Projet : Fouille de Données

Thème : Classification et clustering des tweets en Python.

Réaliser par : Mannai Salim

Objectifs :

- Maitriser l'API de twitter pour l'extraction des tweets
- Maitriser la partie NLP (natural language processing) avec NLTK en Python
- Appliquer les principes de nettoyage des données
- Classer les tweets : regrouper ensemble les tweets qui sont similaires. C'est une étape qui peut être considérée comme une étape

Twitter

Twitter est un service de réseautage social et de micro-blogging sur lequel les utilisateurs publient et interagissent les uns avec les autres via des messages appelés «tweets». Il est classé au 6e rang des sites et applications de réseautage social les plus populaires par Dream Grow en avril 2020 avec une moyenne de 330 millions d'utilisateurs actifs par mois.

Spécifications

Imaginons que vous avez un compte Twitter, et que vous lez suivre les tweets (texte très court) sur ce réseau social. Vu le nombre colossal de Tweets, et faute de temps, vous n'avez pas la possibilité de les lire tous. Pour cela, vous avez besoin d'une application qui va jouer le rôle d'assistantet qui va vous effectuer un résumé de toutes ces informations. Une des approches qu'on peut utiliser estde le classer sous former de groupes de sorte à ce qu'on présente à l'utilisateur un seul Tweet de chaque groupe. Pour cela, on doit procéder en trois grandes étapes :

1. Prétraitement des tweets

Dans cette étape, l'objectif est d'éliminer le texte inutile des tweets tels que les #, les noms des utilisateurs, les url, …

2. Traitement des tweets : NLP (Natural LanguageProcessing) On doit procéder à l'analyse du tweet en respectant les différentes étapes du NLP (Natural LanguageProcessing). La bibliothèque à utiliser est NLTK en Python.
3. Classification des tweets Etant donné un ensemble de tweets, l'objectif est de les résumer sous formes de groupes de sorte à ce que les Tweets qui sont dans le même groupe soient similaires. Ainsi, l'utilisateur pourra par la suite lire juste un Tweet de chaque groupe (le Tweet qui est le centroïde de groupes).

Réalisation:

Libraries

Les bibliothèques suivantes seront utilisées tout au long du projet.

In [1]:
```
#pip install tweepy
#!python -m spacy download en_core_web_sm
#!pip install spacy
```

In [2]:
```
import pandas as pd
import spacy
import en_core_web_sm
import tweepy
import numpy as np
import datetime
import csv
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.model_selection import train_test_split
import nltk
from nltk.tokenize import RegexpTokenizer, WhitespaceTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
import string
from string import punctuation
import collections
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import jaccard_score
from sklearn.feature_extraction.text import CountVectorizer
%matplotlib inline
```

Base de données

On va télécharger les Tweets à partir de Twitter en utilisant l'API de twitter. Pour cela, on doit obetenir un compte « Twitter Developer ».

