M1
INFORMATIQUE

RAPPORT PROJET JAVA

BOUE HUGO – N°2167945
DELAGE ANTHONY – N°2161572

TABLE DES MATIERES

SUJET ET SPECIFICATIONS

Tout d'abord, notre projet s'inscrit dans le cadre du cours de programmation en java et vise à exploiter les compétences et connaissances acquises au cours du semestre. Dans cette logique, le projet proposé se présente comme la réalisation d'un outil permettant d'explorer et d'analyser les données de Twitter. En effet, Twitter est un réseau social massivement utilisé et de nombreuses données peuvent être récoltées à partir de ce dernier et analysées. Deux jeux de données, obtenus grâce à l'API de Twitter, ont été mis à disposition pour permettre la réalisation de notre outil. Il s'agit de tweets en français comportant des mots clés sur deux sujets, à savoir le climat et le football (lors de la coupe du monde féminine en 2019).

La réalisation de l'outil a pour but d'atteindre plusieurs objectifs. Pour commencer, afin de réaliser un outil d'analyse efficace, il est nécessaire que ce dernier permette l'import des données ainsi que leur stockage. Outre ce premier aspect, l'outil proposé pouvait prendre différentes orientations, à savoir la fouille de texte, la fouille de graphe, ou le reporting. Dans notre cas, nous avons opté pour la création d'un outil de reporting. Ainsi, l'analyse privilégié dans notre travail est une analyse temporelle. L'objectif de notre outil est d'obtenir des visualisations dynamiques des données en fonction du temps ainsi que quelques statistiques. Un enjeu majeur de notre travail est de permettre à l'utilisateur d'obtenir la visualisation qu'il souhaite et de rendre cette dernière dynamique.

Pour répondre à ces objectifs, nous avons donc créé une application permettant plusieurs fonctionnalités. La première, et la plus primitive, est la lecture des données sur un laps de temps. Nous avons également travaillé sur l'observation du nombre de tweets à travers le temps, ou encore sur la fréquence des hashtags. Enfin, nous nous sommes intéressés à la popularité des utilisateurs et à l'évolution de cette dernière.

ANALYSE

Afin de réaliser ce projet, nous avons travaillé avec Eclipse sous Windows et avec Netbeans sur Mac. Ce constat est lié au fait que nous n'avons pas réussi à faire fonctionner Java FX avec Eclipse sur Mac. Toutefois, cela nous a permis de voir que notre programme pouvait être compilé sur les deux IDE.

DIAGRAMME DE CLASSES SIMPLIFIE

CLASSE TWEET

CONSTRUCTUEUR ET ACCESSEURS

Tout d'abord, la classe ***Tweet*** comprend un constructeur (*Tweet()*) et plusieurs accesseurs (*getDate(), getRetweet(), getMess(), getHashtag()*). Le constructeur permet de construire un objet de type ***Tweet*** avec toutes les caractéristiques associées. Les accesseurs quant à eux permettent d'accéder à certains champs d'un tweet dans le but de les utiliser dans les traitements des autres classes.

MUTATEURS ET AUTRES METHODES

La classe ***Tweet*** contient également un mutateur pour le champ hashtag et permet ainsi de modifier ce dernier. Les méthodes *toSting()* et *toStringPartiel()* permettent quant à elle d'afficher un tweet. Tandis que la première permet d'obtenir une chaîne de caractères avec tous les champs d'un tweet, la seconde est utilisée pour l'affichage des tweets dans l'interface de notre application et ne contient pas le champ id_tweet dans la chaîne renvoyée. Enfin, la méthode *Gethastagliste()* consiste à découper le message du tweet en un tableau de caractères. On parcourt ensuite ce tableau afin de déceler le caractère « # » qui définit les hashtags sur Twitter. Une fois qu'on a identifié le début de l'hashtag on ajoute à la chaîne de caractères tous les caractères suivants jusqu'à ce qu'un caractère ne soit pas alphabétique.

CLASSE BASEDETWEETS

La classe ***BaseDeTweets*** contient diverses méthodes. On peut en premier lieur présenter la méthode *initialise()* qui permet d'instancier l'*ArrayList* contenant le jeu de données, c'est-à-dire l'ensemble des tweets.

La méthode *date(choix)* permet d'obtenir le jour du premier tweet du jeu de données ou le jour du dernier.

Nous avons ensuite plusieurs méthodes *lire()*. Il s'agit donc d'une surcharge. La méthode sans paramètre permet d'afficher l'ensemble des tweets. Celle avec un seul paramètre date permet d'afficher les tweets du jour entré en paramètre. Enfin

la dernière, comprenant deux paramètres, permet d'afficher les tweets entre deux date sélectionnées.

La méthode a*joute()* permet d'ajouter un tweet à la liste de tweets.

La méthode *lireRetweet()* est surchargée comme la méthode *lire()*. *lireRetweet(date)* permet d'obtenir les cinq utilisateurs les plus retweetés pour la date « date « tandis que *lireRetweet(d1, d2)* fait le même classement entre deux dates. Les méthodes *lireRetweet(stuser)* et *lireRetweet(stuser, date)* permettent respectivement de suivre le nombre de fois où l'utilisateur entré en paramètre est retweeté sur la période du jeu de données et le nombre de fois où il est retweeté le jour entré en paramètre. La liste des utilisateurs retweetés est obtenue grâce à *listeRetweet()*.

La méthode *nbTweets()* est aussi surchargée. Celle présentant deux dates en paramètre permet d'obtenir le nombre de tweets par jour entre deux jours alors que celle n'ayant qu'une seule date en paramètre donne le nombre de tweets par heure sur une journée.

