



FACULTÉ DES  
SCIENCES

---

## Rapport de fin d'études Projet Tweetostats

---

*Auteur :*

Adrien DIDIER  
Nabil BOUILLIN  
Matéo MEYNIER  
Aurélien TROUCHE

*Tuteur :*

Pascal PONCELET

ANNÉE 2018-2019

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Analyse du sujet et de son contexte</b>	<b>3</b>
2.1	Définition de la vision du projet par le tuteur . . . . .	3
2.2	Définition du contexte . . . . .	3
2.3	Analyse de l'existant . . . . .	4
<b>3</b>	<b>Rapport technique</b>	<b>7</b>
3.1	Conception . . . . .	7
3.1.1	Packages implémentés . . . . .	9
3.1.2	Visualisations implémentées . . . . .	10
3.1.3	Double session . . . . .	15
3.1.4	Optimisation effectuée . . . . .	16
3.1.5	Problèmes résolus . . . . .	21
<b>4</b>	<b>Rapport d'activité</b>	<b>27</b>
4.1	Planification des tâches . . . . .	28
4.1.1	Les outils . . . . .	28
4.2	Bilan et perspectives . . . . .	28
4.2.1	Bilan . . . . .	28
4.2.2	Perspective . . . . .	29
<b>5</b>	<b>Conclusion</b>	<b>30</b>
<b>6</b>	<b>Remerciements</b>	<b>31</b>
<b>7</b>	<b>Table des figures</b>	<b>32</b>
<b>8</b>	<b>Références</b>	<b>33</b>
<b>9</b>	<b>Annexes</b>	<b>34</b>
A	Présentation affichage des sessions . . . . .	34
B	Fonction de stockage de tweets . . . . .	35
C	Fonction séparatrice de mots . . . . .	36
D	Fonction récupération des textes de tweets . . . . .	37
E	Exemple Compte rendu réalisé . . . . .	38

# 1 Introduction

Twitter est l'un des réseaux sociaux les plus utilisés dans le monde grâce à sa capacité à suivre l'actualité en temps réel.

Qu'importe l'évènement, quelqu'un "tweetera" dessus et n'importe quel utilisateur pourra alors suivre cet évènement en direct via la plate-forme. C'est l'une des grandes forces de Twitter et cela explique qu'il soit l'un des réseaux sociaux les plus prisés. Grâce à l'envoi de courts messages textuels et instantanés appelés *tweets*, les 350 millions d'utilisateurs communiquent et toutes ces communications produisent énormément de données.

Ces données sont utilisées par des data miners qui les exploitent pour par exemple, aider une entreprise à réaliser une campagne marketing réussie en analysant les comportements de ces clients sur les réseaux sociaux . Mais l'exploration de données intéressent aussi les chercheurs qui étudient la réaction des utilisateurs sur des domaines ou des évènements donnés.

Durant ce projet de fin de Licence Informatique, nous avons pour objectif d'améliorer et d'enrichir une application web aidant à l'analyse de ces tweets.

Dans un premier temps, nous présenterons notre analyse de l'existant afin de comprendre de quoi nous partions, quels étaient les points à améliorer, ceux à développer et les problèmes à résoudre. Dans un second temps, nous expliquerons d'un point de vue technique les solutions aux problèmes rencontrés. Enfin, un rapport d'activité présentera le déroulement et les méthodes utilisées durant le projet.

## 2 Analyse du sujet et de son contexte

### 2.1 Définition de la vision du projet par le tuteur

M. PONCELET a pour objectif de créer une plateforme web qui assistera les chercheurs dans leurs analyses de tweets. Un site qui a pour but d'analyser les tweets pour un mot clé donné et d'afficher une multitude de visualisations pour aider et faciliter le travail de recherche des chercheurs. Le site doit avoir une prise en main rapide et aisée avec des affichages de données sous différentes formes.

### 2.2 Définition du contexte

De nos jours, le simple fait d'utiliser son smartphone crée des données qui peuvent être analysées. On estime à environ 500 millions, le nombre de tweets envoyés par jour. Et, plus il y a de données sur internet, plus il y a de questions qui se posent. Afin d'élaborer des réponses à ces questions, il faut examiner et exploiter ces données. L'analyse des données se fait en plusieurs étapes :

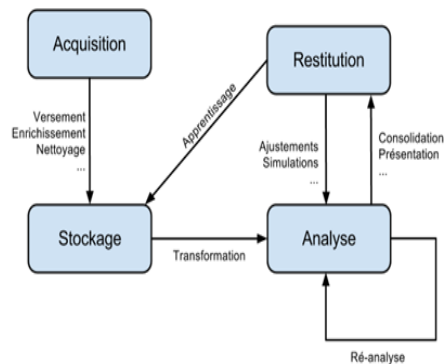


FIGURE 1 – Schéma analyse de données

Nous allons, dans ce projet, développer la partie sur l'acquisition et le stockage pour que l'utilisateur puisse, avec les données nettoyées, faire des analyses plus poussées.

Premièrement, l'acquisition des données. Ces données-là peuvent être acquises, grâce au e-shopping ou encore via des partages de données (Github). Cependant, il existe des lois protégeant une partie de nos données afin d'interdire aux entreprises d'exploiter certaines données des clients.

Ensuite vient la récupération de ces données. C'est ici qu'un trie à lieu. L'analyste doit choisir les données à examiner en fonction de son domaine de recherche. On choisit de stocker ces données dans une base de données. La base de données permet de faciliter les prochaines étapes. Les tailles des bases de données étant

limitées, il faut à ce stade encore trier les données pour ne garder que l'essentiel dans sa base de données.

Après avoir stocké les données qu'il a récupérées, viens la question de la présentation des données. Pour l'entreprise de simples données stockées dans la base de données sont inutilisables. Pour cela, il faut les structurer, par exemple sous forme de graphique circulaire. Cela peut aller de simples courbes aux diagrammes explicites où se trouvent clairement les réponses aux questions posées. Ces visualisations permettent aux personnes extérieures au domaine informatique de pouvoir exploiter ces données.

Enfin, grâce à la structuration des données récoltées, on est capable de faire de l'apprentissage à un programme. On parle de machine learning. Le programme sera capable de répondre à des problèmes précis en n'oubliant aucun des cas possibles. Ceci est possible grâce à des jeux de données qui sont récupérés, par exemple sur Twitter. Plus le jeu de données est complet plus le programme aura des réponses précises et moins il y aura d'erreurs de précision.

## 2.3 Analyse de l'existant

Au lancement du projet, l'équipe a dû analyser le projet de l'an passé pour pouvoir le reprendre, l'améliorer et l'enrichir. Avant de rentrer dans l'aspect technique, nous avons regardé si le site était ergonomique et s'il n'y avait pas de bug lors de son installation et de son fonctionnement.

La première chose que nous avons remarquée était que le projet de l'an passé n'était pas portable sur différentes plateformes. C'est-à-dire qu'il n'était déployable que sous un système Windows. Ensuite, le projet n'étant plus maintenu depuis quasiment un an, certains packages n'étaient plus compatibles avec les dernières versions de Python par exemple.

Le site utilise deux couleurs principales, le blanc et le bleu. La taille de la police d'écriture est correcte. On a donc un site avec un design simple et épuré qui permet de satisfaire amplement les utilisateurs.

En ce qui concerne l'ergonomie, le site possède un système de navigation intuitif qui est facile d'utilisation, le seul petit point noir d'ergonomie est le placement des boutons de connexions. Nous avons remarqué que la plupart des confirmations de connexion sont en dessous à gauche et non à droite (on se retrouve à aller à l'accueil au lieu d'accéder à sa session).

