ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

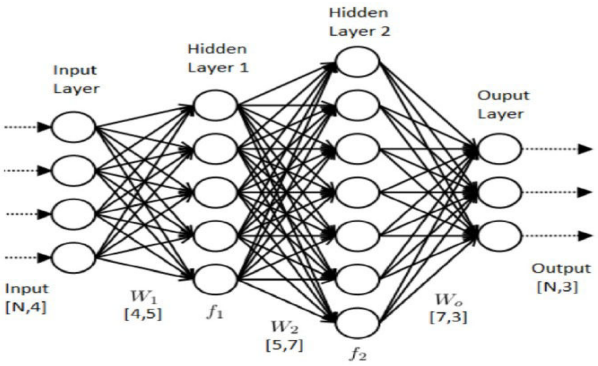
**Σχολή Πολυτεχνική**

**Τμήμα Μηχανικών Η/Υ & Πληροφορικής**

**Εργασία στο πλαίσιο του μαθήματος**

**“Θεωρία αποφάσεων”**

ΜΟΣΧΟΠΟΥΛΟΣ ΝΙΚΟΛΑΟΣ ΑΜ: 1054315
ΝΙΑΡΧΑΚΟΣ ΒΑΣΙΛΗΣ ΑΜ:1058109
ΟΣΜΑΝ ΦΑΤΙΧ ΑΜ:1041847 (236164)




**Πάτρα**
**2021**

# Πίνακας περιεχομένων

# Εισαγωγή

Στην παρούσα αναφορά θα αναλύσουμε την διαδικασία καθώς και τα συμπεράσματα που βγήκαν σε ένα project Sentiment analysis (Εξόρυξη γνώμης, ή ανάλυση αισθήματος) με δεδομένα από δύο dataframes που αφορούν Twitter posts σχετικά με προϊόντα τεχνολογίας τα οποία δόθηκαν στα πλαίσια του μαθήματος Επιχειρησιακή έρευνα.

Ως Sentiment analysis ορίζουμε την υποκατηγορία του (NLP) σκοπός της οποίας είναι να εξορυχθεί η γνώμη(αίσθημα) του κοινού (χρηστών) επάνω σε μία οντότητα. Στην προκειμένη περίπτωση μελετάμε την γνώμη του κοινού πάνω σε προϊόντα τεχνολογίας (κατά κύριο λόγο κινητά και εταιρείες κινητής τηλεφωνίας).

Στόχος μας είναι να παρουσιάσουμε τρεις προσεγγίσεις του προβλήματος και να συγκρίνουμε την αποτελεσματικότητα τους. Για το πρόβλημα αναπτύξαμε τρία προγράμματα στην γλώσσα Python 3.7 , στο πρώτο χρησιμοποιήσαμε το module NaiveBayesClassifier από την βιβλιοθήκη nltk (Natural Language Tool Kit).

# Το Project

## Οι Βιβλιοθήκες

Για τα μοντέλα που κατασκευάσαμε χρησιμοποιήσαμε τις παρακάτω βιβλιοθήκες της python 3.7

### *Sentiment analysis with nltk NaiveBayesClassifier*

```
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tag import pos_tag
from nltk.tokenize import TweetTokenizer,word_tokenize
from nltk import FreqDist,classify,NaiveBayesClassifier
import pandas as pd
import re,string,random
```

Με βάση τα παραπάνω χρησιμοποιήσαμε από την nltk.stem.wordnet το module WordNetLemmatizer η οποία μετατρέπει της λέξεις ενός κειμένου σε μια αντίστοιχης ρίζας η οποία θα έχει μια πιο θεμελιώδη μορφή, σκοπός είναι να μην διαχωρίζονται λέξεις ίδιου νοήματος πχ. hello,hallo,hi. Έπειτα από το nltk.corpus καλούμε την stopwords μέσω της οποίας καλούμε μια έτοιμη λίστα από stopwords της Αγγλικής γλώσσας, μέσω της οποίας καθαρίζουμε τις προτάσεις μας. Από την nltk.tag καλούμε pos_tag μέσω της οποίας γίνεται tokenize των λέξεων και σε κάθε μια αποδίδεται ο τίτλος του αντίστοιχου μέρους του λόγου, με αυτό επιτυγχάνεται ο διαχωρισμός λέξεων όπως το “That’s” → [(“That”,PRT) ,(“is”,Verb)]. Από το nltk.tokenize χρησιμοποιήσαμε την TweetTokenizer και την word_tokenize που και οι δύο κάνουν tokenize τις προτάσεις με την πρώτη να είναι λίγο πιο εξειδικευμένη σε κείμενα από το Twitter. Από την nltk καλώ τις

