Projet TER-L3 : Détection d'opinions Twitter

Membres du groupe :

Cheimae ASSARAR

Nihed BENDAHMAN

Maroua Dorsaf DJELOUAT

Encadrant : Pascal Poncelet

Faculté des Sciences

Université de Montpellier

Année universitaire : 2018-2019

Table des matières

Remerciements

Nous tenons à remercier notre encadrant, Monsieur Pascal Poncelet, pour son aide et son accompagnement tout au long de ce projet car, si nous avons pu atteindre l'objectif fixé pour ce dernier, c'est surtout grâce à lui et à ses judicieux conseils.

Chapitre 1

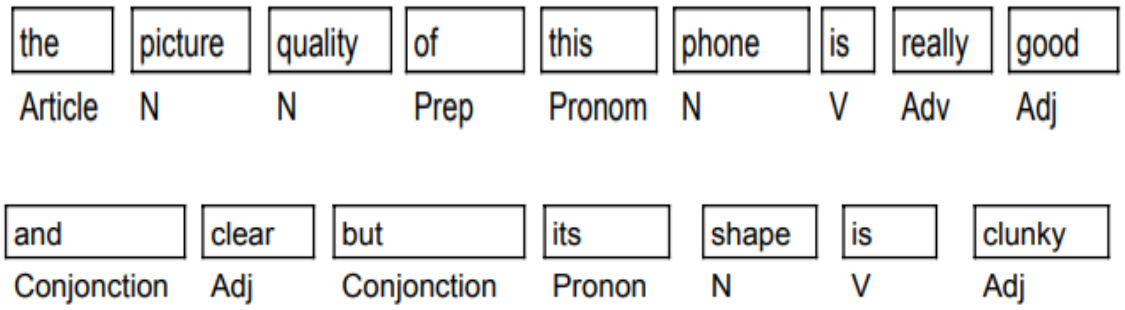
Introduction

1.1 Contexte général

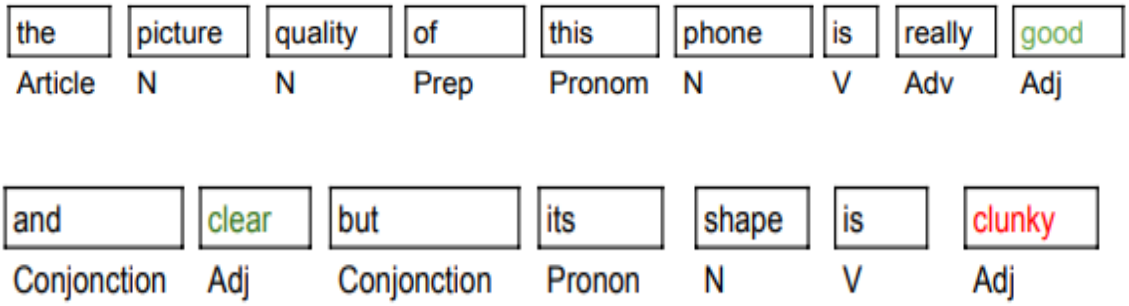
Avec le développement du web et la croissance explosive des réseaux sociaux, l'expression d'opinions sur Internet devient de plus en plus importante : Facebook, Twitter, Forums de discussions, blogs de commentaires, etc. Tous forment le socle d'une fabrique de point de vue. Le grand public donne en direct sa vision des grands sujets de la société. Mais Comment traiter le matériau infiniment massif et varié des idées véhiculées sur la sphère numérique ? [1]

Cette question a poussé les chercheurs de différentes communautés (Fouille de données, Fouille de textes, Linguistique) à s'intéresser de plus près à l'identification et à l'extraction automatique des données d'opinions. Plusieurs méthodes d'analyse ont été découvertes : arbres décisionnels, algortihmes génériques, ... L'une des approches les plus pertinentes consiste à utiliser un ensemble d'adjectifs positifs et négatifs pour au final qualifier un document, et c'est précisemment dans ce cadre que s'inscrit notre TER.[2]

Pour comprendre comment extraire une opinion en se basant uniquement sur des dictionnaires d'adjectifs, prenons un exemple : soit la phrase **"the picture quality of this phone is really good and clear but its shape is clunky"**. Pour classer ce document, nous devons d'abord étiqueter chaque mot de la phrase en fonction de son rôle grammatical à l'aide d'un analyseur morphosyntaxique. En prenant notre exemple, nous obtenons :



Maintenant, pour savoir si notre texte exprime une opinion positive ou négative, nous faisons appel à deux dictionnaires : l'un contenant des adjectifs positifs et l'autre des adjectifs négatifs. Ainsi, nous comparons les adjectifs présents dans notre phrase à ceux présents dans lesdits dictionnaires et déterminons - en fonction du nombre d'occurences de chaque type d'adjectifs - Si le texte est positif ou pas. Dans notre exemple, le nombre d'ajdectifs positifs (en vert) est supérieur au nombre d'adjectifs négatifs (en rouge), et donc l'opinion exprimée est positive :



Cette approche est pratique lorsque les documents sont très généraux mais parfois les termes utilisés dans des domaines spécifiques ne sont pas forcément pertinents. Considérons, par exemple, les deux phrases : **"this is a commercial movie"** et **"this is a commercial product"**. Dans le cas de la première phrase (une opinion exprimée sur un film), l'adjectif "commercial" est négatif. Par contre, dans la seconde phrase (une opinion sur un produit commercial), l'adjectif est plutôt avantageux.

Notre travail consiste à créer, à l'aide de tweets, une approche d'apprentissage automatique d'adjectifs positifs et négatifs pour un domaine donné.

