Twitter workshop: Word Fight

Ce Project à pour but de comparer 2 sujets et de voir la vitesse de publication. Plus un sujet est discuté, plus sa vitesse sera grande

Comment ça fonctionne

Pour faire ça, on utilise l'API Twitter V2, qui propose d'afficher un flux de tweet filtrable correspondant à 1% des tweets.

Les étapes de fonctionnement

Définition des critères de filtrage

Dans l'onglet, l'utilisateur va pouvoir sélectionner les 2 mots qu'il souhaite comparer dans la langue qu'il souhaite.

Traitement du retour de l'API

Lorsque des tweets arrivent depuis l'API, ils passent dans plusieurs streams de transformation qui constituent une pipeline:

- `jsonParser`: convertit le texte en données exploitables
- `tweetsCounter`: compte le nombre de tweets associés à chaque règle de filtrage
- `clientFilter`: ne laisse passer que les tweets correspondant aux règles définies par le client
- `socketStream`: envoie les données au client via WebSocket

Affichage des résultats

Comme le front n'est pas évalué, j'ai décidé de faire simple, en utilisant la librairie ChartJS pour afficher les résultats: elle gère tout (mise à l'échelle, mise à jour des données, légende, axes...) à ma place avec une configuration par défaut qui convient au projet.

Modification des règles de filtrage

Lorsqu'un client modifie ses règles de filtrage, les anciennes règles sont supprimées à condition qu'elles ne soient pas utilisées par un autre client connecté.

Déconnexion d'un client

Lorsqu'un client se déconnecte, les règles de filtrage qui correspondaient à ses choix de sujets sont supprimées afin d'économiser le quota de tweets. Si une règle reste utilisée par un autre client, elle reste active.

Difficultés rencontrées

La séparation entre plusieurs clients m'a posé le plus de difficultés, notamment parce qu'un client correspondait à 2 règles, mais qu'une règle peut correspondre à 1 client comme à 1000. Une autre difficulté

est que 2 règles ne peuvent être identique, même si on leur donne un attribut différend. Il est donc nécessaire de partager une règle entre 2 clients. La troisième difficulté est que lorsqu'un tweet correspond à une règle, l'ID de la règle renvoyé par l'API ne correspond pas *exactement* à celui communiqué à la création de la règle.

J'ai donc utilisé un module (`clientsRules`) chargé de garder en mémoire les correspondances entre les règles et les clients. La correspondance avec l'api ne pouvant pas utiliser les ids fournis par Twitter, il utilise les tags avec un identifiant aléatoire, généré avec uuidv4. Chaque règle est liée à une liste de clients, ce qui permet également de savoir si on peut la détruire une fois que plus aucun client n'en dépend. Un défaut de cette méthode est dans le cas où 2 clients sont sur un même sujet, les résultats peuvent être faussés (voir Known issues). Pour corriger ce bug, il faudrait séparer chaque règle pour chaque client mais en gardant la règle en commun du côté de l'API Twitter, ce qui compliquerait le traitement. Je n'ai pas eu le temps de corriger ce bug.

Making it work

- Clone this repository
- Create a `.env` file at the root of the folder with the following info:

```
TWITTER_BEARER_TOKEN="YourTwitterBearerToken"
```

- Install dependencies: `npm install`
- Run the server: `npm start`
- You can open the client at `localhost:3000`

Known issues

If 2 clients are watching the same topic, the number of tweets won't be since the loading of the topics *for each client*, but since *the first client started watching the topic*, skewing the results.

There is a quite consequent latency between the moment you submit topics and the moment tweets arrive. This is normal and is due to the update delay of the Twitter stream. A message ("Waiting for tweets to arrive...") is displayed during this latency.