FreqDist,classify,NaiveBayesClassifier από τις οποίες η πρώτη μετράει την συχνότητα των λέξεων ,η δεύτερη επιστρέφει την πιο κατάλληλη ταμπέλα (label) για τον δοθέντα χαρακτηρισμό, και η τρίτη καλεί ένα έτοιμο μοντέλο με βάση τον τύπο του Bayes που κάνει classification, χρησιμοποιώντας δεσμευμένες πιθανότητες. Καλώ την pandas υπό το όνομα pd μέσω της οποίας καλούμαι και επεξεργαζόμαστε τα dataframes και τις re,string,random, που κατά αντιστοιχία κάνουν τα εξής, re: regular expressions κόβει και επεξεργάζεται σύμβολα γράμματα και αριθμούς, την χρησιμοποιούμαι για να διαγράψουμε τα URL. Η string για την επεξεργασία string αντικειμένων και την random για την παραγωγή ψευδοτυχαιοποίηση (εδώ θα την χρησιμοποιήσουμε για να ανακατέψουμε τα δεδομένα).

## *Recursive & Simple Neural Network*

```
#Libraries
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords,twitter_samples
from nltk.tag import pos_tag
from nltk.tokenize import TweetTokenizer,word_tokenize
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras import layers
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import re,string
import numpy as np

tt=TweetTokenizer()
```

Κάποιες βιβλιοθήκες είναι κοινές με το προηγούμενο μοντέλο και έτσι δεν θα τις εξηγήσουμε περαιτέρω. Οι υπόλοιπες είναι από την scikit toolkit learn φέρνουμε την train_test_split από την sklearn.model_selection μέσω της οποίας θα διαχωρίσουμε και θα τραβήξουμε τα δεδομένα στο νευρωνικό. Από την keras.preprocessing.text καούμε την Tokenizer μέσω της οποίας μπορούμε να κάνουμε Tokenize τα δεδομένα, από την keras.preprocessing.sequence καλούμε την pad_sequence μέσω της οποίας μετατρέπω το κείμενο που αποτελείτε ήδη από λέξεις υπό μορφή token σε ακέραιους αριθμούς. Καλώ το Sequential module από την keras.models η οποία πάνω στο οποίο χτίσαμε τα νευρωνικά μας και από την ίδια την keras καλώ την layers μέσω της οποίας δημιουργό τα επίπεδα των νευρώνων. Η seaborn χρησιμοποιείτε για να μετρήσουμε τα δύο dataframes και να συγκρίνουμε και να αποτυπώσουμε το πλήθος των αρνητικών και θετικών tweets, ώστε να ελεγχθεί το κατά πόσο τα δεδομένα μπορούν να κάνουν το μοντέλο να είναι biased . Μέσω της numpy δημιουργό ένα διάνυσμα array με τις τιμές 1 για “Positive” και 0 για “Negative” και μέσω της matplotlib.pyplot δημιουργώ τα γραφήματα.

# Δεδομένα

### *Παρουσίαση Δεδομένων*

Τα δεδομένα μας δόθηκαν σε μορφή αρχείων .csv με ονόματα SocialMedia_Positive.csv ,SocialMedia_Negative.csv . Για την εισαγωγή και παρουσίαση των δεδομένων μας χρησιμοποιήσαμε την βιβλιοθήκη pandas. Παρακάτω παρουσιάζονται τα σημεία του κώδικα στα οποία καλούμαι τα δεδομένα.

```
positive_tweets =pd.read_csv("~/Desktop/DT-Datasets-part3/Project7/SocialMedia_Positive.csv")
negative_tweets =pd.read_csv("~/Desktop/DT-Datasets-part3/Project7/SocialMedia_Negative.csv")
```