1.2 Objectif du travail

L'objectif de notre projet est de coder un programme mettant en place un système qui permet d'interroger Twitter et d'analyser ses documents afin de définir deux dictionnaires d'adjectifs positifs et négatifs pour un domaine particulier ; Le domaine en question sera saisi via une plateforme web developpée par nos soins.

Afin d'être plus efficaces nous nous sommes mises d'accord, et ce dès notre première réunion avec notre encadrant, pour découper le travail à réaliser en trois parties :

- ✔ La récupération des Tweets.
- ✔ Le traitement des Tweets.
- ✔ La réalisation de l'interface graphique.

Dans ce qui va suivre, nous présenterons en premier les grandes parties de notre travail - à travers un schéma explicatif - ainsi que les méthodes et outils que nous avons adoptés afin d'organiser ce projet. Après quoi, nous exposerons plus en détails le developpement de notre programme. Pour finir, nous ferons un point sur les difficultés que nous avons rencontrées et les perspectives concernant notre application.

Chapitre 2

Conception

2.1 Présentation de l'architecture générale

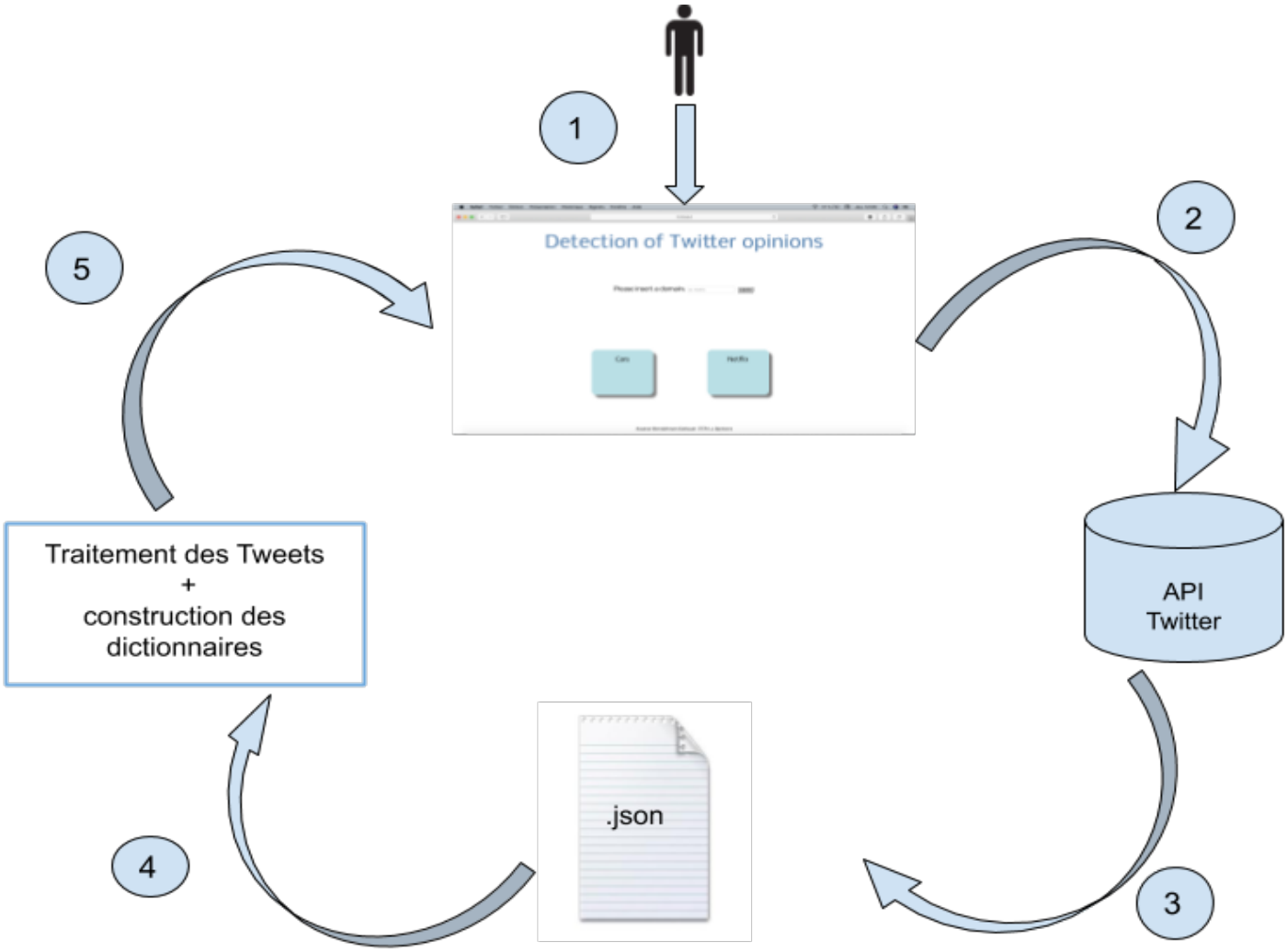


FIGURE 2.1 – Schéma général du travail effectué.

Pour avoir une visions globale de notre travail, nous avons élaboré le schéma ci-dessus et qui se résume de la manière suivante :

D'abord, l'utilisateur accède à la platforme web "Detection of Twitter opinions" et saisit le domaine sur lequel il veut obtenir des dictionnaires (1). Ce domaine est ensuite récupéré par un programme qui lance des requêtes sur l'API Twitter qui, à son tour, retourne des données concernant les tweets (date de publication, le texte du message, etc.), l'auteur (date de création du compte, pseudo...), les entités contenues dans les messages (hashtags, mentions, urls...) et des informations de localisation (pays, timezone, etc) (2) ; Le stockage de ces données se fait dans un fichier "tweets.json" (3).