In [3]:
```
auth = tweepy.OAuthHandler('...', '...')
auth.set_access_token('...', '...')

api = tweepy.API(auth)

public_tweets = api.home_timeline()
for tweet in public_tweets:
    print(tweet.text)
```

```
10) ضد التصور الأسطورى للشيطان) https://t.co/FA8vhsAv54 (https://t.co/FA8vhsAv54)
واشنطن – بومبيو يوجه تهنئة للبنان بعيد الاستقلال دون إشارة للرئيس#
.@sandraregol: "Anne Hidalgo est une véritable socialiste à l'ancienne" http
s://t.co/QCeuylWgTI (https://t.co/QCeuylWgTI)
Maître Caty Richard, avocate d’une partie de la famille Fouillot : "Ce qu'on ch
erchait, c'était quelque chose qui n… https://t.co/qRT7rFQbdW (https://t.co/qRT
7rFQbdW)
Les dents et dodo

🎧 Episode 248 : Le palais en allumettes

Tu veux que je te raconte l’histoire du palais en allum… https://t.co/Ne7mGvD9X
G (https://t.co/Ne7mGvD9XG)
خبير أمني: قانون السجون المصري من أرقى القوانين التي تحترم حقوق الانسان بالعالم

https://t.co/eavcJE6Jsm (https://t.co/eavcJE6Jsm) https://t.co/B4aqScog2u (http
s://t.co/B4aqScog2u)
On taking office, Joe Biden should announce that he will uphold the deal that w
as struck in February with the Talib… https://t.co/KOaYdVlpeT (https://t.co/KOa
YdVlpeT)
.@sandraregol répond à Anne Hidalgo: "Je suis très étonnée que quelqu'un qui se
dit si républicaine n'agisse pas po… https://t.co/D28R0ZLqGI (https://t.co/D28R
0ZLqGI)
واشنطن – وزير الخارجية: الولايات المتحدة ملتزمة بالوقوف إلى جانب الشعب اللبناني في هذه الأيام العصيبة#
«وزير التعليم السعودي: اقتصاديات التعليم ستتغير بعد أزمة «كورونا https://t.co/hvl1wRRDaY (https://
t.co/hvl1wRRDaY)
بالفيديو... رجل ينقذ كلبه الصغير من فك تمساح https://t.co/W3Aox5MLxX (https://t.co/W3Aox5ML
xX)
No normal UK Christmas but families may be able to get together - Sunak http
s://t.co/zNxJaIBpvs (https://t.co/zNxJaIBpvs) https://t.co/4blgHgulij (https://
t.co/4blgHgulij)
اكتشاف مقبرة إسلامية ضخمة في منطقة دفن تعود إلى القرن الثامن بإسبانيا 👇 https://t.co/IlJKNQLy1l (ht
tps://t.co/IlJKNQLy1l)
«بخاخ «وقائي» للأنف يحمي من «كوفيد - 19 https://t.co/eUe3EYr94e (https://t.co/eUe3EYr94
e)
شقيق #محمد_صلاح وزوجته فى جلسة التصوير الرسمية لحفل زفافه
https://t.co/ZylFfIH1D4 (https://t.co/ZylFfIH1D4) https://t.co/wPwi8sqJrM (http
s://t.co/wPwi8sqJrM)
عاجل| البحوث الفلكية: زلزال بقوة 3.5 درجة يضرب منطقة الدلتا

https://t.co/oUoqXqNccb (https://t.co/oUoqXqNccb) https://t.co/Sh6ZL7aJDn (http
s://t.co/Sh6ZL7aJDn)
RT @ajmurgent: عاجل | وزارة الخارجية البريطانية: نشعر بقلق بالغ إزاء اعتقال السلطات المصرية أعضاء من المبا
درة المصرية للحقوق الشخصية
تقرير | البعثة وزعت وثيقة لتحديد آليات اختيار أعضاء #الرئاسي و #الوزراء#

https://t.co/D8HGzkDW9W (https://t.co/D8HGzkDW9W)
```

```
Maître Caty Richard, avocate d'une partie de la famille Fouillot : "Je retiens
 un procès duquel tout le monde resso… https://t.co/96Sf3q4AIz (https://t.co/96
Sf3q4AIz)
```

In [4]:
```
user = api.get_user('twitter')
```

In [5]:
```
print(user.screen_name)
print(user.followers_count)
for friend in user.friends():
   print(friend.screen_name)
```

```
Twitter
58566710
LACity
CityofSeattle
chicago
MiamiBeachNews
CityofMiami
NJGov
inAsburyParkNJ
JerseyCity
OscarTheGrouch
```

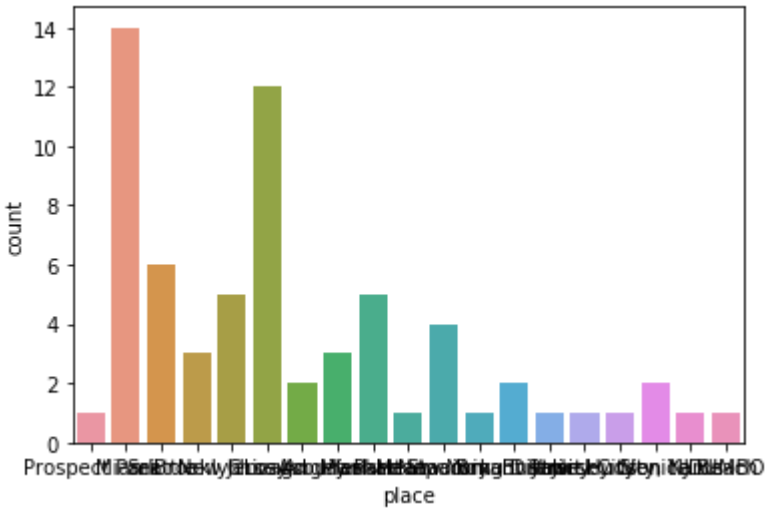
- Maintenant on va sauvgarder les tweets dansun fichier csv

In [6]:
```
filename = 'Datasets/twitter_data_analysis'+(datetime.datetime.now().strftime("%Y
with open (filename, 'w', newline='',encoding="utf-8") as csvFile:
    csvWriter = csv.writer(csvFile)
    csvWriter.writerow(['date', 'TweetId','Tweet','created_at','geo','place','coo
    #using tweepy Cursor
    for tweet in tweepy.Cursor(api.user_timeline , id="Twitter").items(11000):
        csvWriter.writerow([datetime.datetime.now().strftime("%Y-%m-%d  %H:%M"),
```