Enfin, avec la méthode *hashtagliste(d1, d2),* on parcourt la liste des tweets correspondant à la période. Si ceux-ci contiennent des hashtags, on parcourt la liste des hashtags correspondant à la liste *hash*. Si l'hashtag est déjà présent dans *hash* on incrémente de 1 le tableau *occurrence[]* à l'index du hashtag sinon on ajoute le hashtag à la liste *hash*. Ensuite il s'agit de retourner les 10 hashtags les plus utilisés sur la période, on parcourt donc la liste *hash* 10 fois et on retient celui qui a le maximum d'occurrences en le mettant dans la liste finale « *Hashtag* » et on réinitialise son compteur pour ne plus le prendre en compte lors des prochains passages dans la liste.

CLASSE PROJET

La classe Projet est la classe principale de notre programme. C'est elle qui permet d'obtenir l'interface de notre outil.

Pour commencer la méthode main permet de lancer l'application. La première fenêtre est générée grâce aux méthodes start() et construitScene(). La fonction charger() quant à elle permet d'effectuer le chargement des données depuis les fichiers source. La seconde fenêtre, correspondant au menu de l'application, est obtenue avec construitMenu(). Ensuite, les différentes fonctionnalités de l'application, à savoir l'affichage des données, l'analyse sur le nombre de tweets, l'analyse sur les retweets, et celle sur les hashtags sont associées à une fenêtre spécifique et respectivement aux méthodes affichage(), nbTweets(), Retweets() et

HashtagStage(). Pour terminer, la méthode menuB() permet de construire la barre de menu présente dans toutes les fen^êres excepté la fenêtre d'accueil.

CONCEPTION

ORGANISATION ET REPARTITION DES TACHES

Avant de commencer concrètement la réalisation de notre outil, nous avons fait un point pour savoir dans quelle direction aller et avoir une organisation cohérente. Il est ressorti de cette discussion trois principales étapes de construction du projet :

1- La création des classes
2- L'import des données
3- Création des méthodes pour l'analyse

A noter que cet enchaînement a été effectivement respecté par la suite.

Par ailleurs, la répartition des tâches a consisté principalement à ce que chacun s'occupe d'un ou de plusieurs aspects du programme en particulier. Par exemple, tandis que l'un travaillait sur l'analyse des hashtags, l'autre s'intéressait au nombre de tweets. Pour suivre l'avancement de chacun, des échanges téléphoniques réguliers, voire quotidiens, ont été réalisés afin de faire le point sur l'avancée du projet, de clarifier certains aspects ou encore de s'entraider.

Lors de la réalisation de notre travail, nous avons en réalité découpé l'étape 3 présentée précédemment en deux étapes : programmation sans interface graphique et programmation avec interface graphique. Effectivement, nous avons fait le choix de travailler dans un premier temps sans se préoccuper de l'aspect graphique de notre outil. Ce choix nous a permis de nous concentrer sur le fonctionnement des méthodes permettant l'analyse des données afin de bien en comprendre toute la portée et de ne pas nous encombrer l'esprit avec d'autres aspects. Ces méthodes ont donc été testées dans un premier temps en utilisant la console comme cela sera expliqué dans la suite du rapport. Une fois les méthodes relativement maîtrisées, nous avons commencé à introduire l'aspect interface graphique. Cette introduction nous a amené à adapter certaines méthodes pour les rendre utilisables avec l'interface. On peut notamment donner comme simple exemple le remplacement des « *System.out.println* ». De même, c'est à ce moment que nous avons commencé à travailler sur l'aspect « visualisation » de notre outil de reporting et ainsi à réfléchir à la manière d'utiliser les méthodes existantes pour cette visualisation.

Pour la fin du projet, nous avons fait d'autres points afin de résoudre les derniers problèmes, d'épurer le code, notamment en enlevant tous les éléments devenus inutiles et en rajoutant des commentaires si nécessaires. Cela a aussi été l'occasion de réfléchir sur le rapport et de nous répartir les différentes parties à rédiger.

CHOIX DE MODELISATION

Notre projet a nécessité trois classes. La classe ***BaseDeTweets*** est en relation avec la classe principale ***Projet*** et la classe ***Tweets*** est en relation avec la classe ***BaseDeTweets***.

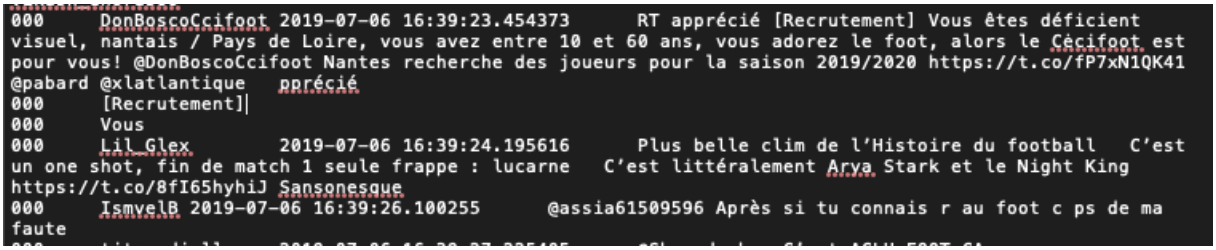
Pour gérer la collection de tweets de la classe ***BaseDeTweets***, nous avons opté pour une *ArrayList*. Un objet de type *TreeSet* aurait aussi été envisageable dans la mesure où il permet d'avoir une liste triée sans duplication. Toutefois, après consultation des fichiers de données, nous avons remarqués que les tweets été déjà classés dans l'ordre chronologique et nous avons donc fait le choix de *l'ArrayList* pour sa facilité d'utilisation. De même, sachant que notre programme ne permet ni ajout, ni suppression de tweets de la liste, il n'est donc pas possible de troubler l'ordre chronologique déjà établi. L'utilisation d'un *TreeSet* ne nous a donc pas semblé nécessairement justifiée.

