Rapport:

E-Réputation

Macdonald’s France

Elaboré Par:

Mohamed IhebBousnina

MontassarThabti

FediBayoudh

MounaDridi

Samar Laajili

Sarah Abidli

Moez Abid

Table des matières:

Atelier I :

Définition des objectifs&Sélection des sources

1. Introduction

2. Définition des objectifs
   2.1 Obecjtifs Métier
   2.2 Objectifs Techniques

3. Collecte des données
   3.1 TrustPilot
   3.2 TripAdvisor
   3.3 Twitter

1. Introduction

Le community management n'est pas seulementune nouvelle discipline ou un nouveau métier. C'estune nouvelle manière d'appréhender la communication qui s'esquisse, de concevoir les rapports entre l'entreprise et ses clients.

Figure 1.1 Image illustrative du Concepte

2. Définition des objectifs

Comme tous les projets Data-Science nous avons deux types d'objectifs. Les objectifs métier qui représentent les fonctionnalités de base de notreprojet, et les objectifs techniques qui sont les objectifs Data-Science pour répondre à sesbesoins.

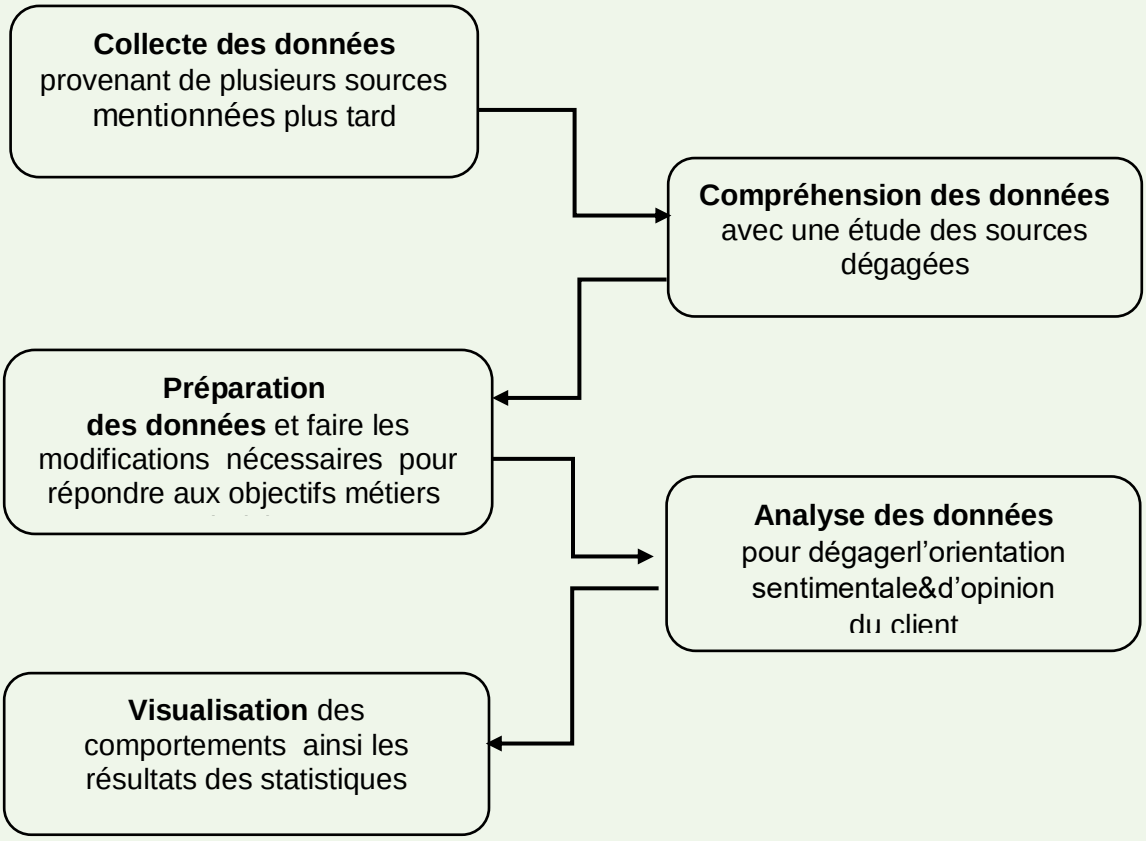
2.1 Objectifs Métier

- Maintenirune bonne e-réputation
- Protégerl'image de l'enseigne
- Analyse des avispositifs et négatifs
- Gérerunecrise

- Améliorer la veilleconcurrentielle

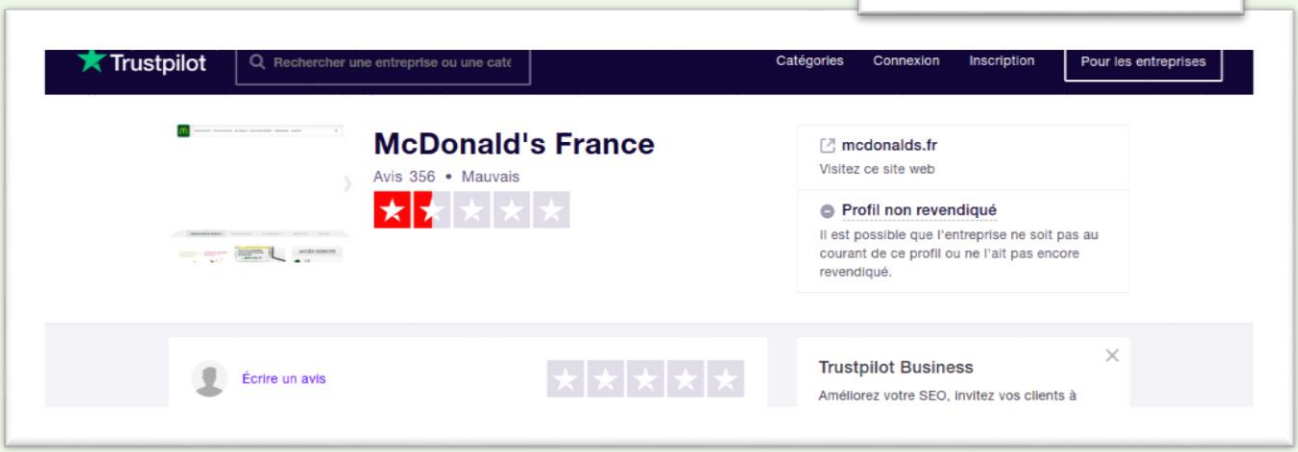
2.2 Objectifs Techniques

Pour les objectifs techniques nous allons se rapprocher de la méthodologie CRISP-DM connues en Data-Science.



3. Collecte des données

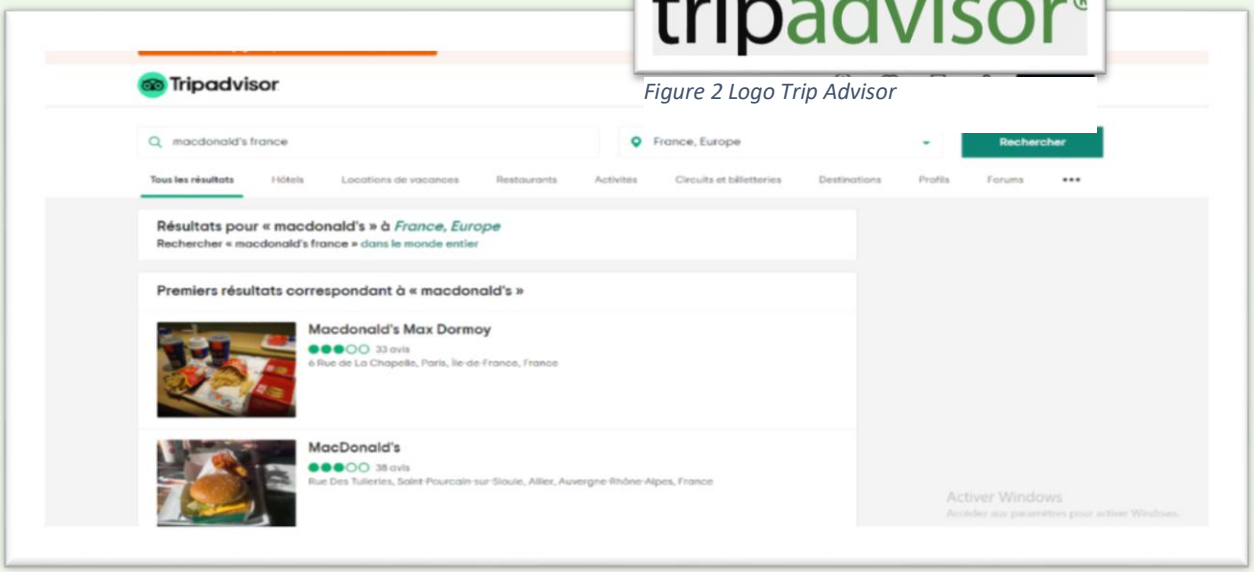
3.1 TrustPilot

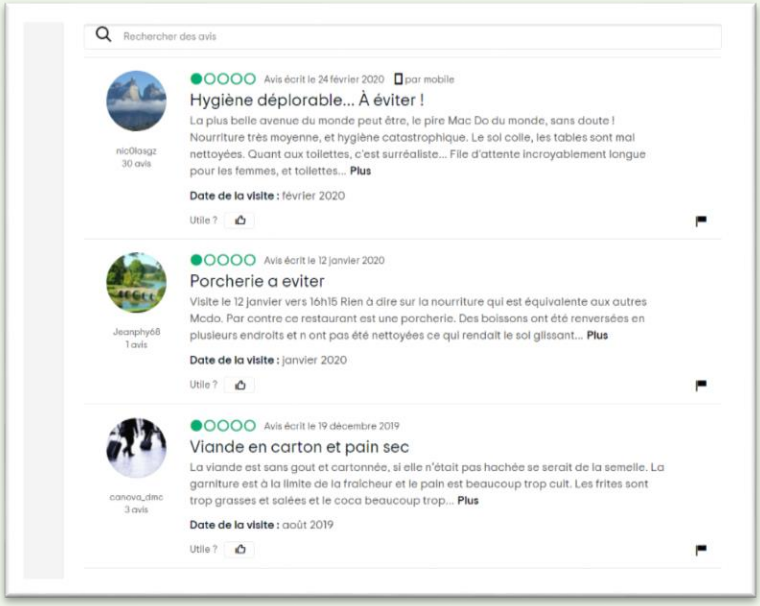


⇨ Nous avonschoisi Trustpilot car ilestl'un des meilleurs sites d'avis de consommateurs. Il hébergeenviron 900 avis pour macdonald's France.