# Connectez-vous

Email

Mot de passe

PAGE D'ACCUEIL

CONNEXION

FIGURE 2 – Interface de connexion

L'utilisateur n'est pas submergé d'information, l'ensemble des différents choix, qu'il s'agit de s'inscrire, de consulter une session de recherche, de paramétrer sa recherche, sont contenus sur une seule page, au premier plan, sans devoir descendre dans la page.

Une session sur le site, correspond à une recherche qui va être faite, où qui a été faite.

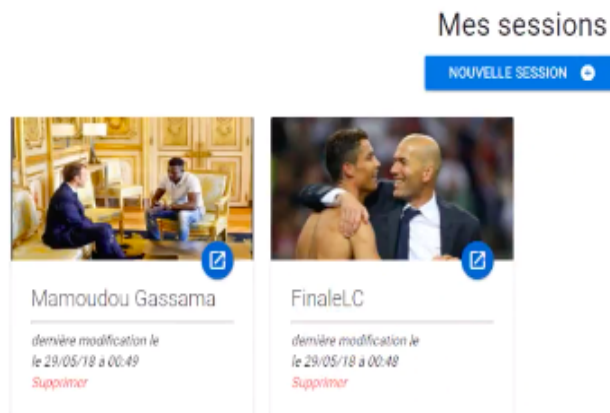


FIGURE 3 – Interface des sessions

Lors d'une recherche, l'utilisateur peut choisir, le mot clé à rechercher, le nombre de tweets à récupérer, il peut mettre un intervalle de temps grâce une date début/fin et heure début/fin. Il peut également choisir la langue de récupérations des tweets ainsi que sa localisation (exemple France). Si l'utilisateur ne spécifie pas un nombre de tweets à récupérer, alors il pourra choisir dans la

page d'après de faire une récupération dite à la volée. Il peut également revenir sur une session de recherche précédente pour pouvoir l'enrichir davantage grâce à une nouvelle récupération.

En ce qui concerne la visualisation des données, le site possède un WordCloud et un histogramme avec différent intervalle de temps (2 jours à 30 secondes). Le site permet également de pouvoir consulter les tweets qui ont été récupérés. On peut également importer ses sessions pour pouvoir les partager à une personne(importation).

Du côté de la base de données, le site possède des tables. Une table "user" qui permet de stocker les informations de l'utilisateur. Une table "tweet" permettant de conserver les tweets récupérés. Une table "sessions" capable d'archiver les sessions des différents utilisateurs. Ainsi que deux autres tables structurant la base de données.

Durant notre phase de test, nous avons remarqué deux problèmes d'optimisations. Premièrement, le site est extrêmement long, il met environ 1 minute pour afficher le WordCloud avec seulement 10 000 tweets et ensuite il affiche l'histogramme. Puis, il met également beaucoup de temps pour afficher les tweets que l'on a récupérés. Deuxièmement, la base de données est hébergée en ligne sur un hébergeur très éloigné de la fac des sciences. Ce qui a pour conséquence que le site n'est utilisable qu'avec une bonne connexion. Les connexions dites publiques (eduroam) ne permettent pas son fonctionnement.

Étant donné que le but de notre projet, nous n'avons pas pu laisser ces problèmes de côté.

## 3 Rapport technique

### 3.1 Conception

Dans cette partie, nous allons expliquer grâce à deux diagrammes UML, les caractéristiques du projet Tweetostats. Tout d'abord, nous détaillerons le diagramme de cas d'utilisation qui décrit les relations homme machine. Ensuite, nous enchaînerons avec le diagramme de séquence qui a pour but d'expliquer le cheminement du programme pendant son fonctionnement lors de la création d'une session.

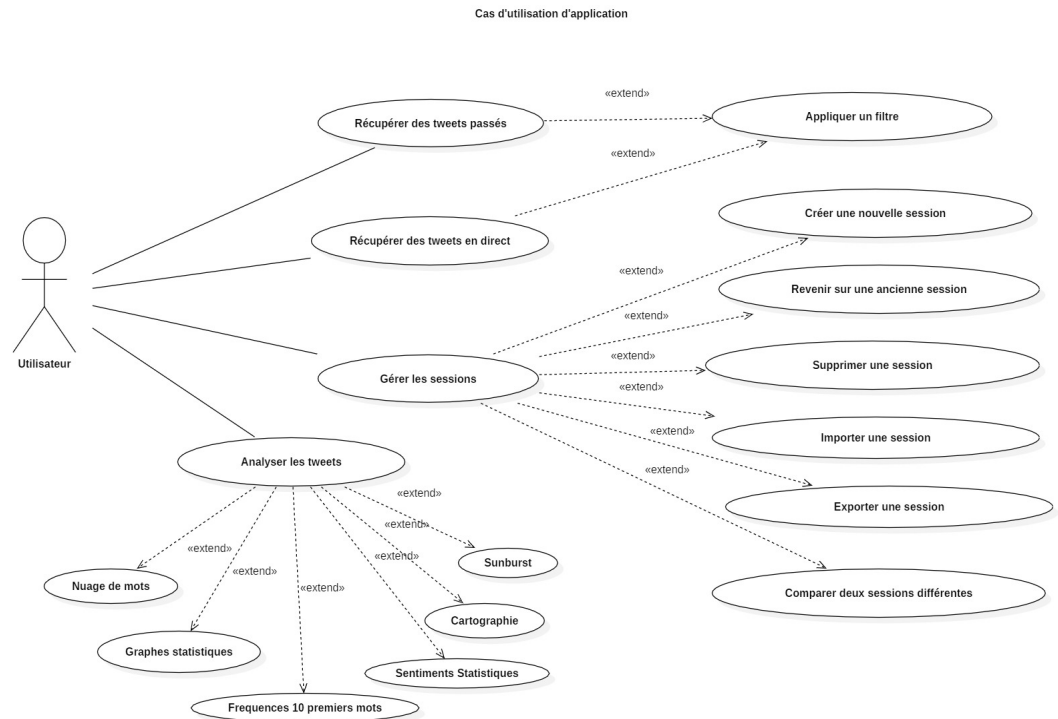


FIGURE 4 – Diagramme de cas d'utilisation

Les tweets sont récupérés selon des critères spécifiques que l'utilisateur aura imposés lors du remplissage du formulaire prévu. En effet, l'utilisateur peut récupérer des tweets en appliquant un ou plusieurs filtres : par mots-clés, par utilisateur twitter, par langue ou encore par géolocalisation. Il est possible de traiter des tweets en temps réel, mais aussi des tweets passés, l'utilisateur a donc différentes interfaces en adéquation avec ses besoins.



