UNIVERSITE IBN TOUFAIL

FACULTE DES SCIENCES

Master Génie Logiciel pour Le Cloud

ANALYSE DES SENTIMENTS AVEC LSTM

Présenté par : Abdellah BIROUK & Ayoub MANÇOUR BILLAH

Encadré par : Younes Chihab

Année Universitaire 2020-2021

Table de matières

Liste des figures

I. INTRODUCTION

Dans ce projet nous allons vous présenter une reconnaissance émotionnelle en utilisant un texte (Sentiment Analysis using text), ce projet est un système de reconnaissance des émotions multimodal qui est construit pour extraire des informations sur les émotions de la saisie de texte. Le système de reconnaissance des émotions classe les émotions selon deux types de base : positif ou négatif. Si la valeur d'intensité d'émotion de l'émotion actuellement reconnue est inférieure à un seuil prédéfini, la sortie d'émotion est déterminée comme étant neutre.

Le système de reconnaissance des émotions proposé peut détecter les émotions à partir du texte par l'algorithme **LSTM** (long short-term memory) pour évaluer l'approche acoustique, un drame diffusé, y compris le contenu textuel. Lors de la sélection des fonctionnalités, un ensemble initial de fonctionnalités acoustiques contenant 33 fonctionnalités est d'abord analysé et la reconnaissance multimodale d'émotion à partir du texte est extraite.

II. Définitions

1. Le Deep Learning

A présent que nous avons précisé comment fonctionnent les réseaux de neurones de manière générale, nous allons aborder le domaine du Deep Learning.

Cette famille d'algorithmes a permis de faire des progrès importants dans les domaines de la classification des textes et du traitement du langage par exemple.

Les modèles de Deep Learning sont bâtis sur le même modèle que les perceptrons multicouches précédemment décrits. Cependant, il convient de souligner que les différentes couches intermédiaires sont plus nombreuses.

Chacune des couches intermédiaires va être subdivisée en sous partie, traitant un sous problème, plus simple et fournissant le résultat à la couche suivante, et ainsi de suite.

2. Analyse des sentiments

L'analyse de sentiment est l'étude computationnelle et sémantique des parties de textes en fonction des opinions, des sentiments et des émotions exprimés dans le texte. Généralement l'expression « analyse des sentiments » est utilisée pour désigner la tâche de classification automatique des unités de texte en fonction de leur polarité (positive, négative, neutre).

3. La polarité et l’intensité de l’opinion

La polarité peut être définie par des catégories telles que « positif », « neutre » et « négatif » La polarité d’une opinion exprime la positivité, la négativité ou une information de cette dernière. On dit d’une opinion positive qu’elle possède une polarité positive, et inversement, on dit d’une opinion négative qu’elle possède une polarité négative ou neutre possède une information L’intensité décrit à quel point la polarité d’une opinion est forte. Par exemple, dans une opinion à polarité positive, aimer est plus intense qu’apprécier, ou encore, dans une opinion de polarité négative, haïr est plus intense que de ne pas aimer.

4. Domaine d’application d’analyse des sentiments :

- **Politique :** Grâce à l’analyse des sentiments, les décideurs de politique peuvent prendre l'avis des citoyens sur certaines politiques, afin de bénéficier de cette information pour améliorer ou créer une nouvelle politique qui convient avec les citoyens.

- **Prise de décision :** L'opinion et l'expérience des gens sont un élément très utile dans le processus de prise de décision.

- **Domaine de Transport :** Pour assembler et analyser les opinions du public sur le statut de transport.

- **Les systèmes de recommandations :** À travers l'analyse des sentiments on peut classer les opinions des gens de façon positive ou négative, le système définit qui devrait prendre ou pas prendre la recommandation.

- **Domaine médical :** Analyse l'opinion des médecins et des patients sur les médicaments et les services hospitaliers. Ainsi que sur les documents de l'état du patient qui contiennent le diagnostic et la description du résultat d'examen.

- **Domaine d'éducation :** Développer le niveau d'enseignement à travers l'analyse et l'interprétation de l'opinion de l'étudiant par les méthodes d'enseignement pour permettre d'améliorer l'enseignement et l'apprentissage.

