Ingénierie des langues : “Temporal Twitter Summarizer”

Robin Moriniere 21606393
Mohamed Lamine Seck 21711412

Master 1 informatique année 2020 - 2021

I. Introduction

Au cours de notre master 1 d'informatique nous avons dû réaliser plusieurs projets, notamment un projet en ingénierie des langues. Ce projet avait pour but final de créer une interface web affichant les résultats obtenus, après avoir effectué un prétraitement sur un fichier Json (environ 8go), avec pour thème l'élection américaine de 2020 opposant Donald Trump à Joe Biden.

Pour réaliser ce projet nous avons dans un premier temps développé une application qui permettrait d'effectuer un traitement linguistique sur un fichier Json contenant l'ensemble des tweets recensés sur une période donnée (environ 43 millions de tweets). Ce traitement linguistique à pour but d'extraire tous les mots importants tout en supprimant les mots parasites (mots de liaisons, déterminants, etc…). Pour par la suite, compter le nombre d'occurrences de ces mots pour faire des statistiques sur les mots qui sont le plus ressortis.

Dans un second temps nous devions créer une une interface web pour pouvoir afficher les résultats obtenus sur le traitement et l'analyse des tweets. Pour cela nous avons choisi d'afficher les résultats sous forme de graphiques et de nuages de mots.

II. Architecture

En ce qui concerne l'architecture comme annoncé précédemment, nous avons choisi de diviser le projet en 2 sous-projets. L'un s'occupant de la partie traitement avec pour nom : App Traitement. C'est-à-dire l'analyse et le traitement du fichier d'entrée pour produire plusieurs fichiers de sortie contenant les résultats.

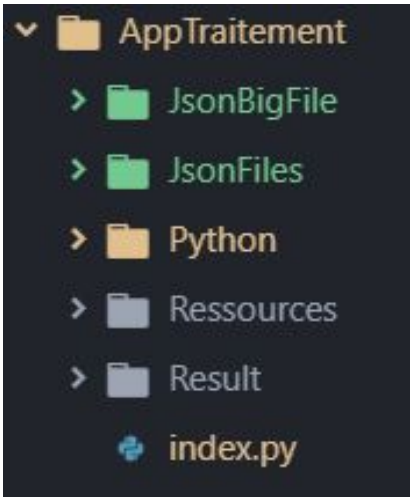
Quant à lui l'autre sous projet devait s'occuper de l'affichage et l'interprétation des résultats sur une page web. Nous avons nommé ce projet : Interface Web.

A. App Traitement

Le projet App Traitement est composé de plusieurs dossiers et fichiers. Dont le fichier index.py qui est le point d'entrée de l'application.

1. Architecture globale

JsonBigFile : Contient le fichier d'entrée (us_election20_tweet_pr.json).



JsonFiles : Contient les fragments du fichier d'entrée pour réduire la taille du fichier qui sera traité.

Python : Contient les classes pythons qui seront appelé par index.py

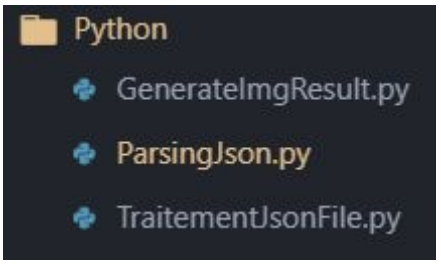
Ressources : Contient les images qui seront utilisées pour créer les graphiques et nuages de mots.

Result : Contient les résultats obtenus après traitement.

2. Dossier Python

Le dossier python contient 3 classes qui seront appelées par le fichier index.

TraitementJsonFile : va permettre de découper le fichier d'entrée en plusieurs fragments pour qu'il soit moins volumineux à traiter. Chacun de ces fragments sera placé dans le dossier **JsonFiles** avec pour nom : **JsonFile** (numéro de fragments) **.json.** Ils sont composés d'un dictionnaire python contenant un certain nombre de tweet définie à l'avance dans index.py.



ParsingJson : va contenir toutes les méthodes essentielles pour le traitement des tweets. C'est-à-dire la suppression des tweets qui ne concerne pas le sujet, la séparation des tweets en fonction qu'il soit sur Donald Trump ou Joe Biden. Mais aussi l'extraction des mots des importants avec la suppression des mots et symboles parasites. À l'issue de ce traitement on obtient deux tableaux contenant les mots les plus représentés parmi tous ces tweets.

GenerateImgResult : va se servir des tableaux obtenus après le traitement pour pouvoir créer des images et nuages de mots qui seront utilisés dans l'interface web pour pouvoir avoir un rendu visuel des résultats.