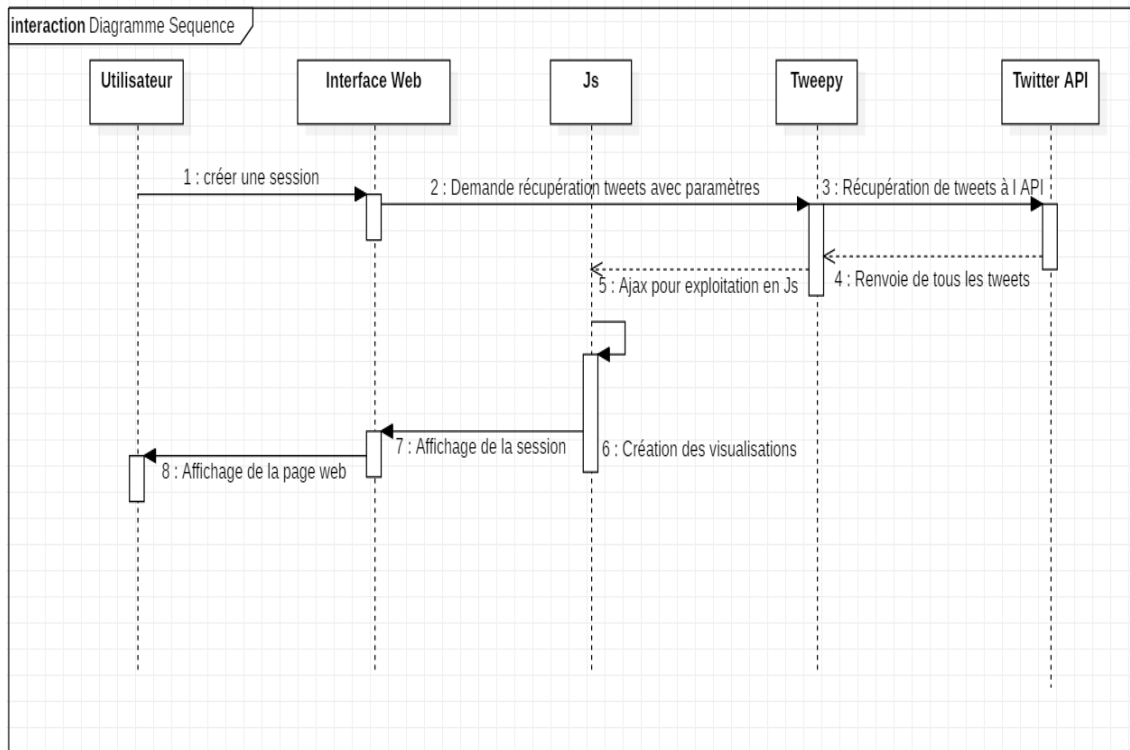


FIGURE 5 – Diagramme de séquence

Après avoir extrait les tweets, il faut pouvoir découvrir ce qu'ils contiennent, c'est pourquoi le programme va créer des visualisations pour pouvoir structurer les données recueillies.

### 3.1.1 Packages implémentés

 <b>Packages Python</b> 	
 <b>Cryptography</b>	Cryptography package est une librairie standard qui propose des outils de cryptographie.
 <b>Tweepy</b>	Tweepy est une bibliothèque Python pour accéder à l'API Twitter.
<b>TextBlob</b>	TextBlob est une bibliothèque pour le traitement des données textuelles. Il fournit une API cohérente pour plonger dans des tâches communes de traitement du langage naturel (NLP) telles que le marquage de partie de la parole, l'extraction de phrase de nom, l'analyse de sentiment, et plus encore.
 <b>Selenium</b>	Selenium est un outil d'automatisation de navigateur web. Il permet donc d'écrire, de manière plus ou moins assistée, des scripts dont l'exécution réalisera automatiquement des actions dans un navigateur web.
<b>Stop_words</b>	Package qui enlève une liste de mots vides qui n'ajoutent pas beaucoup de sens à une phrase.
 <b>Flask</b> web development, one drop at a time	Flask est un framework open-source de développement web en Python.
<b>Flask_PyMongo</b>	PyMongo support pour les applications Flask.
 <b>Pymongo</b>	PyMongo est une distribution Python contenant des outils pour travailler avec MongoDB.

FIGURE 6 – Packages Python

### 3.1.2 Visualisations implémentées





Résumé Visualisation rajoutée	
	Ajout d'une cartographie Leaflet pour géolocaliser les tweets récupérés.
	Ajout d'un histogramme qui affiche les 10 mots les plus fréquents dans les tweets récupérés.
	Ajout d'un sunburst qui affiche en pourcentage des informations comme le nombre de RT de tweets, le nombre de tweets géolocalisé, le nombre de liens ...
	Ajout d'un camembert qui affiche la polarité des tweets récupérés en pourcentage.

FIGURE 7 – Résumé Visualisation Projet

L'un des principaux objectifs de notre projet était de rajouter des visualisations pour les données qu'on récupérerait de Twitter. Suite à une décision commune avec l'équipe de développement et notre tuteur, nous avons donc décidé d'utiliser comme l'équipe précédente D3.js. D3.js est une librairie JavaScript qui permet de manipuler des données et de les afficher sous forme graphique. Cette librairie utilise le format SVG et le langage JavaScript pour proposer énormément de visualisations. L'un des principaux avantages de D3 est sa "rapide" prise en main ainsi que son énorme catalogue de visualisations. Le GitHub de la librairie propose des tutoriels pour apprendre à créer toutes sortes de graphiques et met à disposition de n'importe qui une galerie d'exemples pour comprendre, s'inspirer et créer ses propres visualisations. Nous avons aussi eu recours à Leaflet, une librairie JavaScript de cartographie en ligne pour intégrer au site une carte contenant les localisations des tweets récupérés même si peu de tweets contenaient des coordonnées. Les données du diagramme des sentiments sont obtenues grâce à TextBlob, une librairie Python qui utilise le traitement automatisé du langage pour analyser les sentiments dans les tweets récupérés.

## **Cartographie**

La première visualisation implémentée a été la cartographie. Leaflet nous permet d'avoir une carte interactive sur laquelle on peut placer nos marqueurs de positions. Nous analysons les tweets de la dernière session et récupérons les coordonnées des tweets géo-localisés pour ajouter un par un des marqueurs aux coordonnées récupérées.

Nous pouvons via cette carte, observer si une région particulière est impliquée dans le sujet choisi, même s'il y a un faible nombre de tweets possédant une localisation et que cette localisation est celle du point relais Twitter d'où le fait que les tweets se concentrent souvent au même endroit.

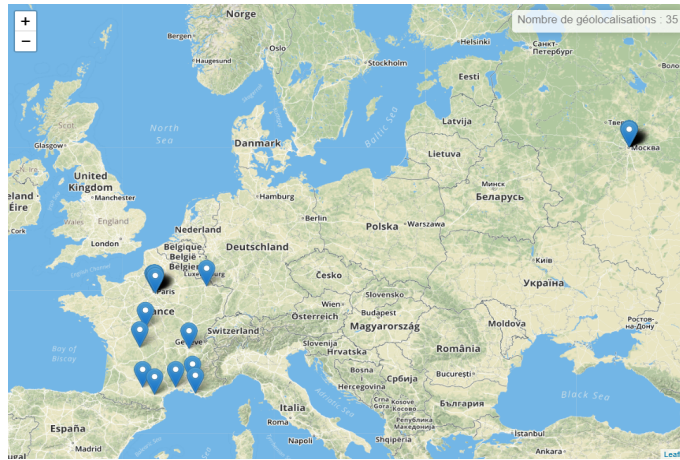


FIGURE 8 – Visualisation cartographique

## Diagramme des sentiments

Nous avons ensuite intégré le diagramme crée via TextBlob qui classe les tweets en fonction de s'ils sont neutres, positifs ou négatifs. On appelle TextBlob sur chacun des tweets de notre session et celui-ci nous renvoie une valeur *polarity* qui est comprise entre -1 et 1 et qui correspond à un sentiment négatif (rouge), neutre(bleu) ou positif (vert).

```
for tweet in tweets_table.find({"session_id": session['last_session']}):
    tweet_text = clean_text(getText(tweet['tweet_object']))
    sentiment_value = TextBlob(tweet_text)
    if sentiment_value.polarity > 0.00:
        positif = positif + 1
    if sentiment_value.polarity == 0:
        neutre = neutre + 1
    if sentiment_value.polarity < 0.00:
        negatif = negatif + 1
```

FIGURE 9 – code TextBlob

On compte ensuite le nombre de valeurs pour chacun des sentiments puis on utilise D3 pour en faire un camembert.

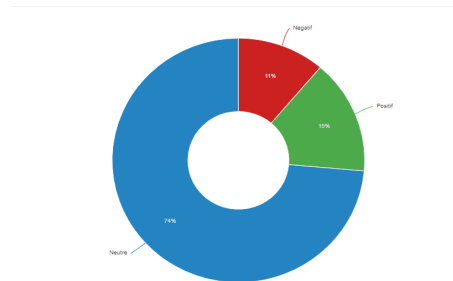


FIGURE 10 – Visualisation sentiments

## Sunburst

Nous voulions avec notre tuteur proposer une visualisation pour des données hiérarchisées. Nous avons porté notre choix sur un *sunburst graph*, un diagramme circulaire à plusieurs niveaux. Pour l'instant, celui-ci n'a qu'un niveau de données mais à l'avenir, des niveaux pourront être rajoutés comme par exemple des sous-sections au paramètre coordonnées en fonction de la localisation des tweets.

