"Machine Learning Classification"

Réalisé par :

- Rajmagan BALAKICHENIN

Année universitaire: 2018-2019

Sommaire

INTRODUCTION

Le but de la matière est comprendre le concept du machine learning avec un apprentissage en mode supervisé. Le projet a pour de comprendre la classification, donc le thème de notre travail est libre.

La problématique du projet de comprendre :
Comment notre programme peut apprendre à prédire les émotions positive et négative ?

Le thème du projet est la reconnaissance de sentiment via des messages textuels, et l'algorithme devra trouver si un message est positive ou négative.
Cependant, ceci est un thème beaucoup trop vaste, il faut rendre l'analyse de sentiments beaucoup plus spécialisé dans un domaine. Dans notre cas, le programme réalisera une analyse de sentiment sur la satisfaction des clients sur les compagnies d'aviations à partir de données trouver sur Tweeter.
Le programme sera réalisé avec le langage Python3.6 avec l'outil Keras, qui facilite la réalisation des algorithmes de deep learning.
Le projet aura aussi 2 corpus de données, c'est-à-dire un corpus constitué de plus 15000 lignes, qui permettra à l'algorithme de s'entraîner et valider son modèle.
L'autre corpus servira de test pour l'algorithme afin de trouver les sentiments de chacun des lignes de ce corpus. Le programme devra aussi prédire en temps réel des phrase entrée par l'utilisateur.

Nous allons commencer pour la partie entraînement et validation, puis continuer par la partie test. Et nous finirons par la conclusion.

ENTRAINNEMENT ET VALIDATION

L' algorithme créera un modèle qui permettra de s'entraîner grâce à des données présent dans un corpus, mais aussi de réaliser des validation aussi grâce à un corpus. L'ensemble de l'étape de validation permettra à l'algorithme d'ajuster son modèle grâce à des évaluations fréquentes.
Ceci permet l'algorithme de mettre à jours les hyper paramètres de son modèle avec chaque répétition, et donc d'ajuster le modèle final.

Puisque le corpus est utiliser pour entraîner et valider le modèle, donc la qualité du programme dépendra de la qualité du modèle.
Pour cela il faut que le corpus soit de taille conséquente, c'est-à-dire au grand minimum 10000 lignes. De plus, il avoir avoir qu'à l'esprit que 10000 n'est vraiment vraiment pas grand pour entraîner un modèle. Puisque si le corpus est limité, le modèle sera aussi limité et peu fiable. De plus, il est important d'avoir un corpus qui ne traite que d'un domaine, et sur plusieurs langues.

Dans mon cas, mon corpus est sous format CSV ce qui permet d'avoir accès plus facilement aux données du fichier. Ce documents contient des informations sur la satisfaction des clients sur les compagnies que nous pouvons trouver sur Tweeter.
Ce corpus a été télécharger à partir du site Kaggle, il fait 15738 lignes.

	airline_sentiment	text
2	1	@VirginAmerica What @dhepburn said.
3	1	@VirginAmerica plus you've added commercials to the experience... tacky.
4	1	@VirginAmerica I didn't today... Must mean I need to take another trip!
5	0	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recou
6	0	@VirginAmerica and it's a really big bad thing about it
7	0	@VirginAmerica seriously would pay $30 a flight for seats that didn't have this playing. it's really the only bad thing about flying VA
8	1	@VirginAmerica yes, nearly every time I fly VX this ÃcÂ□Â□ear wormÃcÂ□Â□ wonÃcÂ□Â□t go away :)
9	1	@VirginAmerica Really missed a prime opportunity for Men Without Hats parody, there. https://t.co/mWpG7grEZP
10	1	@virginamerica Well, I didn'tÃcÂ□Â but NOW I DO! :-D
11	1	@VirginAmerica it was amazing, and arrived an hour early. You're too good to me.
12	1	@VirginAmerica did you know that suicide is the second leading cause of death among teens 10-14
13	1	@VirginAmerica I &lt

Le corpus contient 2 colonnes, puisqu'il y a « text » : qui contient tous les satisfactions des clients pour les compagnies aériennes. L'autre colonne « airline_sentiment » permet de dire si le texte était négatif ou pas. C'est-à-dire 0 veut dire que le commentaire du client est négatif et le 1 pour les commentaire non négatif.

Voici un extrait du code permettant d'accéder au information du corpus :

```
#Utiliser le même source de randomness
np.random.seed(7)
tokenizer = Tokenizer()

#ouvrir le dataset
dataset = pd.read_csv("JeuDeDonnee.csv", delimiter=",")
X = dataset.text
Y = dataset.airline_sentiment
Y2 = to_categorical(Y)

tokenizer.fit_on_texts(X)
encoded_docsX = tokenizer.texts_to_matrix(X, mode='count')
```

Pour pouvoir travailler sur le texte, il faut utiliser la tokenisation.
C'est un processus qui permet de découper une phrase en morceau de mot.
Dans notre cas, nous utilisons la word tokenization qui divisera une phrase mot par mot.

