Analyse de sentiments

Récupération des tweets :

Dans un premier temps, je récupère les 160000 tweets du csv et les place dans un data frame panda.

Je modifie les 'polarity' des tweet en 0 pour les tweet négatifs et 1 pour les positifs afin de faciliter la lecture.

Je supprime les colonnes inutiles du data frame à savoir : « id » « date » « query » et « user »

Je supprime les doublon de tweets

```
df = pd.read_csv("archive/training.1600000.processed.noemoticon.csv",
                 names=['polarity', 'id', 'date', 'query', 'user', 'text'],
                 encoding='latin-1')
df.head()
```

	polarity	id	date	query	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

Data pre-processing :

Afin d'obtenir des données utilisable pour le Word Embeding, je commence par supprimer les identification @... et les URL qui polluent les données.

Ensuite vient le nettoyage des tweet. Pour cela, je crée une fonction « clean_tweets(text) : » qui réalise les opérations suivantes :

- Suppression de la ponctuation
- Lower case
- Suppression des Stop Words
- Lemmatization

Je place ces tweets « nettoyés » dans une nouvelle colonne du data frame.

	polarity	text	clean	length
0	0	Awww that s a bummer You shoulda got ...	awww bummer shoulda got david carr third day	44
1	0	is upset that he can t update his Facebook by ...	upset update facebook texting might cry result...	69
2	0	I dived many times for the ball Managed to ...	dived many time ball managed save 50 rest go b...	50
3	0	my whole body feels itchy and like its on fire	whole body feel itchy like fire	31
4	0	no it s not behaving at all i m mad why a...	behaving mad see	16
...	...	...	...	...
1599995	1	Just woke up Having no school is the best fee...	woke school best feeling ever	29
1599996	1	TheWDB com Very cool to hear old Walt interv...	thewdb com cool hear old walt interview	39
1599997	1	Are you ready for your MoJo Makeover Ask me f...	ready mojo makeover ask detail	30
1599998	1	Happy 38th Birthday to my boo of alll time ...	happy 38th birthday boo alll time tupac amaru ...	52
1599999	1	happy charitytuesday	happy charitytuesday	20

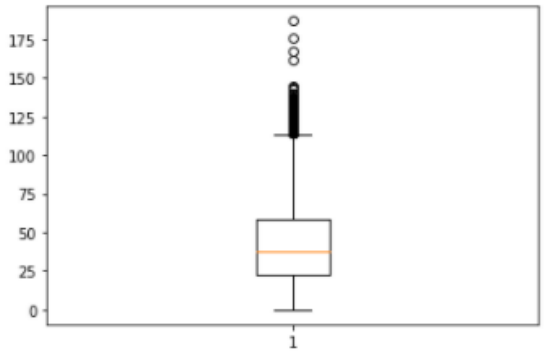
1583691 rows × 4 columns

Visualisation Data :

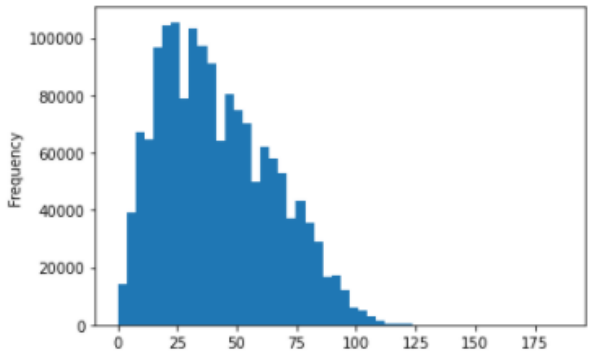
Je réalise deux graphs permettant de visualiser les données et leur longueur.

Ces visualisation me permettrons de choisir mon paramètre de Padding afin de choisir un nombre maximal de mot ni trop grand, ni trop court pour ne pas couper de tweet.

```
Average length of review in training dataset 41.803254549025034
Std Deviation  of review in training dataset 23.497097612068
```




```
<AxesSubplot:ylabel='Frequency'>
```



Préparation des donnée d'entrainement et de test :

Les données doivent maintenant être séparées en donnée d'entrainement et de test.

Avec à la fonction "train_test_split(df.clean ,df.polarity, test_size=0.1, random_state=36)"

Je mélange mes données et choisis de placer 90% de mes donnée en entrainement et 10% en test.

J'applique ensuite un Tokenizer sur mes données de test et d'entrainement , et je transforme mes tweets en liste d'entiers.

Exemple :

['try static app got yet figure supposed work']

[171, 8567, 862, 13, 130, 692, 612, 10]

Padding :

Pour donner ses données à mon RNN, toutes les entrées doivent avoir la même longueur. Pour cela, je dois limiter la longueur de mes tweets à 'max_words' Afin de ne pas tronquer le moindre tweet de ma base de donnée, je choisis d'initier 'max_words' à 200.