Les données collectées sont le nombre de retweets, le nombre de liens et le nombre de coordonnées.

```
for tweet in tweets_table.find({"session_id": session['last_session']}):
    if 'retweeted_status' in tweet['tweet_object']:
        if tweet['tweet_object']['retweeted_status'] is not None:
            compteurrt += 1
    if 'coordinates' in tweet['tweet_object']:
        if tweet['tweet_object']['coordinates'] is not None:
            compteurcoord += 1
    if 'entities' in tweet['tweet_object']:
        if len(tweet['tweet_object']['entities']['urls']) != 0:
            compteurlink += 1
```

FIGURE 11 – code Sunburst

Nous utilisons ensuite D3 pour en faire un sunburst affichant le pourcentage de chaque paramètre lors du survol de sa section.

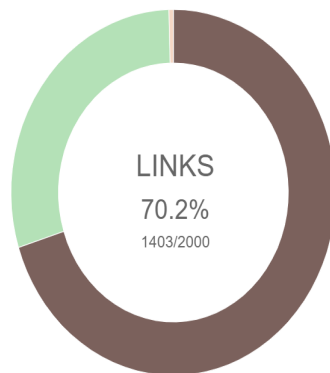


FIGURE 12 – Sunburst

### Diagramme des mots fréquents

La dernière visualisation est un histogramme d3 qui regroupe les fréquences des 10 mots les plus cités. On parcourt la liste des mots et on compte via la fonction Counter le nombre d'occurrences de chacun des mots. On renvoie ensuite les 10 mots les plus récurrents.

Le code étant court, nous le présentons ici :

```
def frequency_word():
    tweets_table = mongo.db.tweets
    all_tweets_words = []
    for tweet in tweets_table.find({"session_id": session['last_session']}):
        for word in tweet['tweet_object']['split']:
            all_tweets_words.append(word)
    count_words = collections.Counter(all_tweets_words)
    return count_words.most_common(10)
```

FIGURE 13 – Code fréquences Mots

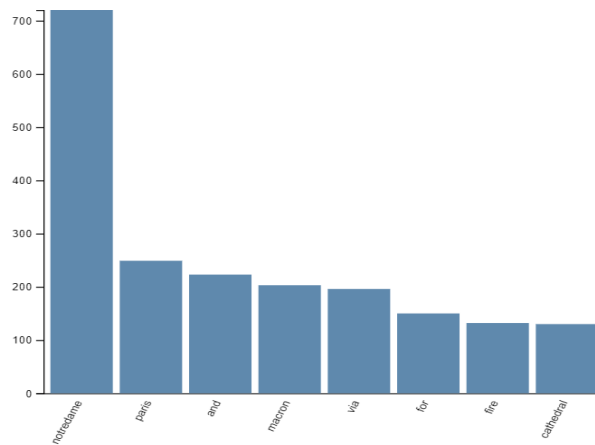


FIGURE 14 – Diagramme fréquence Mots

### 3.1.3 Double session

Une partie importante de notre projet a été consacré à la création d'une double session.

Premièrement, nous avons dû réfléchir à comment nous allions implémenter cette fonctionnalité et surtout quelles visualisations seraient présentes pour différencier deux sessions.

Suite à de nombreuses réflexions, nous avons décidé de choisir toutes les visualisations à l'exception du Wordcloud qui selon nous n'apportait aucun intérêt lors d'une comparaison. Le résultat d'une double session pourrait ressembler à cela (nous avons occulté les graphiques pour avoir une vision globale de la page) :



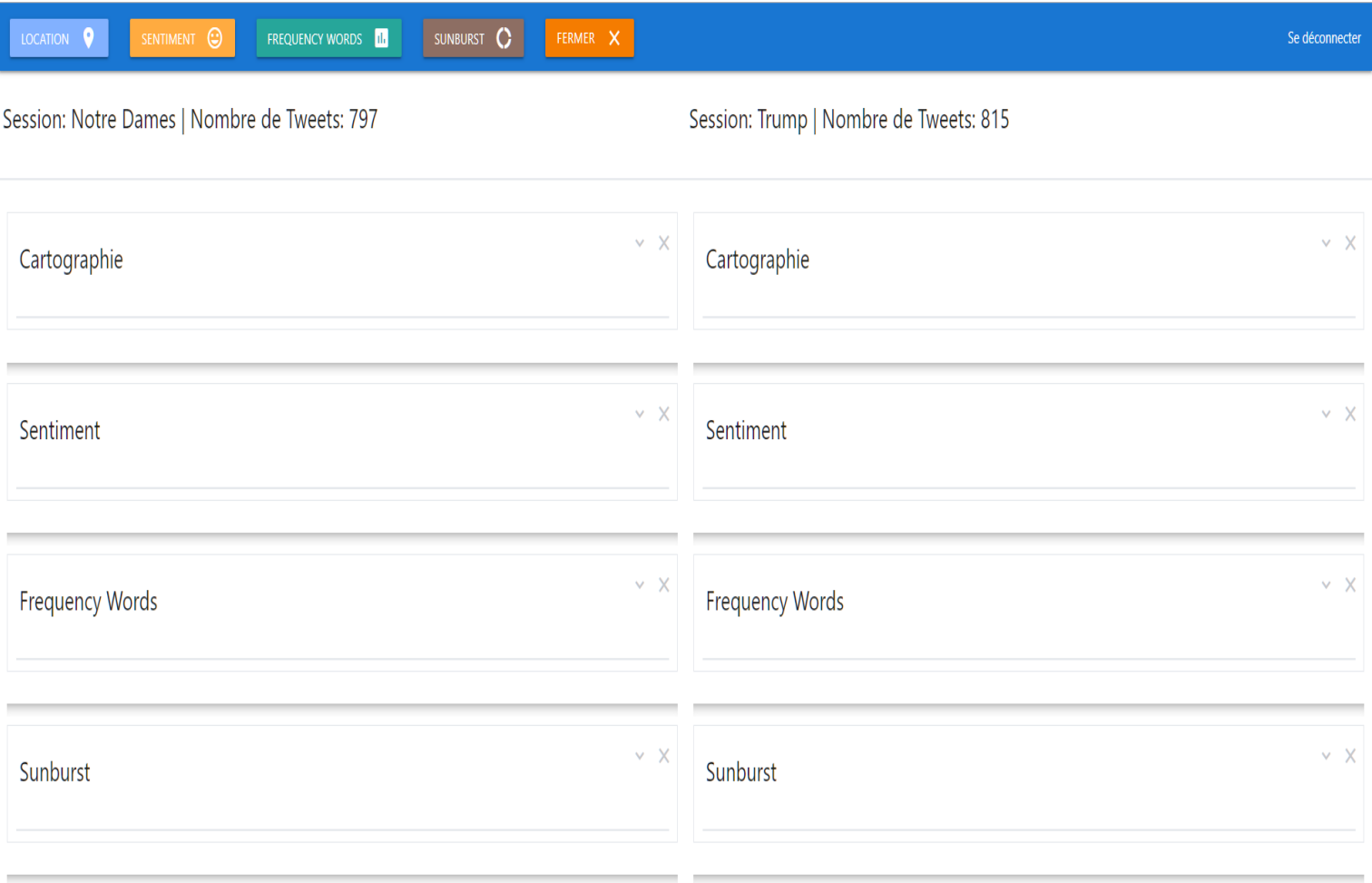


FIGURE 15 – Visualisation d’une double session

3.1.4 Optimisation effectuée



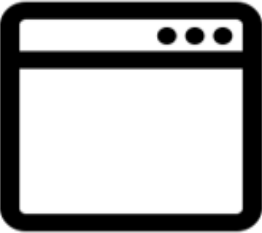

Résumé optimisation du projet	
	Ajout d'un champ dans la base de données afin d'optimiser l'affichage du wordcloud.
 BDD	Rapatriement de la base de données en local.
	Ajout système de fenêtre pour chacune des visualisations pour optimiser l'interface utilisateur.
	Ajout la possibilité d'afficher plus de paramètres de recherche en dessous de chaque session.