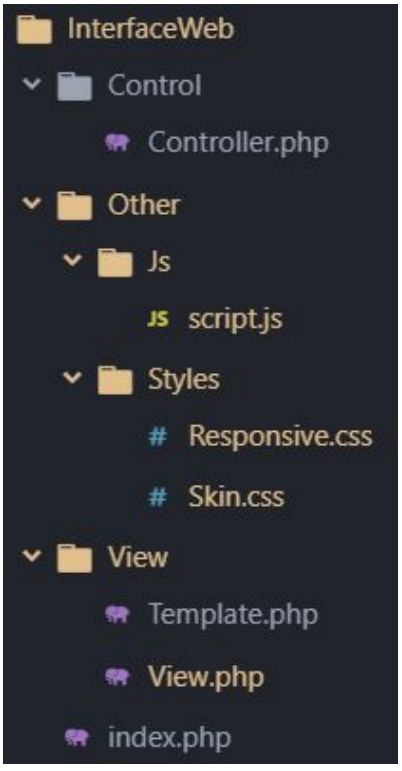
B. Interface Web

Pour ce qui est de l'architecture de l'interface web nous avons choisi de partir sur un modèle MVC simplifié car il n'y a qu'une seule page à afficher et aussi parce que nous avons préféré nous concentrer principalement sur la partie traitement qui représente le cœur du projet.

1. Architecture global

Le point d'entrée de l'application correspond au fichier index.php qui s'occupe d'appeler les classes dont il a besoin.

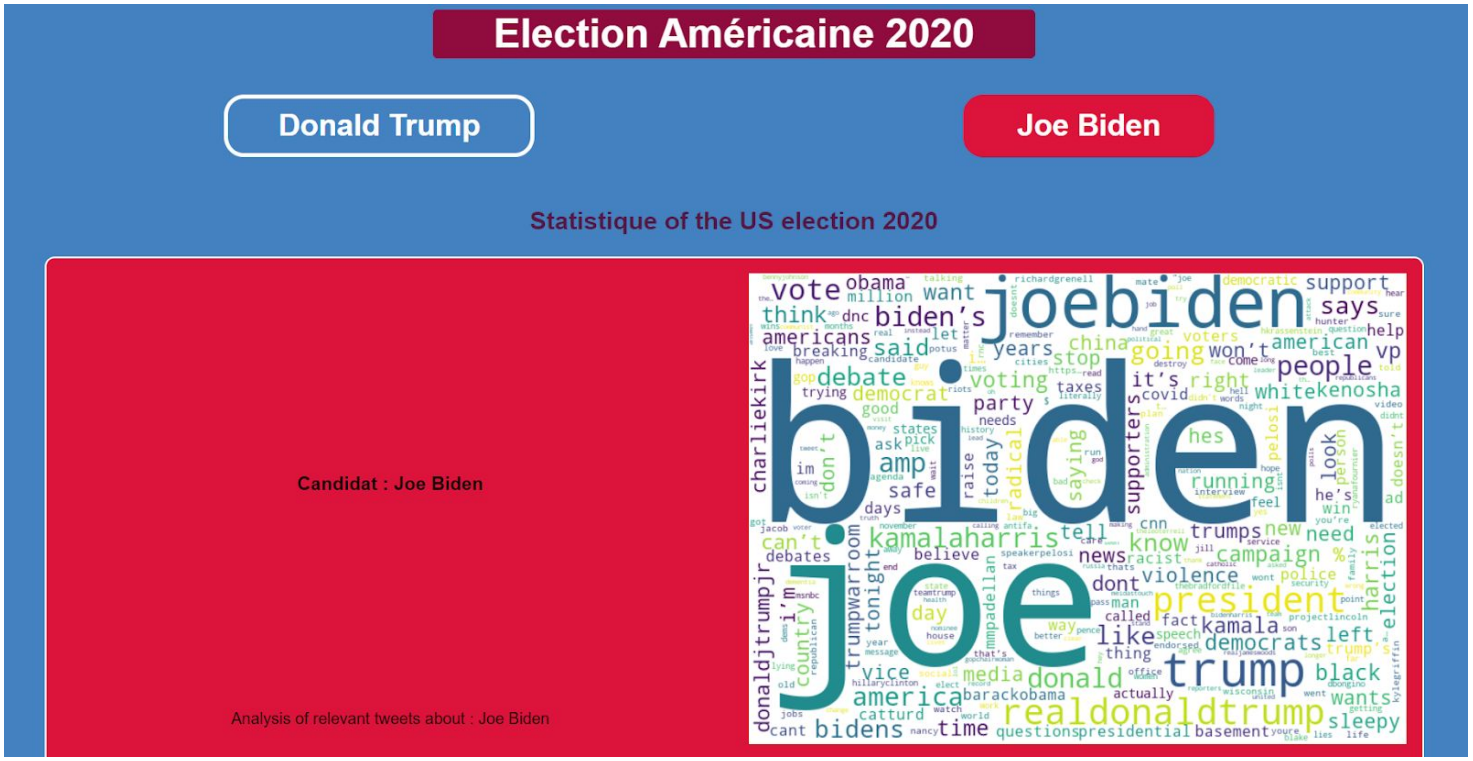
Control : Le dossier Control contient la classe Controller.php. C'est elle qui va se charger d'effectuer toutes les actions nécessaires pour récupérer les résultats obtenus dans l'autre projet, pour ensuite les transférer à la vue.