Les tweets plus court que 200 mots auront un certain nombre de 0 ajouté avant le tweet afin d'obtenir un array d'une taille de 200.

<u>Exemple :</u>

```
[171, 8567,  862,   13,  130,  692,  612,   10]
                      ↓
[   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
     0,    0,    0,    0,    0,  171, 8567,  862,   13,  130,  692,
   612,   10]
```

Modèle RNN :

Je crée mon model de RNN grâce à keras.Sequential() en ajoutant les layers dont j'ai besoin.

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 200, 32)           320000
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 373,301
Trainable params: 373,301
Non-trainable params: 0
_________________________________________________________________
None
```

C'est un model simple constitué d'1 embedding, 1 LSTM et 1 dense layer. Un total de 373,301 paramètres vont être entrainé

Le model va être copilé avec une fonction loss et un optimizer spécifique ainsi qu'une metric 'accuracy'.

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

Il faut ensuite entrainer le model en utilisant les données d'entrainement 'X_train' et 'Y_train'. Et en les comparant avec 'X_valid' et 'Y_valid'.

```
# Fitting the model
X_valid, y_valid = X_train[:batch_size], y_train[:batch_size]
X_train2, y_train2 = X_train[batch_size:], y_train[batch_size:]
model.fit(X_train2, y_train2, validation_data=(X_valid, y_valid), batch_size=batch_size, epochs=num_epo
```

```
Epoch 1/3
22270/22270 [==============================] - 2010s 90ms/step - loss: 0.4871 - accuracy: 0.7615 - va
l_loss: 0.4358 - val_accuracy: 0.7969
Epoch 2/3
22270/22270 [==============================] - 2928s 132ms/step - loss: 0.4459 - accuracy: 0.7886 - v
al_loss: 0.4669 - val_accuracy: 0.7344
Epoch 3/3
22270/22270 [==============================] - 2013s 90ms/step - loss: 0.4305 - accuracy: 0.7969 - va
l_loss: 0.4250 - val_accuracy: 0.7500

<tensorflow.python.keras.callbacks.History at 0x296b802f130>
```

Une fois le model entrainé, nous pouvons le tester avec nos données de test 'X_test' et 'Y_test'.

On obtient alors une accuracy d'environ 80%.

```
Entrée [40]: # Evaluation du model
             scores = model.evaluate(X_test, y_test, verbose=0)
             print('Test accuracy:', scores[1])
```

```
Test accuracy: 0.7887541651725769
```

Pipeline :

La dernière étape est de réaliser une pipeline permettant d'entrer un tout nouveau tweet dans l'algorithme, d'effectuer sur celui-ci tous les traitements puis de renvoyer la prédiction de la polarité du tweet.

```
def pipeline(tweet):
    tweet = clean_tweets(tweet)
    #print(tweet)
    tweet = tk.texts_to_sequences([tweet])
    #print(tweet)
    tweet = pad_sequences (tweet, maxlen=max_words)
    #print(tweet)
    polarity = model.predict([tweet])
    if polarity>=0.5 :
        polarity = 'positive'
    else :
        polarity = 'negative'
    return polarity
```

La fonction suivante renvoie 'positive' ou 'negative' en fonction du tweet en entrée.

API YELP :

L'analyse de sentiment sur des vrais reviews est performant.

	review	polarity
0	Today was delivery day and we were pretty exci...	positive
1	I had an issue with the undercarriage cover. T...	positive
2	Horrible customer service:1) Placed order via ...	negative
3	DREADFUL CUSTOMER SERVICE AND EXPERIENCE My mo...	negative
4	The service center here is TERRIBLE. I had a t...	negative
5	I recently brought a brand new Tesla to fix so...	negative
6	Unfortunately, as a recent new Tesla owner, I ...	negative
7	Great experience overall. Took a little long f...	positive
8	I bought a Model X with a warranty from Tesla ...	negative
9	Don't take vehicle delivery at this SF service...	negative
10	This dealership has owed me $21,000 for the pa...	negative
11	Adding to the bad reviews of this location...I...	negative
12	I had a bad experience. Technician names Adam ...	negative
13	During my first time I went into the San Franc...	positive
14	Elite '2021	positive
15	Love the technology. Hate the service. The per...	negative
16	Elite '2021	positive
17	Unfortunately I'm here to corroborate the othe...	negative
18	Great car (Model 3), absolutely incompetent st...	negative
19	Waited 1yr prior to purchase bcuz of the flaws...	negative

Amélioration :

Afin d'améliorer l'accuracy de mon model, il aurait été possible de rajouter des word embedding pré entrainés afin d'augmenter la taille des données d'entrée. Avec la database Glove par exemple.

Un autre façon d'améliorer le résultat serait de construire un model plus performant, en rajoutant des layers, changeant le nombre d'epoch, le batch size…

Une partie importante des tweet est neutre, il est donc normal que l'accuracy ne soit pas parfaite pour celle-ci.