La classe principale, quant à elle, utilise un objet de type ***BaseDeTweets*** afin de permettre de réaliser tous les traitements. Nous avons décidé de ne créer qu'un objet de ce type (clause *static*) qui contient soit le jeu de données sur le foot soit le jeu de données sur le climat suivant le choix de l'utilisateur. Ce choix est lié au fait que l'utilisateur ne va s'intéresser qu'à un seul jeu de données à la fois. Il aurait été aussi envisageable de créer un second objet de type ***BaseDeTweets*** pour pouvoir stocker les deux jeux de données simultanément. Ainsi, l'utilisateur pourrait travailler sur le premier jeu de données, puis passer à l'autre et enfin revenir au premier sans avoir à charger de nouveau des données qu'il aurait déjà utilisées. Pour gérer cet aspect, nous avons préféré n'utiliser qu'un seul objet en raison du grand nombre de données à stocker et avons fait en sorte que le fichier ne soit pas rechargé et que la liste soit conservée si l'utilisateur retourne sur la page d'accueil et choisit de nouveau le même jeu de données que précédemment.

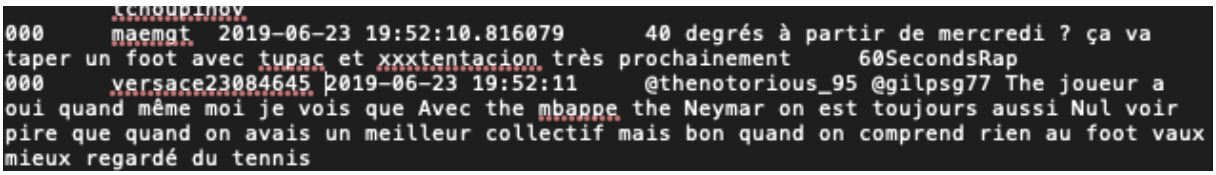
PROBLEMES RENCONTRES

Au cours de la conception de notre application, nous avons rencontré quelques difficultés ou problème.

Le premier problème qui est apparu est en lien avec l'import des données. En effet, lors des premiers tests nous avons remarqué que celui-ci générait des erreurs à l'exécution. Ces erreurs n'entraînaient pas un arrêt du fonctionnement du programme mais en regardant le dernier tweet ajouté à la liste, nous pouvions voir que l'ensemble du fichier sélectionné n'était pas parcouru. En cherchant le dernier tweet en question dans le fichier, nous pouvions voir que la ligne suivante dans le fichier était incomplète et ne correspondait pas à un tweet. Voici un exemple :



Pour pallier à ces lignes inexploitables, nous avons donc rajouter des blocs *Try-Catch* dans le code de notre importation afin de ne pas prendre en compte ces lignes et de continuer le parcours du fichier. Lors de la lecture d'une de ces lignes, la console affiche « Tweet non utilisable ».



De plus, comme le montre la capture d'écran ci-dessus, certaines lignes ont une date ne correspondant pas au format de date que nous avons conservé et qui correspond à celui des autres lignes du fichier. Là encore, l'exception est gérée avec un bloc *Try-Catch* et la console affiche « Erreur lors du formatage de la date » au moment de la lecture de ces lignes. La gestion des exceptions avec l'ajout de ces blocs, a résolu le problème rencontré et permis le parcours complet des fichiers pour le chargement des données

Par ailleurs, la gestion des caractères avec accent a posé problème. La solution a été de passer tous les chaine de caractères en UTF-8 dans les méthodes *toString()* et *toStringPartiel()* de la classe Tweet.

Une autre des principales difficultés rencontrées est lié à l'utilisation de *BarCharts*. Effectivement, nous avions réussi à créer les graphiques souhaités mais cette création était accompagnée de quelques problèmes d'affichage. En effet, lors de la première apparition du graphique, les valeurs étaient correctes mais les labels de l'axe des abscisses étaient tous regroupés au début de l'axe. En cliquant une deuxième fois sur le bouton permettant l'affichage, le problème était résolu et les labels s'affichaient correctement. Après quelques essais infructueux pour régler ce problème, nous avons décidé de ne nous en occuper qu'à la fin du projet. Finalement, nous avons réussi à le résoudre après plusieurs recherches sur internet et la consultation de l'API de java.

De la même manière, nous souhaitions ajouter à certains de nos diagramme une ligne représentant la moyenne afin d'observer plus facilement les valeurs se situant de part et d'autre de celle-ci. Toutefois, malgré nos recherches nous n'avons pas réussi à concrétiser cette idée et avons dû l'abandonner pour nous concentrer sur les autres aspects du projet.

Pour terminer, un des obstacles rencontrés est lié à l'affichage des données. Nous avons souhaité que cet affichage soit plaisant pour l'utilisateur en distinguant chaque élément d'un tweet par une couleur. Nous sommes parvenus à cela pour l'affichage des tweets d'une partie d'une journée mais le traitement permettant cela était trop lourd pour l'affichage complet ou d'un jour entier en raison du grand nombre de données. C'est pourquoi, après tests, nous n'avons conservé l'affichage coloré que pour le premier cas.