In [7]:
```
tweet_df= pd.read_csv('Datasets/twitter_data_analysis2020-11-22-13.csv')
# Affichage de la taille du dataset (n_lignes and n_colonnes)
print('Dataset size:',tweet_df.shape)
print('Columns are:',tweet_df.columns)
tweet_df.info()
sns.countplot(x = 'place', data = tweet_df)
```

```
Dataset size: (3218, 8)
Columns are: Index(['date', 'TweetId', 'Tweet', 'created_at', 'geo', 'place',
'coordinates',
       'location'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3218 entries, 0 to 3217
Data columns (total 8 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   date         3218 non-null   object
 1   TweetId      3218 non-null   int64
 2   Tweet        3218 non-null   object
 3   created_at   3218 non-null   object
 4   geo          0 non-null      float64
 5   place        66 non-null     object
 6   coordinates  0 non-null      float64
 7   location     3218 non-null   object
dtypes: float64(2), int64(1), object(5)
memory usage: 201.2+ KB
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b232c5c88>
```



In [8]:
```
df  = pd.DataFrame(tweet_df[['TweetId', 'Tweet']])
```

Prétraitement

Les tweets contiennent des objets inutiles tels que des hashtags, des mentions, des liens et des signes de ponctuation qui peuvent affecter les performances d'un algorithme et doivent donc être supprimés. Tous les textes sont convertis en minuscules pour éviter que les algorithmes

n'interprètent les mêmes mots avec des cas différents comme différents.

In [9]:
```
string.punctuation
```

Out[9]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

- Supprimez les hashtags, les mentions et les caractères indésirables.

In [10]:
```
def remove_punct(text):
    text  = "".join([char for char in text if char not in string.punctuation])
    text = re.sub('[0-9]+', '', text)
    return text

df['Tweet_punct'] = df['Tweet'].apply(lambda x: remove_punct(x))
df.head(10)
```

Out[10]:

	TweetId	Tweet	Tweet_punct
0	1329561340596391936	RT @shesooosaddity: if you had a twitter befor...	RT shesooosaddity if you had a twitter before ...
1	1329217044391342082	@CloudNaii 40404	CloudNaii
2	1329216472711827458	@issahairplug drink water replaced good morning	issahairplug drink water replaced good morning
3	1329107688916135936	@Ne_ThatGuy we're taking oomf to the Fleets	NeThatGuy were taking oomf to the Fleets
4	1329104797727940612	@_JusJust_ remember "I dedicate my 500th Tweet...	JusJust remember I dedicate my th Tweet to
5	1329104643062902788	@ambr_ncole they're tourists	ambrncole theyre tourists
6	1329101613940797441	@PhallonXOXO proof you're doing it right 😊	PhallonXOXO proof youre doing it right 😊
7	1328838299419627525	some of you hating...\n\nbut we see you Fleeti...	some of you hating\n\nbut we see you Fleeting 😊
8	1328684389388185600	That thing you didn’t Tweet but wanted to but ...	That thing you didn’t Tweet but wanted to but ...
9	1328426768009736192	@quakerraina this is art	quakerraina this is art

- Tokenisation, lemmatisation et suppression des mots vides

In [11]:
```
def tokenization(text):
    text = re.split(' ', text)
    return text

df['Tweet_tokenized'] = df['Tweet_punct'].apply(lambda x: tokenization(x.lower()
df.head()
```

Out[11]:

	TweetId	Tweet	Tweet_punct	Tweet_tokenized
0	1329561340596391936	RT @shesooosaddity: if you had a twitter befor...	RT shesooosaddity if you had a twitter before ...	[rt, shesooosaddity, if, you, had, a, twitter,...
1	1329217044391342082	@CloudNaii 40404	CloudNaii	[cloudnaii,]
2	1329216472711827458	@issahairplug drink water replaced good morning	issahairplug drink water replaced good morning	[issahairplug, drink, water, replaced, good, m...
3	1329107688916135936	@Ne_ThatGuy we're taking oomf to the Fleets	NeThatGuy were taking oomf to the Fleets	[nethatguy, were, taking, oomf, to, the, fleets]
4	1329104797727940612	@_JusJust_ remember "I dedicate my 500th Tweet...	JusJust remember I dedicate my th Tweet to	[jusjust, remember, i, dedicate, my, th, tweet...

In [12]:
```
stopword = nltk.corpus.stopwords.words('english')
```

In [13]:
```
stopword.extend(['old', 'new', 'age', 'lot', 'bag', 'top', 'cat', 'bat', 'sap', '
             'mob', 'map', 'car', 'fat', 'sea', 'saw', 'raw', 'rob', 'win', 'can'
             'use', 'pea', 'pit','pot', 'pat', 'ear', 'eye', 'kit', 'pot', 'pen',
              'lid', 'log', 'pr', 'pd', 'cop', 'nyc', 'ny', 'la', 'toy', 'war', '
              'pay', 'pet', 'fan', 'fun', 'usd', 'rio',':)', ';)', '(:', '(;', '
             'thank', 'https', 'since', 'wanna', 'gonna', 'aint', 'http', 'unto',
             'dont', 'done', 'cant', 'werent', 'https', 'u', 'isnt', 'go', 'theyr
            'weve', 'theyve','googleele' , 'goog', 'lyin', 'lie', 'googles', 'goo
             'msft','microsoft', 'google', 'goog', 'googl','goog','https'])
```