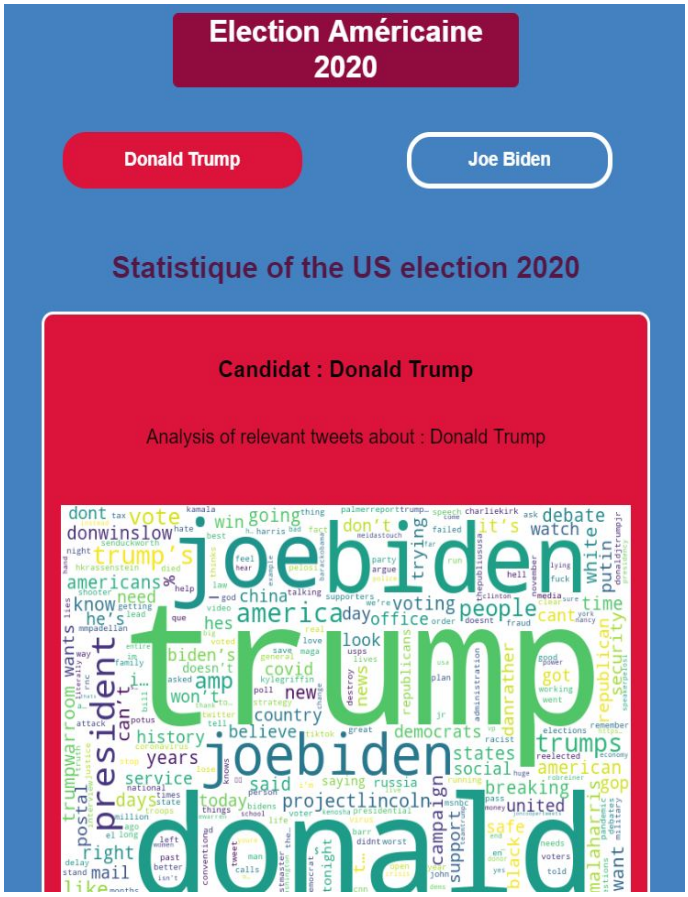


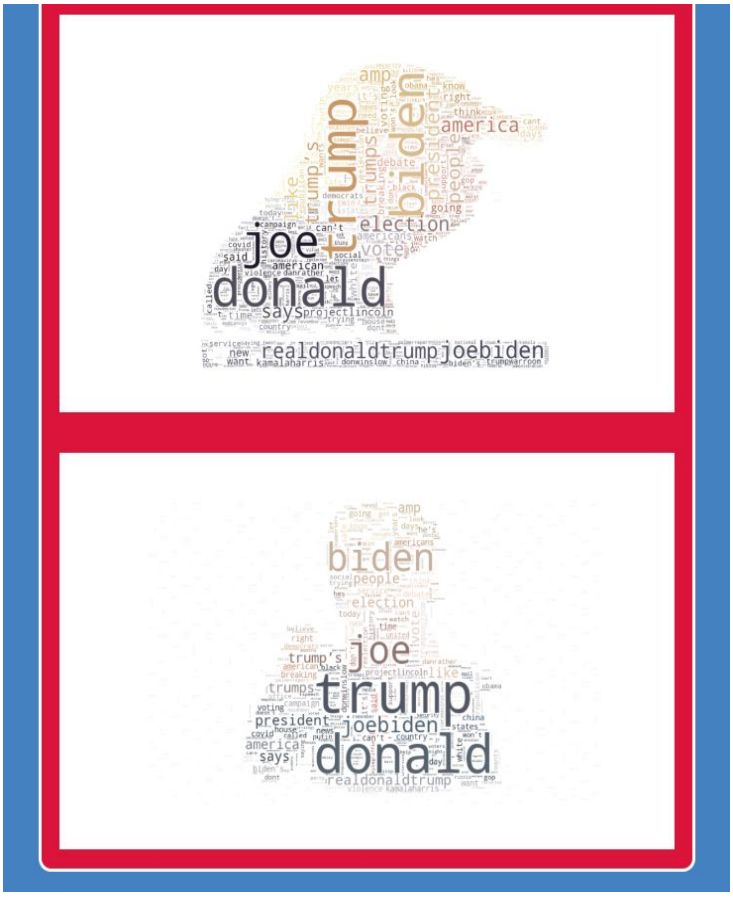
Other : Le dossier Other contient tous les éléments de design de notre site web. C'est-à-dire le Javascript pour éléments interactifs mais aussi le Css pour la mise en forme avec notamment une partie dédiée pour le design responsive.

