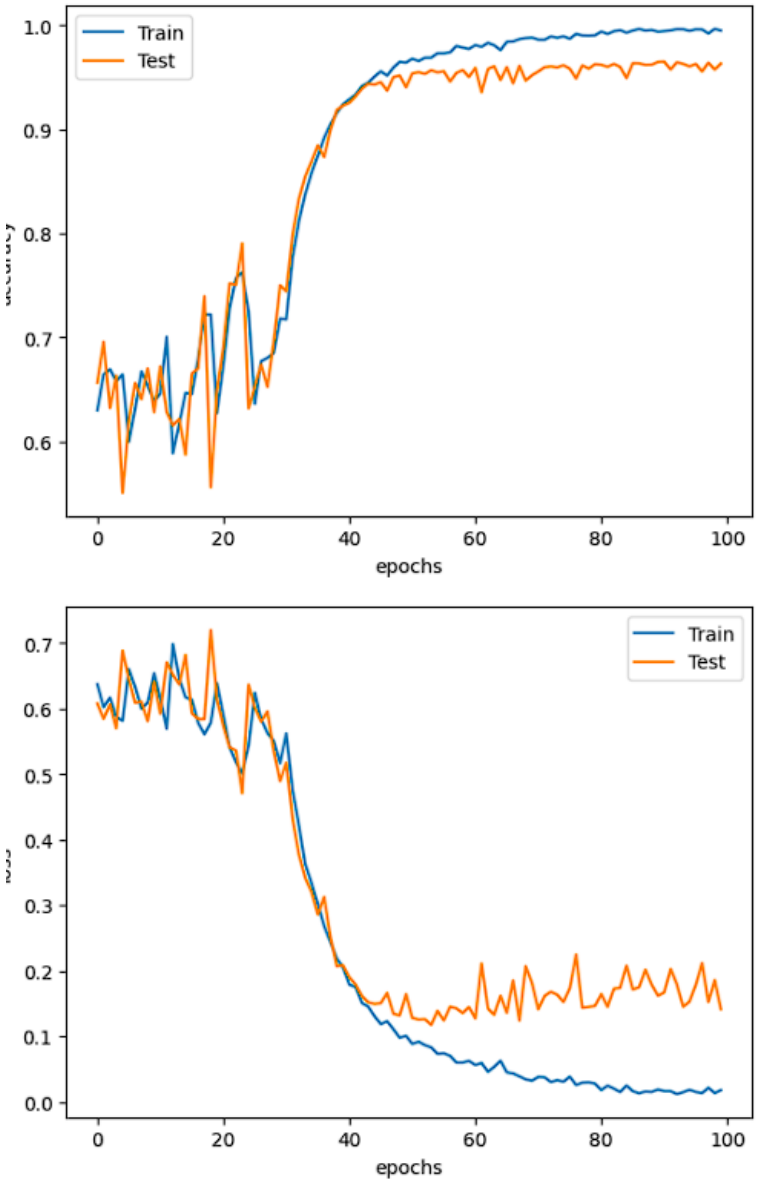
Στο LSTM στρώμα έχουμε για την είσοδο: 4*128*(100+1)= 51712 και για την επαναληπτική

σύνδεση: 4 * 128 * (128 + 1) = 65.536, άρα 51712+65.536 =117248

Για το πρώτο Dense Layer έχουμε 256 νευρώνες με 128 βάρη και μια πόλωση, άρα 256*(128+1) =33024 και για το δεύτερο έχουμε 1 νευρώνα με 256 βάρη και μία πόλωση άρα, 1*(256+1)= 257.

Έτσι έχουμε συνολικό αριθμό παραμέτρων 20104229 και οι 150529 είναι εκπαιδεύσιμοι καθώς έχουμε ορίσει οι παράμετροι του embedding layer να μην χρησιμοποιηθούν.

Το μοντέλο εκπαιδεύτηκε με 256 batches σε 100 εποχές. Έπειτα έγινε προβολή μέσω διαγραμμάτων του accuracy και του loss του train και test set αντίστοιχα. Τα αποτελέσματα φαίνονται παρακάτω :



Από την παραπάνω εικόνα μπορούμε να συμπεράνουμε το εξής: Το μοντέλο αποδίδει αρκετά καλά στα δεδομένα. Τα δύο σετ δεδομένων στην 40η εποχή αγγίζουν το υψηλότερο accuracy το οποίο είναι περίπου στο 95%. Μετά το πέρας της 40ης εποχής η εκπαίδευση συνεχίζεται ομαλά χωρίς προβλήματα over ή under fitting. Όμοια συμπεράσματα παρατηρούνται και μέσω του διαγράμματος του loss, καθώς έχουμε πτώση του training και test loss αντίστοιχα σε κάθε iteration.