In [14]:
```
def remove_stopwords(text):
    text = [word for word in text if word not in stopword]
    return text

df['Tweet_nonstop'] = df['Tweet_tokenized'].apply(lambda x: remove_stopwords(x))
df.head(10)
```

Out[14]:

	TweetId	Tweet	Tweet_punct	Tweet_tokenized	Tweet_nonstop
0	1329561340596391936	RT @shesooosaddity: if you had a twitter befor...	RT shesooosaddity if you had a twitter before ...	[rt, shesooosaddity, if, you, had, a, twitter,...	[rt, shesooosaddity, twitter, , rt]
1	1329217044391342082	@CloudNaii 40404	CloudNaii	[cloudnaii,]	[cloudnaii,]
2	1329216472711827458	@issahairplug drink water replaced good morning	issahairplug drink water replaced good morning	[issahairplug, drink, water, replaced, good, m...	[issahairplug, drink, water, replaced, good, m...
3	1329107688916135936	@Ne_ThatGuy we're taking oomf to the Fleets	NeThatGuy were taking oomf to the Fleets	[nethatguy, were, taking, oomf, to, the, fleets]	[nethatguy, taking, oomf, fleets]
4	1329104797727940612	@_JusJust_ remember "I dedicate my 500th Tweet...	JusJust remember I dedicate my th Tweet to	[jusjust, remember, i, dedicate, my, th, tweet...	[jusjust, remember, dedicate, th, tweet]
5	1329104643062902788	@ambr_ncole they're tourists	ambrncole theyre tourists	[ambrncole, theyre, tourists]	[ambrncole, tourists]
6	1329101613940797441	@PhallonXOXO proof you're doing it right 😉	PhallonXOXO proof youre doing it right 😉	[phallonxoxo, proof, youre, doing, it, right, 😉]	[phallonxoxo, proof, right, 😉]
7	1328838299419627525	some of you hating...\n\nbut we see you Fleeti...	some of you hating\n\nbut we see you Fleeting 😉	[some, of, you, hating\n\nbut, we, see, you, f...	[hating\n\nbut, see, fleeting, 😉]
8	1328684389388185600	That thing you didn’t Tweet but wanted to but ...	That thing you didn’t Tweet but wanted to but ...	[that, thing, you, didn’t, tweet, but, wanted,...	[thing, didn’t, tweet, wanted, didn’t, got, cl...
9	1328426768009736192	@quakerraina this is art	quakerraina this is art	[quakerraina, this, is, art]	[quakerraina, art]

On vas utiliser la bibliothèque NLTK pour effectuer une analyse de chaque tweet et le transformer en un ensemble de mots en suivant les différentes étapes de base du processus NLP (Natural Language Processing)

- Stemming et Lammitization

In [15]:
```
ps = nltk.PorterStemmer()
def stemming(text):
    text = [ps.stem(word) for word in text]
    return text

df['Tweet_stemmed'] = df['Tweet_nonstop'].apply(lambda x: stemming(x))
df.head()
```

Out[15]:

	TweetId	Tweet	Tweet_punct	Tweet_tokenized	Tweet_nonstop	Tweet_
0	1329561340596391936	RT @shesooosaddity: if you had a twitter befor...	RT shesooosaddity if you had a twitter before ...	[rt, shesooosaddity, if, you, had, a, twitter,...	[rt, shesooosaddity, twitter, , rt]	[rt, shes t
1	1329217044391342082	@CloudNaii 40404	CloudNaii	[cloudnaii,]	[cloudnaii,]	[c
2	1329216472711827458	@issahairplug drink water replaced good morning	issahairplug drink water replaced good morning	[issahairplug, drink, water, replaced, good, m...	[issahairplug, drink, water, replaced, good, m...	[iss dri rep
3	1329107688916135936	@Ne_ThatGuy we're taking oomf to the Fleets	NeThatGuy were taking oomf to the Fleets	[nethatguy, were, taking, oomf, to, the, fleets]	[nethatguy, taking, oomf, fleets]	[nethat o
4	1329104797727940612	@_JusJust_ remember "I dedicate my 500th Tweet...	JusJust remember I dedicate my th Tweet to	[jusjust, remember, i, dedicate, my, th, tweet...	[jusjust, remember, dedicate, th, tweet]	[jusjust, dedic,

In [16]:
```
wn = nltk.WordNetLemmatizer()

def lemmatizer(text):
    text = [wn.lemmatize(word) for word in text]
    return text

df['Tweet_lemmatized'] = df['Tweet_nonstop'].apply(lambda x: lemmatizer(x))
df.head()
```

