Webscrapping & Data Processing

Jérémy CLAUSSE - Victor CAZAUX

La problématique :

Le processus :

Design Process: The Basics

1 - Scraping

Collecter des données via Twitter/Article de presse

2 - Traitement des données

Nettoyer les données.
Extraire les sentiments.
Sortir un fichier geojson.

3 - Add BDD

Injecter dans elastic search nos données transformées.

4 - Add bucket

Injecter dans un bucket en ligne nos données transformées.

5 - Mapping des résultats

Visualiser nos données sous forme de carte interactive grâce à Kibana.

1 - Scraping

Quoi ?

- Des tweets (pas de RT),
- entiers (280 chars),
- en français,
- qui ont un keyword = “Ville”,
- postés dans un rayon suivant les coordonnées de la ville désirée.

Comment ?

- Grâce à Tweepy (librairie Python gratuite)
- l’API de twitter
- un compte twitter Developer (pour pouvoir utiliser l’API de twitter)

1 - Scraping

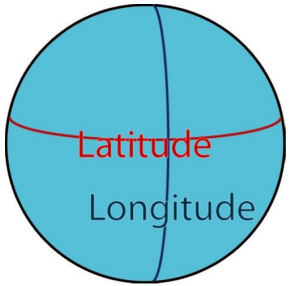

```
page = requests.get("https://photo.capital.fr/les-grandes-villes-francaises-ou-vous-avez-le-plus-de-chance-d-avoir-un-emploi-stab
```

	Villes
0	Rennes
1	Angers
2	Grenoble
3	Strasbourg
4	Montpellier
5	Lille

1 - Scraping

“Ville”

Nominatim



Paris	('48.8588897', '2.3200410217200766')
Marseille	('43.2961743', '5.3699525')
Lyon	('45.7578137', '4.8320114')
Toulouse	('43.6044622', '1.4442469')
Nice	('43.7009358', '7.2683912')

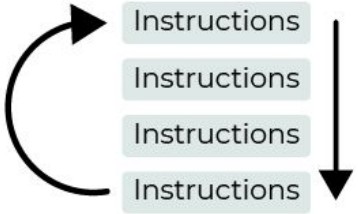
```
url = 'https://nominatim.openstreetmap.org/search/' + urllib.parse.quote(entité) +'?format=json'
```

1 - Scraping

Connexion à l'API twitter :

```
consumer_key = "4YTu
consumer_secret = "U
access_token = "14480
access_token_secret =

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth,wait_on_rate_limit=True)
```

On boucle sur les 50 première villes :



jusqu'à avoir 50 tweets/ville

```
tweets = tweepy.Cursor(api.search_tweets, q=f'{ville} -filter:retweets', geocode=f"{lat},{long},10km",
                       tweet_mode='extended',lang='fr').items(count)
tweets_list+=[[tweet.created_at,tweet.id,ville_list[i][0],tweet.full_text] for tweet in tweets]
```

Contrainte de l'api twitter accès gratuit :

```
92.0 %

Rate limit reached. Sleeping for: 845

94.0 %
96.0 %
```

1 - Scraping

	Date	Tweet ID	Ville	Tweet
0	2023-01-09 17:32:40	1612502631351738368	Paris	Kennedy - Flashback\n Un Temps D'avance •20...
1	2023-01-09 17:32:35	1612502609360990210	Paris	Passage du Grand Cerf, Marais district, Paris ...
2	2023-01-09 17:32:28	1612502577836396582	Paris	Retrouvez la @StarAcademyTF1 en showcase et sé...
3	2023-01-09 17:32:24	1612502564980854802	Paris	@tovwara @DamienRieu @univ_paris_cite Oui et a...
4	2023-01-09 17:31:45	1612502400610275342	Paris	La cour d'appel de Paris a condamné Patrick et

2 - Traitement des données


```
temp = tweet.lower()
temp = re.sub("@[A-Za-z0-9_]+","", temp)
temp = re.sub("#[A-Za-z0-9_]+","", temp)
temp = re.sub(r'http\S+', '', temp)
temp = re.sub(r'^b\s([RT]+)?', '', temp)
```

Hugging Face

cmarkea / **distilcamembert-base-sentiment**

un très beau projet d'innovation technologique...	[{'label': '5 stars', 'score': 0.7745557427406...