FIGURE 16 – Résumé Optimisation Projet

Le plus gros problème concernant l'optimisation (RAM), portait sur le nuage de mot. Lorsque l'on voulait rouvrir une session déjà créée, l'affichage du nuage de mots était bien trop long. Le site lançait de nouveau les fonctions de traitement de texte. Ces fonctions permettent de retirer les mots de liaison, compter l'occurrence d'un mot et pour finir afficher le WordCloud.

Pour résoudre ce problème, nous avons rajouté un attribut dans la base de données ("split"). Cet attribut contient un tableau de mots récupéré par rapport aux tweets de la session. Nous avons également modifié la fonction "word\_splitter" afin qu'elle ne récupère plus que 20 % des mots qui présentent le plus grand nombre d'occurrences sur l'ensemble des tweets récupérés. Lors de la première recherche, le site stocke dans sa base de données les mots qui sont passés dans

les fonctions de traitement de texte.

Ainsi, lorsque l'utilisateur voudra de nouveau accéder à une session passée, le site n'aura plus qu'à chercher les mots déjà traités. Le temps gagné est assez conséquent, on a pu le constater sur une session de 30 000 tweets où le temps d'affichage du Wordcloud passe de 3min environ à 1min actuellement. Ce temps d'affichage reste quand même un problème car les autres visualisations ne sont pas disponibles pendant le temps de chargement. Nous précisons dans la partie sur les problèmes rencontrés comment nous avons réussi à contourner ce problème.

```

  _id: ObjectId("5cc01c63e6f12d271481211c")
  session_id: "5cc01c62e6f12d271481211b"
  tweet_object: Object
    id_str: "1120964861100535808"
    favorite_count: 1
    place: null
    retweeted: false
    in_reply_to_status_id: 1120936471328899072
    source: "<a href='\"http://twitter.com/download/iphone\"' rel='\"nofollow\"'>Twitter fo..."
    display_text_range: Array
      lang: "sv"
      is_quote_status: false
    user: Object
      truncated: false
      full_text: "@PDefer Dubbelmoralen från denna feministiska ikon är bedövande, ena d..."
    metadata: Object
      favorited: false
    entities: Object
      in_reply_to_user_id: 2621084793
      created_at: "Wed Apr 24 08:17:04 +0000 2019"
      geo: null
      contributors: null
      split: Array
      coordinates: null
      retweet_count: 0
      in_reply_to_status_id_str: "1120936471328899072"
      in_reply_to_user_id_str: "2621084793"
      in_reply_to_screen_name: "PDefer"
      id: 1120964861100535808
```

FIGURE 17 – Visualisation champ de la base de données (champ split)

Également, pour améliorer l'ensemble du service, un rapatriement de la base de données en local a été effectué. Désormais, le programme ne devra plus passer par le service web pour récupérer d'anciennes sessions. Nous avons rapatrié la base de données car sur des connexions publiques, comme "eduroam" à la FAC de Montpellier par exemple, le projet TER n'était pas capable de fonctionner, car la connexion n'était pas assez puissante (ou trop sécurisé). De plus le fait d'avoir une base de données en local nous permet d'effectuer des échanges bien plus rapides avec celle-ci, ce qui nous a permis de gagner quelques microsecondes de traitement sur l'affichage et la création de visualisation.

Pour le confort utilisateur, nous avons dû repenser le système d'affichage afin que l'utilisateur ne soit pas submergé d'informations, pour cela nous avons optimisé l'interface utilisateur en ajoutant la possibilité de fermer, de réduire, ou d'ouvrir une visualisation à la guise de l'utilisateur. En effet le bouton "croix" nous permet de cacher totalement la visualisation, pour la rouvrir il faudra cliquer sur le bouton de la visualisation dans la barre de navigation pour que celle-ci réapparaisse. Le bouton réduire, quant à lui, permet aussi de cacher la visualisation mais aussi de la l'ouvrir de nouveau. On peut comparer ce système au système de fermeture de fenêtre d'un ordinateur Windows par exemple.



FIGURE 18 – Visualisation boutons de réduction et de fermeture

Nous avons aussi ajouté la possibilité d'afficher plus de paramètres de recherche en dessous de chaque session. Nous avons fait cela pour donner la possibilité à l'utilisateur de se rappeler en cas d'oubli ou autre de tous les paramètres qu'il a pu fournir.



Trump

---

**Nombre de tweets :** 8  
**Mot(s) clé(s) :** Trump  
**Type de sessions :**  
dated\_tweets  
**Dernière modification le**  
le 07/05/19 à 14:51

**ID de la session :**  
5cd17f1ce6f12d0b60036db2  
**Nombres de tweets à**  
**récupérer :** 100  
**Date de début :** le 07/05/19  
**Date de fin :** le 18/05/2019  
**Heure de début :** à 13:00  
**Heure de fin :** à 18:00  
**Language :** en  
[Supprimer](#)

FIGURE 19 – Visualisation paramètres d'une session

### 3.1.5 Problèmes résolus





<b>Problèmes Rencontrés</b>	
 <b>Bcrypt</b>	Clés et tokens par défaut pour se connecter à l'API Twitter. Les utilisateurs ne peuvent utiliser leurs propres clés et tokens à cause du package bcrypt qui ne permet pas le déchiffrement une fois le chiffrement exécuté.
<b>Formulaire Session</b> 	La version 0.100.2 du framework css materialize entraine des erreurs d'affichage lors de la sélection des paramètres du formulaire pour créer une session de tweets passés ou en direct.
 <b>Localisation Tweet</b>	La plupart des tweets récupérés n'ont pas de coordonnées de géolocalisation puisque les utilisateurs la désactivent dans leurs paramètres Twitter.
 <b>Chargement</b>	Temps de chargement trop long lorsque l'on ouvre une session avec énormément de tweets. La page web n'est même plus fonctionnelle pendant un laps de temps.
<b>Doubles Sessions</b>	Avec la visualisation double sessions, nous obtenions deux fois l'affichage de la première session sélectionnée.

FIGURE 20 – Résumé Problèmes du Projet





<b>Solutions trouvées</b>	
 <b>Cryptography</b>	Utilisation du package cryptography pour utiliser du chiffrement asymétrique afin de permettre aux utilisateurs de pouvoir utiliser leurs propres clés et tokens pour se connecter à l'API Twitter.
<b>Formulaire Session</b> 	Mise à jour du framework css materialize vers la version 1.0.0 et modification des classes css dans le formulaire pour créer une session de tweets passés ou en direct.
 <b>Localisation Tweet</b>	Ajout d'un popup sur la carte pour afficher le nombre de tweets géolocaliser afin de prévenir l'utilisateur lorsqu'aucun tweet n'a de coordonnées de géolocalisation.
 <b>Chargement</b>	Modification de l'ordre d'affichage des visualisations.
<b>Doubles Sessions</b>	Ajout d'une fonction qui va chercher le bon id pour chacune des sessions afin d'afficher les deux sessions en chargeant d'abord les données de la première session puis ceux de la deuxième session.