ASPECTS COMPLEXES ET NOUVEAUTES

IMPORT DES DONNEES

Notre projet repose sur l'utilisation de données récoltées sur Twitter. Il nous a donc semblé pertinent d'expliquer la manière dont nous avons importées les données depuis les fichiers. Cet import des données est réalisé via la méthode *charger(fichier)* de la classe principale prenant en paramètre le nom du fichier voulu. La méthode consiste dans un premier temps à lire toutes les données du fichier. Chaque ligne est ensuite découpée en cinq morceaux correspondant aux cinq champs associés à un tweet (son identifiant, sa date, l'utilisateur, le message, et le nom de l'utilisateur retweeté s'il s'agit d'un retweet. Si le découpage n'est pas possible un message d'erreur est affiché à la console. Il s'agit ensuite de transformer le champ correspondant à la date en *LocalDateTime* suivant le format de date définit. Si cette transformation n'est pas possible, un message d'erreur est aussi affiché dans la console. Dans le cas contraire, si aucune des deux erreurs présentées n'a lieu, le

tweet est ajouté dans le liste avec chaque morceau correspondant à un champ. Une fois qu'il n'y a plus de ligne à lire, si le traitement précédent a bien eu lieu, signifiant ainsi que le fichier source a été trouvé, une fenêtre de dialogue s'affiche pour indiquer que les données ont été chargées. Dans le cas contraire, la console indiquera qu'il y a eu une erreur lors du chargement des données.

NOUVEAUTES

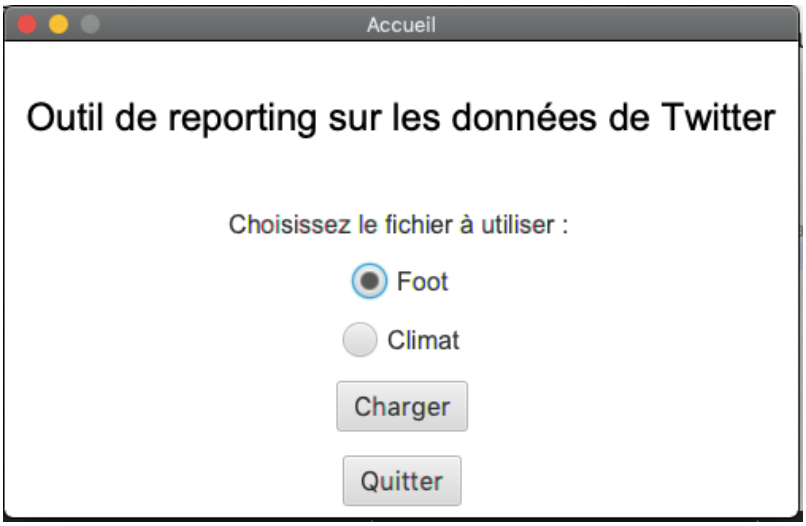
La réalisation de ce projet, nous a amené à utiliser des « nouveautés » par apport aux exercices du T.D. La première nouveauté est l'utilisation de nouveaux objets pour le stockage des collections et notamment l'utilisation de *HashMap* et *TreeMap*. Ces derniers correspondent à des tableaux associatifs sans duplication pour les clefs. Le recours à ces tableaux s'est notamment avéré nécessaire pour la réalisation des graphiques ou faire des classements. Pour chacun de ces exemples, nous avions besoin d'associer une clef à une valeur. Par exemple, nous devions associer un jour au nombre de tweets sur cette journée. Pour apprendre à utiliser ces objets nous avons regardé des exemples/tutoriels sur internet ainsi que la documentation de l'API de java.

L'autre principale nouveauté et la réalisation de diagrammes. Comme évoqué précédemment, nous avons utilisé pour cela des *BarChart*. Là encore, nous avons eu recours à des exemples/tutoriels et à l'API pour comprendre son fonctionnement et l'adapter à notre application.

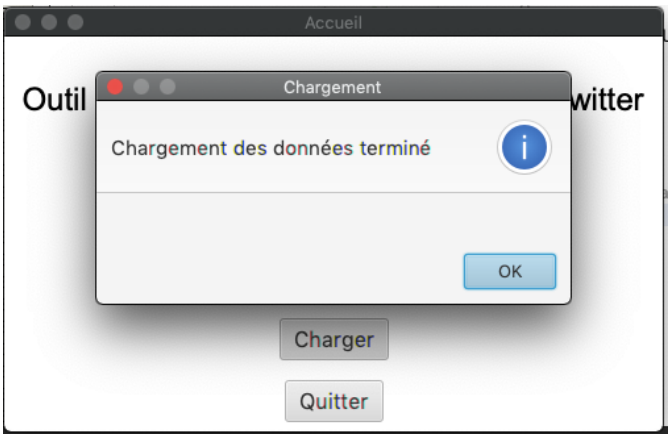
Nous avons également ajouté des raccourcis claviers pour les items de la barre de menu afin d'améliorer le confort de l'utilisateur.

EXEMPLE COMPLET D'UTILISATION

Lors du lancement de l'application, l'utilisateur tombe sur la fenêtre d'accueil. Il doit alors choisir le jeu de données sur lequel il veut travailler et le charger.