Out[16]:

	TweetId	Tweet	Tweet_punct	Tweet_tokenized	Tweet_nonstop	Tweet_
0	1329561340596391936	RT @shesooosaddity: if you had a twitter befor...	RT shesooosaddity if you had a twitter before ...	[rt, shesooosaddity, if, you, had, a, twitter,...	[rt, shesooosaddity, twitter, , rt]	[rt, shes t
1	1329217044391342082	@CloudNaii 40404	CloudNaii	[cloudnaii,]	[cloudnaii,]	[c
2	1329216472711827458	@issahairplug drink water replaced good morning	issahairplug drink water replaced good morning	[issahairplug, drink, water, replaced, good, m...	[issahairplug, drink, water, replaced, good, m...	[iss dri rep
3	1329107688916135936	@Ne_ThatGuy we're taking oomf to the Fleets	NeThatGuy were taking oomf to the Fleets	[nethatguy, were, taking, oomf, to, the, fleets]	[nethatguy, taking, oomf, fleets]	[nethat o
4	1329104797727940612	@_JusJust_ remember "I dedicate my 500th Tweet...	JusJust remember I dedicate my th Tweet to	[jusjust, remember, i, dedicate, my, th, tweet...	[jusjust, remember, dedicate, th, tweet]	[jusjust, dedic,

L'ensemble de données après le prétraitement:

In [17]:
```
df.Tweet_lemmatized
```

Out[17]:
```
0                    [rt, shesooosaddity, twitter, , rt]
1                                            [cloudnaii, ]
2       [issahairplug, drink, water, replaced, good, m...
3                       [nethatguy, taking, oomf, fleet]
4                [jusjust, remember, dedicate, th, tweet]
                                   ...
3213       [themegaboi, keeping, brain, thinking, around]
3214    [guillaumetc, hamillhimself, chrisevans, combo...
3215                          [ksjize, hi, dograte, got]
3216               [insomniacookies, cc, mecookiemonster]
3217    [mnoir, amp, guaranteed, good, morning, good, ...
Name: Tweet_lemmatized, Length: 3218, dtype: object
```

On va mettre les tweets propres dans un nouveau fichier csv

In [18]:
```
df.Tweet_lemmatized.to_csv('Datasets/new_tweets_clean.csv',index = False)
```

In [19]:
```
# remove the hashtags, mentions and unwanted characters.
new_tweet_df= pd.read_csv('Datasets/new_tweets_clean.csv')
print('Dataset size:',new_tweet_df.shape)
print('Columns are:',new_tweet_df.columns)
new_tweet_df.info()
new_tweet_df.head()
```

```
Dataset size: (3218, 1)
Columns are: Index(['Tweet_lemmatized'], dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3218 entries, 0 to 3217
Data columns (total 1 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   Tweet_lemmatized  3218 non-null   object
dtypes: object(1)
memory usage: 25.3+ KB
```

Out[19]:

	Tweet_lemmatized
0	['rt', 'shesooosaddity', 'twitter', '', 'rt']
1	['cloudnaii', '']
2	['issahairplug', 'drink', 'water', 'replaced',...
3	['nethatguy', 'taking', 'oomf', 'fleet']
4	['jusjust', 'remember', 'dedicate', 'th', 'twe...

Vectorisation

- Les données nettoyées en une seule ligne en passant new_tweet_df dans le CountVectorizer

In [20]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X=cv.fit_transform(new_tweet_df.Tweet_lemmatized)
print(X)
```