View : Le dossier View va quant à lui contenir deux fichiers. Le premier étant le template de notre page web c'est-à-dire la structure générale de notre page en HTML. Ce fichier sera appelé par le second fichier qui n'est autre que notre vue. Elle aura la tâche de réceptionner les informations transmises par le contrôleur pour ensuite les ajouter au template.

2. Rendu final







III. Problème

Durant le développement de ce projet nous nous sommes heurtés à quelques problèmes qui nous ont retardés dans l'avancée de notre projet.

A. Taille du fichier d'entrée

En effet, nous n'avions pas prévu lors de la conception de notre projet que le fichier d'entrée serait aussi conséquent. De ce fait, nous avons dû recommencer notre projet en utilisant un langage de programmation plus performant car nous étions partis sur un développement en Php qui n'est absolument pas optimisé pour ce genre de tâche.

C'est pour cette raison que nous nous sommes penché sur Python qui permet d'avoir de meilleures performances sur le calcul tout en gardant une simplicité accrue.

B. Mémoire surchargé

Malgré l'utilisation d'un langage de programmation plus performant nous nous sommes confronté à un autre problème d'envergure qu'est la gestion de la mémoire. Durant la quasi-totalité du développement nous travaillions sur un échantillon de données pour obtenir les résultats rapidement et pouvoir debugger tout aussi rapidement. De ce fait, lors des phases finales de tests nous avons pu constater que python gardait en mémoire les variables utilisées et donc cela produisait un crash de l'application à cause d'une surcharge de la mémoire.

Pour pallier ces problèmes nous avons tout d'abord commencé par réduire la taille des fragments pour que le calcul soit moins lourd en mémoire. Puis dans un second temps nous avons cherché s'il existait une fonction native de python permettant de supprimer une variable non utilisée de la mémoire. Par chance cette fonction existe, il s'agit de la fonction **del** suivie de la variable que l'on veut détruire. Il existe aussi la fonction **.clear()** qui permet de remettre à zéro un dictionnaire python.

C. Optimisation du temps de traitement

Un des autres enjeux dans notre projet était le temps de calcul, et malheureusement lorsque nous avons augmenté le nombre de fichiers traités le temps de calcul est devenu très long.

Pour réduire le temps d'exécution de notre programme nous avons utilisé le module **Time** de python pour encapsuler chacune de nos fonctions afin d'avoir un détail précis de chaque temps d'exécution de nos fonctions.

Par la suite, nous avons dû reprendre la structure de chaque fonction pour effectuer des optimisations minimes ou majeures en fonction de son temps d'exécution et des solutions qu'il était possible de faire.

IV. Conclusion

Améliorations possibles :

- *Optimisation du temps de calcul*
    - Tester une approche différente pour le traitement :

Essayer un traitement ligne par ligne du fichier d'entrée plutôt qu'un traitement bloc par bloc utilisé dans notre projet qui nous oblige à créer des fichiers intermédiaires.

    - Utiliser un langage plus performant comme le C/C++

- *Interface graphique*
    - Ajouter plus de fonctionnalités
    - Faire design plus adapté et soigné

- *Fusionner les deux projets*
    - Un seul projet
    - Automatisation du traitement
    - Actualisation de l'interface web après chaque traitement
    - Utilisation de l'api de Twitter
        - Récupération automatique des tweets

Malgré cela nous sommes tout de même satisfaits du rendu final. Ce projet nous a permis de mettre en relation plusieurs langages de programmation sur un même projet (Python, Php, HTML, Css, Js).

Mais aussi d'un autre côté de voir et comprendre les enjeux de la programmation sur des fichiers conséquents (plusieurs giga), notamment avec les problèmes de gestion de la mémoire lors de la mise en œuvre de ce projet.

Cela nous aura aussi permis d'utiliser nos connaissances dans d'autres matières comme par exemple les cours de Parallélisme qui nous ont bien servi pour optimiser le temps de calcul de notre programme.