Il faut maintenant préparer les données d'entraînement et de validation, dans notre cas nous utiliserons qu'un corpus, mais nous la diviserons en 2. 80% du corpus servira pour l'entraînement du modèle, soit 12 591 et le reste servira pour la validation du modèle, soit 3 147.

```
#Diviser le jeu de donnée 80% pour l'entrainement et 20% pour valider
X_train, X_test, Y_train, Y_test = train_test_split(encoded_docsX, Y2, test_size=0.20,
random_state=7)

nbre_ligne = len(tokenizer.word_index) + 1

# create model
model = Sequential()
model.add(Dense(1000, input_dim=nbre_ligne, activation='relu'))
model.add(Dense(500, activation='relu'))
model.add(Dense(250, activation='relu'))
model.add(Dense(2, activation='sigmoid'))
```

Le modèle a 2 sorties qui sont 0 et 1. La fonction sigmoid permet d'obtenir une réponse entre 0 et 1. Notre modèle se nomme « model », nous allons l'utiliser pendant toute la suite de notre programme. Il créera notre modèle.
Lors de la compilation du modèle, il faut indiquer comment corriger les erreurs, pour cela nous allons utiliser 'adam' pour réaliser les descente du gradiant:

```
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

En suite, pour connaître la fiabilité du modèle, il faut utiliser perte et la précision des valeurs :

```
#Evaluer le model
loss, accuracy = model.evaluate(X_test, Y_test, verbose=False)
print("Loss: ", loss, " Accuracy: ", accuracy)
```

La perte devra être inférieur à 0.35, cependant dans notre cas il est de 0.45, cela est du au manque d'échantillon de notre corpus. Plus la précision est élever et plus les pertes sont bas, plus le modèle sera performant. Cependant, il faut faire attention au sur apprentissage.

Ensuite, il faut valider notre modèle, pour cela il faut essayer de faire prédire à notre sentiment de chaque ligne de notre corpus de validation.

```
#Permet de prédire les émotions dans la partie Validation
dataset_Y_prediction = model.predict_classes(X_test)

#Enregistrer les prédictions dans un fichier (validation)
n=0
f = open("validation.txt", 'w', encoding = 'utf-8')
for i in dataset_Y_prediction:
    f.write("Pour ligne: ")
    f.write(str(n))
    f.write(" valeur: ")
    f.write(str(i))
    f.write('\n')
    n=n+1