- **Marketing :** Du côté d'entreprises, on permet au fournisseur plus de connaissances à propos des besoins des consommateurs. Du côté client, il peut donner son opinion, s'inspirer des opinions des autres et aussi comparer les produits avant de les acheter.

III. RÉALISATION

Durant la réalisation de ce projet nous avons été menés à l'utilisation du langage de programmation PYTHON ainsi que plusieurs bibliothèques de ce langage.

- **Langages et Programmes utilisés :**

5. Python

Python est un langage de programmation interprété, multi paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions.

Figure 1. Logo Python

6. JupyterLab

Figure 2. Logo JupyterLab

JupyterLab est un environnement de développement interactif basé sur le Web pour les blocs-notes, le code et les données Jupyter. JupyterLab est flexible : configurez et organisez l'interface utilisateur pour prendre en charge un large éventail de flux de travail dans les domaines de la science des données, de l'informatique scientifique et de l'apprentissage automatique. JupyterLab est extensible et modulaire : écrivez des plugins qui ajoutent de nouveaux composants et s'intègrent aux composants existants.

7. Anaconda

Figure 3. Logo Anaconda

Anaconda est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique, qui vise à simplifier la gestion des paquets et de déploiement.

8. Google Colab

Google Colab est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur.

Figure 4. Logo Google Colab

- **Outils et Bibliothèques utilisés :**

9. Pandas

Pandas est une bibliothèque écrite pour le langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles. Pandas est un logiciel libre sous licence BSD.

Figure 5. Logo Pandas

10. NumPy

NumPy est une bibliothèque pour langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Figure 6. Logo NumPy

11. Matplotlib

Matplotlib est une bibliothèque du langage de programmation Python destinée à tracer et visualiser des données sous formes de graphiques. Elle peut être combinée avec les bibliothèques python de calcul scientifique NumPy et SciPy.

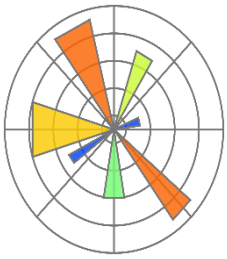


Figure 7. Logo Matplotlib

12. NLTK

Natural Language Toolkit (NLTK) est une bibliothèque logicielle en Python permettant un traitement automatique des langues, développée par Steven Bird et Edward Loper du département d'informatique de l'université de Pennsylvanie. En plus de la bibliothèque, NLTK fournit des démonstrations graphiques, des données-échantillon, des tutoriels, ainsi que la documentation de l'interface de programmation (API).

Figure 8. Logo NLTK

13. KERAS

Keras est une bibliothèque open source écrite en python. La bibliothèque Keras permet d'interagir avec les algorithmes de réseaux de neurones profonds et d'apprentissage automatique, notamment Tensorflow, Theano, Microsoft Cognitive Toolkit ou PlaidML. Conçue pour permettre une expérimentation rapide avec les réseaux de neurones profonds, elle se concentre sur son ergonomie, sa modularité et ses capacites d'extension.

Figure 9. Logo KERAS

14. LSTM

Un réseau Long short-term memory (LSTM), en français réseau récurrent à mémoire court et long terme ou plus explicitement réseau de neurones récurrents à mémoire court-terme et long terme, est l'architecture de réseau de neurones récurrents la plus utilisée en pratique qui permet de répondre au problème de disparition de gradient. Le réseau LSTM a été proposé par Sepp Hochreiter et Jürgen Schmidhuber en 1975.