3.2 TripAdvisor

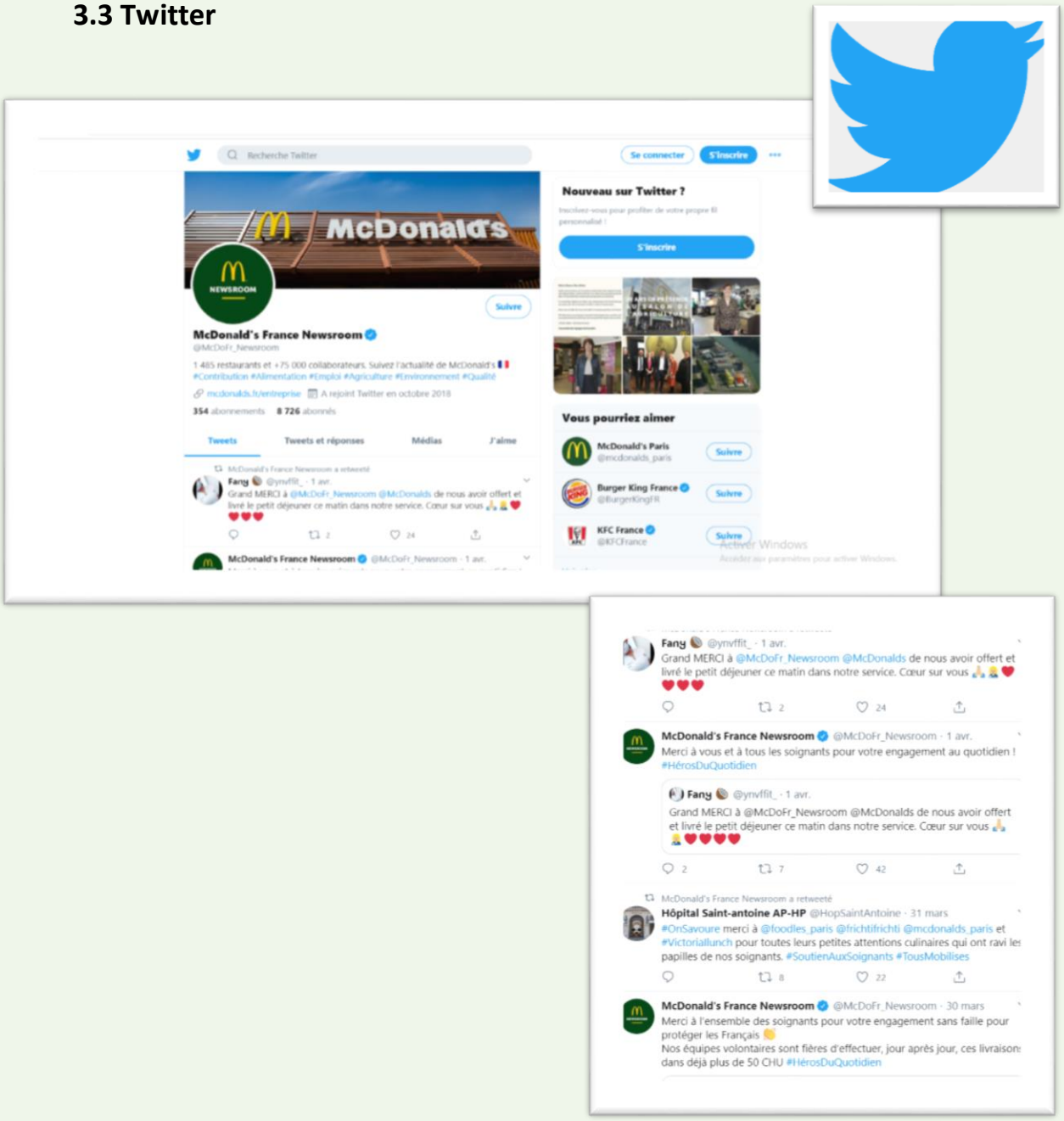
Figure 2 Logo Trip Advisor





⇨ Nous avonschoisiaussiTripAdvsor qui estl'un des sites web qui offrent des conseilstouristiques ainsi que les avis des consommateurs sur des restaurants dans le monde . Dans notre cas il y a unevariété énorme sur les avis du restaurant mcdonald's France ce qui nous amène à une analyse bien précise

3.3 Twitter



⇨ Et finalement Twitter qui estl'un des réseaux sociaux qui réserve un espace de commentaires où l'utilisateur peut donner son avis sur les restaurants.
Il englobe environ1355 commentaires sur macdonald's France. Un nombre très important pour les analyses

Atelier II:

Topic Modeling &LDA

1. Collecte&compréhension des données
2. Prétraitement des données
3. Modélisation
4. Evaluation & Test du modèle
5. Interprétation

Introduction

Le Topic Model est un modèleprobabilistepermettant de déterminer dessujetsouthèmesabstraits dans un document.Donc dans cet atelier nous allonsprésenter les données sous forme de “data-frames”, faire le prétraitement nécessaire tel que le Tokénization, l’élimination de stopwords… , faire la modélisation et finalement nous allons faire quelquesinterprétation sur les résultatstrouvés.

1. Compréhension des données

- **Twitter**

```
from twitterscraper import query_tweets
```

```
INFO: {'User-Agent': 'Opera/9.80 (X11; Linux i686; Ubuntu/14.10) Presto/2.12.388 Version/12.16'}
```

```
import pandas as pd
import datetime
from twitterscraper import query_tweets
# https://twitter.com/search-advanced
list_of_tweets = query_tweets('Mcdonalds France',
         begindate=datetime.date(2010, 1, 1),
         enddate=datetime.date(2020, 4, 10),
         lang='fr')
# Convert list of tweets to DataFrame
tweets_df = pd.DataFrame([vars(x) for x in list_of_tweets])
```

```
tweets_df=pd.read_csv('twitterData.csv')
```

```
tweets_df.shape
```

```
(1154, 22)
```

```
tweets_df.head(2)
```

	Unnamed: 0	has_media	hashtags	img_urls	is_replied	is_reply_to	likes	links	parent_tweet_id	replies	...	screen_name	text	
0	0	False	['McDonalds']	[]	False	False	0	[]	NaN	0	...	Teresathotel	RT @Ludovic_Freitas: La pub gay-friendly de #M...	class="Twe js-tweet-
1	1	False	['McDonalds']	[]	False	False	0	[]	NaN	0	...	sebastiendurand	La pub gay-friendly de #McDonalds France étonn...	class="Twe js-tweet-

2 rows × 22 columns

⇨ A partir du scraping , on a pu obtenir ce data frame qui contient les avis des consommateursainsi que la date de publication , le nombre de like, dislike , nb de “reply” …

- **TripAdvisor**

```
import gensim
from gensim.utils import simple_preprocess
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from gensim.corpora import Dictionary
from gensim.models import TfidfModel, LdaMulticore
import nltk
nltk.download('wordnet')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package wordnet to C:\Users\stef
[nltk_data]     info\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to C:\Users\stef
[nltk_data]     info\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
trustpilot = pd.read_csv('McDonalds_Champs_Elysees-Paris_Ile_de_France.csv')
trustpilot.head(3)
```

	review_body	review_date
0	Have to admit I am not a great fan of McDonald...	February 29, 2020
1	Food was fine. Staff were undesirably rude!Fam...	February 15, 2020
2	We went here to enjoy the Golden Arches fries....	November 5, 2019

⇨ Ce Data frame contient 2 colonnes:

- **Reviw_body** qui décritl'avis d'un tel client
- **Reviw_date** qui indique la date du commentaire

- **Trustpilote**

```
#L'url de base
base_url = 'https://fr.trustpilot.com/review/mcdonalds.fr'
r = rq.get(base_url)
soup = bsoup(r.text)
num_pages = 18
# ajouter dans url_liste l'ensemble des liens des pages pour faire le scrapping
url_list = ["{}?page={}".format(base_url, str(page)) for page in range(1, num_pages + 1)]

#enregistrer l'ensemble des codes des emojis
emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\u2600-\u26FF\u2700-\u27BF"
         u"\U00002702-\U000027B0"
         u"\U000024C2-\U0001F251"
                           u"\U0001f92e"
                           "]+", flags=re.UNICODE)

x=''
i=0
j=0
liste_commentaire= []
liste_avis=[]
for url_ in url_list:
    print ("Processing {}...".format(url_))
    r_new = rq.get(url_)
    soup_new = bsoup(r_new.content, 'html.parser')
    results = soup_new.find(class_="company-profile-body")
    job_elems = results.find_all('div', class_="review-content__body")
    job_elems2 = results.find_all('div', class_="star-rating star-rating--medium")
    for job_elem in job_elems2:
        liste_avis.append( "\n".join([img['alt'] for img in job_elem.find_all('img', alt=True)]).replace(" étoile :","").replace(" étoiles :",""))
        j=j+1

    for job_elem in job_elems:
```

	commentaire	avis	index
0	\n\nles mcdonald's en france sont pour la...\n\n...	4 bien	0
1	\n\nresponsable mc do hautain\n\n\n ...	1 mauvais	1
2	\n\ncorona virus\n\n\n corona v...	1 mauvais	2
3	\n\néviter mcdo peipin\n\n\n at...	1 mauvais	3
4	\n\nmc do anet\n\n\n bonjour mc...	1 mauvais	4
...	...	...	...
336	\n\nmon fast-food préféré\n\n\n ...	4 bien	336
337	\n\ndegueulasse\n\n\n dégueulas...	1 mauvais	337
338	\n\ngenial a part les sodas\n\n\n ...	5 excellent	338
339	\n\npas toujours de glace\n\n\n ...	4 bien	339
340	\n\nmac do venette nul\n\n\n pl...	1 mauvais	340

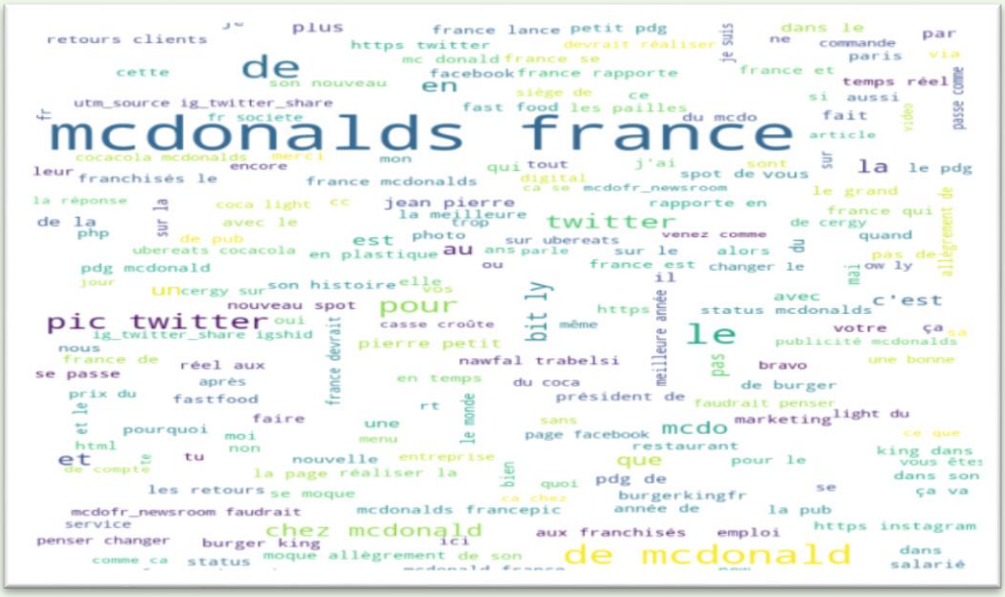
341 rows × 3 columns

- ⇨ Ce data frame contient les **commentaires**ainsi que les **avis**
- ⇨ Si l'avisest bien (4) çaveut dire que le commentaireest de satisfaction (ex: mon fast food préféré). Il est à noter que les notes sont à l'échelle de 5 (1 est la mauvaise note et 5 signifie que le restaurant est excellent)