FIGURE 21 – Résumé Solutions du Projet

Le premier problème rencontré concerne l'identification des utilisateurs, au début du projet, lorsque les utilisateurs se connectent au site, les clés et les tokens utilisés pour pouvoir récupérer des tweets sont définis par défaut dans le code. On n'utilise pas les clés et les tokens entrés par l'utilisateur lors de son inscription. Cela est dû à l'utilisation du package "bcrypt" qui lors de l'inscription d'un utilisateur, encrypte les clés avec les tokens dans la base de données. Le package python bcrypt n'a pas de fonctions pour pouvoir décrypter une clé, on ne peut donc plus récupérer celles-ci lorsque les utilisateurs se connectent. Pour résoudre ce problème nous avons donc utilisé un système de clé asymétrique (clé privée / clé public) défini dans le projet grâce au package python "cryptography". Les utilisateurs peuvent maintenant s'inscrire en utilisant leurs tokens et leurs clés de leur compte Twitter.



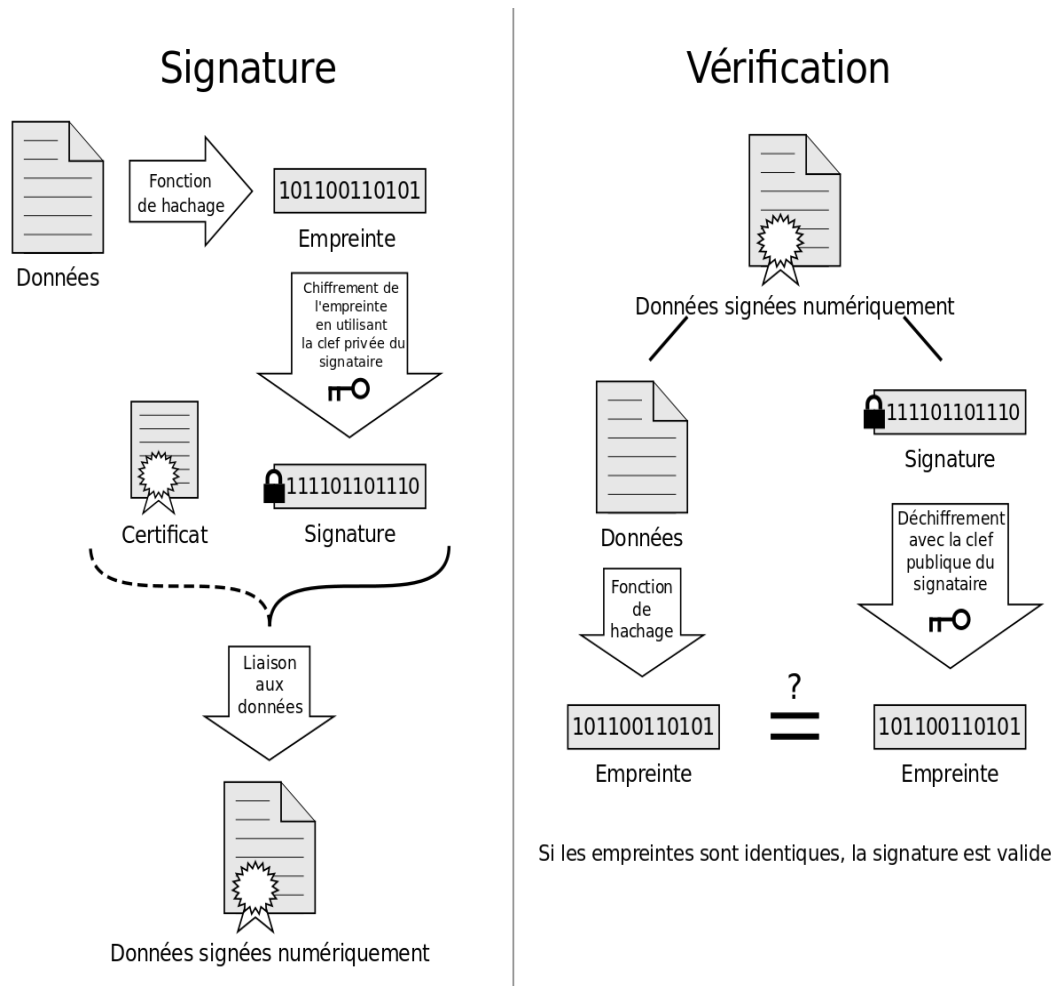


FIGURE 22 – Fonctionnement encrypte/décrypte

Par la suite nous avons pu découvrir, à travers notre analyse de l'existant, des erreurs au niveau des champs du formulaire de création de sessions de tweets directs ou passés. Les champs pour sélectionner les dates, les heures ou encore les langues n'étaient pas fonctionnels. Ces problèmes étaient dus à une version trop ancienne du framework css materialize. Nous avons ainsi effectué une mise à jour de ce framework et modifié l'intégralité du css du projet pour que les champs du formulaire puissent fonctionner sur chacun des navigateurs web.

En ce qui concerne la cartographie, le nombre de tweet géolocalisé est assez faible, voire inexistant. Pour que l'utilisateur ne pense pas que le service de géolocalisation ne fonctionne pas, nous avons rajouté un compteur de tweet

géolocalisé. Grâce à celui-ci l'utilisateur connaît directement le nombre de localisation.



FIGURE 23 – Nombre de géolocalisation

Nous avons aussi eu des problèmes de temps de chargement. Malgré le fait que le Wordcloud charge plus rapidement cela ne permettait pas à l'utilisateur de naviguer vers les autres visualisations. Pour résoudre ce léger souci, nous avons décidé de changer l'ordre de rafraîchissement des visualisations pour permettre à l'utilisateur de voir les autres graphiques, tableau... le temps que le nuage de mots charge et puisse être visionné.

Lors du développement de la visualisation en doubles sessions, nous avons eu un conflit d'information. Les deux sessions avaient les mêmes informations. Le problème lors de l'affichage d'une double session était dans l'appel des fonctions. Les fonctions ont été développées pour l'affichage d'une seule session et non de plusieurs. Du coup, les fonctions affichées toujours la dernière session qui était en mémoire.

On se retrouve avec la première session qui était bonne, mais la seconde session avec les données de la première.

Pour résoudre ce problème, le groupe a dû faire en sorte que pour chaque appel de fonction, la fonction aille bien chercher les bonnes données grâce au passage en paramètre de l'id de la session voulu. Nous avons donc rendu toutes les fonctions "générique" afin que la double session soit totalement fonctionnelle.

```
function refresh_tweet_polarity(start, session) {  
    if (start) {  
        if (session != null && session == 2) {  
            var pol = '#polarity_panel' + session;  
            console.log(pol);  
            var polarity = $(pol);  
            polarity.find('svg').remove();  
            $('#loading_circle_polarity2').show();  
        }  
        else {  
            var polarity = $('#polarity_panel');  
            polarity.find('svg').remove();  
            $('#loading_circle_polarity').show();  
        }  
    } else {  
        ajax_tweet_polarity(session);  
    }  
}
```

FIGURE 24 – Exemple d'une fonction générique

## 4 Rapport d'activité

Lors de la réalisation de ce projet, nous avons formé une équipe qui est composée de quatre personnes : Nabil BOUILLIN, Adrien DIDIER, Matéo MEYNIER et Aurélien TROUCHE.

Nous venions tous de l' IUT de Montpellier et avions l'habitude de travailler en équipe lors des différents travaux de groupes des années précédentes, ainsi nous avons les mêmes méthodes de travail et d'organisation.

Ce projet a débuté lors de notre premier rendez-vous avec notre tuteur, Pascal PONCELET.

Nous avons alors décidé de mettre en place des rendez-vous avec notre tuteur toutes les deux semaines pour avoir des directives et un suivi poussé tout au long du projet.

Lors des premiers rendez-vous, nous avons dû prendre du temps pour analyser ce qui avait été fait précédemment dans le projet.