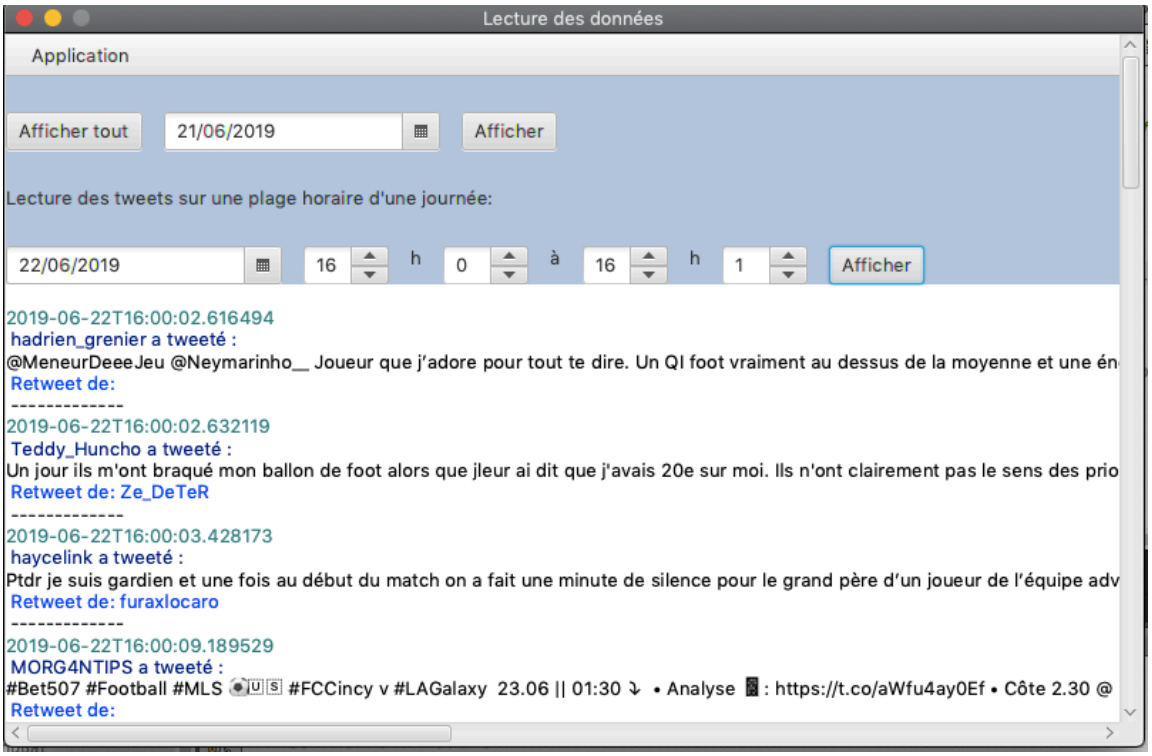
Si le chargement des données s'est déroulé correctement le pop-up suivant apparait à l'écran :



Il a alors accès aux différentes fonctionnalités proposées par l'application grâce à l'ouverture d'une nouvelle fenêtre. Cette dernière est aussi dotée d'une barre de menu permettant de retourner à l'accueil ou de quitter l'application. Cette barre est la même pour toutes les prochaines fenêtres de l'application avec le rajout de l'item « *Menu* » permettant de fermer la fenêtre actuelle et de revenir sur la fenêtre présentée ici.

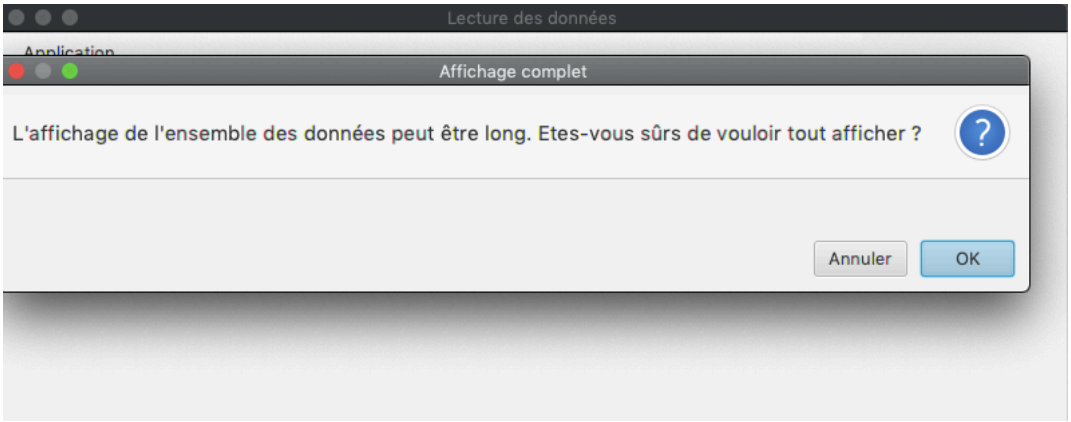
LIRE LES DONNEES

Dans la fenêtre « *Lecture des données* », l'utilisateur peut visualiser le jeu de données. Il peut visualiser l'ensemble des tweets d'une journée en choisissant la date et cliquant sur le bouton « *Afficher* » en haut de la page, ou encore visualiser les tweets postés sur une certaines plage horaire d'un jour en particulier en utilisant la seconde ligne de contrôle.