```
  (0, 4699)     2
  (0, 4921)     1
  (0, 5754)     1
  (1, 932)      1
  (2, 2792)     1
  (2, 1380)     1
  (2, 5962)     1
  (2, 4587)     1
  (2, 1923)     1
  (2, 3721)     1
  (3, 3860)     1
  (3, 5364)     1
  (3, 4125)     1
  (3, 1743)     1
  (4, 3028)     1
  (4, 4575)     1
  (4, 1221)     1
  (4, 5432)     1
  (4, 5740)     1
  (5, 181)      1
  (5, 5658)     1
  (6, 4267)     1
  (6, 4410)     1
  (6, 4635)     1
  (7, 2062)     1
  :     :
  (3213, 285)   1
  (3213, 3079)  1
  (3213, 5486)  1
  (3214, 5991)  1
  (3214, 5766)  1
  (3214, 6015)  1
  (3214, 148)   1
  (3214, 1997)  1
  (3214, 2021)  1
  (3214, 876)   1
  (3214, 974)   1
  (3215, 1935)  1
  (3215, 2123)  1
  (3215, 1336)  1
  (3215, 3163)  1
  (3216, 3557)  1
  (3216, 789)   1
  (3216, 2749)  1
  (3217, 1923)  2
  (3217, 3721)  1
  (3217, 5740)  1
  (3217, 3955)  1
  (3217, 189)   1
  (3217, 3682)  1
  (3217, 1991)  1
```

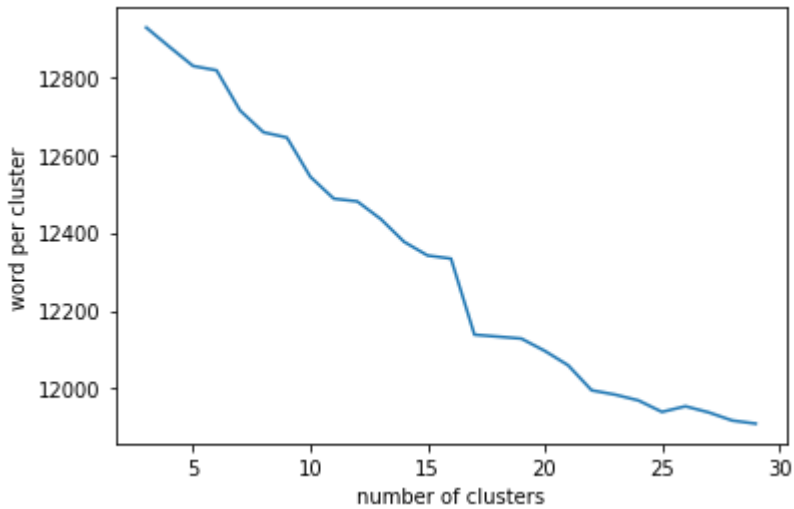
Classification des tweets

- Cette approche utilise la technique de création d'un ensemble de mots qui peuvent être classés en toute confiance comme appartenant à une catégorie particulière .
- On va Utiliser l'algorithme K-Means pour classer les Tweets en 30 classes.

In [21]:
```
from sklearn.cluster import KMeans
wcss=[]
for i in range(3,30):
    Kmeans=KMeans(n_clusters=i,init='k-means++',max_iter=300,n_init=10,random_sta
    Kmeans.fit(X)
    wcss.append(Kmeans.inertia_)
```

```
Initialization complete
Iteration  0, inertia 19487.000
Iteration  1, inertia 13009.733
Iteration  2, inertia 13009.635
Converged at iteration 2: center shift 0.000000e+00 within tolerance 6.594548
e-08
Initialization complete
Iteration  0, inertia 16441.000
Iteration  1, inertia 13091.284
Iteration  2, inertia 13090.433
Converged at iteration 2: center shift 0.000000e+00 within tolerance 6.594548
e-08
Initialization complete
Iteration  0, inertia 22485.000
Iteration  1, inertia 12982.834
Iteration  2, inertia 12981.419
Converged at iteration 2: center shift 0.000000e+00 within tolerance 6.594548
e-08
Initialization complete
```

In [22]:
```
import matplotlib.pyplot as plt
plt.plot(range(3,30),wcss)
plt.xlabel('number of clusters')
plt.ylabel('word per cluster')
plt.show()
```



In [23]:
```
true_k=30
Kmeans=KMeans(n_clusters=true_k,init='k-means++',n_init=1)
Kmeans.fit(X)
```

Out[23]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=30, n_init=1, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [24]:
```
print("Top terms per cluster:")
order_centroids = Kmeans.cluster_centers_.argsort()[:, ::-1]
terms = cv.get_feature_names()
for i in range(true_k):
    print("Cluster %d:" % i)
    print("-----------------------")
    for ind in order_centroids[i, :10]:
        print(' %s' % terms[ind])
    print()
print("\n")
```

```
Top terms per cluster:
Cluster 0:
-----------------------
 right
 person
 anyways
 andrearussett
 luckyatuanya
 see
 hazy
 havishaf
 haw
 hawyeehoran

Cluster 1:
-----------------------
 tweet
 like
 time
```

- pour chaque cluster on va afficher un seul tweet
- on a choisi d'afficher la tweet qui a un plus grand score

In [25]:
```
i=0
j=0
while i<28:
    while True:
        Y=cv.transform([new_tweet_df.Tweet_lemmatized[j]])
        prediction=Kmeans.predict(Y)
        if i == prediction:
            print("Tweet of cluster "+str(prediction)+" : "+df.Tweet[i])
            print ("----------------------------------------------")
            print("\n")
            j=0
            break
        j+=1
    i+=1
```

```
Tweet of cluster [0] : RT @shesooosaddity: if you had a twitter before 2020 r
t this
----------------------------------------------


Tweet of cluster [1] : @CloudNaii 40404
----------------------------------------------


Tweet of cluster [2] : @issahairplug drink water replaced good morning
----------------------------------------------


Tweet of cluster [3] : @Ne_ThatGuy we're taking oomf to the Fleets
----------------------------------------------