Le projet était fonctionnel mais étant très peu commenté, nous avons uniquement le rapport de l'an passé et le code brut pour assimiler le projet, de quelle manière il fonctionnait, la récupération des tweets via Tweepy et les liens établis entre les différentes fonctions. Surtout qu'aucun de nous n'avait de connaissances en Python or Tweepy est une librairie Python. Mais peu à peu, nous avons commencé à comprendre le fonctionnement global du projet et nous avons avancé rapidement dans le projet.

Lors des rendez-vous, nous présentions les différentes avancées faites depuis le dernier rendez-vous ainsi que les problèmes rencontrés.

M. Poncelet nous conseillait ensuite sur la voie à prendre pour résoudre nos problèmes ainsi que sur les prochaines fonctionnalités qu'il voulait voir implémenter. Après chaque réunion, nous faisons un résumé des points abordés ainsi que des tâches à faire pour la prochaine fois sous la forme d'un compte rendu qui était ensuite déposé ensuite sur le Google Drive et envoyé au tuteur. Cela nous permettait de savoir précisément ce qu'on devait réaliser durant les deux semaines et de garder un oeil sur les différentes avancées.

Nous utilisions aussi l'application Messenger pour discuter du projet et de nos avancées personnelles via un groupe dédié au projet. Cela nous a permis d'instaurer dès le début du projet une bonne communication au sein du groupe afin de pouvoir répondre aux besoins de notre tuteur.

## 4.1 Planification des taches

### 4.1.1 Les outils

Les outils utilisés pour ce projet sont multiples. En effet, les informations étaient diffusées par mail entre le tuteur et notre groupe d'étudiants. En ce qui concerne les discussions internes au groupe, nous faisons des réunions de projet à la bibliothèque universitaire durant nos pauses repas quelques fois par semaine et nous avons, comme dit précédemment un groupe Messenger pour discuter tous ensemble lorsque nous étions à distance.

En ce qui concerne le développement, celui-ci a été fait sur l'IDE Pycharm sous Windows, Linux et OSX. Cette IDE nous a permis de faciliter l'utilisation la version 3.5.4 de Python et du logiciel Git pour pouvoir commit nos modifications directement sur Github. Pycharm permettait de lancer facilement le projet car il gère de nombreux langages ainsi que le serveur Flask sur lequel tourne la plateforme web actuellement.

Nous avons aussi utilisé Google Drive pour pouvoir regrouper l'ensemble des comptes rendus des réunions (7 au total) pour pouvoir les partager avec les membres du groupe ainsi que le tuteur.

Également, le groupe s'est beaucoup servi de la bibliothèque JavaScript D3js pour pouvoir chercher et définir les nouvelles visualisations à implémenter dans le projet.

Pour l'écriture de ce rapport, nous avons utilisé un éditeur Latex en ligne qui se nomme Overleaf. Cette plateforme gratuite offre la possibilité de rédiger des documents de manière collaborative. Nous avons ainsi pu rédiger et mettre en forme rapidement notre rapport de projet.

## 4.2 Bilan et perspectives

### 4.2.1 Bilan

Dans l'ensemble, le projet est beaucoup plus fonctionnel qu'en janvier. Le rajout de visualisations permet d'avoir un projet plus abouti en proposant à l'utilisateur plusieurs visions des données récupérées ainsi que plus d'informations sur celles-ci. L'ensemble du projet a été optimisé grâce au rapatriement de la base de données et à la réécriture de certaines fonctions Python et JQuery. L'affichage des visualisations est donc plus rapide et moins pénalisant pour l'utilisateur surtout au niveau du wordcloud qui ralentissait le site lors de son chargement. L'ajout de la double session est aussi une avancée car cela permet la comparaison de deux sessions entre elles.

### 4.2.2 Perspective

Comme dit précédemment, nous avons beaucoup optimisé le programme, mais nous pensons qu'une nouvelle optimisation devra être faite l'an prochain. N'ayant pas eu le temps de modifier l'ensemble de nos fichiers comportant des requêtes ajax pour optimiser le chargement des visualisations. Les étudiants qui poursuivront le projet devront ainsi essayer d'implémenter un système de multi processus pour les requêtes Ajax du projet.

Il faudra également continuer à enrichir le nombre de visualisations possible pour rendre le site encore plus apte à être utilisé pour des recherches notamment un streamgraph qui est une visualisation intéressante à proposer d'après M. Poncelet. Enrichir les données du Sunburst via l'ajout de sous-niveaux comme la langue des tweets, leur localisation par pays ou si ce sont des réponses serait aussi une voie d'amélioration pour le rendre plus pertinent.

## 5 Conclusion

Notre objectif était d'améliorer le projet "*TweetoStats*" débuté l'année dernière par des étudiants de L3 Informatique. Un projet qui offre la possibilité d'extraire et d'étudier des tweets provenant de Twitter sur un sujet précis via différentes visualisations. Ce projet sera amené à être, les années suivantes, amélioré afin de proposer un outil fonctionnel et optimisé pour les chercheurs de l'UM de Montpellier. Nous avons donc amélioré ce projet en respectant les attentes et les conseils de notre tuteur tout en optimisant grandement sa vitesse d'exécution.

"*TweetoStats*" a été une expérience enrichissante. Nous avons consolidé nos connaissances en JavaScript et HTML et appris un nouveau langage de programmation (Python). Ce projet nous a également permis d'acquérir une nouvelle expérience dans un projet en groupe ce qui a renforcé notre organisation et nos méthodes de travail. Cette expérience nous a permis également d'aiguiser notre esprit d'analyse.

## 6 Remerciements

Nous tenons à remercier notre encadrant de TER Pascal PONCELET pour sa disponibilité et son aide durant la durée du projet Il a toujours su exprimer clairement ses besoins en tant que tuteur et nous conseiller sur la voie à suivre pour les réaliser.

Nous tenons aussi à remercier toutes les personnes qui ont pu, de près ou de loin, contribuer à la concrétisation de celui-ci.



## 7 Table des figures

### Table des figures

1	Schéma analyse de données . . . . .	3
2	Interface de connexion . . . . .	5
3	Interface des sessions . . . . .	5
4	Diagramme de cas d'utilisation . . . . .	7
5	Diagramme de séquence . . . . .	8
6	Packages Python . . . . .	9
7	Résumé Visualisation Projet . . . . .	10
8	Visualisation cartographie . . . . .	12
9	code TextBlob . . . . .	12
10	Visualisation sentiments . . . . .	13
11	code Sunburst . . . . .	13
12	Sunburst . . . . .	14
13	Code fréquences Mots . . . . .	14
14	Diagramme fréquence Mots . . . . .	15
15	Visualisation d'une double session . . . . .	16
16	Résumé Optimisation Projet . . . . .	17
17	Visualisation champ de la base de données (champ split) . . . . .	18
18	Visualisation boutons de réduction et de fermeture . . . . .	19
19	Visualisation paramètres d'une session . . . . .	20
20	Résumé Problèmes du Projet . . . . .	21
21	Résumé Solutions du Projet . . . . .	22
22	Fonctionnement encrypte/décrypte . . . . .	24
23	Nombre de géolocalisation . . . . .	25
24	Exemple d'une fonction générique . . . . .	26
25	Interface des sessions . . . . .	34
26	Fonction stockage tweets . . . . .	35
27	Fonction word_splitter . . . . .	36
28	Fonction getText tweets . . . . .	37
29	Compte Rendu . . . . .	38

## 8 Références

### Références


- [1] Documentation API Twitter, Tweepy :  
[https://github.com/AdrienL3/TER\\_Twitter](https://github.com/AdrienL3/TER_Twitter)  
Consulté le : 24/01/2019
- [2] Documentation API Twitter, Tweepy :  
<https://www.tweepy.org>  
Consulté le : 30/01/2019
- [3] Documentation Langage Python :  
<https://www.python.org>  
Consulté le : 30/01/2019
- [4] Documentation du Framework CSS Materialize :  
<https://materializecss.com>  
Consulté le : 30/01/2019
- [5] Documentation Cartographie Leaflet :  
<https://leafletjs.com>  
Consulté le : 14/02/2019
- [6] Documentation Base de Données MongoDB :  
<https://docs.mongodb.com>  
Consulté le : 20/03/2019
- [7] Depot Github Template flask-gentelella :  
<https://github.com/afourmy/flask-gentelella>  
Consulté le : 27/02/2019
- [8] Documentation Python :  
<https://dev.to>  
Consulté le : 01/02/2019
- [9] Documentation du Module Textblob :  
<https://textblob.readthedocs.io/en/dev>  
Consulté le : 10/03/2019
- [10] Documentation Analyse de tweets :  
<https://datascienceplus.com>  
Consulté le : 01/02/2019
- [11] Documentation visualisation D3.js :  
<https://d3js.org>  
Consulté le : 24/01/2019
- [12] Documentation SunBurst faite par Kerryrodden :  
<https://bl.ocks.org/kerryrodden>  
Consulté le : 02/04/2019
- [13] Documentation implémentation Leaflet :  
<https://www.supinfo.com>  
Consulté le : 14/02/2019

## 9 Annexes

### A Présentation affichage des sessions

# Mes sessions

NOUVELLE SESSION +

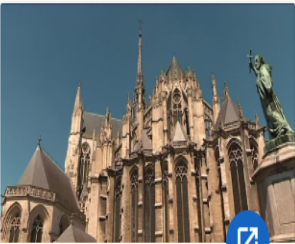


## Paris

**Nombre de tweets :** 300  
**Mot(s) clé(s) :** Paris  
**Type de sessions :** dated\_tweets  
**Dernière modification le**  
le 09/05/19 à 21:36

**ID de la session :**  
5cd481098356692c182e2921  
**Nombres de tweets à récupérer**  
: 300  
**Date de début :** le 09/05/19  
**Heure de début :** à 00:00  
**Heure de fin :** à 00:00  
[Supprimer](#)

☐ Cochez deux sessions




## Cathédrale

**Nombre de tweets :** 500  
**Mot(s) clé(s) :** Cathédrale  
**Type de sessions :** dated\_tweets  
**Dernière modification le**  
le 09/05/19 à 21:35

**ID de la session :**  
5cd480b58356692c182e272c  
**Nombres de tweets à récupérer**  
: 500  
**Date de début :** le 09/05/19  
**Heure de début :** à 00:00  
**Heure de fin :** à 00:00  
[Supprimer](#)

☐ Cochez deux sessions




## Gilets Jaunes

**Nombre de tweets :** 2000  
**Mot(s) clé(s) :** Gilets Jaunes  
**Type de sessions :** dated\_tweets  
**Dernière modification le**  
le 09/05/19 à 21:22

**ID de la session :**  
5cd47b7f8356692c182e1d65  
**Nombres de tweets à récupérer**  
: 2000  
**Date de début :** le 09/05/19  
**Heure de début :** à 00:00  
**Heure de fin :** à 00:00  
[Supprimer](#)

☐ Cochez deux sessions



## France

**Nombre de tweets :** 455  
**Mot(s) clé(s) :** France  
**Type de sessions :** stream  
**Dernière modification le**  
le 09/05/19 à 21:11

**ID de la session :**  
5cd47aec8356692c182e1b9d  
**Date de début :** le 09/05/19  
[Supprimer](#)

☐ Cochez deux sessions

FIGURE 25 – Interface des sessions

## B Fonction de stockage de tweets

```
def stock_tweets(tweet, stream):  
    if stream:  
        tweets_table = mongo.db.tweets  
        data = tweet_json  
        tweet_text = getText(data)  
        tweet_text = clean_text(tweet_text)  
        words = tweet_text.split(" ")  
        stop_words = get_stop_words('fr') + get_stop_words('en')  
        words = [word for word in words if word not in stop_words and len(word) > 2]  
        field = {'split': words}  
        data.update(field)  
        tweets_table.insert({'session_id': session['last_session'], 'tweet_object': data})  
    else:  
        tweets_table = mongo.db.tweets  
        data = tweet  
        tweet_text = getText(data)  
        tweet_text = clean_text(tweet_text)  
        words = tweet_text.split(" ")  
        stop_words = get_stop_words('fr')  
        words = [word for word in words if word not in stop_words and len(word) > 2]  
        field = {'split': words}  
        data.update(field)  
        if tweets_by_session_id(session['last_session']).count() > 0:  
            for tw in tweets_by_session_id(session['last_session']):  
                if tweet['id_str'] == tw['tweet_object']['id_str']:  
                    return 0  
            tweets_table.insert({'session_id': session['last_session'], 'tweet_object': data})  
            return 1  
        else:  
            tweets_table.insert({'session_id': session['last_session'], 'tweet_object': data})  
            return 1
```

FIGURE 26 – Fonction stockage tweets

## C Fonction séparatrice de mots

```
def word_splitter(words):  
    new_words = []  
    word_counter = collections.Counter(words)  
    total_word = len(word_counter)  
    counter_display_word = round(0.2 * total_word)  
    word_counter = word_counter.most_common(counter_display_word)  
    for word in range(len(word_counter)):  
        if word_counter[word][1] > 2:  
            if word != '':  
                new_words.append({'text': word_counter[word][0], 'size': word_counter[word][1]})  
    return new_words
```

FIGURE 27 – Fonction word\_splitter

## D Fonction récupération des textes de tweets

```
def getText(data):  
    # Try for extended text of original tweet, if RT'd (streamer)  
    try: text = data['retweeted_status']['extended_tweet']['full_text']  
    except:  
        # Try for extended text of an original tweet, if RT'd (REST API)  
        try: text = data['retweeted_status']['full_text']  
        except:  
            # Try for extended text of an original tweet (streamer)  
            try: text = data['extended_tweet']['full_text']  
            except:  
                # Try for extended text of an original tweet (REST API)  
                try: text = data['full_text']  
                except:  
                    # Try for basic text of original tweet if RT'd  
                    try: text = data['retweeted_status']['text']  
                    except:  
                        # Try for basic text of an original tweet  
                        try: text = data['text']  
                        except:  
                            # Nothing left to check for  
                            text = ''  
    return text
```

FIGURE 28 – Fonction getText tweets

## E Exemple Compte rendu réalisé

### Compte rendu numéro 3

Projet : Tweetostats

Membres du groupe :

- Nabil BOUILLIN
- Adrien DIDIER
- Mateo MEYNIER
- Aurélien TROUCHE

Date/Lieu : 20/02/2019 | Universités UM2 Bâtiment informatique

Points abordés :

- Visualisation cartographique avec la géolocalisation
- Mots de passes, cryptage grâce à un chiffrement asymétrique
- Serveur mLab, rapatrier la base de donnée en local pour améliorer l'expérience utilisateur.
- Correction de bugs liés au formulaire (sélection de la langue).

Décisions :

- Enrichir la visualisation des données grâce à un sunburst, stream graph, stack graphe.
- Chercher d'autre visualisation possible et utile pour le projet
- Pouvoir afficher et fermer les visualisation que l'on souhaite
- Prévoir une fonctionnalité de comparaison entre plusieurs recherche avec des graphes adaptés pour la comparaison.
- Essayer d'enrichir le sunburst avec un premier niveau qui contiendra pour le moment, le nombre de tweet, de retweet, le nombre de tweet qui contiennent une URL et/ou une géolocalisation.

Lien du github: [https://github.com/AdrienL3/TER\\_Twitter](https://github.com/AdrienL3/TER_Twitter)

FIGURE 29 – Compte Rendu