Après quoi vient le traitement des tweets lus dans le fichier "tweets.json". Ce traitement comprend le nettoyage des tweets, mais aussi l'extraction ainsi que le filtrage des adjectifs contenus dans les tweets. À la fin, nous nous retrouvons avec deux dictionnaires : l'un contenant des adjectifs positifs et l'autre des adjectifs négatifs (ce processus sera expliqué plus en détail dans la partie "développement du projet") (4).

Une fois le traitement terminé, nous affichons les résultats (les dictionnaires d'adjectifs ainsi que quelques exemples de tweets récupérés) sur la page web (5).

Chapitre 3

Organisation du projet

3.1 Organisation du travail

Pour l'organisation du travail, nous avons privilégié le travail de groupe. Ainsi, nous nous retrouvions plusieurs fois par semaine (en particulier les après-midis) pour des séances de travail communes.

Tout au long du projet, nous effectuions des réunions régulières avec M. Poncelet afin de faire le point sur l'avancement de notre projet. Ces réunions nous ont permis de bénéficier de ses conseils et de son aide pour résoudre les difficultés que nous avons rencontrées lors du développement. Suite à chaque réunion, nous rédigions un compte-rendu afin d'établir un bilan sur notre avancement et de résumer les points abordés.

3.2 Outils de travail collaboratif

Pour collaborer avec l'ensemble du groupe, nous avons ouvert un dépôt git commun du serveur "GitLab UM" hébergé par le service Informatique de la Faculté des Sciences de Montpellier (SIF). Ce depôt nous a été d'une très grande utilité quant à la gestion des versions de code et le travail collaboratif simultané.

Pour ce qui est de la communication, nous avons créé un groupe sur Télégram, une application de messagerie sécurisée hébérgée sur le cloud [3] qui nous a permis de communiquer les informations importantes concernant le projet et de discuter de nos disponibilités pour les séances de travail en commun.

3.3 Répartition du travail dans le temps

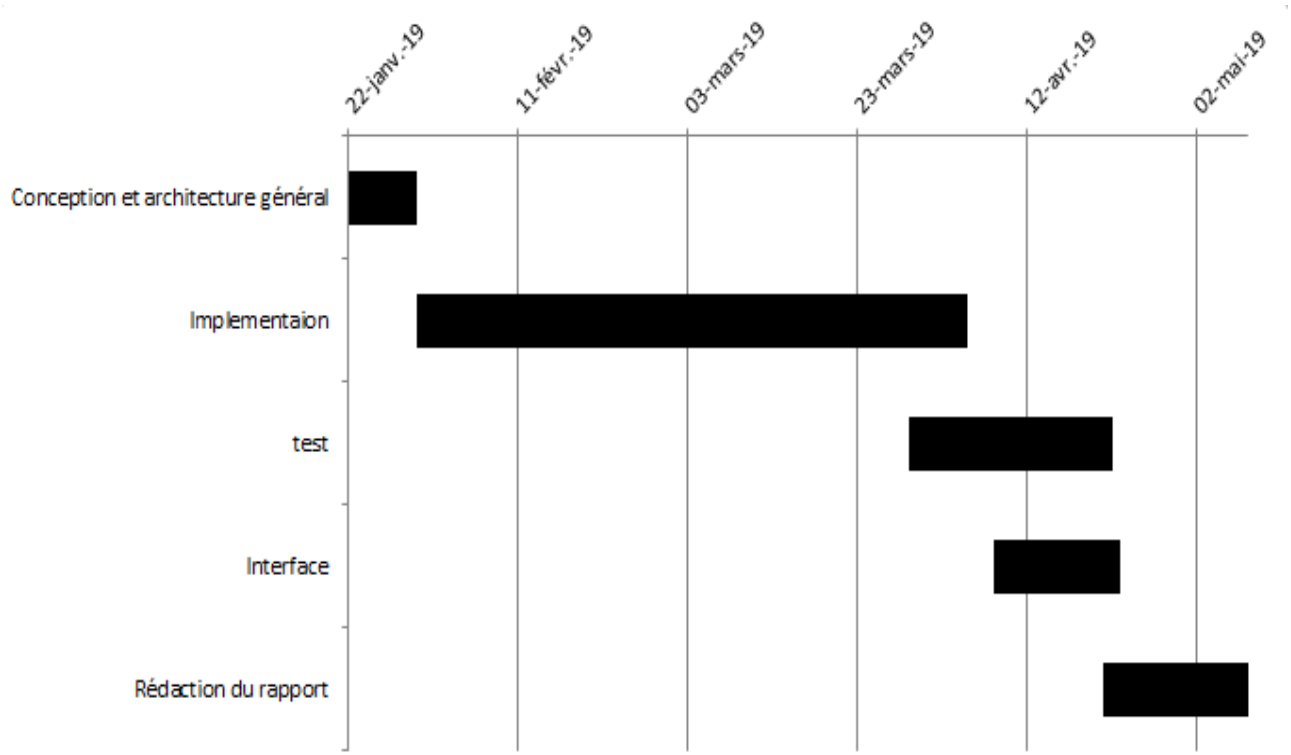


FIGURE 3.1 – Diagramme de Gantt.

Chapitre 4

Développement du travail

4.1 Première étape : Traitement des données

Disclaimer : l'integralité de ce projet a été codé en Python, conformément à ce qui a été demandé par M. Poncelet. Pour ce qui de l'analyseur morphosyntaxique, nous avons utilisé la fonction "pos_tag()" de la librairie Python "nltk", également sous les conseils de notre encadrant.

Comme cela a été expliqué un peu plus haut dans ce rapport (cf. "Objectif du travail"), le développement de notre application a été découpé en trois étapes principales. Notre encadrant nous a conseillé de commencer par la deuxième étape, à savoir le traitement des tweets et la création des dictionnaires d'adjectifs.

