Détection du cyberharcèlement sur Twitter - Documentation

Prérequis 1
Versions utilisées 1
Twitter 1
Installation 2
Fonctionnement 3
Premier filtrage 3
Principe 3
Utilisation 3
Second filtrage 4
Principe 4
Utilisation 4
Traitement des résultats 5
Statistiques générales 5
Tweets 5
Auteurs 5
Affichage des tweets 7
Nos résultats 9

Prérequis

Versions utilisées

Ce TIPE a été testé sous Python 3.8.12.

Les modules ont été utilisés avec les versions suivants :

- sqlite3 : 2.6.0
- requests : 2.26.0
- json : 2.0.9

Twitter

Il est nécessaire d'avoir un accès à l'API de Twitter.

Ce TIPE a été réalisé avec l'accès "Elevated".

Installation

- Téléchargez les fichiers
- Placez-les dans un même dossier
- Modifiez le fichier **config/api.py** en renseignant vos clés API

Fonctionnement

Toutes les commandes sont à exécuter depuis le fichier **main.py**.

Premier filtrage

Principe

Le premier filtrage consiste à récupérer, grâce à l'API de Twitter, un flux de tweets en temps réel. Ce flux est filtré : seuls les tweets contenant au moins un des mots (ou expressions) présents dans le fichier **config/lists.py** sont affichés.

Chaque tweet récupéré est un tweet qui vient d'être posté.

Une fois le tweet récupéré, celui-ci est stocké dans la base de données **database.db**, ainsi que son auteur.

Utilisation

```
Stream.start()
```

- Paramètres :
  - `showTweets` (bool) : désactivé par défaut, permet d'afficher les tweets récupérés.
  - `insertInDatabase` (bool) : activé par défaut, permet d'insérer les tweets récupérés dans la base de données "database.db"
  - `resetRules` (bool) : désactivé par défaut, permet de mettre à jour les règles définies pour le stream (si des changements ont été effectués par exemple.)

- Limite API (avec un accès "Elevated") :
  - 50 requêtes / 15 minutes
  - 2M tweets / mois

Lors de la première éxécution de ce filtrage, il est nécessaire de passer le paramètre resetRules sur la valeur "True".

Second filtrage

Principe

Chaque tweet provenant du premier filtrage a été stocké dans la base de données **database.db**, avec un "score" valant "NULL".

Le "score" correspond à une valeur entre 0 et 100, indiquant la probabilité que le tweet soit effectivement haineux (100 étant la plus haute probabilité). Un score de "NULL" signifie que le tweet n'a pas encore été traité.

Le second filtrage permet d'attribuer à un tweet son score. Pour ce faire, nous avons attribué à chaque mot/expression des listes présentes dans le fichier config/lists.py une valeur entre 0 et 100. Plus la valeur est élevée, plus le mot ou l'expression a une probabilité élevée d'être lié à un commentaire haineux.

Par exemple, le mot "merde" a une probabilité plus faible d'être lié à un message haineux que l'expression "fils de pute".

Une moyenne pondérée est ensuite effectuée : chaque mot détecté est ajouté au score total du tweet, sachant que le nombre d'apparition du mot compte.

Si le score du tweet est strictement inférieur à 40, celui-ci est supprimé de la base de données. Sinon, le score du tweet dans la base de données est édité.

Utilisation

```
Filtering.start()
```

- Paramètres :
    - `showOutput` (bool) : désactivé par défaut, permet d'afficher les termes détectés.

Traitement des résultats

Toutes les fonctions du module **Stats** ont, en plus de leurs paramètres respectifs, un paramètre `readable` (bool), défini sur True par défaut, qui rend les données lisibles pour tout le monde. La fonction affiche alors seulement du texte, et ne renvoie rien.

Statistiques générales

```
Stats.general()
```

Affiche le nombre total de tweets et d'auteurs dans la base de données.

Tweets

```
Stats.tweetsNumberInAList()
```

Affiche le nombre de tweets correspondant à une liste particulière.

- Paramètres :
  - `list` (str) : Une des listes du fichier **config/lists.py**

Auteurs

```
Stats.authorsWithMoreThanXTweets()
```

Affiche le nombre d'auteurs apparaissant au moins X fois dans la base de données.

- Paramètres :
  - **`tweetsNumber`** (int)

```
Stats.topAuthorsUsernames()
```

Affiche la liste des noms d'utilisateurs des auteurs qui apparaissent le plus de fois dans la base de données.

- Paramètres :
  - `authorsNumber` (int) : 10 par défaut, représente le nombre d'auteurs à afficher.
  - `showTweetsNumber` (bool) : désactivé par défaut, permet d'afficher le nombre de tweets de l'auteur apparaissant dans la base de données.

```
Stats.topAuthorsSensitiveTweetsPercentage()
```

Analyse les derniers tweets des auteurs, et affiche le pourcentage de tweets qui sont détectés par le second filtrage.

- Paramètres :
    - `authorsNumber` (int) : 10 par défaut, représente le nombre d'auteurs à afficher.

Affichage des tweets

Le module **Search** permet de chercher et d'afficher des tweets provenant de la base de données.

Les fonctions de ce module n'affichent que du texte.

```
Search.random()
```

Affiche un tweet aléatoire.

```
Search.byID()
```

Cherche un tweet avec son ID et l'affiche (s'il existe).

- Paramètres :
  - `id` (int) : ID à chercher.

```
Search.byListName()
```

Affiche un ou plusieurs tweets ayant été détectés dans une liste particulière.

- Paramètres :
  - `list` (int) : une des listes du fichier **config/lists.py**.
  - `limit` (int) : 1 par défaut, représente le nombre de tweets à afficher.

```
Search.byText()
```

Affiche un ou plusieurs tweets contenant un mot ou une expression en particulier.

- Paramètres :
  - `text` (str) : texte à chercher.
  - `limit` (int) : 1 par défaut, représente le nombre de tweets à afficher.

```
Search.byScore()
```

Affiche un ou plusieurs tweets ayant un score précis.

- Paramètres :
  - **`score`** (float) : score à chercher.
  - `limit` (int) : 1 par défaut, représente le nombre de tweets à afficher.

```
Search.byAuthor()
```

Affiche un ou plusieurs tweets ayant été écrit par un auteur en particulier.

- Paramètres :
  - **`username`** (str) : nom d'utilisateur de l'auteur à chercher.
  - `limit` (int) : 1 par défaut, représente le nombre de tweets à afficher.

Nos résultats

A venir…