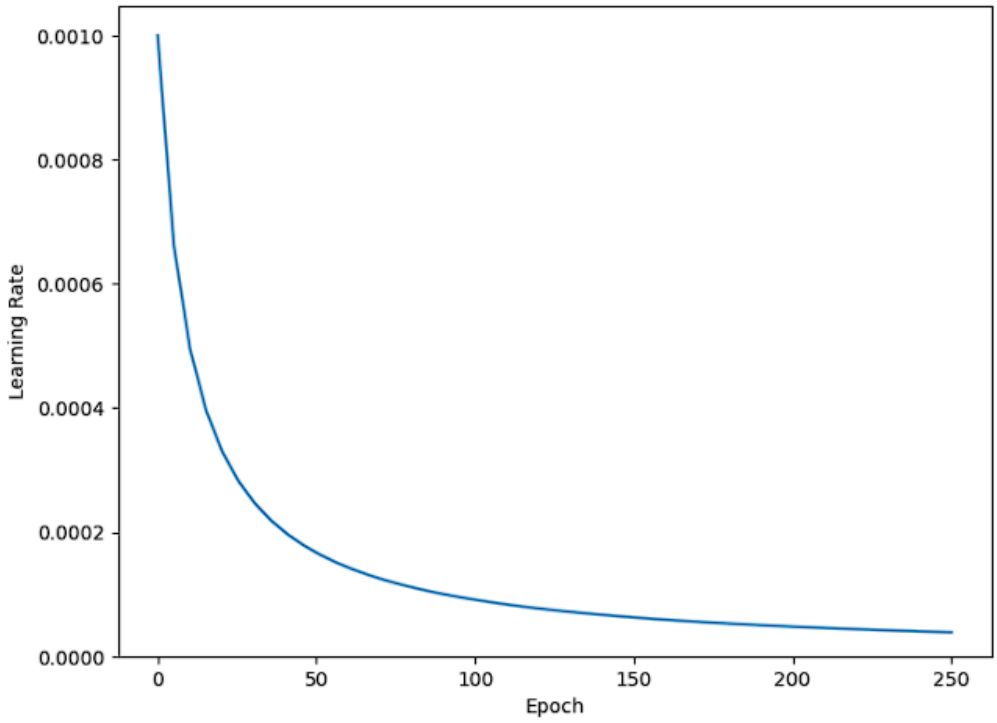
6. Compare Models

Στην συνέχεια ακολούθησαν και άλλες εκπαιδεύσεις και πειραματισμοί άλλων δυο μοντέλων κάνοντας χρήση SimpleRNNs και LSTMs με ίδια αρχιτεκτονική. Τα μοντέλα αυτά εκπαιδευτήκαν

την πρώτη φορά με την default τιμή του βελτιστοποίητή Adam και την δεύτερη φορά κάνοντας χρήση ενός χρονοδιαγράμματος μάθησης, έτσι ώστε το δίκτυο αρχικά να μαθαίνει γρήγορα και στη συνέχεια όσο προχωρούσε η εκπαίδευση του να γίνεται πιο προσεκτικό. Χρησιμοποιήθηκε η κλάση `InverseTimeDecay` και ορίστηκε ότι η έναρξη θα γίνει με έναν αρχικό ρυθμό συνέχειας και στις 10 εποχές αυτός θα μειώνει ώστε στις 10 εποχές να γίνει 1/2 του αρχικού, στις 20 έως 1/3 του αρχικού, κ.ο.κ

```
lr_schedule = tf.keras.optimizers.schedules.InverseTimeDecay(
  0.001,
  decay_steps=steps_per_epoch * 10,
  decay_rate=1,
  staircase=False)
```

```
step = np.linspace(0,10000)
lr = lr_schedule(step)
plt.figure(figsize = (8,6))
plt.plot(step/steps_per_epoch, lr)
plt.ylim([0,max(plt.ylim())])
plt.xlabel('Epoch')
_ = plt.ylabel('Learning Rate')
```



Εικόνα 14 Learning Rate Schedule

Οι αρχιτεκτονικές των μοντέλων φαίνονται παρακάτω :

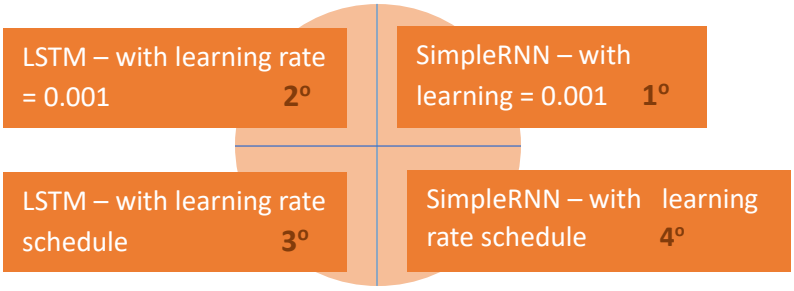
```
model = Sequential([
    Embedding(vocab_size+1, 100, weights=[embedding_matrix], trainable=False),
    Dropout(0.2),
    LSTM(128,return_sequences=True),
    Dropout(0.2),
    LSTM(128),
    Dropout(0.1),
    Dense(512),
    Dropout(0.2),
    Dense(256),
    Dropout(0.1),
    Dense(256),
    Dense(1, activation='sigmoid')
])
```