Pour éviter les répétitions, nous n'allons expliquer que l'élaboration du dictionnaire d'adjectifs positifs (la construction du dictionnaire d'adjectifs négatifs se fait de manière analogue).

Nous avons commencé par créer deux tableaux "Tpositif" et "Tnegatif" (que nous avons initialisé avec des adjectifs d'opinion positifs (resp. négatifs) généraux), ainsi que deux dictionnaires "dicoPositif" et "dicoNegatif" qui ont servi comme structures intermédiaires et dans lesquelles nous avons stocké par la suite les adjectifs que nous récupérions au fur et à mesure. Les dictionnaires ont comme clés les adjectifs présents dans les tableaux "Tpositif" et "Tnegatif", et comme valeurs des tableaux dans lesquels sont stockés tous les nouveaux adjectifs (ces tableaux sont donc vides au lancement du programme).

Tpositif = { *good, nice, great, wonderful, best, amazing, funny, awesome* }
Tnegatif= { *bad, wrong, horrible, sad, terrible, boring, poor, negative* }

dicoPositif = { *good :[], nice :[], great :[], wonderful :[], best :[], amazing :[], funny :[], awesome :[]* }
dicoNegatif= { *bad :[], wrong :[], horrible :[], sad :[], terrible :[], boring :[], poor :[], negative :[]* }

À noter que les résultats finaux de notre travail sont stockés dans "Tpositif" et "Tnegatif".

Comme nous n'avons pas encore récupéré de véritables tweets à ce moment là, nous avons d'abord tester notre programme sur des phrases que nous avons nous même construites. Aussi, nous avons débuté par le "nettoyage" des chaines de caractères et qui consiste à enlever les émojis, les URL, les hashtags, et autres caractères spéciaux à l'aide de fonctions spécialement conçues pour cela.

Après quoi, nous avons implémenté une fonction "remplissage" qui parcourt chaque chaîne de caractères (tweet) à la recherche d'adjectifs et remplit le tableau "dicoPositif" avec ce qu'elle trouve.

Cette fonction commence tout d'abord par découper la phrase en mots. Ensuite, elle applique l'analyseur morphosyntaxique sur cette phrase pour n'en garder au final que les adjectifs. Pour chacun des adjectifs gardés, la fonction vérifie si il est stocké ou non dans "Tpositif" ; Le cas cohérent, l'ensemble des adjectifs présents dans la phrase est stocké dans dicoPositif selon la clé qui correspond à l'adjectif présent dans le tableau "Tpositif".

Pour mieux comprendre le principe de "remplissage", prenons comme exemple la phrase :

"This dress is very nice, it makes me feel pretty and it's so comfortable."

Après l'application de l'analyseur morphosyntaxique sur cette phrase, nous nous retrouvons avec trois adjectifs, à savoir : *nice, pretty* et *comfortable*. Comme l'adjectif *nice* apparaît dans "Tpositif", la clé "nice" dans "dicopositif" se vera attribuer les adjectifs "pretty" et "comfortable" :

dicoPositif = { *good :[], nice :[pretty, comfortable], ...* }

Si un adjectif est précédé par une négation (no, not, neither...nor) nous concaténons le mot exprimant la négation et l'adjectif et nous les stockons dans "dicoPositif". Nous avons aussi traité de la même manière les adjectifs précédés par des adverbes tels que *so, really, very*, etc. Ainsi dans notre exemple précédent, nous obtiendrons plutôt :

dicoPositif = { *good :[], nice :[pretty, **so** comfortable], ...* }

Une fois toutes les chaînes à notre disposition traîtées de cette manière, nous procédons au "filtrage" des adjectifs recupérés. Nous parcourons le tableau d'adjectifs associé à chaque clé de "dicopositif" et vérifions, pour chaque adjectif associé à cette clé :

- ➜ S'il apparaît avec au moins cinq autres adjectifs.
- ➜ Si son nombre d'occurrences total est supérieur à dix.
- ➜ S'il n'apparaît pas déjà dans "Tpositif".

Si l'adjectif vérifie ces trois conditions, il est rajouté à "dicoPositif"(comme clé) mais aussi à "Tpositif" ; sinon, nous le supprimons.

Une fois l'implémentation de ces différentes fonctions terminée, nous avons procédé à la récupération des tweets depuis l'API Twitter (cette partie sera expliqué plus en détails à la suite de ce rapport) grâce à une fonction "récupérationTweets" et, une fois les tweets en notre possession, nous avons encapsulé l'ensemble des fonctions "récupérationTweets, remplissage et filtrage" dans une fonction récursive, elle même s'arrêtant quand il n'y a plus de nouveaux adjectifs découverts. Cette fonction récursive est pour ainsi dire le "main" de notre programme.

En dernier, nous avons mis en place un filtre final qui sert à vérifier si un adjectif n'appartient qu'à un et un seul tableau seulement (soit "Tpositif", soit "Tnegatif"). Si il est présent dans les deux tableaux, alors nous comptons son nombre d'occurences : s'il apparaît dans un tableau plus que dans un autre, nous le conservons dans le tableau où son nombre d'occurences est le plus élevé. Si par contre la différence du nombre d'occurences entre les deux tableaux est moindre (<15) nous le supprimons et de "Tpositif" et de "Tnegatif".

Voici, pour résumer, un schéma explicatif :

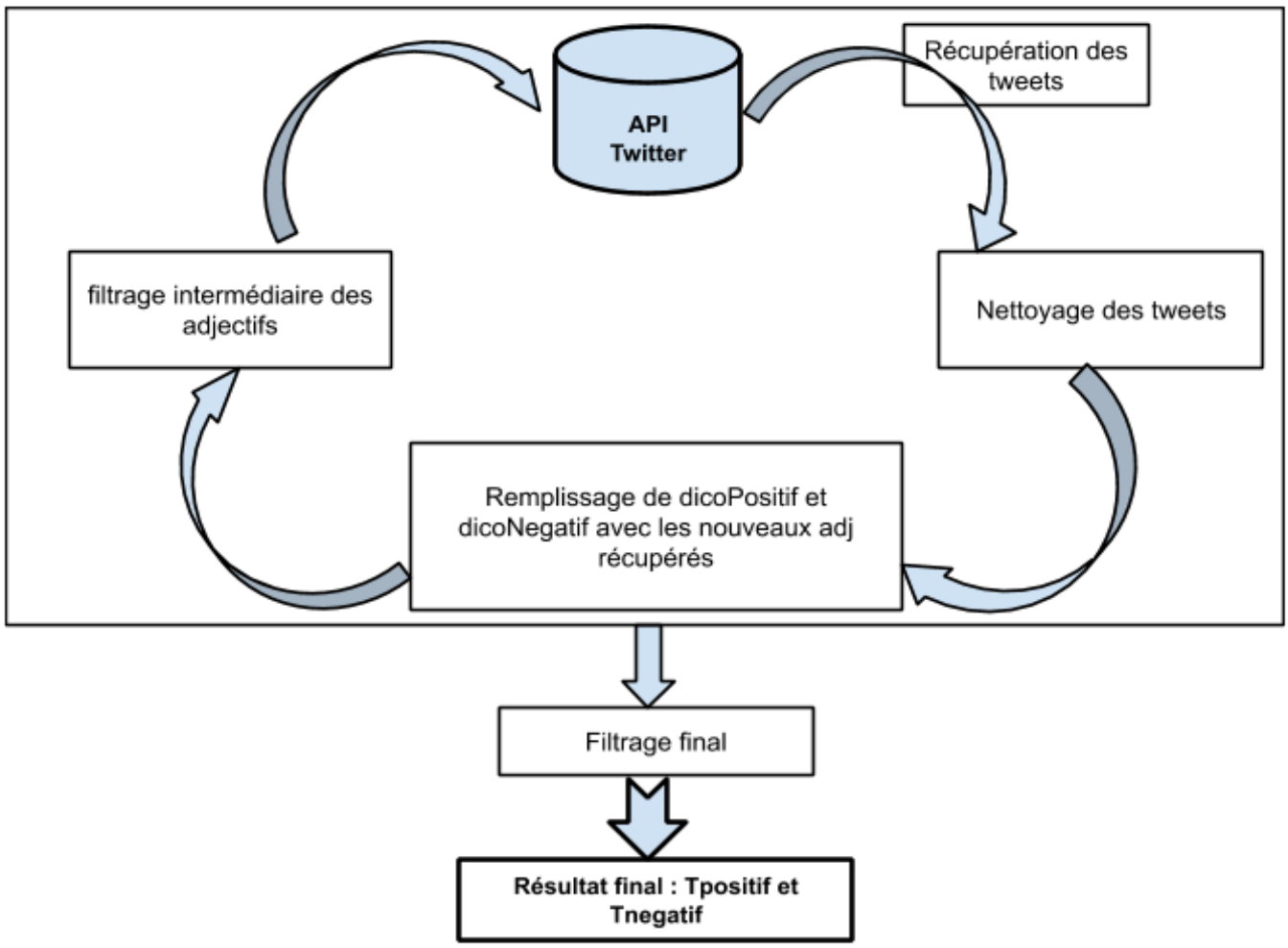


FIGURE 4.1 – Schéma résumant le déroulement de notre programme.

Après avoir obtenu les deux tableaux d'adjectifs définitifs (à savoir "Tpositif" et "Tnegatif"), nous avons procédé à la détection d'opinions dans nos phrases. Pour ce faire, nous avons implémenté une fonction "opinionTweet" qui prend les deux tableaux d'adjectifs et une chaîne de caractères et qui, de cette chaîne, extrait les adjectifs et attribue un poids à chacun d'entre eux :

- ➜ Si l'adjectif figure dans le tableau "Tpositif", un poids de "2" lui est attribué.
- ➜ Si l'adjectif figure dans "Tpositif" mais qu'il est précédé par une négation, son poids sera de "-1".
- ➜ Si l'adjectif est dans "Tpositif" et est précédé par un adverbe, son poids sera de "3".

Soit comme exemple la phrase :

"This book is so good ! It is full of plot twists which makes it interesting and not repetitive at all."

À supposer que l'ensemble des adjectifs présents dans la phrase, à savoir : *so good, full, interesting, not repetitive* soit présent dans le tableau "Tpositif". Nous obtenons, pour chacun de ces adjectifs, les poids suivants :

{ *so good : 3, full : 2, interesting : 2, not repetitive : -1* }

En faisant la somme de tous ses poids nous obtenons : 3 + 2 + 2 -1 = 6 > 0, et donc l'opinion exprimée par notre phrase est positive.

4.2 Deuxième étape : récupération des tweets

La première chose à réaliser fut la création d'une application Twitter, afin d'obtenir des identifiants permettant de se connecter aux APIs de ce dernier, et qui se décomposent en quatre classes [4] :

- ✔ La classe REST : qui permet d'accéder à des fonctionnalités avancées de Twitter (chercher des utilisateurs, des followers, voir les statuts, éditer des informations sur son compte, etc).
- ✔ La classe WEBSITES : autorise l'intégration des fonctions de base de Twitter dans des sites webs.
- ✔ La classe SEARCH : permet d'interroger Twitter pour récupérer des données simples, essentiellement des tweets. La particularité de cette API avancée est de permettre l'accès à de gros volume de données Twitter et d'être moins contrainte par les limites d'accès et d'interrogrations de twitter.
- ✔ La classe STREAMING : permet de communiquer avec Twitter en mode streaming.