Στο πρώτο δίνονται [501 γραμμές επί 3 στήλες] από tweets που έχουν αποτιμηθεί ως θετικά :

```
print("my data positive\n",positive_tweet_tokens)
my data positive
     ID                                      Text Sentiment
0     P1  Who is ready for iPhone 6s and iPhone 6s Plus?...  positive
1     P2  @lucy_larkman there are hacks on how to save h...  positive
2     P3  Hi Rica Cunanan : ricacunanan09 Do u want to g...  positive
3     P4  Hi Ariane Valentine : mainlyboredom Do u want ...  positive
4     P5  Hi eastside_crazii : rthompson672 Do u want to...  positive
..   ...                                     ...       ...
496  P497  Hi solihin : muhdsolihin23 Do u want to get FR...  positive
497  P498  Hi Andi N.  : inlovewithcamd_ Do u want to get...  positive
498  P499  Hi nia : deluxechaelin Do u want to get FREE i...  positive
499  P500  Hi JBM . : TuesdayBleu Do u want to get FREE i...  positive
500  P501  Hi Adam Crandall : adamkcrandall Do u want to ...  positive

[501 rows x 3 columns]
```

Στο δεύτερο δίνονται [501 γραμμές επί 3 στήλες] από tweets που έχουν αποτιμηθεί ως αρνητικά :

```
print("my data negative\n", negative_tweet_tokens)
my data negative
      ID                                      Text Sentiment
0     N1  I've had an IPhone for like 3 years and I've n...  negative
1     N2        I dont need the new iphone but I want it :(  negative
2     N3     fuck the iPhone 6s cus I'm not getting one :(  negative
3     N4  Hopefully will be able to get my iPhone 6s tod...  negative
4     N5  @clydesdalebank I've just bought a new iPhone ...  negative
..   ...                                     ...       ...
496  N497  Does anyone have an extra iPhone 6 case I can ...  negative
497  N498  @dburrows I think it's the first iPhone that I...  negative
498  N499  #HomeOfTechUS @techradar been using an iPhone ...  negative
499  N500                        I need the new iPhone :(  negative
500  N501  @RosieLondoner @calzora yeah I was getting loa...  negative
[501 rows x 3 columns]
```

Οι δομή και των δύο dataframes είναι όμοια όπως φαίνεται και παραπάνω. Οι στήλες και των δύο ονομάζονται "ID", , "Text","Sentiment". Η στήλη  "ID" δεν είναι σημαντική για την μελέτη μας.

## *Επεξεργασία Δεδομένων*

Για να μπορέσουμε να εισάγουμε τα δεδομένα μας μέσα στα μοντέλα που θα χτίσουμε πρέπει πρώτον να τα καθαρίσουμε από τον πιθανό θόρυβο ( λέξεις που δεν προσφέρουν πληροφορία στο μοντέλο π.χ. stopwords (i,and,at, … κ.α.), σημεία στίξης (punctuaries),URL ) και έπειτα να μετασχηματίσουμε τα δεδομένα σε κάτι που να μπορεί να διαβάσει το κάθε μοντέλο.

### *Sentiment analysis with nltk NaiveBayesClassifier*

Για την  επεξεργασία των δεδομένων αρχικά δημιουργούμε κάποια modules:

```
#Define modules for pre-process of data.
def remove_noise(tweet_tokens,stop_words=()):

    cleaned_tokens=[]

    for token,tag in pos_tag(tweet_tokens):
        token=re.sub('http[s]?://(?:[a-zA-Z][0-9]|[$-_@.&+#]|[!*\(\),]|\
                    '(?:%[0-9a-fA-F][0-9a-fA-F]))+','',token)
        token=re.sub("(@[A-Za-z0-9_]+)","",token)

        print("token",token)

        if tag.startswith("NN"):
            pos='n'
        elif tag.startswith("VB"):
            pos='v'
        else:
            pos='a'

        lmtzr=WordNetLemmatizer()
        token=lmtzr.lemmatize(token,pos)

        if len(token)>0 and token not in  string.punctuation and token.lower() not in stop_words:
            cleaned_tokens.append(token.lower())
    return cleaned_tokens

def get_all_words(cleaned_tokens_list):
    for tokens in cleaned_tokens_list:
        for token in tokens:
            yield token

def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
        yield dict([token,True] for token in tweet_tokens)
```