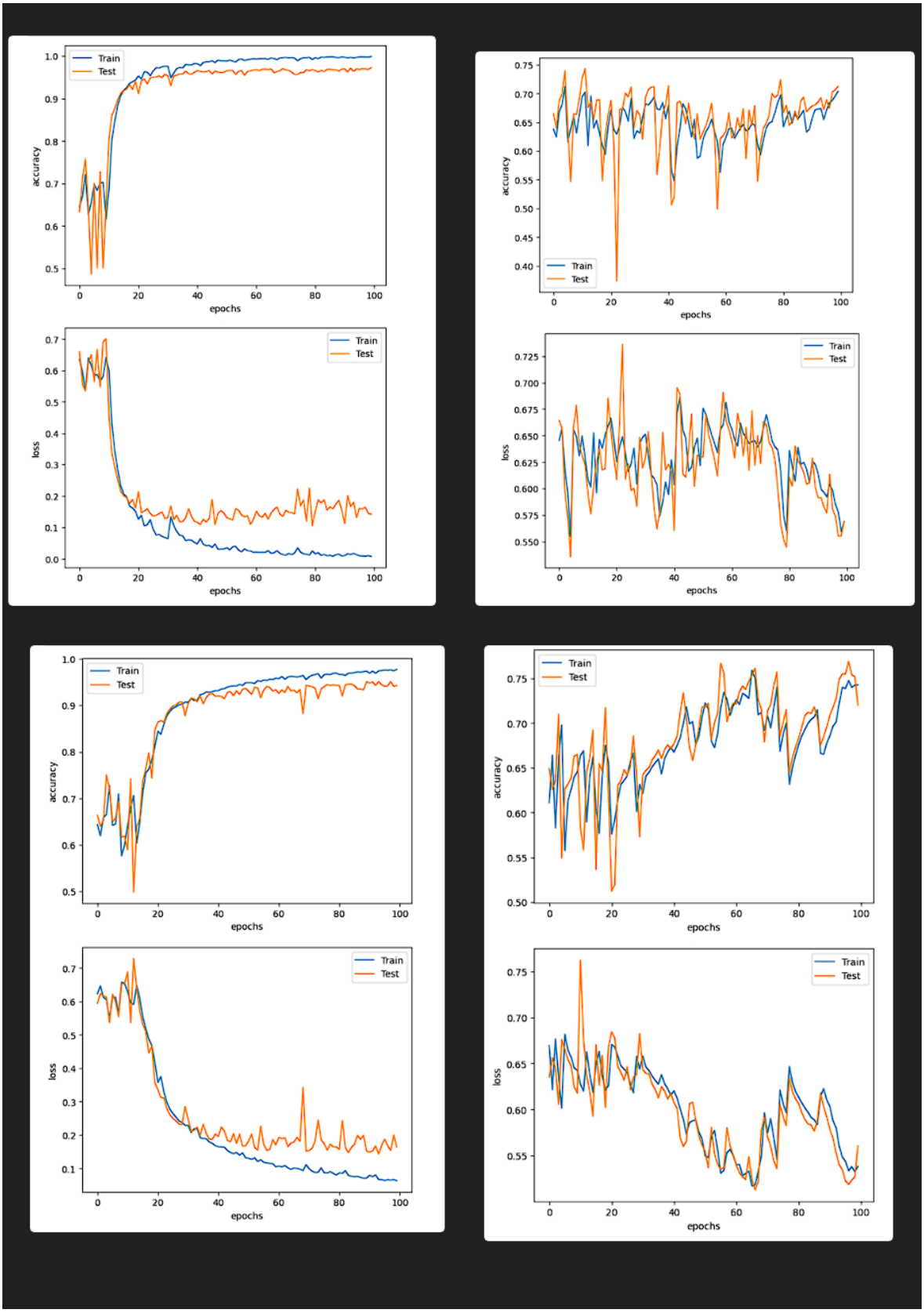
Εικόνα 15 Αρχιτεκτονική LSTM

```
model = Sequential([
    Embedding(vocab_size+1, 100, weights=[embedding_matrix], trainable=False),
    Dropout(0.2),
    SimpleRNN(128, activation='tanh', return_sequences=True),
    Dropout(0.2),
    SimpleRNN(128, activation='tanh', return_sequences=True),
    Dropout(0.1),
    Dense(512),
    Dropout(0.2),
    Dense(256),
    Dropout(0.1),
    Dense(256),
    Dense(1, activation='sigmoid')
])
```