2 - Traitement des données

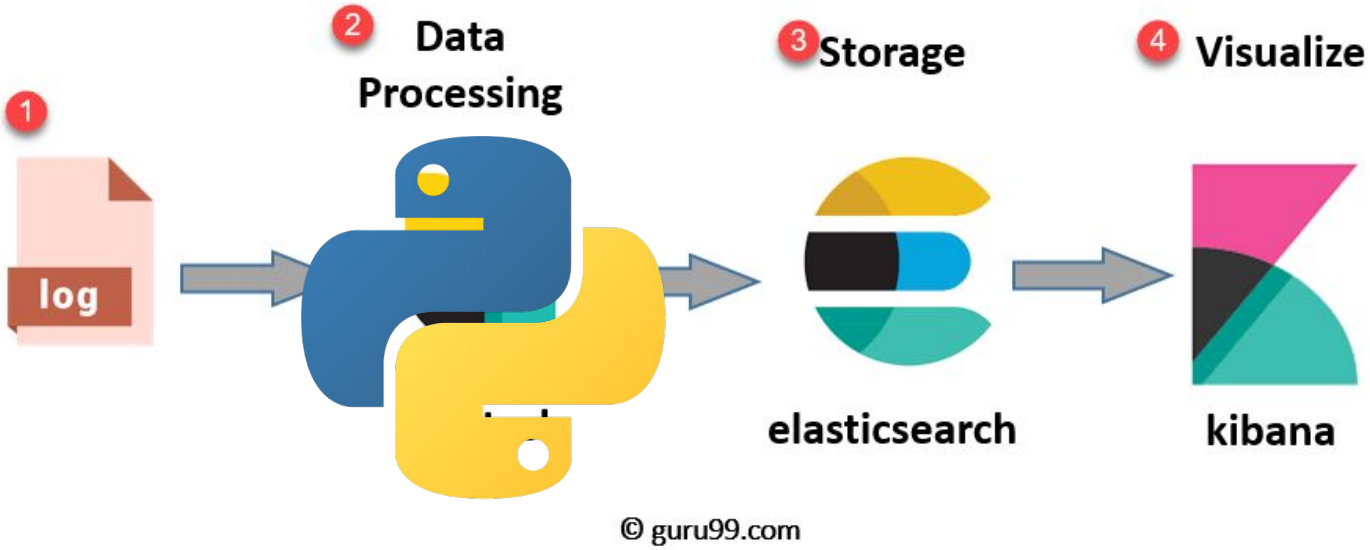
Ville	sentiment_note
Aix en Provence	4.100000
Amiens	3.400000
Angers	3.580000
Annecy	3.920000
Argenteuil	2.589744
Avignon	3.480000

GEOJSON

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

```
geo_json = to_geojson(df=tweets_df, lat='lat', lon='long',
                      properties=['Date','Tweet ID','Ville','Tweet','sentiment_note','moyenne'])
```

3 - Add BDD



- Trop de data pour un csv (1M+ tweets)
- Beaucoup de fonctionnalités de filtres/pré nlp pour les queries
- Facile à mettre en place