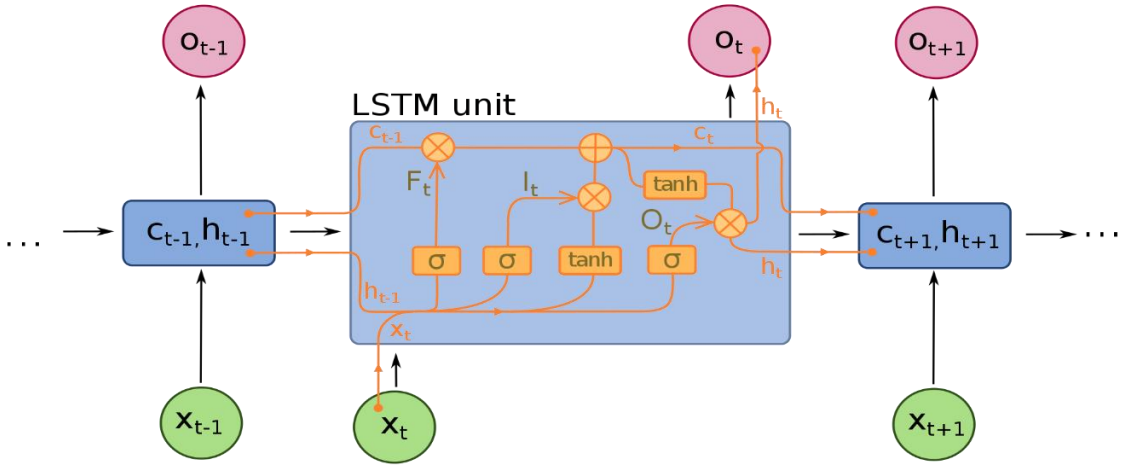


Figure 10. Schéma d'un réseau LSTM

15. TensorFlow

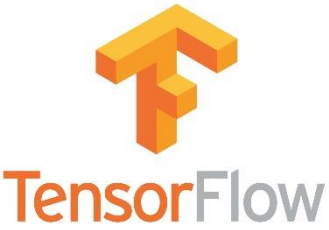


Figure 11. Logo TensorFlow

TensorFlow est un framework de programmation pour le calcul numérique qui a été rendu Open Source par Google en Novembre 2015. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des framework les plus utilisés pour le Deep Learning et donc les réseaux de neurones. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multidimensionnelles, appelées Tenseurs (Tensor). Un Tensor à deux dimensions est l'équivalent d'une matrice. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow : Gmail, Google Photos, Reconnaissance de voix

IV. Test et fonctionnement :

Dans ce chapitre, on va traiter les différentes phases de notre méthode d'analyse des sentiments. Durant ce travail, on a fait face à plusieurs phases :

1. Préparation des données (dataset) :

Nous considérons d'abord comment diviser les données. Nous avons choisi de diviser les données en trois morceaux : train, développement, test.

1) **Training :** L'échantillon de données utilisé pour l'apprentissage.
2) **Ensemble de développement** (ensemble de validation croisée de blocage) : échantillon de données utilisé pour ajuster les paramètres d'un classificateur et fournir une évaluation impartiale d'un modèle.
3) **Ensemble de test :** échantillon de données utilisé uniquement pour évaluer performance d'un modèle final.