Une fois notre application déclarée, nous nous retrouvions avec une clé d'accès à l'API et un jeton d'accès. Il s'agit respectivement des champs "consumer key / consumer secret" et "access token / access token secret" ; Ces clés ont permis à notre application de s'authentifier et d'emmettre des requêtes sur les adjectifs positifs et négatifs [5].

Au début, nous avons manipulé essentiellement la classe STREAMING en réalisant un programme Python qui interroge Twitter avec Tweepy - une librairie Python open-source, hébergée sur GitHub. Nous avons donc commencé par créer une classe héritant de "StreamListener" ainsi qu'un objet Stream (dans Tweepy, une instance de tweepy.Stream établit une session de diffusion en continu et route les messages vers l' instance StreamListener) et avons établi la connexion à Twitter à l'aide de ce dernier.

Pour ce qui est du stockage des resultats, nous avons fait en sorte de n'extraire que le champs "texte" de chaque tweets ; ces extraits ont été enregistrés dans un fichier ".JSON" que nous avons parcouru et traité en utilisant le programme expliqué un peu plus haut dans ce rapport (cf. partie "traitement des données").

Toutefois, nous nous sommes très vite apperçue que notre code ne donnait pas le résultat escompté. En effet, la classe STREAMING s'est avérée contrainte par des limites d'accès et d'interrogation de Twitter ; nous ne pouvions recupérer que très peu de tweets à la fois. Voilà pourquoi, après plusieurs recherches sur le sujet, nous avons fini par élaborer un script utilisant la classe SEARCH, qui était beaucoup plus malléable que ça prédécessrice.

L'interrogation de l'API Twitter grâce à SEARCH s'est faite avec des requêtes du style :

nomDomaine adjectif -RT

Où "-RT" est une option qui empêche de récupérer les retweets.

Au préalable et afin de diminuer les chances de nous retrouver avec des phrases qui n'ont pas de sens, nous devions vérifier que l'adjectif sur lequel nous lancions une requête ne soit pas égal au domaine (afin d'éviter les répétitions inutiles).

4.3 Troisième étape : création de l'interface

En ce qui concerne l'interface graphique, nous nous sommes aidées du framework Flask qui est un framework open-source de développement web en Python.

Notre travail étant plus focalisé sur la fouille de données et l'interface web n'étant qu'un moyen pour récupérer le domaine entré par l'utilisateur, nous avons opté pour une plateforme plutôt sobre et codée uniquement en HTML et CSS. Ainsi, nous avons comme page d'accueil de notre application :

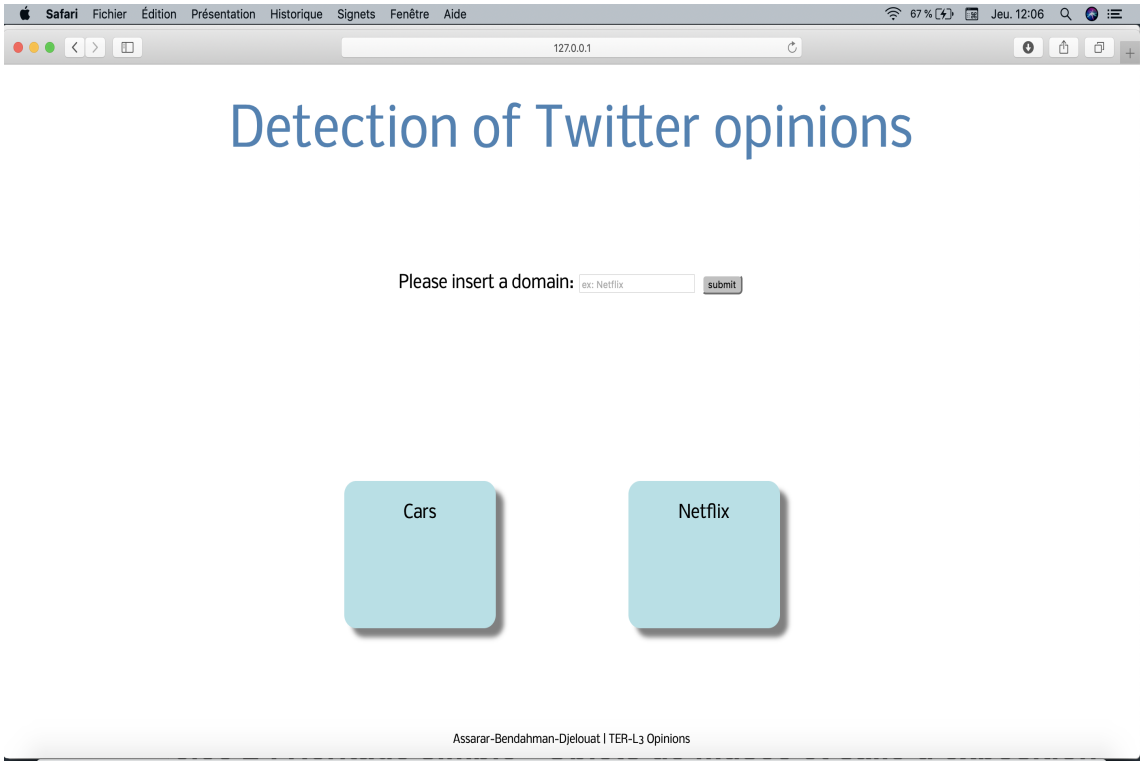


FIGURE 4.2 – Page d'acceuil de notre plateforme web.