3 - Add BDD

```
client.indices.create(index="tweets", body=tweets)
```

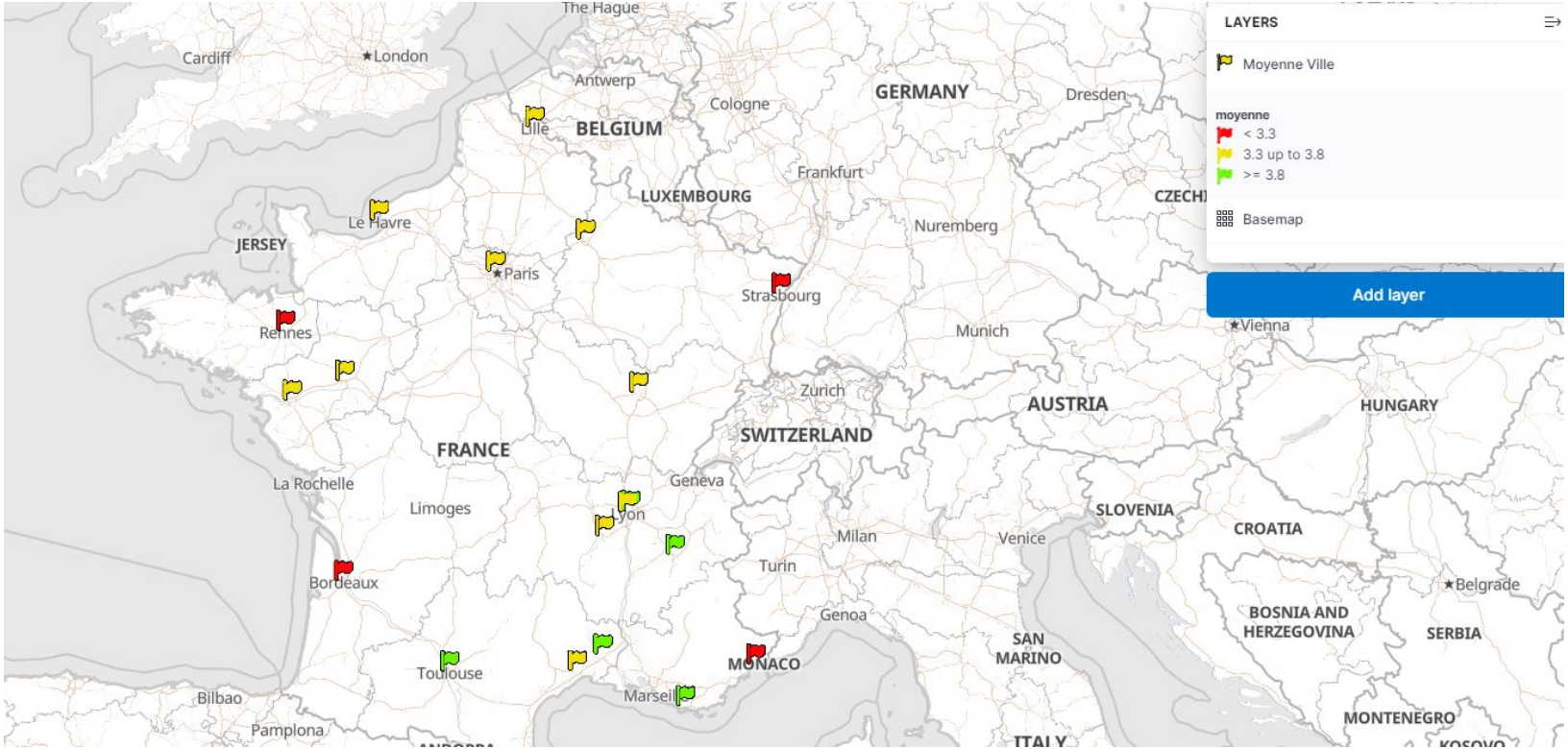
```
tweets = {
   "settings": {
        "index": {"number_of_replicas": 2},
        "analysis": {
            "filter": {
                "ngram_filter": {
                    "type": "edge_ngram",
                    "min_gram": 2,
                    "max_gram": 15,
                },
            },
            "analyzer": {
                "ngram_analyzer": {
                    "type": "custom",
                    "tokenizer": "standard",
                    "filter": ["lowercase", "ngram_filter","asciifolding","elision"], #e
                    "char_filter":"html_strip"
             }
            },
        }
    },

    "mappings": {
        "properties": {
            'Date': {'type': 'keyword'},
            'TweetID': {'type': 'keyword'},
            'Username': {'type': 'keyword'},
             'Ville': {'type': 'text', "analyzer": "standard",
                     "fields": {
                        "keyword": {"type": "keyword"},
                        "ngrams": {"type": "text", "analyzer": "ngram_analyzer"}},
                    },

             'Tweet': {"type": "text", "analyzer": "standard",
                     "fields": {
                        "keyword": {"type": "keyword"},
                        "ngrams": {"type": "text", "analyzer": "ngram_analyzer"}},

                      },
            'Coords': {'type': 'keyword'},
            }
```

```
colums = ["Date", "TweetID", "Username", "Ville","Tweet", "cords"]
index_name = "tweets"

with open("tweetswebscrap.csv", "r",encoding="utf8") as fi:
    reader = csv.DictReader(
        fi, fieldnames=colums, delimiter=",", quotechar='"'
    )


    # This skips the first row which is the header of the CSV file.
    next(reader)

    actions = []
    for row in reader:


        action = {"index": {"_index": index_name, "_id": row["TweetID"]}
        doc = {
             "Date": row["Date"],
            "TweetID": row["TweetID"],
            "Username": (row["Username"]),
            "Ville": row["Ville"],
            "Tweet": row["Tweet"],
            "cords": row["cords"]


        }
        actions.append(json.dumps(action))
        actions.append(json.dumps(doc))

    #print(actions)
    print("CONNEXION")
    client.bulk(operations="\n".join(actions),request_timeout=100)
```