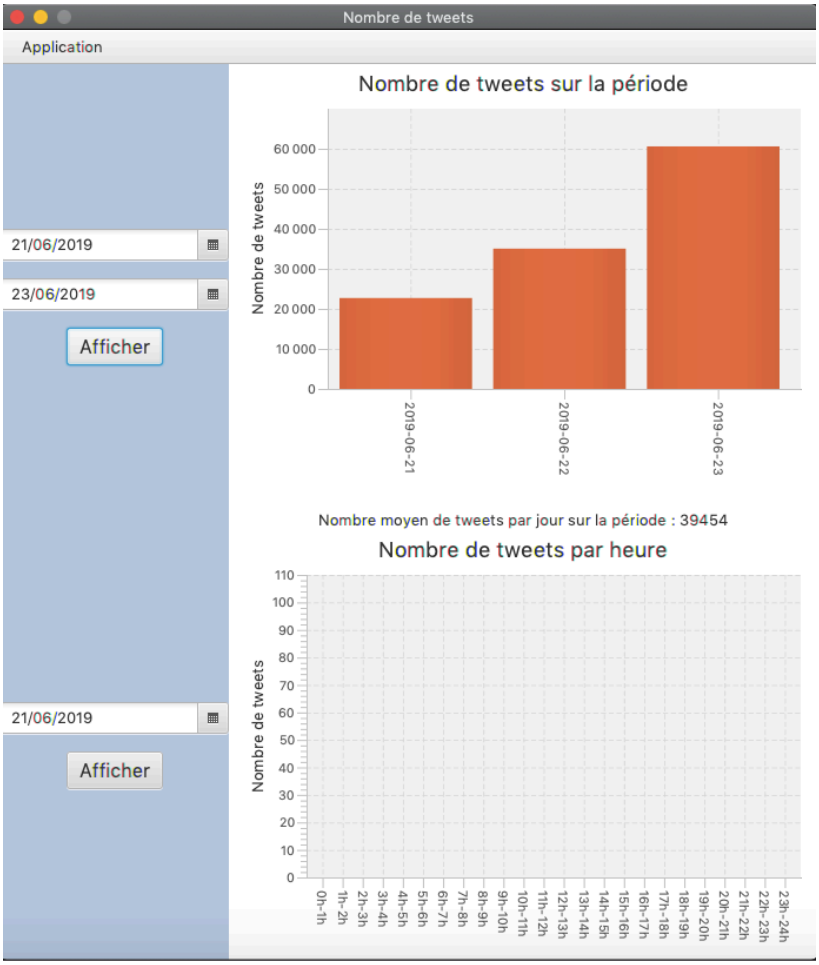
Il est aussi possible d'afficher l'ensemble du jeu de données. Une fenêtre de dialogue s'affiche alors pour prévenir que l'affichage peut prendre du temps. L'affichage n'a alors lieu que si l'utilisateur clique sur « *OK* ».

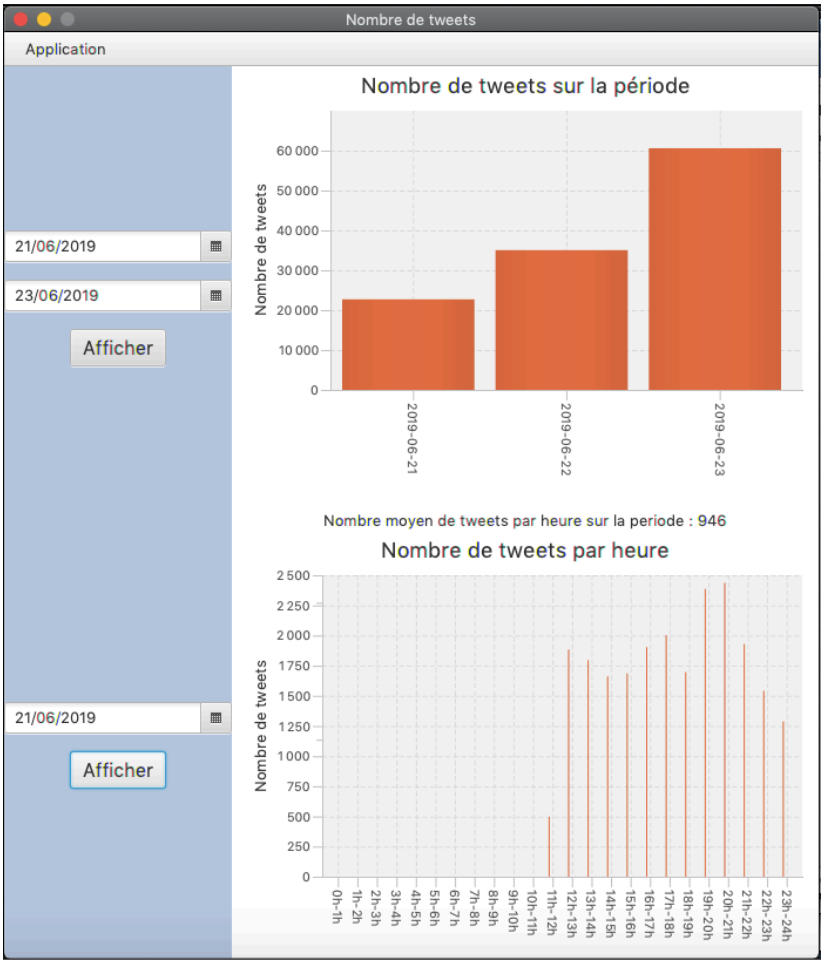
En raison du grand nombre de données contenues dans les deux fichiers proposées nous ne conseillons pas d'afficher l'ensemble des tweets ou l'ensemble des tweets sur un jour en raison du temps que l'affichage peut prendre



NOMBRE DE TWEETS

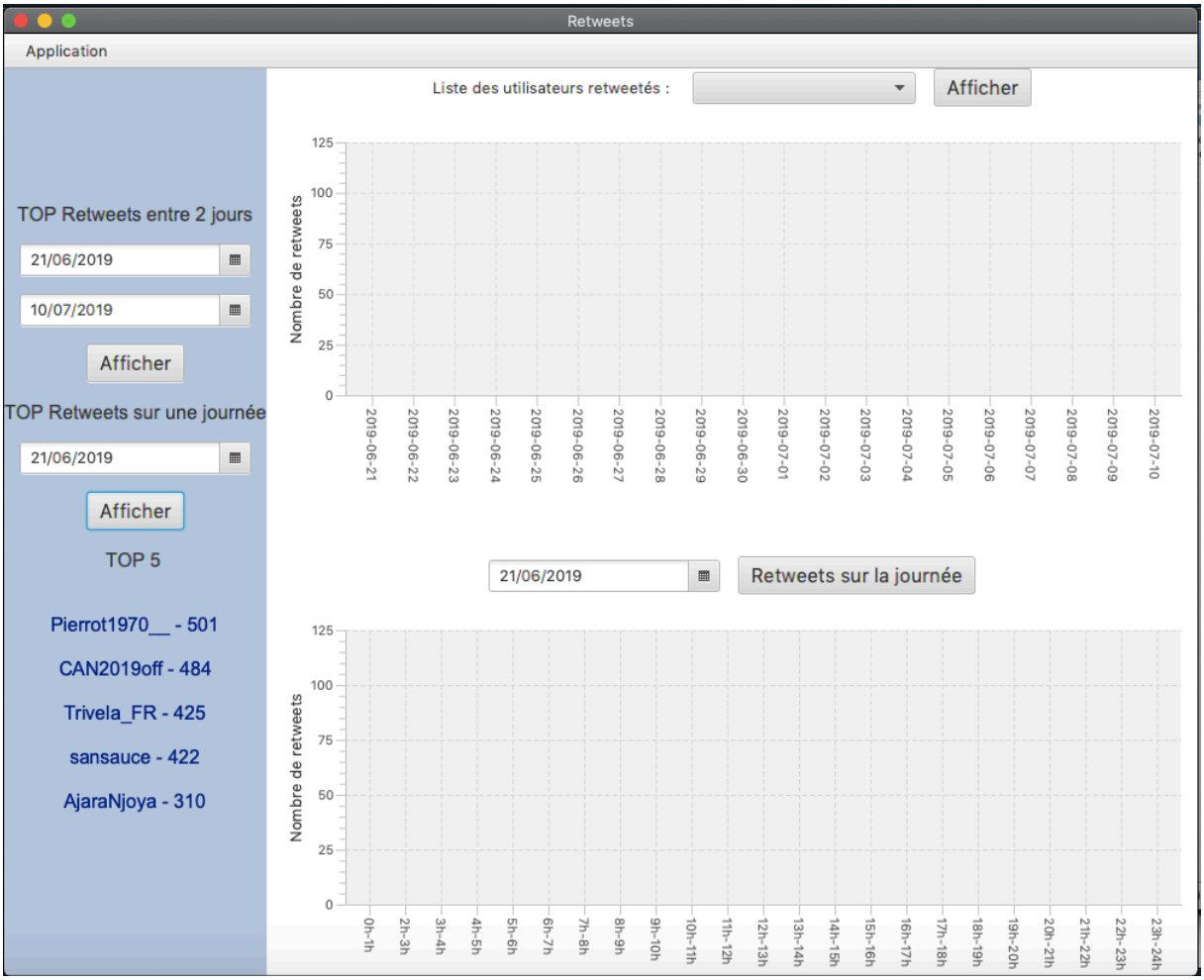
Dans la fenêtre s'ouvrant en cliquant sur le bouton « *Nombre de tweets* », l'utilisateur a deux possibilités. Tout d'abord il peut choisir de visualiser le nombre de tweets par jour sur la période qu'il souhaite ainsi que le nombre moyen de tweets par jour. Sinon, il a la possibilité de visualiser heure par heure le nombre de tweets sur le jour choisi.




Nombre de tweets
Application
21/06/2019
23/06/2019
Afficher
Nombre de tweets sur la période
Nombre de tweets
60 000
50 000
40 000
30 000
20 000
10 000
0
2019-06-21
2019-06-22
2019-06-23
Nombre moyen de tweets par heure sur la periode : 946
21/06/2019
Afficher
Nombre de tweets par heure
Nombre de tweets
2 500
2 250
2 000
1750
1500
1250
1000
750
500
250
0
0h-1h
1h-2h
2h-3h
3h-4h
4h-5h
5h-6h
6h-7h
7h-8h
8h-9h
9h-10h
10h-11h
11h-12h
12h-13h
13h-14h
14h-15h
15h-16h
16h-17h
17h-18h
18h-19h
19h-20h
20h-21h
21h-22h
22h-23h
23h-24h

RETWEETS

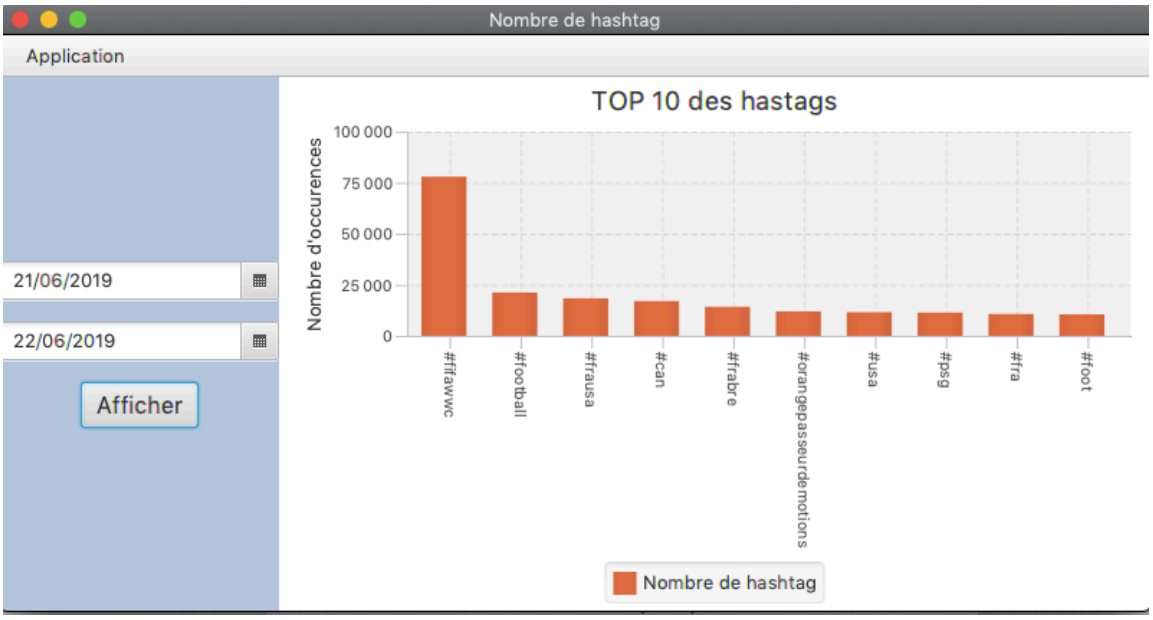
Dans la fenêtre *Retweets*, l'utilisateur peut regarder les « twittos » les plus retweetés sur une période ou un jour donné (partie droite de la fenêtre).