Nous avons également fait en sorte que l'utilisateur puisse consulter les dictionnaires et les tweets de domaines déjà traités ("Cars" et "Netflix" sur la capture précédente).

Pour ce qui est de la page des résultats, elle devrait s'afficher comme suit :

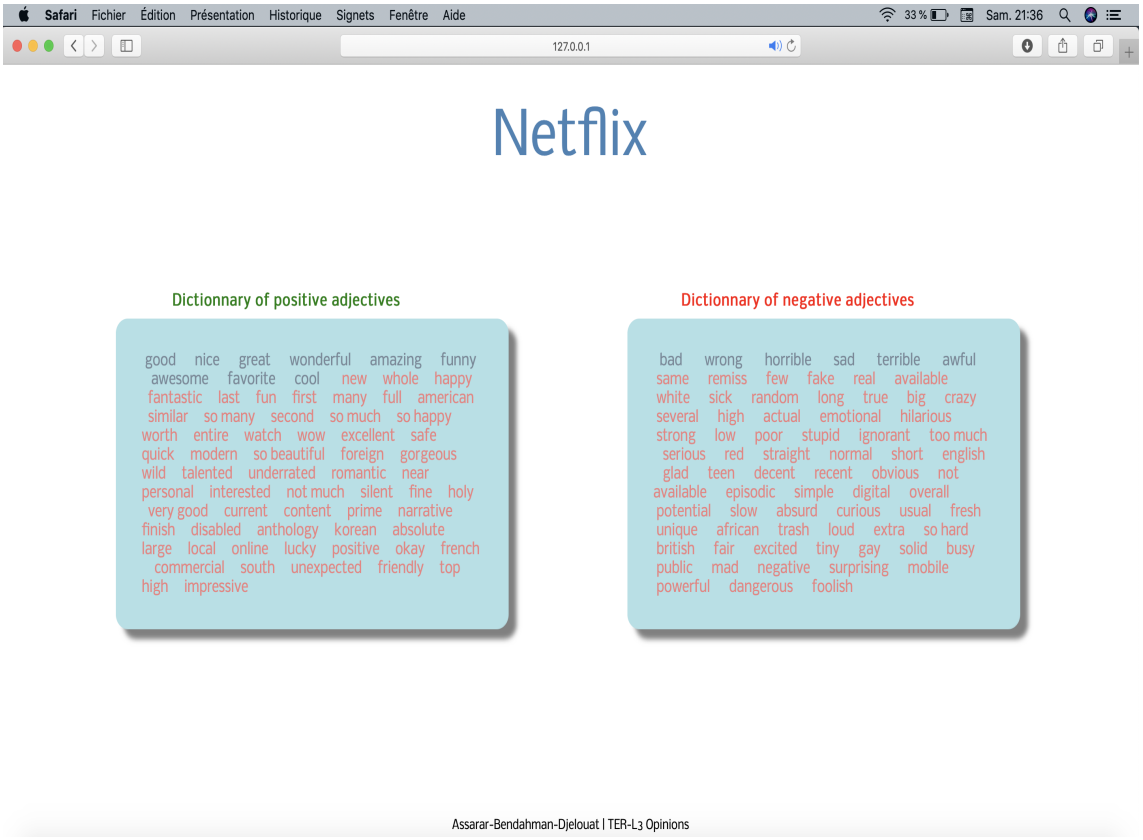


FIGURE 4.3 – Affichage des deux dictionnaires d'adjectifs positifs et négatifs.

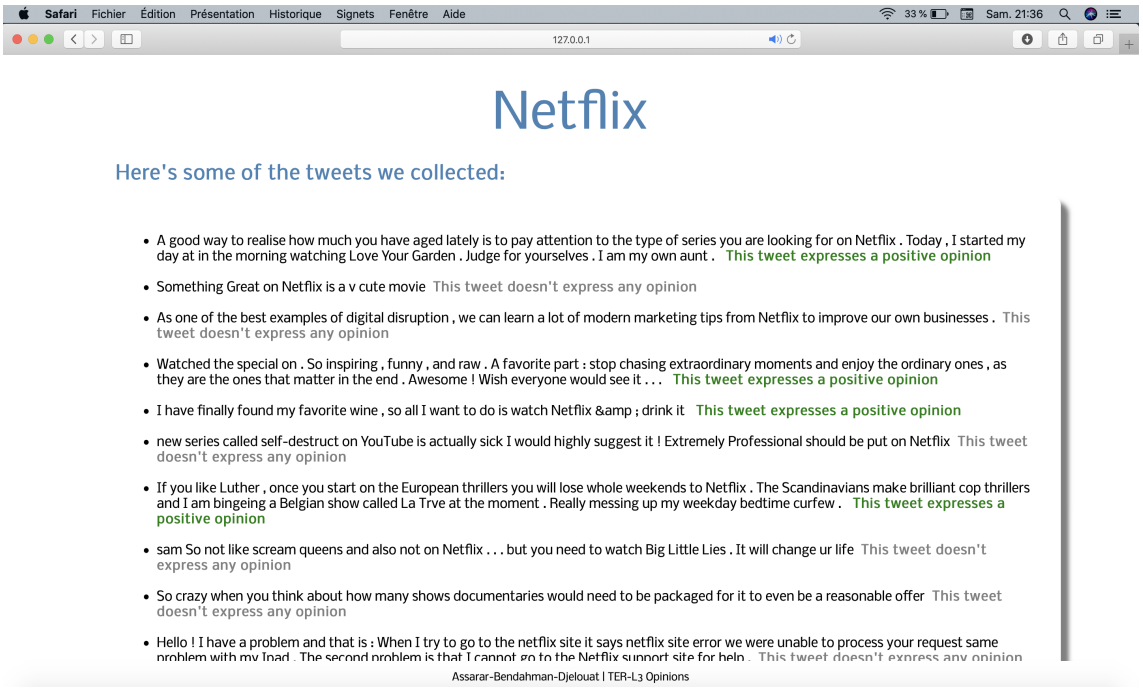


FIGURE 4.4 – Affichage de quelques exemples de tweets.