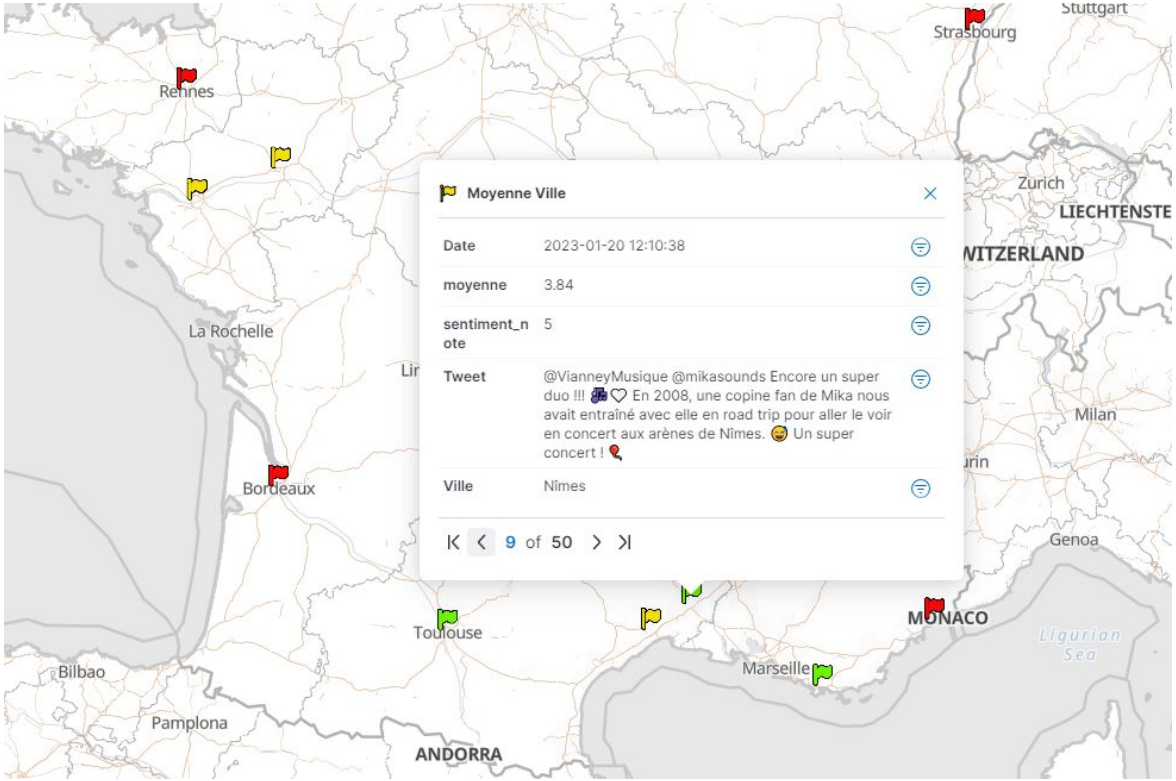
4 - Add bucket

- pour archiver la grosse quantité de data scrapée (scrap twitter gratuitement c’est long)
- sécurité supplémentaire si la BDD crash