f.close()
```

Il faut maintenant passer à la partie du test de modèle.

TEST

Cette partie permet d'évaluer le modèle et ne se fait qu'après les tests de validation. Cette partie permettra de savoir si le modèle est prêt à être utilisé dans le monde réel .

Dans partie nous allons réaliser 2 type de tests :

Dans le premier test, le modèle dois essayer de prédire les sentiments des phrases d'un corpus conçu pour les tests de 1 153 lignes.

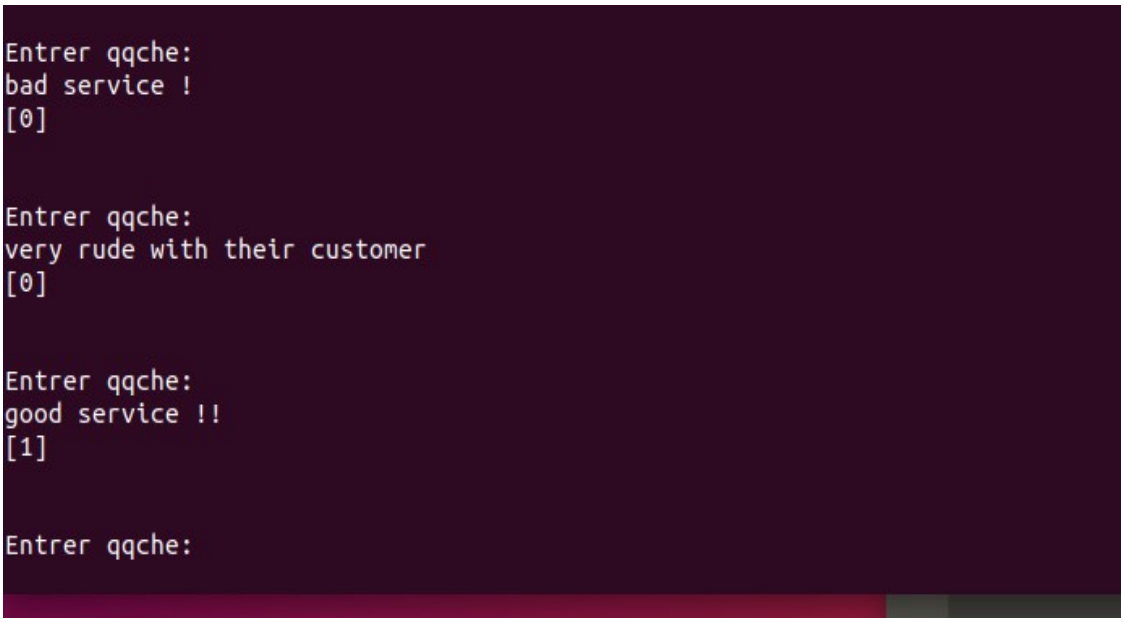
text
@AmericanAir no worries, loved flying with you guys. Thanks!
@AmericanAir you have the worst reps at Jacksonville airport. So rude and totally not helpful at all!
@AmericanAir Understood. Integration takes time, but no excuse for poor customer service
@malhoit at least you got a response from @AmericanAir via Twitter. I went 0 for 2 yesterday.
@AmericanAir I am, called Paris office this morning again, still waiting. It is in Miami but apparently tag was
@AmericanAir Can I get some assistance? Flight Cancelled Flightled (today) from PHX to DFW. Not that I'm
@AmericanAir would also like say kind move on adding the points !
@AmericanAir ok, I just received an email with my registration from your team. Thanks a lot
@AmericanAir wasn't just a delay. Your counter wouldn't take a valid CAC card as a valid ID which is neede
@AmericanAir all right, but can you give me an email to write to ?
@AmericanAir is DFW accepting any arrivals or departures today?
@AmericanAir Is it true, as FAs are saying, that you have no way to collect money from passengers (after d
@AmericanAir I called reservation at 1 am and I'm still waiting for someone to call me back
@AmericanAir a confirmed flight. I'm so done! Thanks for nothing!
@AmericanAir Not necessary. I am confident the excellent in-flight staff will make the appropriate report.

Nous comprenons quand une phrase contient des sentiments négatives ou non, le modèle arrive à comprendre ce corpus de test, alors il peut être être utiliser dans le monde réel ;

Dans le deuxième test, le programme réalisera des test en temps réel, c'est à dire que l'utilisateur va entrer un phrase et le modèle essayera de trouver si c'est négatif ou non.

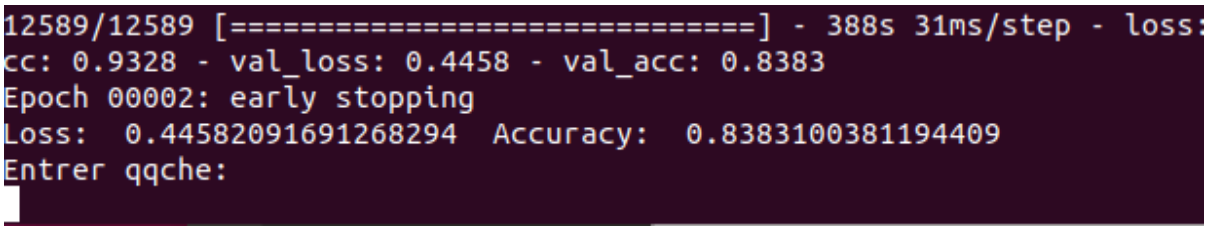
```
#Partie Test en temps réel
while(1):
    print('Entrer qqche: ')
    x = input()
    x_test = [x]
    encoded_docsX2 = tokenizer.texts_to_matrix(x_test, mode='count')
    prediction = model.predict_classes(encoded_docsX2)
    print(prediction)
    print("\n")
```

Voici un exemple de fonctionnement :



Les deux premières phrases entrées par l'utilisateur sont négatives puisqu'ils sont représentés par des 0. Les autres les autre phrase qui ne sont pas négative par des 1.

Avant de passer au test de prédiction de sentiment en temps réel, nous pouvons remarquer ces valeurs ci-dessous :



Les pertes Loss = 44,58 % et la précision Accuracy = 83,83% indique la qualité du modèle après l'entraînement sur le jeu de donnée. La précision n'est pas à 100%, ce qui évite le sur apprentissage, cependant il manque de fiabilité puisqu'il y a une perte de 44,58%.

CONCLUSION

Le projet a pour but d’analyser les sentiments la satisfaction des clients sur les compagnies aériennes en les classifiant par 0 qui est négative et par 1 qui est positive. Le projet arrive bien à créer un modèle qui va l’entraîner et réaliser des validations qui permettra d’ajuster les hyper paramètres de son modèle. Puis enfin il réalise des tests, pour qu’elle soit applicable dans la vraie vie. Cependant après l’entraînement du modèle, notre programme affiche une perte de 44%, or il faut au moins être en dessous de 35% pour être un fiable. Donc, il faut rendre encore plus fiable notre modèle.

Le problème vient donc de la qualité du corpus, puisqu'il faut des données encore plus conséquentes, mais qui soit toujours en lien avec la satisfaction des clients sur les compagnies aériennes. Il faut aussi construire un meilleur modèle en essayant de mieux utiliser les couches.

Nous pouvons utiliser ce programme afin de remplir le corpus déjà existant s'il y avait une meilleur fiabilité. Ce programme peut être coupler avec un scrapper pour chercher des informations sur le web afin de les analyser avec notre programme de reconnaissance de sentiment. Puis de remplir un corpus existant.