2. Prétraitement des données

- **Twitter**

```
[ ] from wordcloud import WordCloud, STOPWORDS
    import matplotlib.pyplot as plt
    import pandas as pd


    comment_words = ''
    stopwords = set(STOPWORDS)

    # iterate through the csv file
    for val in tweets_df.text:

        # typecaste each val to string
        val = str(val)

        # split the value
        tokens = val.split()

        # Converts each token into lowercase
        for i in range(len(tokens)):
            tokens[i] = tokens[i].lower()

        comment_words += " ".join(tokens)+" "

    wordcloud = WordCloud(width = 800, height = 800,
                    background_color ='white',
                    stopwords = stopwords,
                    min_font_size = 10).generate(comment_words)

    # plot the WordCloud image
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
```



➔Le "WordCloud" est une représentation visuelle des données textuelles sous forme d'un nuage.Les mots qui sont représentés en grande taille sont les mots les plus fréquents, dont on peut citer ,pour twitter: fast-food, burger king, Paris, restaurants.
Ces mots représentent le centre d'intérêt des peuples français envers Macdonald's France

```
[ ] import gensim
    from gensim.utils import simple_preprocess
    from gensim.parsing.preprocessing import STOPWORDS
    from nltk.stem import WordNetLemmatizer, SnowballStemmer
    from nltk.stem.porter import *
    import numpy as np
    np.random.seed(2018)
    import nltk
    nltk.download('wordnet')
    stemmer = PorterStemmer()

    [nltk_data] Downloading package wordnet to C:\Users\stef
    [nltk_data]     info\AppData\Roaming\nltk_data...
    [nltk_data]   Package wordnet is already up-to-date!

[ ] def lemmatize_stemming(text):
        return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))
    def preprocess(text):
        result = []
        for token in gensim.utils.simple_preprocess(text):
            if token not in gensim.parsing.preprocessing.STOPWORDS and len(token) > 3:
                result.append(lemmatize_stemming(token))
        return result

[ ] for i in range(0,tweets_df['text'].shape[0]):
        doc_sample = tweets_df['text'][i]
        words = []
        for word in doc_sample.split(' '):
            words.append(word)
        print(words)
        print('\n\n tokenized and lemmatized document: ')
```

```
tokenized and lemmatized document:
['The', 'French', "McDonald's", 'is', 'great!', '#McDonalds', '#France']

tokenized and lemmatized document:
['Repost:', 'The', 'Asterix', 'ad', 'by', 'McDonalds', 'France', 'http://bit.ly/9OrsPb']

tokenized and lemmatized document:
['waaah', 'Mcdonalds', '(France)', 'sucks.', "it's", 'a', 'scam.', 'Now', 'the', 'sauces', 'cost', '20cents', 'LOL\neven', 'KFC', '-_-"', 'this', 'is', 'called', 'the', 'Economic', 'Crisis', 'FTL', 'wa
```

```
dictionary = gensim.corpora.Dictionary(processed_docs)
count = 0
for k, v in dictionary.iteritems():
    print(k, v)
    count += 1
    if count > 10:
        break
```

```
0 elgin
1 foodstyl
2 franc
3 game
4 heat
5 mcdonald
6 shoot
7 swelter
8 unglam
9 definit
10 employ
```

```
dictionary.filter_extremes(no_below=15, no_above=0.5, keep_n=100000)
```

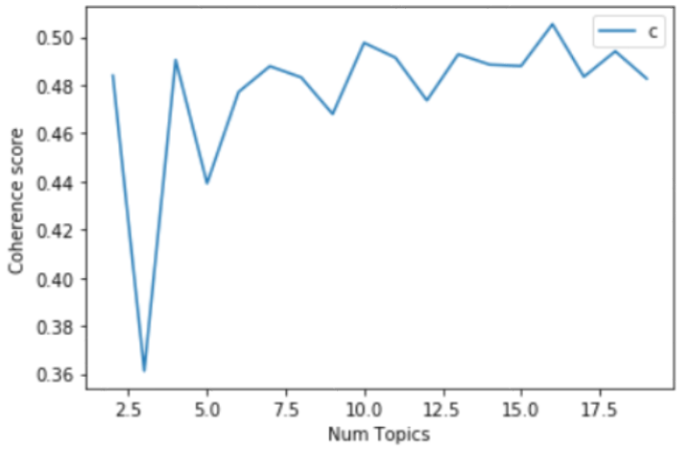
```
bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
bow_corpus[20]
```

```
[(7, 1)]
```

```
bow_doc_20 = bow_corpus[20]
for i in range(len(bow_doc_20)):
    print("Word {} (\"{}\") appears {} time.".format(bow_doc_20[i][0], dictionary[bow_doc_20[i][0]], bow_doc_20[i][1]))
```

```
Word 7 ("sell") appears 1 time.
```

```
plt.plot(model_results['Topics'], model_results['Coherence'])
plt.xlabel("Num Topics")
plt.ylabel("Coherence score")
plt.legend(("coherence_values"), loc='best')
plt.show()
```



- **TripAdvisor**

```
[ ] from wordcloud import WordCloud, STOPWORDS
    import matplotlib.pyplot as plt
    import pandas as pd


    comment_words = ''
    stopwords = set(STOPWORDS)

    # iterate through the csv file
    for val in trustpilot.review_body:

        # typecaste each val to string
        val = str(val)

        # split the value
        tokens = val.split()

        # Converts each token into lowercase
        for i in range(len(tokens)):
            tokens[i] = tokens[i].lower()

        comment_words += " ".join(tokens)+" "

    wordcloud = WordCloud(width = 800, height = 800,
                    background_color ='white',
                    stopwords = stopwords,
                    min_font_size = 10).generate(comment_words)

    # plot the WordCloud image
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
```

➔De même pour TripAdvisor ce nuage présente les mots clés les plus utilisés dans les commentaires du people français envers Macdo-France (Coffe, Food, clean, order, toilet, meal..)

```
[ ]  stem = SnowballStemmer('french')
     sp =stopwords.words('french')

     def tokenize_text(text):
         return simple_preprocess(text)

     def stemm_text (text):
         return stem.stem(text)

     def preprocess(text):
         results = []
         for word in tokenize_text(text.lower()):
             if word not in sp :
                 results.append(stemm_text(word))
         return results
```

```
[ ]  processed_docs = trustpilot['review_body'].map(preprocess)
     processed_docs[:10]
```

```
0    [hav, to, admit, am, not, great, fan, of, mcdo...
1    [food, was, fin, staff, wer, undesirably, rud,...
2    [we, went, her, to, enjoy, the, golden, arche,...
3    [when, visited, this, mcdonald, was, excited, ...
4    [this, restaurent, was, mad, uprad, from, the,...
5    [the, wait, for, the, women, toilet, was, min,...
6    [do, you, know, how, they, call, doubl, bigmac...
7    [for, busy, locat, servic, was, fast, and, fri...
8    [staff, appear, disorganized, and, unmotivated...
9    [had, two, chees, and, egg, mcmuffin, coffe, a...
Name: review_body, dtype: object
```

	review_body	review_date
0	[hav, to, admit, am, not, great, fan, of, mcdo...	February 29, 2020
1	[food, was, fin, staff, wer, undesirably, rud,...	February 15, 2020
2	[we, went, her, to, enjoy, the, golden, arche,...	November 5, 2019
3	[when, visited, this, mcdonald, was, excited, ...	November 2, 2019
4	[this, restaurent, was, mad, uprad, from, the,...	October 2, 2019

```
[ ]  dictionary = gensim.corpora.Dictionary(processed_docs)
     count = 0
     for k, v in dictionary.iteritems():
         print(k, v)
         count += 1
         if count > 10:
             break
```

```
0 actively
1 admit
2 almost
3 alway
4 am
5 ample
6 an
7 and
8 any
9 anymor
10 arc
```

```
[ ]  dictionary.filter_extremes(no_below=15, no_above=0.5, keep_n=100000)
```

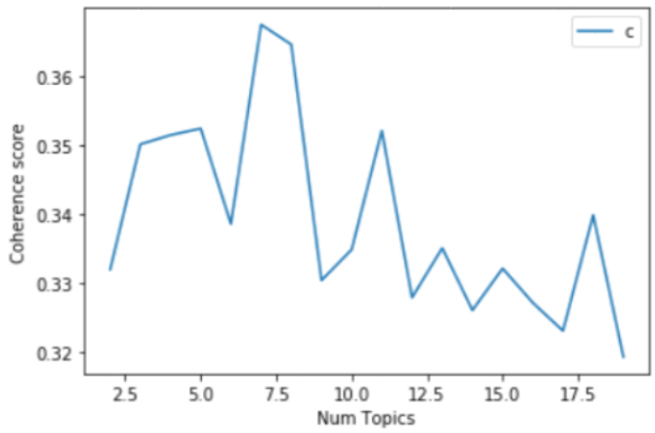
```
[ ]  bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
     bow_corpus[20]
```

```
[(3, 1),
```

```
[ ]  bow_doc_10 = bow_corpus[10]
     for i in range(len(bow_doc_10)):
         print("Word {} (\"{}\") appears {} time.".format(bow_doc_10[i][0], dictionary[bow_doc_10[i][0]], bow_doc_10[i][1]))
```

```
Word 0 ("almost") appears 1 time.
Word 1 ("alway") appears 1 time.
Word 2 ("am") appears 2 time.
Word 3 ("an") appears 2 time.
Word 4 ("any") appears 2 time.
Word 5 ("arc") appears 1 time.
```

```
[ ]  plt.plot(model_results['Topics'], model_results['Coherence'])
     plt.xlabel("Num Topics")
     plt.ylabel("Coherence score")
     plt.legend(("coherence_values"), loc='best')
     plt.show()
```



- **Trust-Pilot**

```
[ ]  #la liste des ponctuations pour les enlevers plus tard
     punctuations = list(string.punctuation)
     punctuations.append('’')
     punctuations.append("’’")
     punctuations.append("…")
     punctuations.append("``")
     punctuations.append("-")
     punctuations.append("🤣")
     punctuations.append("🤣🤣🤣")
     punctuations.append("😂")
     punctuations.append(".")

[ ]  word_tokenize(x) ;
     final =  [i.strip("".join(punctuations)) for i in word_tokenize(x) if i not in punctuations] ;


     #On créé notre set de stopwords final qui cumule ainsi les 100 mots les plus fréquents du corpus ainsi que l'ensemble de stopwords par défaut présent dans la librairie NLTK
     sw = set()
     sw.update(tuple(nltk.corpus.stopwords.words('french')))

[ ]    filtered_sentence =[]
     for w in final:
         if w not in sw :
             filtered_sentence.append(w)
     #print(filtered_sentence)

[ ]  Stemmer = FrenchStemmer()
     #'Il n'existe pas de fonction de lemmatisation de corpus français dans NLTK
     wordnet_lemmatizer = WordNetLemmatizer()
     Finalsteem = []
     #print("{0:20}{1:20}".format("mot","steem"))
     for w in filtered_sentence:
```

```
[ ]  processed_docs = df['commentaire'].map(preprocess)

     df['avis'].value_counts()
```

```
1 mauvais      277
2 bas           30
5 excellent     17
3 moyen         10
4 bien           7
Name: avis, dtype: int64
```

```
[ ]
```

```
[ ]  dictionary = gensim.corpora.Dictionary(processed_docs)
     count = 0
     for k, v in dictionary.iteritems():
         print(k, v)
         count += 1
         if count > 10:
             break
```

```
0 anné
1 augment
2 chaqu
3 dommag
4 franc
5 léger
6 mcdonald
7 menus
8 plupart
9 pour
10 prix
```