Στο πρώτο module αρχικά διαγράφουμε τα URL και τα ονόματα (αυτό μπορεί να γίνει και μέσω του Tweetokenizer μέσω του strip_handles=True), έπειτα κάνουμε tokenize τα δεδομένα μέσω της pos_tag καί έπειτα μετατρέπουμε τα pos σε n,v,a για NN(nouns) , VB(verbs),anythink else, αντίστοιχα και τέλος καθαρίζουμε τα stopwords και τα punctuations.
Στο δεύτερο module καλούμε όλες τις λέξεις 1 προς 1 ξεχωριστά ως strings.
Στο τρίτο module δημιουργούμε ένα dictionary στο οποίο σε κάθε λέξη στις προτάσεις της λίστας ( argument ), αποδίδουμε την λογική τιμή True. Έτσι προετοιμάζουμε τα δεδομένα μας για την εισαγωγή τους στο μοντέλο.

Παρακάτω δείχνουμε την εφαρμογή των συναρτήσεων αυτών μέσα στο βασικό πρόγραμμα. Αρχικά καλούμε την εντολή για την έναρξη του προγράμματος:

```
if __name__ == '__main__':
```

έπειτα καλούμε την λίστα με τα Αγγλικά stopwords,

```
stop_words=stopwords.words("english")
```

```
positive_cleaned_tokens_list = []
negative_cleaned_tokens_list = []

for tokens in positive_tweet_tokens:
    positive_cleaned_tokens_list.append(remove_noise(tokens,stop_words))

for tokens in negative_tweet_tokens:
    negative_cleaned_tokens_list.append(remove_noise(tokens,stop_words))

all_pos_words=get_all_words(positive_cleaned_tokens_list)
freq_dist_pos=FreqDist(all_pos_words)                    print('see',freq_dist_pos.most_common(10))
```

Δημιουργώ τις λίστες με τα καθαρισμένα δεδομένα. Στο κείμενο η λέξη tokens μπορεί να μεταφράζεται ως πρόταση ενώ το token ως λέξη. Στα positive_cleaned_tokens_list και negative_cleaned_tokens_list τοποθετούμε τις λίστες με τα καθαρισμένα δεδομένα. Στην all_pos_words καλώ όλες τις λέξεις από τα positive_cleaned_tokens_list και μετά μετράμε την συχνότητα εμφάνισης και αποτυπώνουμε τις 10 πιο συχνές.

```
positive_tokens_for_model = get_tweets_for_model(positive_cleaned_tokens_list)
negative_tokens_for_model = get_tweets_for_model(negative_cleaned_tokens_list)

positive_dataset = [(tweet_dict, "Positive")
                    for tweet_dict in positive_tokens_for_model]

negative_dataset = [(tweet_dict, "Negative")
                    for tweet_dict in negative_tokens_for_model]

dataset = positive_dataset + negative_dataset

random.shuffle(dataset)
```

Παραπάνω χρησιμοποιούμε την συνάρτηση get_tweets_for_model για να κανονικοποιήσουμε τα δεδομένα μας, και έπειτα στην μεταβλητή ενώνουμε τις δύο λίστες. Τέλος με την

random.shuffle(dataset) ανακατεύουμε τα δεδομένα, για να αποφύγουμε να υπάρξει προκατάληψη λόγο σειράς εμφάνισης των δεδομένων.

## *Recursive & Simple Neural Network*

Για την  επεξεργασία των δεδομένων αρχικά δημιουργούμε κάποια modules:

```
df= pd.concat([positive_tweets,negative_tweets])
sns.set_theme()
sns.countplot(x='Sentiment',data=df)
plt.show()



stop_words=stopwords.words("english")



positive_cleaned_tokens_list = []
negative_cleaned_tokens_list = []

for tokens in positive_tweet_tokens:
    positive_cleaned_tokens_list.append(remove_noise(tokens,stop_words))

for tokens in negative_tweet_tokens:
    negative_cleaned_tokens_list.append(remove_noise(tokens,stop_words))









dataset =  positive_cleaned_tokens_list + negative_cleaned_tokens_list

max_ln = len(dataset[0])
for sentence in dataset:
    print(sentence)
    if max_ln < len(sentence):
        max_ln = len(sentence)

print(max_ln)
```