Tweet of cluster [4] : @_JusJust_ remember "I dedicate my 500th Tweet to:___
_"
```

Tweets par catégorie

Nous souhaitons créer une dataframe contenant le nombre total de tweets par catégorie. Une base de données 4D avec la colonne d'index remplie d'utilisateurs et 3 autres colonnes contenant le nombre total de tweets de l'utilisateur dans les classes sociales, culturelles, sanitaires et économiques. Cela peut être réalisé d'abord en créant un bloc de données contenant les scores Jaccard pour chaque tweet pour chaque catégorie, puis en attribuant un tweet à une catégorie en fonction du score le plus élevé et enfin en regroupant les tweets par nom d'utilisateur et somme des tweets.

Ensembles de mots

Le bloc ci-dessous représente des mots liés à l'économie,social,culture et senté.

In [26]:
```
economy_related_words = "agriculture infrastructure capitalism trading service se
social_related_words = " emotion excuse shield creative persistence  enthusiastic
culture_related_words = "arts humanities philosophy literature music painting bel
health_related_words =  "asthma band aid bandage be allergic to be constipated be
```

In [27]:
```
nlp = en_core_web_sm.load()
tokenizer = RegexpTokenizer(r'\w+')
lemmatizer = WordNetLemmatizer()
stop = set(nltk.corpus.stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
w_tokenizer = WhitespaceTokenizer()
def furnished(text):
    final_text = []
    for i in w_tokenizer.tokenize(text):
        if i.lower() not in stop:
            word1 = lemmatizer.lemmatize(i)
            final_text.append(word1.lower())
    return " ".join(final_text)
df.Tweet = df.Tweet.apply(furnished)
```

Tout comme les tweets, ils doivent subir un pré-traitement. La fonction fournie utilisée sur les tweets est appliquée sur les sets.

In [28]:
```
economy = furnished(economy_related_words)
social = furnished(social_related_words)
culture = furnished(culture_related_words)
health = furnished(health_related_words)
```

Les doublons sont également supprimés:

In [29]:
```
string1 = economy
words = string1.split()
economy = " ".join(sorted(set(words), key=words.index))
economy
string1 = social
words = string1.split()
social = " ".join(sorted(set(words), key=words.index))
social
string1 = health
words = string1.split()
health = " ".join(sorted(set(words), key=words.index))
health
string1 = culture
words = string1.split()
culture = " ".join(sorted(set(words), key=words.index))
culture
```

Out[29]:
```
'art humanity philosophy literature music painting belief ethos intellectual ac
hievement principle activity visual fine art, music, lifestyle custom tradition
habit background civilisationuk civilizationus heritage more society value way
life convention development ethnicity ethnology folklore folkways grounding hum
anism idea knowledge science community nation race people origin ancestry ethni
c group lineage state population extraction pedigree clan tribe living national
ity identity descent style parentage colorus cultural colouruk attainment polit
y social order world heredity root racial type strain human mankind humankind r
ubric prescription rule past history ethnos situation condition naturalisationu
k allegiance political home confederation body politic country affiliation resi
dence native land enfranchisement minority naturalizationus national status beh
aviouruk position regime conduct routine behaviorus populace fate lot existence
station citizenry doctrine essence circumstance manner personage business kind
kin progeny environment play daily acting mode everyday region realm standard s
et empire commonwealth republic federation sovereignty organizationus instituti
on citizen entity public union kingdom organisationuk fatherland motherland sov
ranty homeland resident inhabitant democracy territory power superpower domain
micronation sovereign dominion principality monarchy nation-state re publica co
mmonality general collective klatch fold klatsch denizen burgher'
```

In [30]:
```
def jaccard_similarity(query, document):
    intersection = set(query).intersection(set(document))
    union = set(query).union(set(document))
    return len(intersection)/len(union)
def get_scores(group,tweets):
    scores = []
    for tweet in tweets:
        s = jaccard_similarity(group, tweet)
        scores.append(s)
    return scores
e_scores = get_scores(economy, df.Tweet.to_list())
s_scores = get_scores(social, df.Tweet.to_list())
c_scores = get_scores(culture, df.Tweet.to_list())
h_scores = get_scores(health, df.Tweet.to_list())
```

In [31]:
```
# create a jaccard scored df.
data  = {'names':df.TweetId.to_list(),       'economic_score':e_scores,
         'social_score': s_scores, 'culture_score':c_scores, 'health_scores':h_sc
scores_df = pd.DataFrame(data)
#assign classes based on highest score
def get_classes(l1, l2, l3, l4):
    econ = []
    socio = []
    cul = []
    heal = []
    for i, j, k, l in zip(l1, l2, l3, l4):
        m = max(i, j, k, l)
        if m == i:
            econ.append(1)
        else:
            econ.append(0)
        if m == j:
            socio.append(1)
        else:
            socio.append(0)
        if m == k:
            cul.append(1)
        else:
            cul.append(0)
        if m == l:
            heal.append(1)
        else:
            heal.append(0)