Nous avons d'abord les dictionnaires qui s'affichent, puis un échantillon de tweets un peu plus bas dans la page, avec, pour chaque tweet, un commentaire disant si la phrase exprime une opinion positive/négative ou bien si elle n'exprime aucune opinion.

Chapitre 5

Conclusion

5.1 Difficultés rencontrées

La difficulté majeure que nous avons rencontré pendant le développement de notre projet fut la manipulation des données récupérées depuis l'API Twitter. En effet, cette API retourne plusieurs dizaines de champs avec plusieurs options paramétrables pour chaque tweet, or ce qui nous intéressait fut seulement le champs texte avec l'option "tronquée" à faux. Il nous a donc fallu faire des recherches pour comprendre la signification des options et savoir comment extraire seulement ce que nous voulions.

5.2 Perspectives

L'application que nous avons développé consiste à analyser et traiter le langage humain, or le discours humain n'est pas toujours précis, il est souvent ambigu et sa structure linguistique peut dépendre d'un grand nombre de variables complexes, notamment largo, les dialectes régionaux et le contexte social[6]. Par conséquence l'algorithme peut parfois se tromper dans la qualification d'un tweet et le considérer comme un tweet d'opinion.

Une des perspectives que nous avons sur notre projet est d'ajouter une option de confirmation du résultat du traitement par l'utilisateur. Une autre perspective consiste à affiner encore le traitement des dictionnaires pour éliminer les mots qui ne sont pas des adjectifs et qui figurent encore dans nos dictionnaires (les adjectifs de couleurs, certains verbes...).

5.3 Apports personnels

Ce projet nous a permis de consolider nos connaissances sur le langage Python et a surtout été un moyen de découvrir un peu plus le domaine de l'intelligence artificielle ; domaine qui nous intrigue et nous fascine depuis un moment déjà. Nous sommes pour ainsi dire amplement satisfaites d'avoir choisi ce sujet.

Au niveau de la gestion du travail en équipe, nous avons réussi à bien nous répartir les tâches afin de réaliser nos objectifs dans les temps et l'ambiance générale du groupe était très bonne. Une belle expérience à renouveler ! [7]

Annexe A

Manuel d'installation de l'application

A.1 CRÉATION DE L'APPLICATION TWITTER :

➢ Allez sur le site : https ://twitter.com/ ?lang=fr

➢ Si vous n'avez pas de compte, créez-en un ; sinon, identifiez-vous et accédez aux paramètres de votre compte.

➢ Une fois cela fait, cliquez sur l'onglet "applications" qui apparaît à gauche de l'écran.

➢ Définissez les données de votre application Twitter : saisissez le nom, la description aini que l'URL de votre site pour l'application.

➢ Générez vos clés en cliquant sur "Create my access token", en veillant à bien choisir le type d'accès à Twitter (lecture seulement, lecture et écriture...).

➢ Récupérez les 4 clés : Consumer Key, Consumer Secret, OAuth Access Token, OAuth Access Token Secret.

A.2 EXÉCUTION DU PROGRAMME :

➢ Allez au fichier "fonctionsAux.py" et remplissez les champs "Consumer Key, Consumer Secret, OAuth Access Token, OAuth Access Token Secret" (ligne 20 à 23) par les clés précédemment récupérées.

➢ Pensez à installer le "package installer" de Python "pip" si vous ne l'avez pas déjà, en tapant la commande : pip install -U pip

➢ Installez les packets :

- numpy.
- nltk.
- tweepy.
- contractions.

- unicode.
- flask.

Avec la commande : pip install <package>

➢ Exécutez le programme "ProgrammePrincipal.py".

➢ Copiez l'URL qui va s'afficher sur votre terminal dans la barre de recherche de votre navigateur. Une page "Detection of Twitter opinions" devrait s'afficher.

➢ Une fois le domaine saisi et après un moment, les dictionnaires d'adjectifs positifs/négatifs ainsi que quelques exemples de tweets devraient s'afficher. Si vous préférez voir des dictionnaires/tweets sur des domaines déjà traités, cliquez sur l'un des liens en bas de la page d'accueil ("Cars" ou "Netflix").

Bibliographie

[1] **Comment sont gouvernées nos opinions sur Internet :** https://www.lexpress.fr/culture/comment-sont-gouvernees-nos-opinions-sur-internet_2037029.html

[2] **Détection d'opinions : merci Tweeter ! :** https://moodle.umontpellier.fr/pluginfile.php/630234/mod_resource/content/0/TERL3_Tweet_Opinions.pdf

[3] **Définition de l'application Télégram :** https://fr.wikipedia.org/wiki/Telegram_(application)

[4] **TD/TP : API Twitter :** https://www.lifl.fr/~derbel/pje/api/

[5] **Tutoriel – Utiliser l'API Twitter pour collecter des tweets avec Talend (et sans coder !) :** http://www.erwanlenagard.com/general/tutoriel-utiliser-lapi-twitter-pour-collecter-des-tweets-sans-coder-avec-tale

[6] **Que signifie le traitement du langage naturel (TLN, ou NLP) :** https://www.lemagit.fr/definition/Traitement-du-langage-naturel-TLN

[7] **Rapport de projet long informatique :** http://www-soc.lip6.fr/~cecile/enseig/PL/projet_long_li_rousseau.pdf