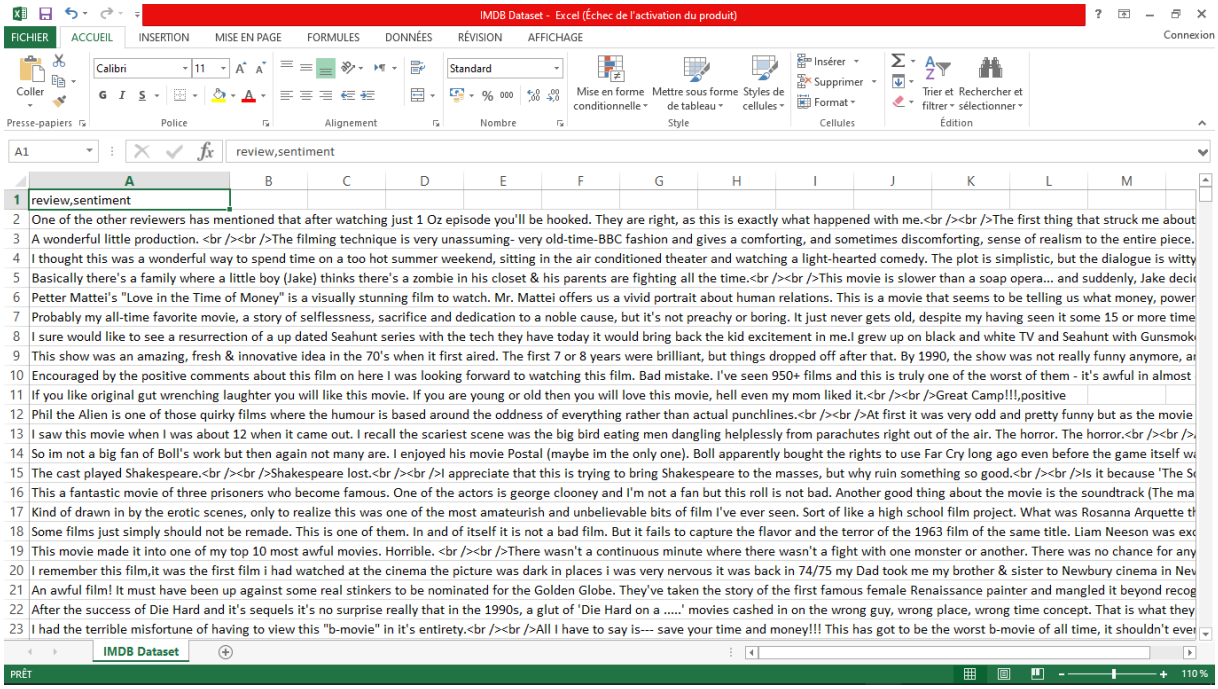
2. Word Embeddings

Un mot embedding est une représentation apprise pour le texte où les mots qui ont le même sens ont une représentation similaire. C'est cette approche de représenter des mots et des documents qui peut être considérée comme l'une des principales clés de l'apprentissage en profondeur sur les problèmes de traitement du langage naturel.

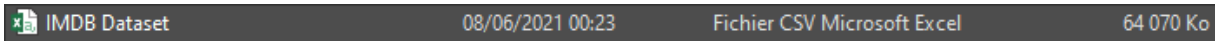
3. Base de données (Dataset):

On a choisi comme base de données (dataset) une liste d’avis sur plusieurs films. On a +49000 avis (positif / négatif) sur les films.

Voici ci-dessous une partie de notre dataset :



La taille du fichier (‘IMDB Dataset.csv’) est 64mo :



4. Réalisation :

a) Importation des bibliothèques et des plates-formes

Durant cette phase, on a importé les différentes bibliothèques et plates-formes tels que pandas, tensorflow, numpy…

Entrée [144]:
```
import string
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords

import tensorflow as tf
import tensorflow as tf
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
from keras.layers import GlobalMaxPooling1D
from keras.layers.embeddings import Embedding
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
import matplotlib.pyplot as plt
```

b) Lecture des données

Après, on a fait une lecture de données :

Entrée [145]:
```
# READ THE CSV FILE
movie_reviews = pd.read_csv("./IMDB Dataset.csv")

movie_reviews.isnull().values.any()

movie_reviews.shape
```

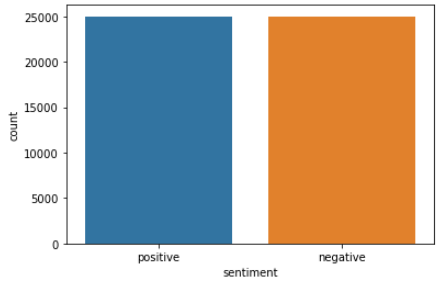
Out[145]:
```
(50000, 2)
```

Entrée [147]:
```
# vérifier les valeurs de sentiment
# 1 est un sentiment positif et 0 est un sentiment négatif
movie_reviews['sentiment'].value_counts()
```

Out[147]:
```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

Entrée [149]:
```
import seaborn as sns
sns.countplot(x='sentiment', data=movie_reviews)
```

Out[149]:
```
<AxesSubplot:xlabel='sentiment', ylabel='count'>
```



c) Prétraitement des données

Dans cette phase, on va traiter nos données (cleaning) en supprimant les ponctuations, les espaces et les différents caractères spéciaux :

Entrée [7]:
```
from string import punctuation
print(punctuation)
```
```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Entrée [8]:
```
#nettoyage des donnée

TAG_RE = re.compile(r'<[^>]+>')

def remove_tags(text):
    return TAG_RE.sub('', text)
```

Entrée [9]:
```
def preprocess_text(sen):
    # Supprimer les balises html
    sentence = remove_tags(sen)

    # Supprimer les signes de ponctuation et les chiffres
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)

    # Suppression d'un seul caractère
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence)

    # Suppression de plusieurs espaces
    sentence = re.sub(r'\s+', ' ', sentence)

    return sentence
```

Entrée [10]:
```
X = []
sentences = list(movie_reviews['review'])
for sen in sentences:
    X.append(preprocess_text(sen))
```

d) Fractionnement des données

On a fait un fractionnement ; train de données : 70%, validation : 10% et le test : 20%.

Entrée [159]:
```
#Fractionnement du entrainement de données:70%    validation:10%    test:20 %
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7,test_size=0.20, random_state=101, shuffle=False)
```

e) Tokenisation des données

La tokenisation est le processus de séparation d'un flux de texte en mots, phrases, symboles et d'autres éléments significatifs.

Entrée [160]:
```
#extraire les jetons du texte le nombre de jetons est de 5000
tokenizer = Tokenizer(num_words=5000)

#création du dictionnaire
tokenizer.fit_on_texts(X_train)

# convertir le texte en une séquence numérique
X_train = tokenizer.texts_to_sequences(X_train)
X_test = tokenizer.texts_to_sequences(X_test)
```

Entrée [106]:
```
# Ajout de 1 à cause de l'index 0 réservé pour l'attribut UNKNOWN
vocab_size = len(tokenizer.word_index) + 1

maxlen = 100

#s'assurer que toutes les séquences d'une liste ont la même longueur.
X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)
```

Entrée [107]:
```
# importer le dictionnaire de mots 'glove.6B.100d'

from numpy import array
from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()
glove_file = open('./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()
```

Entrée [161]:
```
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
```

f) Construction du modèle

Entrée [113]:
```
from keras.layers.recurrent import LSTM
# Initialisation du RNN
model = Sequential()
#dimension vercteur 100
embedding_layer = Embedding(vocab_size, 100, weights=[embedding_matrix], input_length=maxlen, trainable=False)
model.add(embedding_layer)
#128 nombre de neuronnes
model.add(LSTM(128))

# Comme la sortie est 1D, nous utilisons donc unit=1
# L'activation est sigmoïde
model.add(Dense(1, activation='sigmoid'))
#L'optimiseur est Adam
#compiler et adapter le modèle
model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics=['acc'])
```

Entrée [114]:
```
print(model.summary())
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 100, 100)          8737700
_________________________________________________________________
lstm_3 (LSTM)                (None, 128)               117248
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 8,855,077
Trainable params: 117,377
Non-trainable params: 8,737,700
_________________________________________________________________
None
```

g) Formation du modèle

Entrée [115]:
```
#batch_size : nombre d'échantillons qui seront propagés sur le réseau
#utiliser les derniers 20 % des données avant de les mélanger pour la validation
history = model.fit(X_train, y_train, batch_size=128, epochs=6, verbose=1, validation_split=0.2)

score = model.evaluate(X_test, y_test, verbose=1)
```

```
Epoch 1/6
219/219 [==============================] - 35s 159ms/step - loss: 0.5817 - acc: 0.6892 - val_loss: 0.4939 - val_acc: 0.7664
Epoch 2/6
219/219 [==============================] - 34s 155ms/step - loss: 0.4513 - acc: 0.7893 - val_loss: 0.4272 - val_acc: 0.8046
Epoch 3/6
219/219 [==============================] - 34s 155ms/step - loss: 0.4004 - acc: 0.8201 - val_loss: 0.3884 - val_acc: 0.8236
Epoch 4/6
219/219 [==============================] - 34s 153ms/step - loss: 0.3674 - acc: 0.8351 - val_loss: 0.3818 - val_acc: 0.8246
Epoch 5/6
219/219 [==============================] - 33s 153ms/step - loss: 0.3466 - acc: 0.8487 - val_loss: 0.3544 - val_acc: 0.8413
Epoch 6/6
219/219 [==============================] - 33s 152ms/step - loss: 0.3311 - acc: 0.8552 - val_loss: 0.3585 - val_acc: 0.8381
313/313 [==============================] - 7s 24ms/step - loss: 0.3591 - acc: 0.8392
```

h) Score et précision des tests

Entrée [116]:
```
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
Test Score: 0.3590550422668457
Test Accuracy: 0.8392000198364258
```

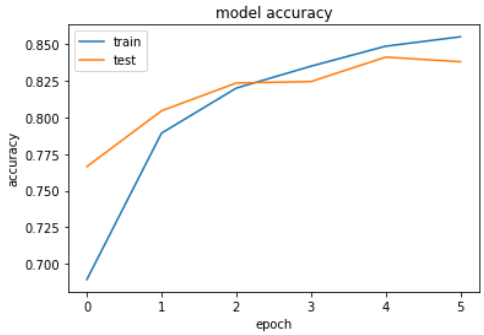
i) Précision et perte du modèle pour l'entraînement et les tests

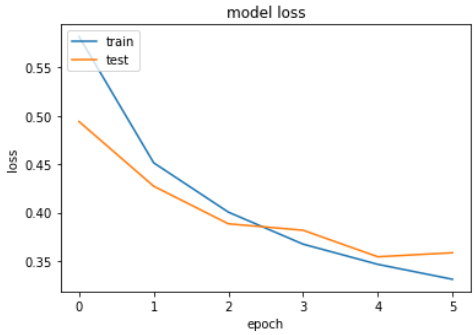
Entrée [117]:
```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])

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()
```





j) Test et résultat final :

- Résultat Négatif :
  On a essayé le 7ème avis de notre nouveau tableau créé X [10] :

Entrée [165]:
```
instance = X[10]
print(instance)
```

```
Phil the Alien is one of those quirky films where the humour is based around the oddness of everything rather than actual punch
lines At first it was very odd and pretty funny but as the movie progressed didn find the jokes or oddness funny anymore Its lo
w budget film thats never problem in itself there were some pretty interesting characters but eventually just lost interest ima
gine this film would appeal to stoner who is currently partaking For something similar but better try Brother from another plan
et
```

Entrée [166]:
```
instance = tokenizer.texts_to_sequences(instance)

flat_list = []
for sublist in instance:
    for item in sublist:
        flat_list.append(item)

flat_list = [flat_list]

instance = pad_sequences(flat_list, padding='post', maxlen=maxlen)

model.predict(instance)
```

Out[166]:
```
array([[0.48003367]], dtype=float32)
```

Entrée [167]:
```
y_pred = model.predict(instance)

if y_pred[0] < 0.5:
    value ='negative review'
else:
    value = 'positive review'
print("Predicted=%s,  %s " % ( y_pred[0], value ))
```

```
Predicted=[0.48003367],  negative review
```

- Résultat Positif :
  Maintenant, on a essayé le 9ème avis de notre nouveau tableau créé X [9] :

Entrée [211]:
```
instance = X[9]
print(instance)
```
```
If you like original gut wrenching laughter you will like this movie If you are young or old then you will love this movie hell even my mom liked it Great Camp
```

Entrée [212]:
```
instance = tokenizer.texts_to_sequences(instance)

flat_list = []
for sublist in instance:
    for item in sublist:
        flat_list.append(item)

flat_list = [flat_list]

instance = pad_sequences(flat_list, padding='post', maxlen=maxlen)

model.predict(instance)
```
Out[212]:
```
array([[0.7111333]], dtype=float32)
```

Entrée [213]:
```
y_pred = model.predict(instance)

if y_pred[0] < 0.5:
    value ='negative review'
else:
    value = 'positive review'
print("Predicted=%s,  %s " % ( y_pred[0], value ))
```
```
Predicted=[0.7111333],  positive review
```

k) Sauvegarde du programme

Entrée [208]:
```
model.save('model-lstm-sentiment-movie.h5')
```

Entrée [209]:
```
lstm_model= tf.keras.models.load_model('model-lstm-sentiment-movie.h5')
```

Entrée [210]:
```
score = lstm_model.evaluate(X_test, y_test, verbose=1)
```
```
313/313 [==============================] - 7s 22ms/step - loss: 0.3585 - acc: 0.8440
```

V. CONCLUSION

Durant ce projet nous avons étudié une méthode de reconnaissance textuelle des sentiments. Cette application qui repose sur l'apprentissage profond (Deep Learning), le langage Python et le réseau Long short-term memory (**LSTM**) peut extraire des sentiments depuis des informations textuelles (. Cette reconnaissance textuelle permet d'obtenir une vue d'ensemble sur l'opinion du public au sujet de certains thèmes, cela aide les entreprises à extraire des informations à partir des données du web social.