PJE
#Currentmood

Quentin Van de Kadsye Jérôme Tanghe

Mardi 15 décembre 2015

Table des matières

I – Le projet

Il est parfois intéressant pour les entreprises de connaître l'humeur générale des gens concernant un sujet donné. Twitter étant une plateforme où l'on peut s'exprimer librement, c'est donc un emplacement de choix pour récolter ce type d'information. Se pose alors le problème suivant : comment connaître rapidement l'humeur des personnes sur un sujet donné ?

#Currentmood est un programme tentant de répondre à ce besoin. Écrit en Java, il permet d'estimer l'humeur d'un ou plusieurs messages publiés sur Twitter (*tweet*), à l'aide d'une des trois méthodes proposées :

— **Mots-clés :** utilise une liste de mots prédéfinis dans des fichiers pour déterminer l'humeur du tweet.
— **KNN :** évalue l'humeur d'un tweet en fonction de l'humeur de k autres tweets, en recherchant dans une base de données de tweets dont on connaît déjà l'humeur ceux qui contiennent les mêmes mots.
— **Classification bayésienne :** évalue la probabilité d'humeur d'un tweet en calculant la probabilité que les mots qu'il contient appartiennent à cette humeur à partir de la base de données.

II – API Twitter et gestion des tweets

1 – La classe `CMTwitter`

Afin de communiquer avec l'API de Twitter, nous avons utilisé la librairie **Twitter4J** [1] qui propose les fonctionnalités dont nous avons besoin pour mener à bien le projet. Pour faciliter son implémentation, nous avons également créé une classe, `CMTwitter`, s'interfaçant entre notre application et Twitter4J, comme le montre la figure 2.1.

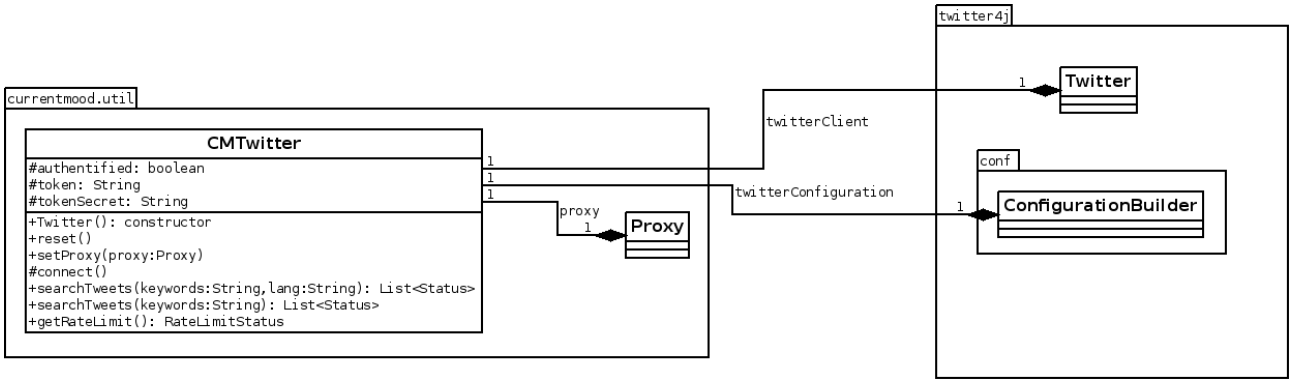

FIGURE 2.1 – Diagramme de classe montrant comment nous interfaçons Twitter4J

Cela nous permet d'utiliser l'API de Twitter à l'aide de trois méthodes principales seulement au lieu d'une dizaine :

— `setProxy()` afin de donner les paramètres proxy à Twitter4J ;
— `connect()` afin d'établir la connexion avec l'API de Twitter ;
— `searchTweets()` afin d'utiliser la fonction de recherche de l'API de Twitter.

La méthode `reset()`, quant à elle, est utilisée par les méthodes précédentes afin de permettre d'effectuer plusieurs requêtes. En effet, nous avons pu remarquer que l'objet `ConfigurationBuilder` périme après chaque requête, nous obligeant à le recréer avant d'effectuer une nouvelle requête.

L'utilisation de cette classe s'effectue en trois étapes principales.

Tout d'abord, on configure si nécessaire les paramètres proxy à l'aide de la méthode `setProxy()`. Cette dernière donne lesdits paramètres à l'objet `ConfigurationBuilder`.

1. twitter4j.org

Ensuite, on appelle la méthode `connect()` qui utilise l'objet `ConfigurationBuilder` pour obtenir une instance de `Twitter`.

C'est cette instance qui est ensuite utilisée par `searchTweets()` qui instancie un objet de Twitter4J permettant d'obtenir des tweets correspondant à la recherche. Cette méthode retourne une liste d'objets `Status` provenant de la librairie Twitter4J.

2 – La classe `Tweet`

Nous nous sommes rapidement rendus compte que Twitter4J ne proposait aucune classe implémentant l'interface `Status` que nous devons manipuler. De plus, les objets que nous en obtenons contient de nombreuses informations qui ne nous sont pas utiles, tandis que d'autres nous étaient nécessaires mais n'y étaient pas présents.

Nous avons donc créé une classe indépendante de Twitter4J, `Tweet`, qui réponde à ce besoin. Elle est utilisée dans toute l'application afin de contenir chaque tweet.

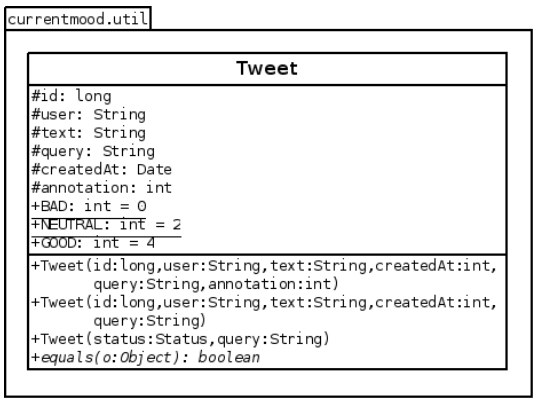

FIGURE 2.2 – La classe `Tweet`

Cette classe est surtout composée d'accesseurs permettant d'accéder à chaque propriété du tweet. Ses deux premiers constructeurs permettent de créer un objet à partir de données déjà connues (typiquement lors de l'ouverture d'un fichier CSV), tandis que le troisième permet d'obtenir un objet `Tweet` à partir d'un objet `Status` généré par Twitter4J.

Nous avons également surchargé la méthode `equals()` de la super-classe `Object` afin de permettre la comparaison de l'objet courant avec un autre objet `Tweet`. Cela nous sera nécessaire pour certaines actions par la suite.

3 – Le fichier CSV

Pour gérer la base de données, il a été décidé de sauvegarder les messages annotés dans un fichier CSV [2]. Ce type de fichier a pour principal avantage d'être relativement léger et facile à

2. *Comma-separated values*

lire par programmation.

Les données à sauvegarder étant globalement celles contenues dans les objets `Tweet`, l'ordre de sauvegarde sera donc le suivant :

1. Le numéro d'identification du tweet
2. Le nom de l'auteur du tweet
3. Le contenu du tweet
4. La date et l'heure du tweet
5. La recherche qui a permi de trouver le tweet
6. L'annotation du tweet