Κάποια μέρη του κώδικα είναι ίδια με την περίπτωση του BayesNaivesclassifier method, για λόγους οικονομίας και επομένως δεν θα επεξηγηθούν. Στην df χρησιμοποιούμε το module pd.concat, μέσω του οποίου ενώνουμε κάθετα τα δεδομένα μας . Με την λούπα της sns και την εντολή plot() συγκρίνουμε και μεταφέρουμε σε γράφημα το πλήθος των δεδομένων για να δείξουμε ότι τα δεδομένα μας δεν είναι biased ( Εικόνα1). Έπειτα τοποθετούμε στην μεταβλητή max_len το μήκος  της πρώτης γραμμής του dataset και με ένα loop συγκρίνω τις γραμμές με σκοπό να βρω την μέγιστη σε μήκος (πλήθος λέξεων) πρόταση στην περίπτωση μας το μέγιστο μήκος μεταξύ των προτάσεων είναι 24, αυτή η τιμή θα χρησιμοποιηθεί στο νευρωνικό.

```
y=df['Sentiment']
y=np.array(list(map(lambda x: 1 if  x== 'positive' else 0,y)))

x_train,x_test,y_train,y_test = train_test_split(dataset,y, test_size=0.20,random_state=42)

max_words=max_ln
tokenizer=Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(x_train)
x_train=tokenizer.texts_to_sequences(x_train)
x_test=tokenizer.texts_to_sequences(x_test)
print (x_train)
print(x_test)

vocab_size=len(tokenizer.word_index)+1
print(vocab_size)

x_train=pad_sequences(x_train,padding='post',maxlen=max_words)
x_test=pad_sequences(x_test,padding='post',maxlen=max_words)
```

Στην μεταβλητή y τοποθετούμε την στήλη με το όνομα “Sentiment” και μέσω της numpy φτιάχνουμε ένα διάνυσμα λίστα που από τα στοιχεία του y όπου αντιστοιχεί την τιμή 1 σε κάθε γραμμή του “positive” και 0 σε κάθε γραμμή του “negative” . Με την εντολή split χωρίζουμε τα δεδομένα σε τιμές x_train,x_test με στοιχεία από το dataset για την ακρίβεια το 80% πηγαίνει στο x_train και το 20% των δεδομένων πηγαίνει στο x_test.

Ονομάζω μια μεταβλητή την max_words και εκχωρώ την max_ln. Έπειτα κάνουμε tokenize τα δεδομένα με μέγιστο μέγεθος πρότασης τα 24 και με την fit_on_texts αποδίδουμε έναν ακέραιο σε κάθε ένα από τα token (λέξεις ) μετράω το πλήθος των λέξεων και το εκχωρώ στο vocab_size. Μετατρέπουμε λοιπόν τις προτάσεις (tweets) σε ισομήκης λίστες των 24 στοιχείων, συμπληρώνοντας τα κενά με μηδενικά μετά το πέρας των λέξεων.

# Τα Μοντέλα

### *NaiveBayesClassifier*

Παρακάτω φαίνεται το κομμάτι του κώδικα που αφορά την εισαγωγή των δεδομένων την εκπαίδευση και την ανάλυση του μοντέλου.

```
train_value= 0.8*len(dataset)
test_value = 0.2*len(dataset)+1
train_dataset=dataset[:int(train_value)]
test_dataset=dataset[int(test_value):]



classifier = NaiveBayesClassifier.train(train_dataset)

print("Accuracy is:", classify.accuracy(classifier, test_dataset))

print(classifier.show_most_informative_features(10))

custom_tweet = "I ordered just once from TerribleCo, they screwed up, never used the app again."
custom_tokens = remove_noise(word_tokenize(custom_tweet))
print(custom_tweet, classifier.classify(dict([token, True] for token in custom_tokens)))
```

Στα train_value τοποθετούμε το  80% του πλήθους του dataset και στην test_value το 20% του πλήθους του dataset και έπειτα κάλουμε τα αντίστοιχα ποσοστά του dataset στις μεταβλητές train_ dataset  και test_ dataset. Με στο όνομα classifier καλούμε το έτοιμο μοντέλο του Naive Bayes Classifier και το εκπαιδεύουμε. Έπειτα τυπώνουμε και υπολογίζουμε την ακρίβεια του μοντέλου η οποία υπολογίζεται με βάση τα δεδομένα υπό την μεταβλητή  test_ dataset. Στην επόμενη γραμμή τυπώνουμε τις 10 λέξεις που είχαν μεγαλύτερη βαρύτητα στο αποτέλεσμα.

Στις τελευταίες γραμμές καταχωρούμε και ελέγχουμε μία πρόταση με βάση το μοντέλο μας.

## Simple Neural Network

Παρακάτω φαίνεται το κομμάτι του κώδικα που αφορά την εισαγωγή των δεδομένων την εκπαίδευση και την ανάλυση του μοντέλου.

```
model1 = Sequential()
embedding_layer = layers.Embedding(vocab_size, max_words, input_length=max_words, trainable=False)
model1.add(embedding_layer)
model1.add(layers.Flatten())
model1.add(layers.Dense(1, activation='relu'))
model1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
print(model1.summary())
history = model1.fit(x_train, y_train, batch_size=34, epochs=100, verbose=1, validation_split=0.2)
score = model1.evaluate(x_test, y_test, verbose=1)
```

Εδώ εισάγουμε το νευρωνικό μας. Το μοντέλο μας περιέχει τρία layers το embedding που μετασχηματίζει τα δεδομένα μας από λίστα που περιέχει λίστες ακεραίων σε πίνακες, το flatten layer που μας επιστρέφει τα δεδομένα σε vectors, και τέλος το Dence layer που δίνει ένα output και ενεργοποιείτε μέσω της συνάρτησης Rectifier linear Unit. Τυπώνουμε ένα summary του μοντέλου.Εκπαιδεύουμε το μοντέλο μας 100 φορές “εποχές” τραβώντας τα δεδομένα σε πακέτα των 34 γραμμών.

```
print("Test Score:", score[0])
print("Test Accuracy:", score[1])

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()
```

Στο τελευταίο μέρος του κώδικα παρουσιάζουμε την αποτελεσματικότητα και τα σφάλματα του μοντέλου τόσο σε νούμερα όσο και σε γραφήματα.

## Recursive Neural Network

Παρακάτω φαίνεται το κομμάτι του κώδικα που αφορά την εισαγωγή των δεδομένων την εκπαίδευση και την ανάλυση του μοντέλου.

```
model1 = Sequential()
embedding_layer = layers.Embedding(vocab_size, max_words, input_length=max_words, trainable=False)
model1.add(embedding_layer)
model1.add(layers.LSTM(128,recurrent_dropout=0.5))

model1.add(layers.Dense(1, activation='sigmoid'))
model1.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

print(model1.summary())

history = model1.fit(x_train, y_train, batch_size=128, epochs=100, verbose=1, validation_split=0.2)

score = model1.evaluate(x_test, y_test, verbose=1)
```

Το νευρωνικό σε αυτήν την περίπτωση είναι παρόμοιο με το προηγούμενο με μόνη διαφόρα το ότι αντί για το flatten layer, έχουμε ένα LSTM layer Long short-term Memory, μέσω του οποίου διατηρούνται οι μνήμες από τα προηγούμενα περάσματα, το οποίο ενεργοποιήτε μέσω της συνάρτησης tanh και ενεργοποιεί την αναδρομή μέσω της σιγμοειδούς συνάρτησης. Το Dense layer ενεργοποιείτε από την σιγμοειδή επίσης. Ο υπόλοιπος κώδικας είναι ίδιος.

```
predict=model1.predict_classes(x_train)
for i in range(100,105):
    print('%s => %d (expected %d)' % (x_train[i].tolist(), predict[i], y[i]))

print("Test Score:", score[0])
print("Test Accuracy:", score[1])

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()
```

Τέλος τυπώνουμε όπως και παραπάνω την αποτελεσματικότητα του μοντέλου τόσο σε γραφήματα όσο και σε αριθμούς, και επίσης εφαρμόζω το μοντέλο στα κείμενα του dataframe που βρίσκονται στις γραμμές 101 έως 106.