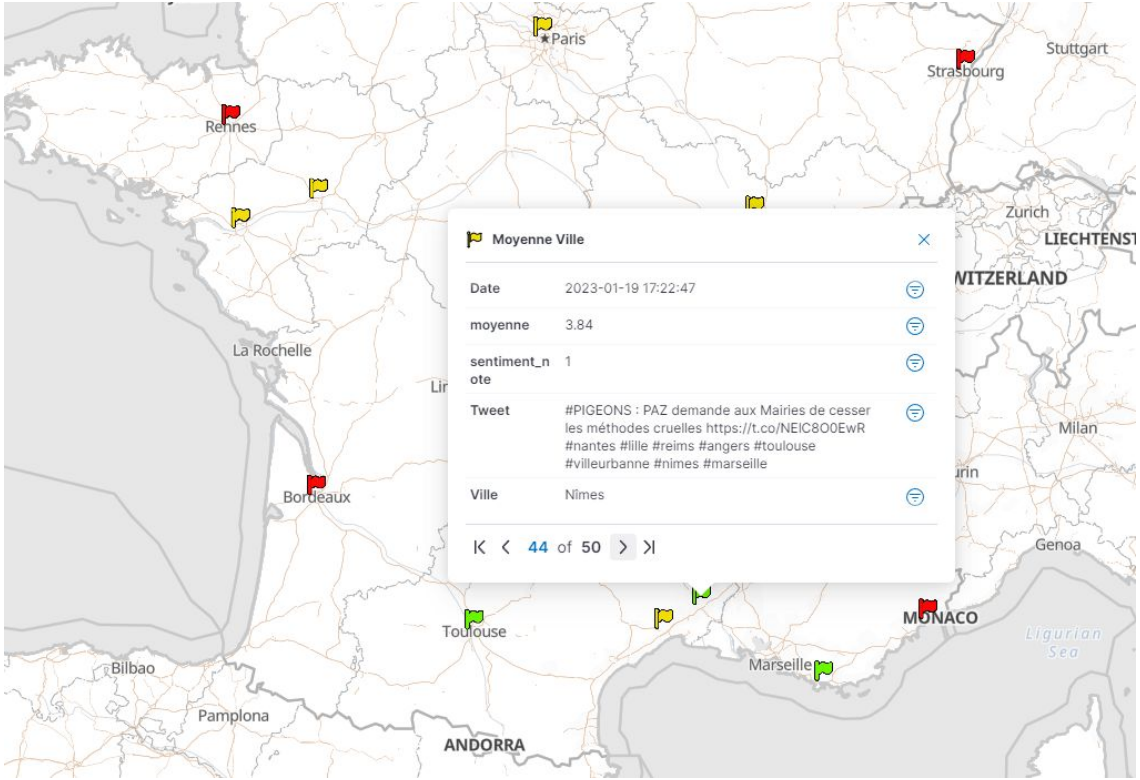
5 - Mapping des résultats



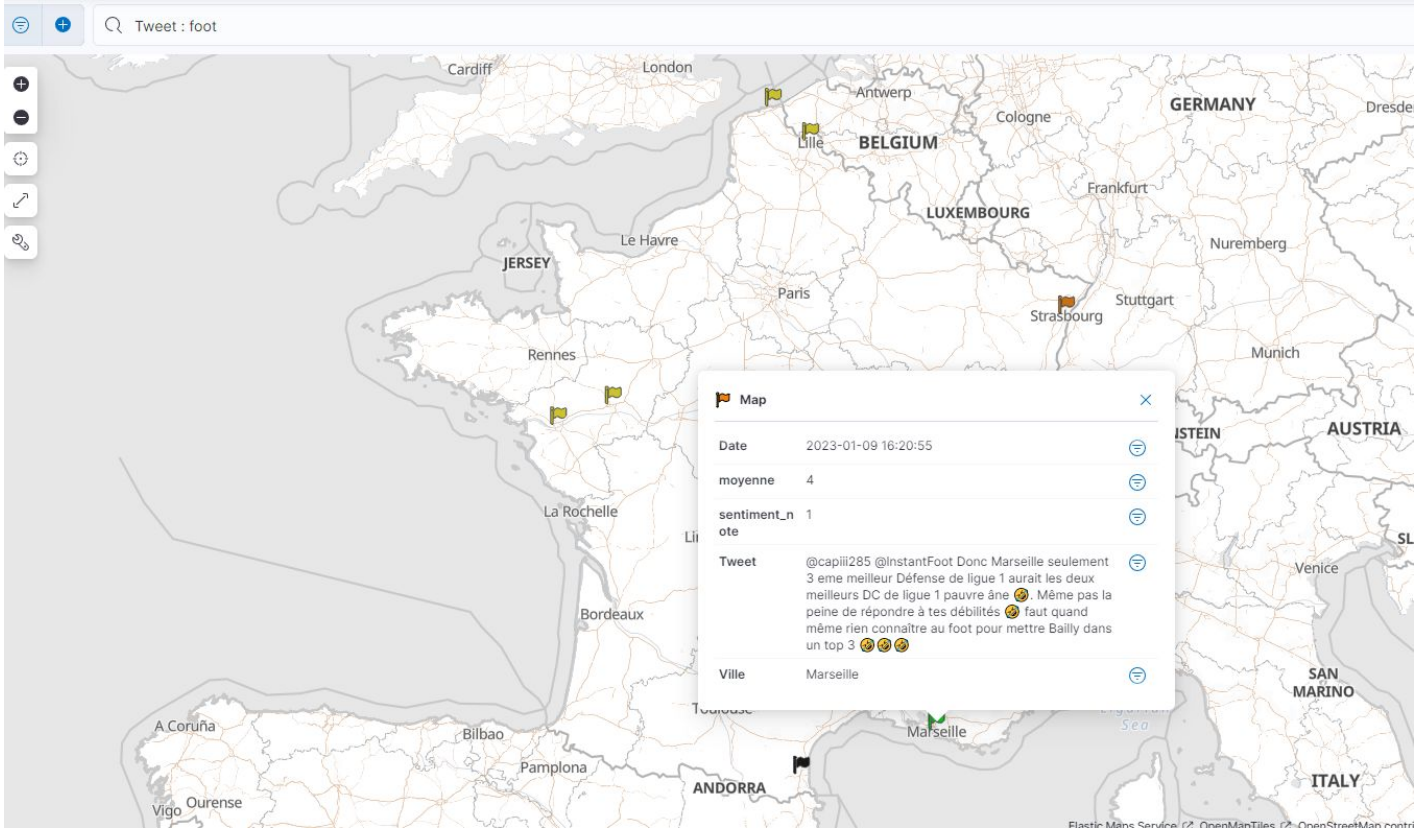
5 - Mapping des résultats



5 - Mapping des résultats



5 - Mapping des résultats



6 - Conclusion

=