➔ 277 personnes jugent que le service de macdonalds France est mauvais . Contre seulement 17 qui ont jugé qu'il s'agit d'un excellent service . Des résultats restent à valider avec l'opinion mining (l'atelier suivant) . Mais on peut avoir une idée globale.

```
[ ] dictionary.filter_extremes(no_below=10, no_above=0.5, keep_n=100000)
```

```
[ ] """Convert document (a list of words) into the bag-of-words format = list of (token_id, token_count) 2-tuples. Each word is assumed to be a tokenized and normalized
    """

    bow_corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
    bow_corpus[0]
```

```
[(0, 1), (1, 2), (2, 1), (3, 2), (4, 1), (5, 1), (6, 1), (7, 2), (8, 1)]
```

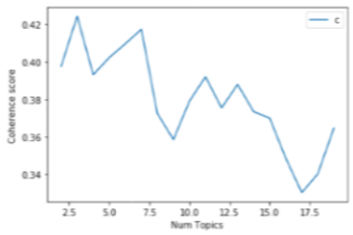
```
[ ] #pour les commentaires
    bow_doc_300 = bow_corpus[0]
    for i in range(len(bow_doc_300)):
        print("Word {} (\"{}\") appears {} time.".format(bow_doc_300[i][0],
                                                         dictionary[bow_doc_300[i][0]],
    bow_doc_300[i][1]))
```

```
Word 0 ("chaqu") appears 0 time.
Word 1 ("mcdonald") appears 1 time.
Word 2 ("menus") appears 2 time.
Word 3 ("pour") appears 3 time.
Word 4 ("prix") appears 4 time.
Word 5 ("rapid") appears 5 time.
Word 6 ("servic") appears 6 time.
Word 7 ("sont") appears 7 time.
Word 8 ("tres") appears 8 time.
```

```
[ ]
```

```
[ ] from gensim import corpora, models
    tfidf = models.TfidfModel(bow_corpus)
    corpus_tfidf = tfidf[bow_corpus]
    from pprint import pprint
```

```
] from gensim.models import CoherenceModel
  import matplotlib.pyplot as plt
  import seaborn as sns
  %matplotlib inline

  topics_range = range(2, 20, 1)
  model_results = { 'Topics': [],
                    'Coherence': []
                  }

  for k in topics_range:
      lda_model_test = gensim.models.LdaMulticore(corpus=bow_corpus, id2word=dictionary, num_topics=k, random_state=42, chunksize=100, iterations=25, passes=50, workers=3)
      coherence_model_lda = CoherenceModel(model=lda_model_test, texts=processed_docs, dictionary=dictionary, coherence='c_v')
      cv = coherence_model_lda.get_coherence()
      model_results['Topics'].append(k)
      model_results['Coherence'].append(cv)

  plt.plot(model_results['Topics'], model_results['Coherence'])
  plt.xlabel("Num Topics")
  plt.ylabel("Coherence score")
  plt.legend(("coherence_values"), loc='best')
  plt.show()
```



Explication sur les prétraitements des données sur les sites :

- Tout d'abords les traitements sont similaires dans les 3 sites :

  Après avoir faire le scrapping et organiser le résultat dans un Dataframe , on a utilisé la bibliothèque genim pour faire nos prétraitement on a commencer par rendre les mots en minuscule pour faciliter le traitement , enlever toute sorte de ponctuation , emojis, etc. Ensuite pour chaque paragraphe obtenu (les commentaires traités) , on sépare les mots et on les a mis dans une liste , et la génération des dictionnaires .

  - Le résultat final est une liste qui contient un ensemble des listes des mots séparés , On a pu tester le nombre d'occurrence pour chaque mot dans une liste .

    Finalement , on a pu générer un plot pour déterminer le nombre optimal des topics .

- **Pour trustpilot le nombre optimal est : 3**
- **Pour twitter le nombre optimal est : 15**
- **Pour tripadvisor le nombre optimal est : 7**

3. Modélisation

- **Twitter**

```
for idx, topic in lda_model.print_topics(-1):
    print('Topic: {} \nWords: {}'.format(idx, topic))
```

```
Topic: 0
Words: 0.200*"instagram" + 0.082*"pari" + 0.050*"igshid" + 0.050*"think" + 0.036*"euro" + 0.035*"breakfast" + 0.030*"work" + 0.029*"price" + 0.025*"salad" + 0.024*"holiday"
Topic: 1
Words: 0.078*"come" + 0.054*"fri" + 0.045*"market" + 0.034*"news" + 0.033*"french" + 0.032*"potato" + 0.031*"commerci" + 0.029*"emoji" + 0.026*"campaign" + 0.025*"patrick"
Topic: 2
Words: 0.132*"ebay" + 0.080*"glass" + 0.071*"anim" + 0.071*"mickey" + 0.071*"kingdom" + 0.071*"mous" + 0.071*"celebr" + 0.070*"disney" + 0.070*"world" + 0.061*"soutkn"
Topic: 3
Words: 0.085*"twitter" + 0.054*"time" + 0.040*"cool" + 0.030*"say" + 0.030*"poster" + 0.027*"shoot" + 0.027*"photograph" + 0.024*"tweet" + 0.024*"thank" + 0.023*"follow"
Topic: 4
Words: 0.077*"design" + 0.074*"friendli" + 0.058*"coffe" + 0.051*"advertis" + 0.045*"cup" + 0.041*"play" + 0.034*"great" + 0.033*"campaign" + 0.031*"twitter" + 0.029*"youtub"
Topic: 5
Words: 0.152*"bread" + 0.094*"america" + 0.091*"american" + 0.087*"burger" + 0.075*"featur" + 0.075*"promot" + 0.062*"home" + 0.056*"index" + 0.054*"stofftre" + 0.054*"id_thread"
Topic: 6
Words: 0.077*"mcdo" + 0.047*"burger" + 0.047*"html" + 0.041*"macaron" + 0.040*"order" + 0.036*"sell" + 0.033*"homeless" + 0.030*"assault" + 0.030*"king" + 0.029*"twitter"
```

```
lda_model_tfidf = gensim.models.LdaMulticore(corpus_tfidf, num_topics=10, id2word=dictionary, passes=2, workers=4)
for idx, topic in lda_model_tfidf.print_topics(-1):
    print('Topic: {} Word: {}'.format(idx, topic))
```

```
Topic: 0 Word: 0.080*"twitter" + 0.027*"coffe" + 0.026*"design" + 0.026*"cup" + 0.024*"francep" + 0.019*"friendli" + 0.016*"right" + 0.016*"happi" + 0.016*"tbwa_pari" + 0.013*"news"
Topic: 1 Word: 0.044*"wifi" + 0.027*"want" + 0.025*"style" + 0.020*"advertis" + 0.019*"facebook" + 0.019*"song" + 0.019*"tinyurl" + 0.018*"employ" + 0.016*"french" + 0.016*"ebay"
Topic: 2 Word: 0.036*"check" + 0.034*"burger" + 0.032*"realiti" + 0.030*"work" + 0.028*"free" + 0.026*"mcflurri" + 0.025*"comté" + 0.022*"instagram" + 0.020*"wifi" + 0.019*"chicken"
Topic: 3 Word: 0.037*"love" + 0.023*"say" + 0.021*"order" + 0.021*"come" + 0.019*"twitter" + 0.019*"french" + 0.018*"brand" + 0.017*"world" + 0.016*"instagram" + 0.016*"breakfast"
Topic: 4 Word: 0.047*"twitter" + 0.026*"mcdo" + 0.021*"macaron" + 0.020*"statu" + 0.018*"good" + 0.018*"blog" + 0.018*"page" + 0.017*"video" + 0.016*"emoji" + 0.014*"facebook"
Topic: 5 Word: 0.037*"burger" + 0.031*"instagr" + 0.030*"comté" + 0.028*"food" + 0.028*"realiti" + 0.023*"chees" + 0.022*"check" + 0.021*"youtub" + 0.020*"hamburg" + 0.020*"royal"
Topic: 6 Word: 0.052*"anim" + 0.052*"kingdom" + 0.052*"mous" + 0.052*"mickey" + 0.051*"celebr" + 0.051*"disney" + 0.049*"glass" + 0.049*"world" + 0.042*"pinterest" + 0.032*"ebay"
Topic: 7 Word: 0.036*"cool" + 0.023*"french" + 0.023*"mcbaguett" + 0.022*"like" + 0.022*"time" + 0.019*"twitter" + 0.017*"design" + 0.017*"ebay" + 0.014*"termin" + 0.013*"bagel"
Topic: 8 Word: 0.036*"bread" + 0.028*"burger" + 0.028*"american" + 0.023*"campaign" + 0.022*"know" + 0.021*"advertis" + 0.019*"come" + 0.018*"featur" + 0.018*"veggi" + 0.018*"id_thre
Topic: 9 Word: 0.028*"news" + 0.024*"twitter" + 0.020*"great" + 0.019*"nice" + 0.017*"euro" + 0.016*"design" + 0.016*"tell" + 0.016*"serv" + 0.015*"photo" + 0.014*"market"
```

- On a pu générer environ 14 topic , par ailleurs on peut conclure qu'il 14 sujets qui intéressent l'opinion public des français. Par exemple le sujet numéro 1 qui parle de stratégie de marketing et la nouveauté de macdo france : "frensh potato". Il y a aussi le sujet numéro 1 qui parle des prix.
- Avec TIDF : il y a par exemple le sujet numéro 8 qui parle de la nourriture offerte par macdo france : bread , burger ,veggi …

```
for index, score in sorted(lda_model[bow_corpus[20]], key=lambda tup: -1*tup[1]):
    print("\nScore: {}\t \nTopic: {}".format(score, lda_model.print_topic(index, 10)))
```

```
Score: 0.5499900579452515
Topic: 0.156*"burger" + 0.138*"check" + 0.118*"realiti" + 0.111*"comté" + 0.060*"hamburg" + 0.054*"mcdo" + 0.050*"french" + 0.033*"sell" + 0.030*"youtub" + 0.029*"html"

Score: 0.050007160753011703
Topic: 0.096*"french" + 0.068*"world" + 0.045*"pari" + 0.044*"ebay" + 0.044*"burger" + 0.043*"glass" + 0.041*"html" + 0.032*"love" + 0.030*"fri" + 0.029*"anim"

Score: 0.050000566989183426
Topic: 0.140*"design" + 0.069*"coffe" + 0.064*"cup" + 0.055*"friendli" + 0.052*"mobil" + 0.048*"chees" + 0.036*"like" + 0.035*"royal" + 0.035*"twitter" + 0.032*"custom"

Score: 0.050000566989183426
Topic: 0.096*"wifi" + 0.072*"need" + 0.066*"advertis" + 0.063*"order" + 0.063*"food" + 0.053*"sauc" + 0.051*"free" + 0.040*"market" + 0.039*"open" + 0.038*"love"

Score: 0.05000053346157074
Topic: 0.142*"burger" + 0.102*"instagram" + 0.058*"time" + 0.050*"instagr" + 0.041*"think" + 0.036*"fri" + 0.030*"campaign" + 0.027*"serv" + 0.027*"realiti" + 0.025*"check"
```

- Avec une probabilité de 54%, le sujet de bow_corpus[20] est que le burger en réalité n'est pas le même que celui publié dans les réseaux sociaux (exp : youtube)

- **TripAdvisor**

```
[28]: lda_model = gensim.models.LdaMulticore(corpus_tfidf, num_topics=7, id2word=dictionary,chunksize=100, iterations=50, passes=50, workers=3)

[29]: for idx, topic in lda_model.print_topics(-1):
          print('Topic: {} \nWords: {}'.format(idx, topic))

Topic: 0
Words: 0.036*"chees" + 0.033*"cak" + 0.033*"mc" + 0.032*"donald" + 0.024*"bit" + 0.022*"though" + 0.021*"with" + 0.016*"burger" + 0.016*"caf" + 0.014*"out"
Topic: 1
Words: 0.018*"you" + 0.014*"easy" + 0.014*"really" + 0.013*"nic" + 0.013*"mccaf" + 0.013*"busy" + 0.012*"champ" + 0.012*"is" + 0.012*"meal" + 0.012*"your"
Topic: 2
Words: 0.041*"breakfast" + 0.029*"typical" + 0.018*"family" + 0.016*"wer" + 0.014*"lov" + 0.014*"far" + 0.014*"rud" + 0.013*"ye" + 0.013*"such" + 0.013*"just"
Topic: 3
Words: 0.022*"worth" + 0.020*"instead" + 0.017*"do" + 0.016*"they" + 0.015*"don" + 0.015*"you" + 0.015*"world" + 0.014*"rud" + 0.014*"best" + 0.014*"quit"
Topic: 4
Words: 0.021*"enjoy" + 0.018*"nothing" + 0.018*"usual" + 0.017*"tried" + 0.016*"street" + 0.016*"view" + 0.016*"donald" + 0.016*"clean" + 0.015*"sit" + 0.015*"som"
Topic: 5
Words: 0.014*"we" + 0.010*"wer" + 0.010*"that" + 0.009*"had" + 0.009*"you" + 0.008*"at" + 0.008*"ther" + 0.008*"this" + 0.008*"not" + 0.008*"they"
Topic: 6
Words: 0.023*"different" + 0.019*"servic" + 0.019*"tast" + 0.018*"family" + 0.017*"off" + 0.017*"than" + 0.014*"pay" + 0.013*"expect" + 0.013*"great" + 0.012*"mcdonald"
```

- Avec tripAdvisor , on a pu générer un total de 7 sujets (les sujets les plus compréhensibles et clair à partir du scrapping qu'on a fait tout à l'heure).
  Dans notre explication , on va essayer de mentionner seulement les sujets les plus pertinents dans notre contexte.
  Sujet numéro 6 : macdonalds France est un espace familial avec un bon service.
  Sujet numéro 0 : la nourriture offerte par macdo France .
  Sujet numéro 4 : il s'agit d'un espace calme et une vue panoramique et de plus propre .
- Globalement le client est plus ou moins satisfait du service offert par macdonalds. Mais ca ne reste pas une conclusion finale . il faut faire l'analyse des autres sites pour pouvoir connaitre le positionnement de macdo France par rapport à ses concurrents .

- **Trust-Pilot**

```
lda_model = gensim.models.LdaMulticore(bow_corpus, num_topics=3, id2word=dictionary, random_state=42, chunksize=100, iterations=10, passe
for idx, topic in lda_model.print_topics(-1):
    print('Topic: {} \nWords: {}'.format(idx, topic))
```

```
Topic: 0
Words: 0.053*"gobelet" + 0.050*"nouveau" + 0.039*"couvercl" + 0.035*"pour" + 0.031*"tout" + 0.027*"dan" + 0.027*"boisson" + 0.026*"plus" +
0.025*"paill" + 0.022*"renvers"
Topic: 1
Words: 0.042*"nous" + 0.041*"pour" + 0.027*"dan" + 0.026*"servic" + 0.026*"command" + 0.025*"avec" + 0.024*"plus" + 0.022*"vous" + 0.019*"r
estaur" + 0.018*"tout"
Topic: 2
Words: 0.057*"command" + 0.035*"driv" + 0.035*"fois" + 0.030*"chez" + 0.029*"plus" + 0.026*"mcdonald" + 0.021*"sandwich" + 0.019*"toujour"
+ 0.018*"client" + 0.018*"sont"
```

- Avec trustpilot , on a pu générer 3 sujets :
  Sujet 1 : les avis sur les boissons .
  Sujet 2 : le service offert par macdonalds France .
  Sujet 3 : le système de commande chez macdonalds France .

```
bow_vector = dictionary.doc2bow(preprocess("La nourriture était bonne. Le personnel était désagréablement désagréable! Sortie en famille c
for index, score in sorted(lda_model[bow_vector], key=lambda tup: -1*tup[1]):
    print("Score: {}\t Topic: {}".format(score, lda_model.print_topic(index, 4)))
```

```
Score: 0.8657394647598267       Topic: 0.042*"nous" + 0.041*"pour" + 0.027*"dan" + 0.026*"servic"
Score: 0.08635911345481873      Topic: 0.053*"gobelet" + 0.050*"nouveau" + 0.039*"couvercl" + 0.035*"pour"
Score: 0.04790142551064491      Topic: 0.057*"command" + 0.035*"driv" + 0.035*"fois" + 0.030*"chez"
```

« La nourriture était bonne. Le personnel était désagréablement désagréable! Sortie en famille complètement ruinée par un homme grossier, irrespectueux et désagréable À ÉVITER»

- **Avec une probabilité de 0.86 , le sujet passé comme input parle du service .**
  Ce qui est correct .

4. Interprétation globale

D'après les analyses faites (Topic modeling)dans les trois sites : tripAdvisor , Twitter et Trustpilit , On peut conclure que :
Ainsi à travers l'analyse des mots les plus utilisés dans les dictionnaires générés on a pu constaté le positionnement de la marque dans le marché où la majorité montre une insatisfaction totale du service ce qui peut endommager plus au moins la e-reputation de Macdonalds france . Ceci n'empêche pas l'existence de quelques mots positifs mais avec une faible fréquence du coup une faible influence. Pour conclure avec les analyses faites, cette dernière doit revisiter sa stratégie afin de marquer sa place.
Dans l'atelier suivant , on va essayer de faire l'analyse des opinion pour bien confirmer les résultats trouvés .

Atelier III :

Opinion&Sentiment Mining

1. Analyse sur les commentaires
    - TripAdvisor
    - Trustpilot
    - Twitter

2. Analyse sur les avis
    - TripAdvisor
    - Trustpilot
    - Twitter

3. Opinion classification
    - TripAdvisor
    - Trustpilot
    - Twitter

4. Intérpretation globale

Introduction

L'analyse des sentiments s'intéresse à l'orientation d'une opinion par rapport à une ou à un aspect d'une entité. On parle de polarité, elle peut être positive, neutre ou négative.

1. Analyse sur les commentaires

- TripAdvisor

- **Avec vaderSentiment :**

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import time
analyzer = SentimentIntensityAnalyzer()

pos_count = 0
pos_correct = 0


for line in data['review_body']:
    vs = analyzer.polarity_scores(line)
    if not vs['neg'] > 0.6:
        if vs['pos']-vs['neg'] > 0:
            pos_correct += 1
        pos_count +=1


neg_count = 0
neg_correct = 0
x=0

for line in data['review_body']:
    x=x+1
    vs = analyzer.polarity_scores(line)
    if not vs['pos'] > 0.6:
        if vs['pos']-vs['neg'] <= 0:
            neg_correct += 1
        neg_count +=1

print("Nombre de ligne à analyser:{}".format(x)  )
print("Positive accuracy = {}% via {} samples".format(pos_correct/pos_count*100.0, pos_count))
print("Negative accuracy = {}% via {} samples".format(neg_correct/neg_count*100.0, neg_count))
```

The output :

```
Nombre de ligne à analyser:540
Positive accuracy = 68.51851851851852% via 540 samples
Negative accuracy = 31.48148148148148% via 540 samples
```

➔ Contrairement à trustpilot et Twitter , les avis dans tripadvisor sont plutôt positifs avec un pourcentage de 68% ,une majorité faible , qu'à travers laquelle on ne peut pas juger ni la e-réputation qui semble dans ce cas plus ou moins positive ni son emplacement dans le marché .

- TrustPilot

- **Avec vaderSentiment :**

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import time
analyzer = SentimentIntensityAnalyzer()

pos_count = 0
pos_correct = 0


for line in df['commentaire']:
    vs = analyzer.polarity_scores(line)
    if not vs['neg'] > 0.4:
        if vs['pos']-vs['neg'] > 0:
            pos_correct += 1
        pos_count +=1


neg_count = 0
neg_correct = 0
x=0

for line in df['commentaire']:
    x=x+1
    vs = analyzer.polarity_scores(line)
    if not vs['pos'] > 0.4:
        if vs['pos']-vs['neg'] <= 0:
            neg_correct += 1
        neg_count +=1

print("Nombre de ligne à analyser:{}".format(x)  )
print("Positive accuracy = {}% via {} samples".format(pos_correct/pos_count*100.0, pos_count))
print("Negative accuracy = {}% via {} samples".format(neg_correct/neg_count*100.0, neg_count))
```

The output :

```
Nombre de ligne à analyser:342
Positive accuracy = 17.251461988304094% via 342 samples
Negative accuracy = 82.7485380116959% via 342 samples
```

➔ On remarque que 82% de la population française (342 commentaires peut être considérer comme étant un échantillon représentative) pensent que macdonalds n'est pas une enseigne à retenir à cause des différents raisons comme le service (un détail qu'on l'a dégagé lors du TP précédent : Topic modeling)

- Twitter

- **Avec textblob :**

```
[1]: import sys,tweepy,csv,re
     from textblob import TextBlob
     import matplotlib.pyplot as plt
```

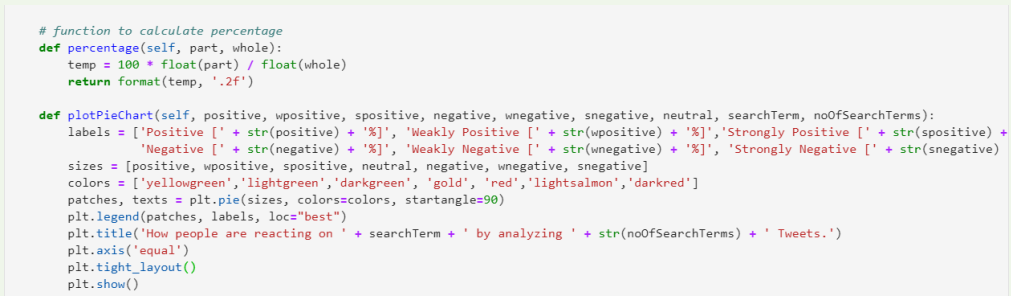
```
[1]: import sys,tweepy,csv,re
     from textblob import TextBlob
     import matplotlib.pyplot as plt


     class SentimentAnalysis:

         def __init__(self):
             self.tweets = []
             self.tweetText = []

         def DownloadData(self):
             # authenticating
             #Twitter API credentials

             consumerKey = 'tiVIII7j3O9bPAbRgdJWmoJWd'
             consumerSecret = 'SR5KZuyzFgXkXvd0tADD8qFDfYzQ0mXBaTTvgDq0SagnHlWLEU'
             accessToken = '490394724-yTbJ770PfbqAqdiejo8IfEEKkIKTTtmTNodZI0ru'
             accessTokenSecret = '3qouPTCCnp6t1tfV6NjcXbJ4ulQ4wJ5WQt9QZzwURuTGo'


             auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
             auth.set_access_token(accessToken, accessTokenSecret)
             api = tweepy.API(auth)

             # input for term to be searched and how many tweets to search
             searchTerm = input("Enter Keyword/Tag to search about: ")
```



```
# input for term to be searched and how many tweets to search
searchTerm = input("Enter Keyword/Tag to search about: ")
NoOfTerms = int(input("Enter how many tweets to search: "))

# searching for tweets
self.tweets = tweepy.Cursor(api.search, q=searchTerm, lang = "fr").items(NoOfTerms)

# Open/create a file to append data to
csvFile = open('result.csv', 'a')

# Use csv writer
csvWriter = csv.writer(csvFile)


# creating some variables to store info
polarity = 0
positive = 0
wpositive = 0
spositive = 0
negative = 0
wnegative = 0
snegative = 0
neutral = 0


# iterating through tweets fetched
for tweet in self.tweets:
    #Append to temp so that we can store in csv later. I use encode UTF-8
    self.tweetText.append(self.cleanTweet(tweet.text).encode('utf-8'))
    # print (tweet.text.translate(non_bmp_map))    #print tweet's text
    analysis = TextBlob(tweet.text)
    # print(analysis.sentiment)  # print tweet's polarity
    polarity += analysis.sentiment.polarity  # adding up polarities to find the average later
```

```
        if (analysis.sentiment.polarity == 0):  # adding reaction of how people are reacting to find average later
            neutral += 1
        elif (analysis.sentiment.polarity > 0 and analysis.sentiment.polarity <= 0.3):
            wpositive += 1
        elif (analysis.sentiment.polarity > 0.3 and analysis.sentiment.polarity <= 0.6):
            positive += 1
        elif (analysis.sentiment.polarity > 0.6 and analysis.sentiment.polarity <= 1):
            spositive += 1
        elif (analysis.sentiment.polarity > -0.3 and analysis.sentiment.polarity <= 0):
            wnegative += 1
        elif (analysis.sentiment.polarity > -0.6 and analysis.sentiment.polarity <= -0.3):
            negative += 1
        elif (analysis.sentiment.polarity > -1 and analysis.sentiment.polarity <= -0.6):
            snegative += 1


    # Write to csv and close csv file
    csvWriter.writerow(self.tweetText)
    csvFile.close()

    # finding average of how people are reacting
    positive = self.percentage(positive, NoOfTerms)
    wpositive = self.percentage(wpositive, NoOfTerms)
    spositive = self.percentage(spositive, NoOfTerms)
    negative = self.percentage(negative, NoOfTerms)
    wnegative = self.percentage(wnegative, NoOfTerms)
    snegative = self.percentage(snegative, NoOfTerms)
    neutral = self.percentage(neutral, NoOfTerms)

    # finding average reaction
    polarity = polarity / NoOfTerms

    # printing out data
    print("How people are reacting on " + searchTerm + " by analyzing " + str(NoOfTerms) + " tweets.")
    print()
```

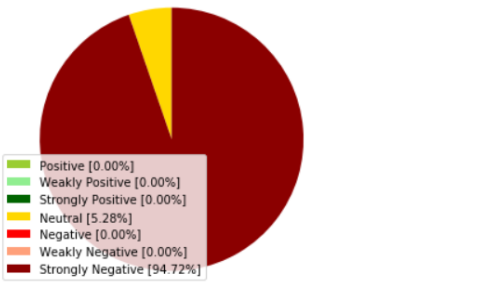

```
    print()
    print("General Report: ")

    if (polarity == 0):
        print("Neutral")
    elif (polarity > 0 and polarity <= 0.3):
        print("Weakly Positive")
    elif (polarity > 0.3 and polarity <= 0.6):
        print("Positive")
    elif (polarity > 0.6 and polarity <= 1):
        print("Strongly Positive")
    elif (polarity > -0.3 and polarity <= 0):
        print("Weakly Negative")
    elif (polarity > -0.6 and polarity <= -0.3):
        print("Negative")
    elif (polarity > -1 and polarity <= -0.6):
        print("Strongly Negative")

    print()
    print("Detailed Report: ")
    print(str(positive) + "% people thought it was positive")
    print(str(wpositive) + "% people thought it was weakly positive")
    print(str(spositive) + "% people thought it was strongly positive")
    print(str(negative) + "% people thought it was negative")
    print(str(wnegative) + "% people thought it was weakly negative")
    print(str(snegative) + "% people thought it was strongly negative")
    print(str(neutral) + "% people thought it was neutral")

    self.plotPieChart(positive, wpositive, spositive, negative, wnegative, snegative, neutral, searchTerm, NoOfTerms)


def cleanTweet(self, tweet):
    # Remove Links, Special Characters etc from tweet
    return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t]) | (\w +:\ / \ / \S +)", " ", tweet).split())
```

```
# function to calculate percentage
def percentage(self, part, whole):
    temp = 100 * float(part) / float(whole)
    return format(temp, '.2f')

def plotPieChart(self, positive, wpositive, spositive, negative, wnegative, snegative, neutral, searchTerm, noOfSearchTerms):
    labels = ['Positive [' + str(positive) + '%]', 'Weakly Positive [' + str(wpositive) + '%]','Strongly Positive [' + str(spositive) +
              'Negative [' + str(negative) + '%]', 'Weakly Negative [' + str(wnegative) + '%]', 'Strongly Negative [' + str(snegative)
    sizes = [positive, wpositive, spositive, neutral, negative, wnegative, snegative]
    colors = ['yellowgreen','lightgreen','darkgreen', 'gold', 'red','lightsalmon','darkred']
    patches, texts = plt.pie(sizes, colors=colors, startangle=90)
    plt.legend(patches, labels, loc="best")
    plt.title('How people are reacting on ' + searchTerm + ' by analyzing ' + str(noOfSearchTerms) + ' Tweets.')
    plt.axis('equal')
    plt.tight_layout()
    plt.show()
```

The output :

```
[3]: objet = SentimentAnalysis()
     objet.DownloadData()

Enter Keyword/Tag to search about:  Mcdonalds France
Enter how many tweets to search:  1155
How people are reacting on Mcdonalds France by analyzing 1155 tweets.

General Report:
Strongly Negative

Detailed Report:
0.00% people thought it was positive
0.00% people thought it was weakly positive
0.00% people thought it was strongly positive
0.00% people thought it was negative
0.00% people thought it was weakly negative
94.72% people thought it was strongly negative
5.28% people thought it was neutral
```



- **Avec vaderSentiment:**

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import time
analyzer = SentimentIntensityAnalyzer()

pos_count = 0
pos_correct = 0


for line in tweets_df['text']:
    vs = analyzer.polarity_scores(line)
    if not vs['neg'] > 0.4:
        if vs['pos']-vs['neg'] > 0:
            pos_correct += 1
        pos_count +=1


neg_count = 0
neg_correct = 0
x=0

for line in tweets_df['text']:
    x=x+1
    vs = analyzer.polarity_scores(line)
    if not vs['pos'] > 0.4:
        if vs['pos']-vs['neg'] <= 0:
            neg_correct += 1
        neg_count +=1

print("Nombre de ligne à analyser:{}".format(x)  )
print("Positive accuracy = {}% via {} samples".format(pos_correct/pos_count*100.0, pos_count))
print("Negative accuracy = {}% via {} samples".format(neg_correct/neg_count*100.0, neg_count))
```

The output :

```
Nombre de ligne à analyser:1155
Positive accuracy = 9.437229437229437% via 1155 samples
Negative accuracy = 90.56277056277057% via 1155 samples
```

- On remarque que les résultats sont plus ou moins similaire .
- Environ 90% de la population française pensent que macdonalds France est une mauvaise enseigne. Un espace à boycotter . Ceci , bien évidemment, a une influence énorme sur sa réputation .

2. Analyse sur les avis

○ TripAdvisor

Après le scrappingdu site “TripAdvisor” nous avonsobtenuce data frame qui contient4 colonnes:

- **Rate_name** qui indiquel’avis du consommateur (ex: 3 of 5 bubbles)
- **Rate_lieu** qui indique le lieu du macdo
- **Rate_nb_review** Le nombre de reviews
- **Rate_nb_url**qui indiquel’url du rate.

```
import pandas as pd
import numpy as np
df = pd.read_csv("nouveau.csv",sep=";")
df
```

	Rate_name	Rate_Lieu	Rate_nb_review	Rate_nb_review_url
0	3 of 5 bubbles	140 avenue des Champs Elysees, Paris, Ile-de-F...	740 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1	3 of 5 bubbles	Disney Village, Marne-la-Vallee, Seine-et-Marn...	831 reviews	https://www.tripadvisor.com/Restaurant_Review-...
2	3 of 5 bubbles	184 rue de Rivoli, Paris, Ile-de-France, France	156 reviews	https://www.tripadvisor.com/Restaurant_Review-...
3	2 of 5 bubbles	2 boulevard Poissonniere, Paris, Ile-de-France...	34 reviews	https://www.tripadvisor.com/Restaurant_Review-...
4	3 of 5 bubbles	19 Place de la Republique, Paris, Ile-de-Franc...	123 reviews	https://www.tripadvisor.com/Restaurant_Review-...
...	...	...	...	...
1015	3.5 of 5 bubbles	9 boulevard Malesherbes, Paris, Ile-de-France,...	13 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1016	4.5 of 5 bubbles	235 avenue des Mouettes, St-Laurent du Var, Fr...	131 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1017	3.5 of 5 bubbles	22 rue de Ponthieu, Paris, Ile-de-France, France	131 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1018	4.5 of 5 bubbles	12 rue d Alsace Lorraine, Nice, French Riviera...	280 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1019	4.5 of 5 bubbles	Arc 2000 Place Haute, Les Arcs, Bourg Saint Ma...	11 reviews	https://www.tripadvisor.com/Restaurant_Review-...

Nous passons maintenant à la phase du nettoyage des données.

```
df['Rate_name'] = np.where(df['Rate_name']=='1 of 5 bubbles ', 'Mauvais', df['Rate_name'])
df['Rate_name'] = np.where(df['Rate_name']=='1.5 of 5 bubbles ', 'Mauvais', df['Rate_name'])

df['Rate_name'] = np.where(df['Rate_name']=='2 of 5 bubbles ', 'Bas', df['Rate_name'])
df['Rate_name'] = np.where(df['Rate_name']=='2.5 of 5 bubbles ', 'Bas', df['Rate_name'])

df['Rate_name'] = np.where(df['Rate_name']=='3 of 5 bubbles ', 'Moyen', df['Rate_name'])
df['Rate_name'] = np.where(df['Rate_name']=='3.5 of 5 bubbles ', 'Moyen', df['Rate_name'])

df['Rate_name'] = np.where(df['Rate_name']=='4 of 5 bubbles ', 'Bien', df['Rate_name'])
df['Rate_name'] = np.where(df['Rate_name']=='4.5 of 5 bubbles ', 'Bien', df['Rate_name'])

df['Rate_name'] = np.where(df['Rate_name']=='5 of 5 bubbles ', 'Excelent', df['Rate_name'])
```

```
df
```

	Rate_name	Rate_Lieu	Rate_nb_review	Rate_nb_review_url
0	Moyen	140 avenue des Champs Elysees, Paris, Ile-de-F...	740 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1	Moyen	Disney Village, Marne-la-Vallee, Seine-et-Marn...	831 reviews	https://www.tripadvisor.com/Restaurant_Review-...
2	Moyen	184 rue de Rivoli, Paris, Ile-de-France, France	156 reviews	https://www.tripadvisor.com/Restaurant_Review-...
3	Bas	2 boulevard Poissonniere, Paris, Ile-de-France...	34 reviews	https://www.tripadvisor.com/Restaurant_Review-...
4	Moyen	19 Place de la Republique, Paris, Ile-de-Franc...	123 reviews	https://www.tripadvisor.com/Restaurant_Review-...
...	...	...	...	...
1015	Moyen	9 boulevard Malesherbes, Paris, Ile-de-France,...	13 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1016	Bien	235 avenue des Mouettes, St-Laurent du Var, Fr...	131 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1017	Moyen	22 rue de Ponthieu, Paris, Ile-de-France, France	131 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1018	Bien	12 rue d Alsace Lorraine, Nice, French Riviera...	280 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1019	Bien	Arc 2000 Place Haute, Les Arcs, Bourg Saint Ma...	11 reviews	https://www.tripadvisor.com/Restaurant_Review-...

⇨ On aattribué des mots plus significatifs pour les valeurs de la colonnerate_name

- Excellent pour les avis entre 4 et 5
- Moyen pour les avis entre 3 et 4
- Mauvais pour les avis entre 1 et 2

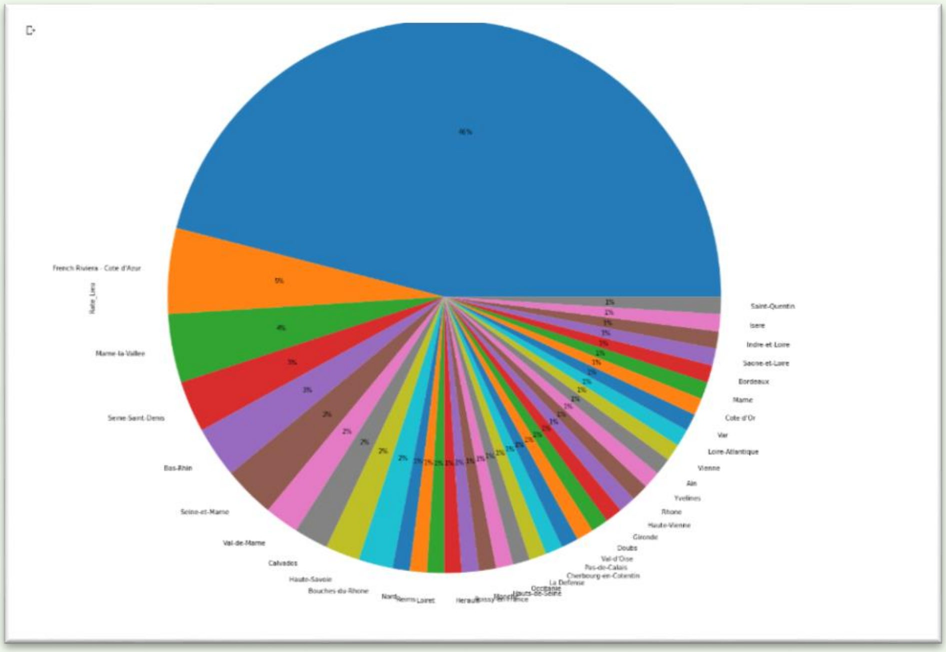
⇨ Ensuite on a groupé le lieu par régions , par exemple deux macdonald's qui se situent dans la mêmerégion on les regroupe.

⇨ On obtient plus visible et plus clair comme la capture au dessous montre :

	Rate_name	Rate_Lieu	Rate_nb_review	Rate_nb_review_url
0	Moyen	Ile-de-France	740 reviews	https://www.tripadvisor.com/Restaurant_Review-...
1	Moyen	Seine-et-Marne	831 reviews	https://www.tripadvisor.com/Restaurant_Review-...
2	Moyen	Ile-de-France	156 reviews	https://www.tripadvisor.com/Restaurant_Review-...
3	Bas	Ile-de-France	34 reviews	https://www.tripadvisor.com/Restaurant_Review-...
4	Moyen	Ile-de-France	123 reviews	https://www.tripadvisor.com/Restaurant_Review-...
...	...	...	...	...
95	Bas	Loire-Atlantique	104 reviews	https://www.tripadvisor.com/Restaurant_Review-...
96	Bas	Seine-Saint-Denis	26 reviews	https://www.tripadvisor.com/Restaurant_Review-...
97	Moyen	Ile-de-France	12 reviews	https://www.tripadvisor.com/Restaurant_Review-...
98	Mauvais	Occitanie	14 reviews	https://www.tripadvisor.com/Restaurant_Review-...
99	Moyen	Val-de-Marne	48 reviews	https://www.tripadvisor.com/Restaurant_Review-...

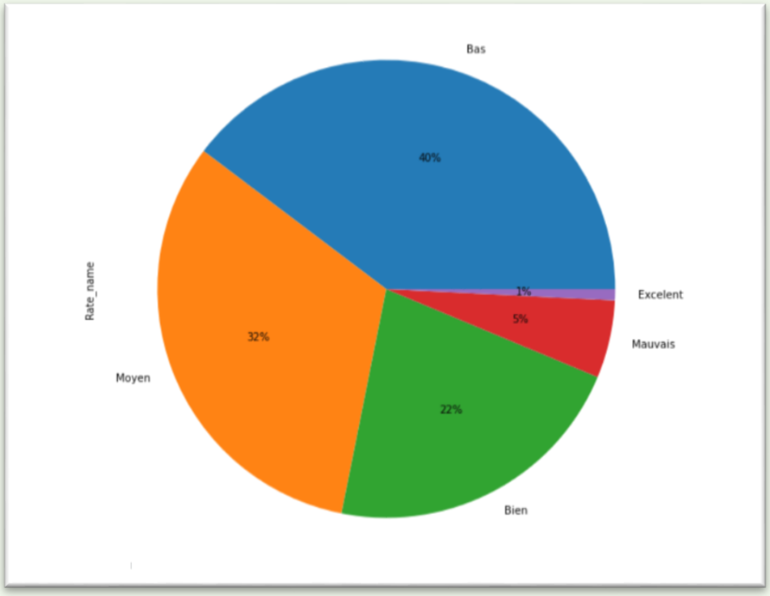
Et finalement on passe à la phase de visualisation .

1) Le tauxd'avisattribués par ville



Ce diagramme circulaire résume la répartition des avis par rapport à la région où se trouve McDonald's envoie marque que la majorité des commentaires sont réservés pour McDonald's Île-de-France 46 % tandis que le reste des régions ont des pourcentages proches du nombre totale d'avis entre 1 % et 5 %

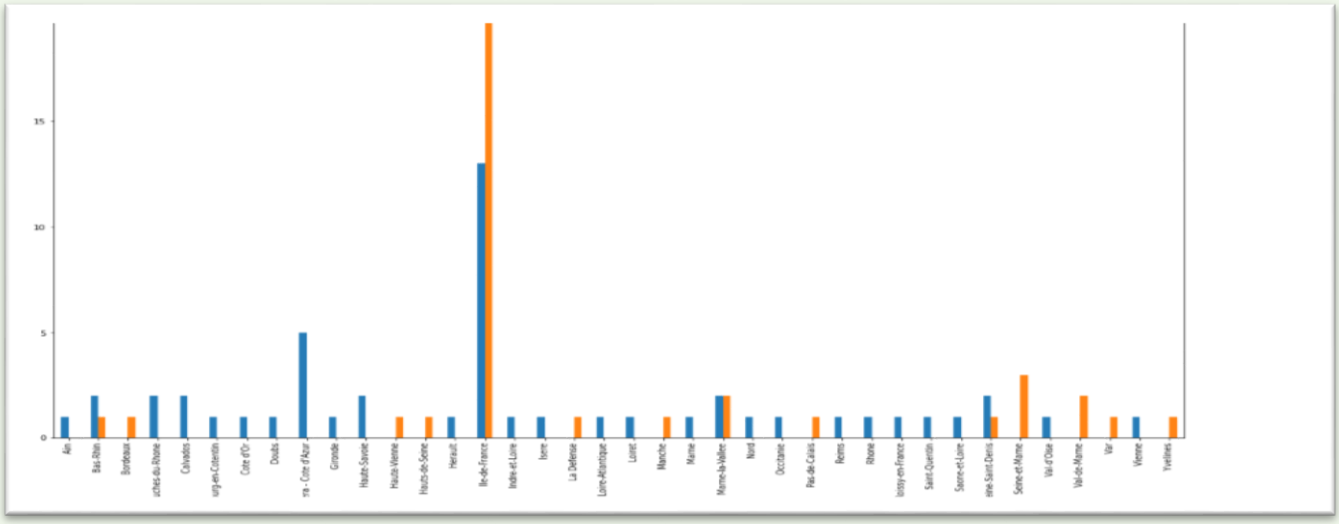
2) Le total des avisattribués



On Remarque que la majorité sont des avis negatifs : bas (40%) ,moyen (32%),mauvais (5%) pour faire un total de 77%. il s'agit d'un pourcentage très élevé pour un restaurant très populaire dans le monde . la E-reputation de Mcdonalds France est très mauvaise ce qui peut nuire à l'image de l'enseigne et cause une crise .

Les avis positifs sont d'ordre 23 % une pourcentage très négligeable par rapport à la dernière pour avoir une influence sur l'opinion public des français .

3) Le total d'avis negatives et positives par ville



D'après ce graphe, on peut conclure que les avis sont généralement positives sauf pour les villes :

- Iles de France : +30 avis positif
- Seine et marne : il n'y a que des avis positifs
- Val de marne : il n'y a que des avis positifs
  - Comme interprétation , on peut juger que e-réputation de macdonalds France dans les villes sitées au dessus est bonne. Mais les avis négatifs restent une majorité pour les autres villes . Mcdonalds doit travailler de plus sur son image en dehors de : Iles de france/ Seine et marne/ Val de marne.

- Trustpilot

De même pour trustpilot nous avonsobtenucedataframe.

```
[ ]  df.head()
```

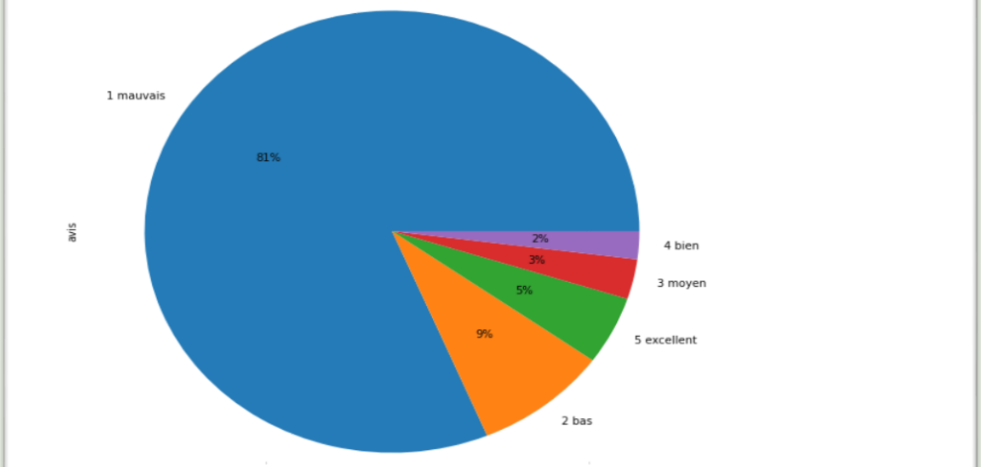
	commentaire	avis	index
0	\n\nconditions d'emploi indécent\n\n\n ...	1 mauvais	0
1	\n\nles mcdonald's en france sont pour la...\n\n...	4 bien	1
2	\n\nresponsable mc do hautain\n\n\n ...	1 mauvais	2
3	\n\ncorona virus\n\n\n corona v...	1 mauvais	3
4	\n\néviter mcdo peipin\n\n\n at...	1 mauvais	4

```
plot_size = plt.rcParams["figure.figsize"]
print(plot_size[0])
print(plot_size[1])

plot_size[0] = 10
plot_size[1] = 10
plt.rcParams["figure.figsize"] = plot_size
```

```
20.0
20.0
```

```
[ ]  df.avis.value_counts().plot(kind='pie', autopct='%1.0f%%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f264bceb9b0>
```



- 81% des avis sont négatifs
- L'opinion publique pense que mcdonald's france n'est pas une bonne enseigne. Un tel opinion peut cause une crise pour la chaine . Il faut que Mcdonalds change de stratégie pour qu'elle s'impose dans le marché français

  - Twitter :

- Pas de note dans le site twitter : on ne peut faire que des analyses sur les commentaires.

➢ **Une interprétation globale sur les notes :**

D'après les analyses faites dans les sites :

L'opinion du peuple français est plus au moins Claire envers cette enseigne. cette dernière montre un mécontentement totale .On peut donc conclure qu'il s'agit d'une e-réputation mauvaise qui va générer une mauvaise e-réputation . ces résultats donnent une chance au concurrent de briller et prendre la place de macdonalds dans le marché français .

3. Opinion classification :

- TripAdvisor :

Classification 1

[108]:
```
from vaderSentiment import vaderSentiment
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

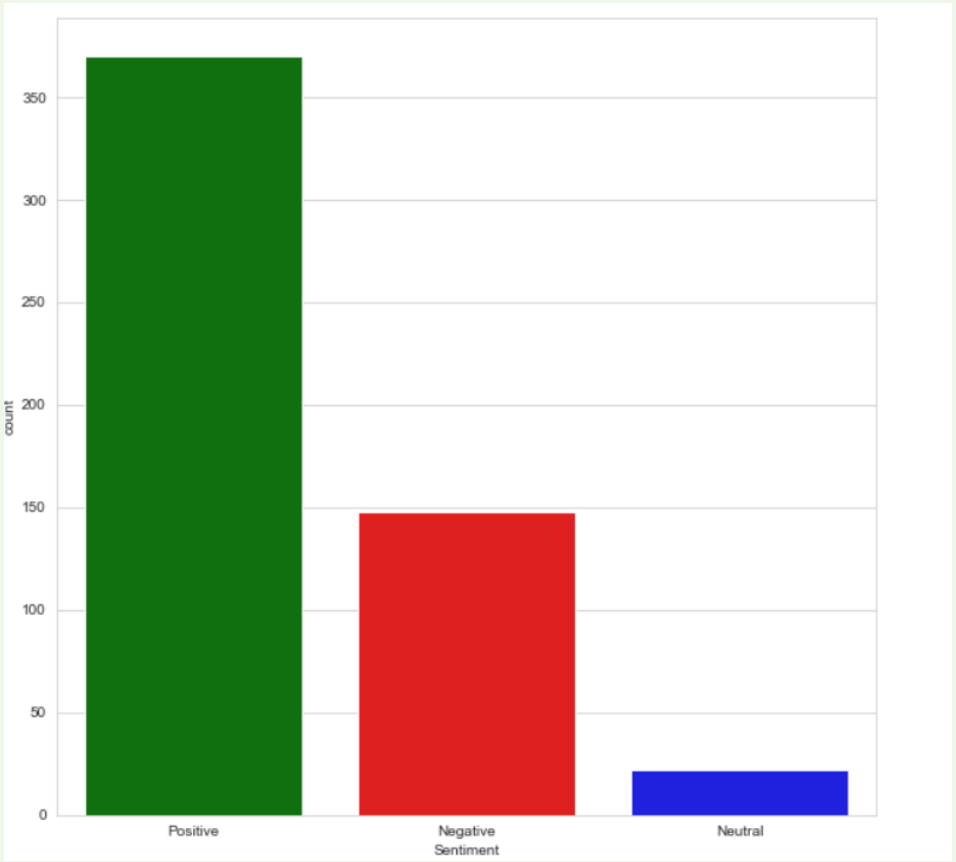
[109]:
```
sent_analyser = SentimentIntensityAnalyzer()
def sentiment(text):
    return (sent_analyser.polarity_scores(text)["compound"])
```

[111]:
```
data["Polarity"] = data["review_body"].apply(sentiment)
```

[112]:
```
def senti(data):
    if data['Polarity'] >= 0.05:
        val = "Positive"
    elif data['Polarity'] <= -0.05:
        val = "Negative"
    else:
        val = "Neutral"
    return val
```

[114]:
```
data['Sentiment'] = data.apply(senti, axis=1)
```

[121]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(10,10))
sns.set_style("whitegrid")
ax = sns.countplot(x="Sentiment", data=data,
palette=dict(Neutral="blue", Positive="Green", Negative="Red"))
```




```
[126]: training=data.loc[201:1154,["review_body","Sentiment"]].values.tolist()

[127]: testing=data.loc[0:200,["review_body","Sentiment"]].values.tolist()

[128]: from textblob import classifiers
       classifier = classifiers.NaiveBayesClassifier(training)
       dt_classifier = classifiers.DecisionTreeClassifier(training)

[129]: print (classifier.accuracy(testing))

       0.7313432835820896

[130]: print (dt_classifier.accuracy(testing))

       0.6417910447761194

[131]: classifier.show_informative_features(3)

       Most Informative Features
              contains(normal) = True           Neutra : Positi =     27.2 : 1.0
             contains(finally) = True           Negati : Positi =     10.9 : 1.0
               contains(worst) = True           Negati : Positi =     10.9 : 1.0

[ ]: classifier.show_informative_features(3)
```

```
Entrée [168]: blob = TextBlob('Excelent hamburger!', classifier=classifier)
              print (blob.classify())

              Positive
```

Classification 2

```
132]: from sklearn.model_selection import train_test_split
      text_counts= cv.fit_transform(data['review_body'])
      X_train, X_test, y_train, y_test = train_test_split(
          text_counts, data['review_body'], test_size=0.3, random_state=1)

133]: from sklearn.naive_bayes import MultinomialNB
      from sklearn import metrics

134]: clf = MultinomialNB()

135]: clf.fit(X_train, y_train)

135]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

136]: predicted= clf.predict(X_test)

137]: print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))

      MultinomialNB Accuracy: 0.3950617283950617

[ ]:
```

- Trustpilot :

###classification

```
[7]: from sklearn.feature_extraction.text import CountVectorizer
     from nltk.tokenize import RegexpTokenizer
     #tokenizer to remove unwanted elements from out data like symbols and numbers
     token = RegexpTokenizer(r'[a-zA-Z0-9]+')
     cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1),tokenizer = token.tokenize)
     text_counts= cv.fit_transform(df['commentaire'])

[8]: from sklearn.model_selection import train_test_split
     X_train, X_test, y_train, y_test = train_test_split(
         text_counts, df['avis'], test_size=0.3, random_state=1)

[9]: from sklearn.naive_bayes import MultinomialNB
     from sklearn import metrics

[11]: clf = MultinomialNB()

[12]: clf.fit(X_train, y_train)

[12]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

[13]: predicted= clf.predict(X_test)

[14]: print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))

      MultinomialNB Accuracy: 0.8058252427184466
```


- Twitter :

```
training=df.loc[201:1154,["text","Sentiment"]].values.tolist()
```

```
testing=df.loc[0:200,["text","Sentiment"]].values.tolist()
```

```
from textblob import classifiers
classifier = classifiers.NaiveBayesClassifier(training)
dt_classifier = classifiers.DecisionTreeClassifier(training)
```

```
print (classifier.accuracy(testing))
```

```
0.8905472636815921
```

```
print (dt_classifier.accuracy(testing))
```

```
0.8905472636815921
```

```
classifier.show_informative_features(3)
```

```
Most Informative Features
 contains('lecafedugeek) = False          Negati : Positi =      1.0 : 1.0
         contains('foi) = False           Negati : Positi =      1.0 : 1.0
    contains('autorout) = False           Negati : Positi =      1.0 : 1.0
```

```
blob = TextBlob('Pas mal les hamburgers!', classifier=classifier)
print (blob.classify())
```

```
Neutral
```

4. Interprétation globale :

Même si les analyses des commentaires de tripadvisor nous à mener à une piste légèrement positive en ce qui concerne la e-réputation de macdonalds France , mais les analyses qui suivent (à travers les analyses des avis etc ..) , ont démontré que l'e-réputation de mcdonalds France est très mauvaise.

Ainsi , cela peut provoquer une crise en premier lieu et sa pourriture au sein du marché français en second lieu . Par conséquence les concurrents ,qui considèrent ca comme étant une opportunité , vont briller .

Il est conseillé que macdonalds revoir ca stratégie de marketing, de vente ainsi son service avec la clientèle .