Εικόνα 16 Αρχιτεκτονική SimpleRNN

Ακολουθούν τα αποτελέσματα των τεσσάρων εκπαιδεύσεων. Αν δούμε την εικόνα με τεταρτημόρια το 1ο αντιστοιχεί στο SimpleRNN μοντέλο με default τιμή learning rate του adam, το 2ο στο LSTM με default τιμή learning rate του adam, το 3ο αντιστοιχεί στο LSTM μοντέλο με το χρονοδιάγραμμα μάθησης και τέλος στο 4ο τεταρτημόριο αντιστοιχεί στο SimpleRNN μοντέλο με το χρονοδιάγραμμα μάθησης.





Εικόνα 17 Σύγκριση Μοντέλων

Από τις παραπάνω εικόνες μπορούμε να συμπεράνουμε τα εξής: το SimpleRNN μοντέλο με default τιμή learning rate του adam, αποδίδει καλά στα δεδομένα με accuracy (train+test) να είναι κοντά στο 68%. Αντίστοιχα το loss του μοντέλου αυτού μέχρι και τις 70 εποχές εμφανίζει αυξομειώσεις, ενώ με το πέρας της 70ης εποχής παρατηρούμε ότι μειώνεται, συνεπώς η διαδικασία της εκπαίδευσης μέχρι και το τέλος της 100ης εποχής διεκπεραιώνεται ομαλά χωρίς προβλήματα over ή under fitting. Από την άλλη το LSTM μοντέλο με παρόμοια αρχιτεκτονική και default τιμή learning rate του adam αποδίδει ακόμα καλύτερα στα δεδομένα με το με accuracy (train+test) να είναι κοντά στο 92%. Από τις πρώτες 15 εποχές το μοντέλο μαθαίνει αρκετά καλά και λίγο πριν την 20η εποχή αγγίζει το 92 % και η διαδικασία της εκπαίδευσης ολοκληρώνεται ομαλά χωρίς προβλήματα over ή under fitting, το οποίο επιβεβαιώνεται και μέσω του διαγράμματος απώλειας, καθώς μέχρι και τις 20 εποχές έχουμε μείωση σε κάθε iteration και μετά το πέρας της 20ης εποχής υπάρχει διατήρηση της απώλειας τόσο στα δεδομένα εκπαίδευσης και ελέγχου αντίστοιχα. Μεταξύ των δύο αυτών μοντέλων βλέπουμε ότι το LSTM προσφέρει καλύτερα και ομαλότερα αποτελέσματα. Έχει τη δυνατότητα να αποθηκεύσει πληροφορίες για μεγάλα χρονικά διαστήματα και να τις ανακαλέσει αργότερα. Οι τέσσερις πύλες του Forget , Input , Output και Cell αλλάζουν την ακριβέστερη ρύθμιση της ροής των πληροφοριών και της λήψης αποφάσεων για το ποιες πληροφορίες πρέπει να διατηρηθούν και ποιες πρέπει να αγνοηθούν. Παρόμοια αποτελέσματα παρατηρούνται και με την χρήση του χρονοδιαγράμματος μάθησης με το LSTM πάλι να είναι καλύτερο αγγίζοντας accuracy της τάξης του 91%. Το μοντέλο μαθαίνει γρήγορα στην αρχή και όσο προχωράει η εκπαίδευση του γίνεται πιο προσεκτικό. Τέλος το SimpleRNN μοντέλο με την χρήση του χρονοδιαγράμματος μάθησης θα λέγαμε ότι τα έχει πάει σχετικά καλά σε σχέση με το LSTM. Το SimpleRNN πιάνει υψηλότερο accuracy της τάξης ~ 75%. Εάν γίνει και σύγκριση μεταξύ των ίδιων μοντέλων, δηλαδή του LSTM με και χωρίς χρονοδιάγραμμα μάθησης θα λέγαμε ότι και τα δύο αποδίδουν παρόμοια αποτελέσματα 90 με 92 %, ενώ μεταξύ των SimpleRNN μοντέλων βλέπουμε ότι το χρονοδιάγραμμα μάθησης προσέφερε και υψηλότερα αποτελέσματα από το μοντέλο με την default τιμή learning rate του adam βελτιστοποιητή αλλά και ομαλότερη εκπαίδευση κάτι που φαίνεται από τα διαγράμματα των train & test τόσο στα αποτελέσματα της εκπαίδευσης όσο και στα διαγράμματα απώλειας.