# Αποτελέσματα

### *NaiveBayesClassifier model*

```
[(':)', 505), ('iphone', 486), ('get', 388), ('6', 378), ('free', 372), ('check', 371), ('hi', 369), ('bi0', 369), ('thx', 369), ('want', 225)]
Accuracy is: 0.9712858926342073
Most Informative Features
                free = True           Positi : Negati =    199.5 : 1.0
               check = True           Positi : Negati =    198.8 : 1.0
                bi0 = True           Positi : Negati =    198.2 : 1.0
                thx = True           Positi : Negati =    198.2 : 1.0
                 :( = True           Negati : Positi =    115.0 : 1.0
               hurry = True           Positi : Negati =     75.8 : 1.0
                 :) = True           Positi : Negati =     61.5 : 1.0
                 hi = True           Positi : Negati =     35.0 : 1.0
              wanna = True           Positi : Negati =     21.2 : 1.0
               rise = True           Negati : Positi =     17.1 : 1.0
```

Στην πρώτη γραμμή φαίνονται οι δέκα πιο συχνές λέξεις π.χ. η λέξη iphone εμφανίζεται 486 φορές , το ποσοστό επιτυχίας είναι περίπου 97,12%. Παρακάτω παρουσιάζονται η 10 πιο σημαντικές λέξεις για το αποτέλεσμα.

```
 I ordered just once from TerribleCo, they screwed up, never used the app again. Negative
```

παραπάνω παρουσιάζεται το παράδειγμα και το αποτέλεσμα που έδωσε.

### *Simple Neural Network*

Το αποτέλεσμα φαίνεται παρακάτω αλλά και στα γραφήματα [Εικόνα 2], [Εικόνα 3].

Το αποτέλεσμα ήταν το μοντέλο να έχει μια ακρίβεια πρόβλεψης της τάξης του 90,04% .

```
Test Score: 0.15747074782848358
Test Accuracy: 0.9004974961280823
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 24, 24)            56136
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 577
=================================================================
Total params: 56,713
Trainable params: 577
Non-trainable params: 56,136
```

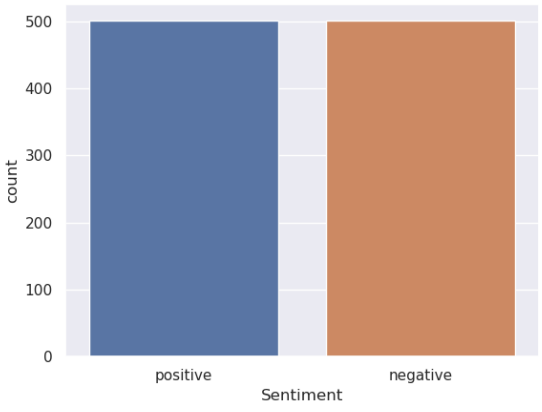
### *Recursive  Neural Network*

Το αποτέλεσμα  φαίνεται παρακάτω αλλά και στα γραφήματα [Εικόνα 4], [Εικόνα 5].

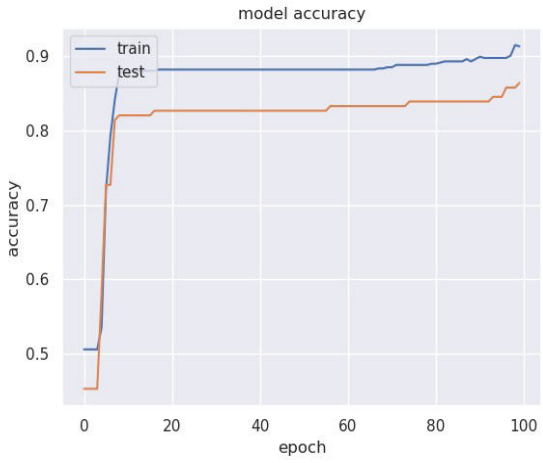
Το αποτέλεσμα ήταν το μοντέλο να έχει μια ακρίβεια πρόβλεψης της τάξης του 98,00% .

```
Test Score: 0.05854669213294983
Test Accuracy: 0.9800994992256165
[1, 12, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] => 1 (expected 1)
[6, 13, 4, 7, 1, 5, 15, 8, 9, 2, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] => 1 (expected 1)
[6, 13, 4, 7, 1, 5, 15, 8, 9, 2, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] => 1 (expected 1)
[1, 12, 12, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] => 0 (expected 1)
[3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] => 0 (expected 1)
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 24, 24)            56136
_________________________________________________________________
lstm (LSTM)                  (None, 128)               78336
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 134,601
Trainable params: 78,465
Non-trainable params: 56,136
```