    return econ, socio, cul, heal
l1 = scores_df.economic_score.to_list()
l2 = scores_df.social_score.to_list()
l3 = scores_df.culture_score.to_list()
l4 = scores_df.health_scores.to_list()
econ, socio, cul, heal = get_classes(l1, l2, l3, l4)
data = {'name': scores_df.names.to_list(), 'economic':econ, 'social':socio, 'cult
class_df = pd.DataFrame(data)
#grouping the tweets by username
new_groups_df = class_df.groupby(['name']).sum()
#add a new totals column
new_groups_df['total'] = new_groups_df['health'] + new_groups_df['culture'] + new
#add a new totals row
new_groups_df.loc["Total"] = new_groups_df.sum()
```

In [32]:
```
scores_df
```

Out[32]:

	names	economic_score	social_score	culture_score	health_scores
0	1329561340596391936	0.375000	0.392857	0.387097	0.387097
1	1329217044391342082	0.290323	0.346154	0.300000	0.300000
2	1329216472711827458	0.655172	0.653846	0.678571	0.678571
3	1329107688916135936	0.580645	0.571429	0.600000	0.600000
4	1329104797727940612	0.470588	0.451613	0.441176	0.484848
...	...	...	...	...	...
3213	1147225147230904321	0.500000	0.535714	0.516129	0.516129
3214	1147223872326029314	0.612903	0.666667	0.689655	0.633333
3215	1147222590580305921	0.533333	0.464286	0.500000	0.551724
3216	1147222445381865472	0.400000	0.423077	0.413793	0.413793
3217	1147221416779157504	0.454545	0.482759	0.468750	0.468750

3218 rows × 5 columns

In [33]:
```
new_groups_df
```

Out[33]:

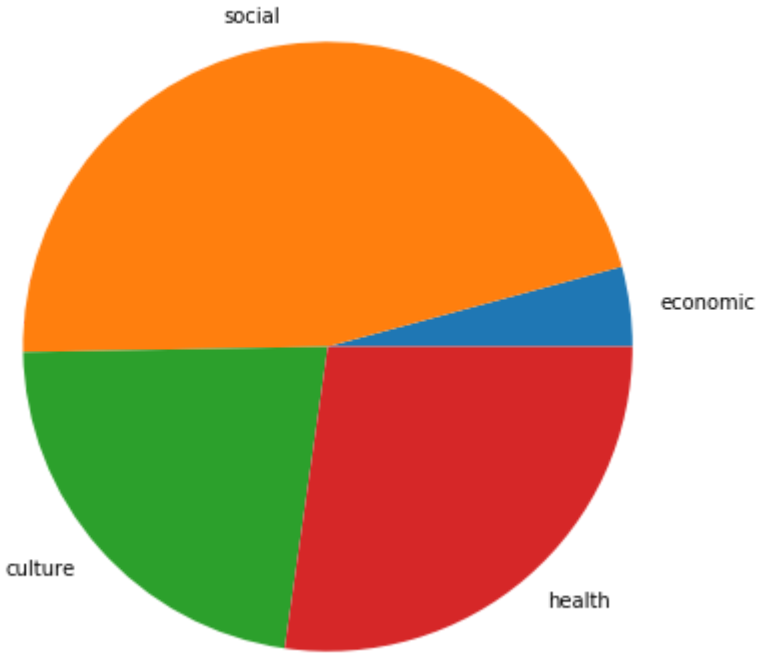
	economic	social	culture	health	total
name					
1147221416779157504	0	1	0	0	1
1147222445381865472	0	1	0	0	1
1147222590580305921	0	0	0	1	1
1147223872326029314	0	0	1	0	1
1147225147230904321	0	1	0	0	1
...	...	...	...	...	...
1329107688916135936	0	0	1	1	2
1329216472711827458	0	0	1	1	2
1329217044391342082	0	1	0	0	1
1329561340596391936	0	1	0	0	1
Total	166	1808	881	1070	3925

3219 rows × 5 columns

Vous trouverez ci-dessous un graphique à secteurs pour montrer les volumes de tweets dans les différentes catégories:

In [34]:
```
fig = plt.figure(figsize =(10, 7))
a = new_groups_df.drop(['total'], axis = 1)
plt.pie(a.loc['Total'], labels = a.columns)
plt.title('A pie chart showing the volumes of tweets under different categories.'
plt.show()
```

A pie chart showing the volumes of tweets under different categories.



le plus grand pourcentage pour le secteur social puisque j'ai mis des suivres pour plusieur page de cinema et music ,ainsi que la santé un un pourcentage Cela pourrait être le résultat de la pandémie actuelle dont tout le monde parle. Les données peuvent être utilisées pour de nombreuses analyses et de belles visualisations, mais l'objectif de l'article est l'analyse de cluster.

In [ ]: