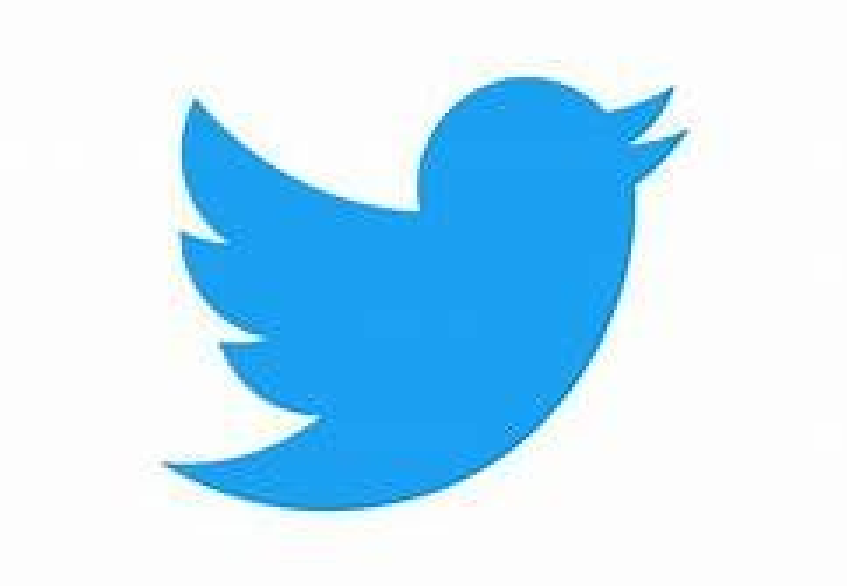
Compte Rendu Projet Twitter

Langages dynamiques -
Master 1 ISD



Sommaire

Introduction

Nous avons réalisé une interface graphique de reporting permettant de faire un rendu graphique synthétique de statistiques sur un ensemble de tweets. Ces messages proviennent de la plateforme Twitter. L'application contient deux parties, une client et l'autre serveur. Ces deux parties interagissant via le protocole HTTP.

Le serveur réalisé entièrement en python est capable de servir des fichiers, gérer des sessions HTTP, reconnaître certaines URL spéciales et déclencher l'exécution de code Python arbitraire lorsqu'on navigue vers ses URLs, mettre à disposition du code Python exécuté .

Le client est entièrement implémenté en pur CSS/HTML/Javascript et fait des requêtes asynchrones au serveur. Il s'agit de l'interface graphique proposant une barre de recherche pour la saisie de mots-clés. L'interface permet de visualiser à l'aide de graphiques le nombre de tweets trouvés, d'une carte du monde la répartition par pays, et un nuage de mots des hashtags et mots-clés associés à ces tweets et d'afficher le texte de ces tweets.

Architecture

L'architecture de l'application "Twitter" se compose en :

- Premièrement, la donnée en JSON est disponible via une URL et à télécharger dans ce dossier.
- Deuxièmement, le back-end contenant les fichiers du serveur et de la recherche en Python.
- Troisièmement, le front-end qui se compose des fichiers HTML et CSS de la page d'accueil de l'application, mais également dans d'un sous dossier les images et un autre les classes Javascript pour ces fichiers.

I. Back-end:

Nous avons commencé par coder un server avec la classe http.server de python. En choisissant CGI comme handler, ça nous permis de faciliter la communication entre le coté server codé en python et le coté client codé en html.
Ainsi, en utilisant un fichier de configuration .htaccess, dans notre répertoire contenant les tweets, nous avons pu imposer à notre serveur de n'éxécuter que les fichiers .py de notre répertoire cgi-bin.

Ce repertoire contient un fichier requette.py qui charge les données depuis un fichier csv puis recherche les tweets dont le texte contient les chaines de carractere recherchées par le client. Le résultat est ensuite formatater en json avant d'etre renvoyé au client.

II. Front-end:

L'architecture se compose tout d'abord d'un fichier connexion.js qui nous a servi à récupérer la requête et faire les différents graphiques, de fichiers.html et .js dans le fichier test qui nous ont servi à à tester les différentes interfaces avant de les intégrer à la véritable page. Le fichier "index.html" est la page officielle de notre application. On y ajoute au fur et à mesure les codes qui ont passé avec succès les tests dans le fichier précédent. Nous avons répertorié dans un seul fichier "style.css" les règles de styles.

On a utilisé la bibliothèque chart.js pour tous les graphiques.

Afin de bien structurer nos requêtes asynchrones avant de l'envoyer au serveur, nous avons un script connexion.js qui se charge d'établir la connexion avec le serveur et de récupérer le résultat qui sera par la suite utilisé pour la visualisation.

Structures de données utilisées

I. Back-end

Cette partie a été entiérement codée en python. La lecture des données à partir d'un fichier csv ainsi que la recherche sur ces données a été réalisé grace à la biblioteque pandas de python.

Et pour finir, le requette de ces requettes en renvoyé sous forme de json.

II. Front-end

Contrairement au back-end nous utilisons uniquement les données sous format JSON ainsi que des tableaux javascript classiques.

Problèmes Rencontrés

I. Back-end

Au début de ce projet, nous avons eu pas mal de confusion pour choisir le handler de notre server entre plusieurs proposés par la classe http.server car la différences n'est pas très explicite.
Nous avons également eu un peu de mal à récupérer le fichier json sur la partie javascript car malgré l'utilisation de JSON.parse(), on avait toujours cet erreur.
Un autre problème a été d'utiliser les promesse. En effet, nous ne voyons pas d'erreur sur notre code, cependant, ce dernier génère des alertes sur notre navigateur.
Nous l'avons donc laissé ce code dans le fichier connexion_promesse.js pour pouvoir avoir avoir une correction si possible et utiliser la partie sans promesses.

II. Front-end

- Côté front-end nous avons rencontré des difficultées lors de la création de la carte du monde car il y avait un décalage.On a fini essayé la projection de Lambert après plusieurs essais on a fait une projection de Mercator.
- On a eu un problème avec le word cloud fallait limiter le nombre de mots

Conclusion

Ce projet nous permis de mieux comprendre la relation client/server et de progresser dans la programmation de langages dynamiques telles que javascript.

Nous avons travaillé ensemble tout au long de ce projet en partageant notre travail de la manière suivante:

GNING Nogaye: Parties back-end(serveur+requette) + requete(Javascript)

Timgad: Partie front-end (Visualisation et graphique)+ requetes