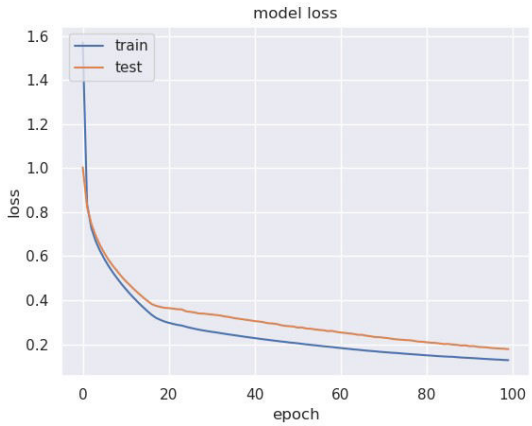
### *Γραφήματα*



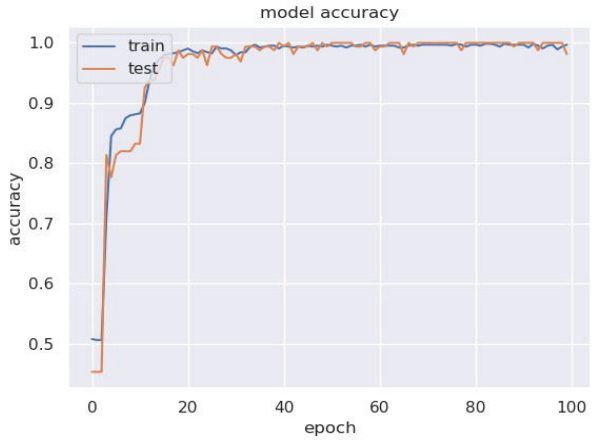

*Εικόνα 2*



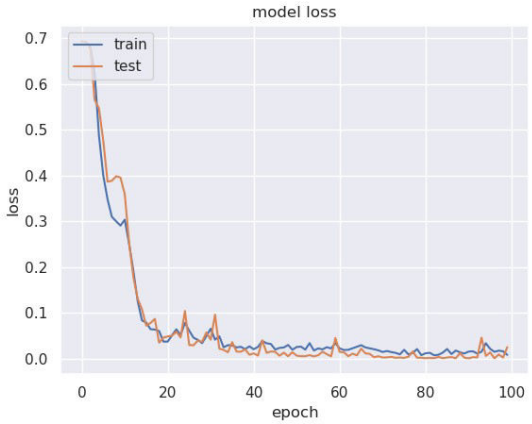

*Εικόνα 1: Simple NN accuracy*




*Εικόνα 3: Simplex NN loss*




*Εικόνα 4: RNN Accuracy*




*Εικόνα 4: RNN loss*

### *Προβλήματα*

Το απλό νευρωνικό παρουσίαζε μια κατάσταση πλήρους τυχαιότητας με Batches :128

Το RNN δοκιμάζοντας μεγαλύτερο πλήθος λέξεων max_words π.χ.100 εμφάνιζε κατάσταση πλήρους τυχαιότητας.

Για λόγους ομοιομορφίας στα νευρωνικά έχουμε κρατήσει κάποια κομμάτια της επεξεργασίας των δεδομένων από το πρώτο μοντέλο τα οποία αποτελούν περίσσεια.

## Συμπεράσματα

Τρέξαμε τα μοντέλα πολλές φορές και με διαφορετικά νούμερα. Το RNN φέρνει τα καλύτερα αποτελέσματα έχοντας σταθεροποιηθεί σε τιμή >98%, ενώ το μοντέλο του Bayes, μας έδινε σταθερά μια υψηλή τιμή.

# References

### *Links*

Shaumik Daityari (How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)) Published onSeptember 26, 2019 :https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltk

https://keras.io/guides/

https://stackoverflow.com/

Jason Brownlee (How to Prepare Text Data for Deep Learning with Keras) Published on Octomber 2, 2017 :https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

Usman Malik *(Python for NLP: Movie Sentiment Analysis using Deep Learning in Keras):* https://stackabuse.com/python-for-nlp-movie-sentiment-analysis-using-deep-learning-in-keras/

https://github.com/keras-team/keras/blob/master/keras/preprocessing/text.py#L160