Il peut aussi sélectionner un « twittos » et afficher le nombre de fois où il a été retweeté par jour sur une période donnée ou bien par heure sur un jour donné.

HASHTAG

Dans la fenêtre *hashtag*, l'utilisateur peut regarder les dix hashtags les plus utilisés sur la période sélectionnée.



VALIDATION

TESTS INDIVIDUELS

Comme évoqué précédemment, nous nous sommes en général concentrés sur un aspect ou une analyse à la fois lors de la conception du programme. Ainsi, lorsque nous avons travaillés sur une méthode nous avons testé son fonctionnement au cours de sa réalisation et à la fin de celle-ci. Pour effectuer cette vérification, nous avons notamment eu recours à des affichages en console. Ces affichages ont de la même manière été réellement utiles quand nous avons commencé à implémenter l'interface graphique. Dans ce cas, l'enjeu a notamment été de vérifier si nous obtenions les mêmes résultats.

Nous avons aussi souvent testé sur des petites périodes (entre les tweets) les méthodes affichant les retweets, hashtags et nombre de tweets afin de savoir si elles fonctionnaient. Puis nous avons testé sur chaque fonctionnalité de prendre la plus grande période comportant des tweets afin de constater les différences de temps de traitement.

TESTS GENERAUX

Vers la fin du projet, nous avons lancé l'application, regardé et testé chaque fenêtre afin de déceler d'éventuels aspects à améliorer du point de vue graphique, de définir es propriétés adéquates pour les fenêtres (notamment la taille) et vérifier de nouveau le bon fonctionnement des méthodes. Nous ne sommes pas infaillibles et il est possible de ne pas se rendre compte directement de certaines erreurs. Lors de cette vérification, nous avons noté les points manquants ou à améliorer pour chaque fenêtre et avons par la suite modifié le code en conséquence. Par exemple, nous avons vu que dans certains cas nous n'avions pas géré l'exception liée à la division par zéro.

Lors de cette phase, nous avons aussi tenté de faire « planter » notre outil. Par exemple, nous avons tenté de définir une date de départ supérieure à la date d'arrivée pour certains de nos traitements. Ce cas provoque un avertissement en console mais ne perturbe pas le fonctionnement du programme en lui-même puisque qu'il n'affiche rien et ne s'arrête pas.

Par ailleurs, ces tests sur le fonctionnement global nous ont permis de remarquer un problème dans notre programme. Le nombre de retweets sur la journée ne fonctionne que sur la journée du premier tweet. Nous avons fait plusieurs tentatives pour corriger la fonction associée *lireRetweet (stuser,date)*, en vain. Ainsi, en l'état, cette fonctionnalité n'est pas opérationnelle.

CONCLUSION : EXTENSIONS ET AXES D'AMELIORATION

En conclusion, à l'heure actuelle, notre programme est fonctionnel. Toutefois, nous avons conscience que ce dernier reste perfectible et que plusieurs axes d'amélioration et extensions pourraient être envisagés à l'avenir. Il n'est pas impossible que la complexité de certains algorithmes, notamment celui sur le traitement des hashtags, puissent être réduites afin de réduire le temps de traitement. Il est également envisageable d'améliorer l'aspect visuel de l'outil. En effet nous avons essayé de rendre ce dernier agréable d'utilisation mais l'aspect « ergonomique » est passé au second plan par rapport à la réflexion sur l'analyse des données. De même, il pourrait être intéressant de rendre l'outil encore plus dynamique, ou de chercher des librairies se prêtant au reporting. Nous avions commencé à chercher des librairies pouvant être utiles à notre projet mais nous nous sommes rendus compte que nous pouvions finalement le réaliser sans. Enfin, il est totalement concevable de faire fonctionner l'outil sur d'autres jeu de données de Twitter à condition que ces derniers respectent la structure des données actuelle.

Il est également possible d'imaginer par la suite plusieurs extensions à notre outil. Par exemple, il pourrait être intéressant de présenter davantage de statistiques simples autre que la moyenne. De plus, il semblerait pertinent de permettre la visualisation de l'évolution de la présence d'un hashtag de la même manière que nous l'avons fait pour le nombre de tweets ou encore les utilisateurs retweetés.

Pour finir, outre les améliorations et extensions possibles présentées précédemment nous pensons avoir réaliser un outil fonctionnel et efficace qui permet une exploitation des jeux de données présentés et répond au sujet en utilisant des concepts propres à la programmation orientée objet vus en cours mais en incluant également quelques nouveautés.