Rapport Projet Encadré

Analyse de sentiments sur Twitter

Arthur Assima

Nordine El ammari

Université de Lille

Récupération de l’application

Notre application est disponible sur le GitLab de l’université à l’adresse suivante :

https://gitlab-etu.fil.univ-lille1.fr/elammari/pje_c_assima_elammari

Sommaire

Description du projet

I. Description de la problématique

Le projet consiste à créer une application capable de donner le sentiment général sur Twitter à propos d'un sujet en particulier.

Le sentiment d'un tweet peut être représenté par trois classes : positif, négatif, neutre. Afin de déterminer le sentiment à propos d'un sujet, une requête à l'API Twitter est faite avec le sujet en tant que mot clef. Un certain nombre de tweets à propos du sujet sont ainsi récupérés. Un compte Twitter développeur à du être créé au préalable pour avoir accès à l'API.

Une fois les tweets récupérés, il faut alors les analyser pour les répartir dans les différentes classes et ainsi en déduire le sentiment dominant à propos du sujet.

II. Description de l'architecture de notre application

L'application est divisée en plusieurs classes :

- Interface.java : Contient l'ensemble des composants graphiques de l'interface excepté pour la création des « lignes » pour l'affichage des tweets et des boutons d'annotation.

- Request.java : Contient toutes les méthodes nécessaires afin de récupérer un tweet via l'API Search de Twitter grâce à la librairie Twitter4J. Permet également d'activer le proxy ou non.

Classification.java : Contient l'ensemble des méthodes permettant la classification par mots-clés ainsi que pour la classification KNN.

Bayes.java : Contient l'ensemble des méthodes permettant les classifications Bayésiennes (Fréquence, Présence, Bi-gramme, Uni-gramme)

PieChart.java : Classe permettant d'afficher un diagramme en camembert pour voir la tendance des tweets d'une recherche à l'aide de la librairie JfreeChart.

Description du travail réalisé

I. API Twitter

Pour pouvoir interroger l'API Twitter, nous avons utilisés la librairie twitter4j. Cette librairie donne accès au singleton TwitterFactory, qui permet d'accéder à l'API Twitter ainsi qu'aux classes suivantes :

• Query : requête

• QueryResult : résultat d'une requête (ensemble de Status) • Status : tweet et toutes les informations associées

Nous avons choisi pour notre conception d'application de toujours gardé les QueryResult afin d'effectué des traitements sur les tweets.

Pour choisir le nombre de tweets récupérés nous avons utilisés la méthode de la classe Query setCount(int nb) et pour nous assurer que la langue des tweets serait le français nous ajoutons à la fin de chaque requêtes « Lang:fr ».

II. Préparation de la base d'apprentissage

a) Nettoyage des données

Afin de nettoyer les tweets avant de les ajoutés dans la base d'apprentissage nous utilisons la méthode nettoyage de la classe Interface. Cette méthode utilise la méthode String.replaceAll() qui attend en paramètre une expression régulière ainsi que la chaine de caractère qui va remplacer les champs correspondants à la regex.

b) Construction de la base

Notre base d'apprentissage est concrète c'est à dire qu'elle est uniquement stockée dans un fichier CSV appelé requests.csv. Donc, à chaque besoin des données de la base nous lisons le fichier afin d'extraire celles-ci.

Pour sa construction nous utilisons directement l'interface graphique, nous annotons à la main chaque tweet à l'aide de boutons radios puis avec le bouton Ajouter à la base tout en bas de l'interface. Un évènement déclenche l'enregistrement des tweets annotés dans la base.

III. Algorithmes de classification

a) Mots-clés et KNN

L'algorithme de classification par mots-clés consiste à compter le nombre de mots positifs et le nombre de mots négatifs pour cela nous disposons de deux fichiers texte contenant un ensemble de mots de chaque tendance.

Si le tweet contient plus de mots positifs alors on l'annote comme positif, s'il contient plus de mots négatifs alors on l'annote comme négatif et s'il y a autant de mots positifs que de mots négatifs alors on l'annote comme neutre.

L'algorithme KNN consiste à considérer la distance entre le tweet à annoter et les tweets dans la base et à en garder les k les plus proches.

La distance se calcule ainsi :

D(t1, t2) = (Nombre total de mots – nombre de mots communs) / nombre total de mots)

Une fois que l'algorithme permettant de garder les k plus proches voisins a été exécuté il faut maintenant déterminer la classe du tweet en regardant la classe majoritaire dans les k tweets les plus proches.

b) Bayes

Il y a plusieurs variantes à l'algorithme de classification bayésienne nous en avons codé toutes les variantes qui sont des compositions de : présence, fréquence, uni-grammes et/ou bi-grammes.

L'algorithme de Bayes repose sur des probabilités la formule de Bayes pour la classification par présence est celle-ci :

$P(neutral|t) = \{Produit\ m \in t\}\ P(m|neutral) \cdot P(neutral)$

Avec P(m|neutral) étant le nombre d'occurrence du mot m dans les tweets de la classe neutral divisé par le nombre total de mots des tweets de la classe neutral

Pour la fréquence c'est la même formule sauf qu'on met a la puissance du nombre d'occurrences du mot m dans le tweet à évaluer l'expression P(m|neutral).

Et, enfin pour les bi-grammes il s'agit juste de considérer un ensemble de deux mots comme étant un seul mot et d'appliquer les mêmes algorithmes.

IV. Interface graphique

Notre interface contient à son lancement un champ permettant d'y renseigner le sujet de notre recherche et un bouton permettant de lancer la recherche ainsi qu'une indication sur le nombre de requêtes restantes.

Une fois la recherche lancée tous les tweets sont affichés avec le nom d'utilisateur, le contenu du tweet ainsi que la date et l'heure de l'émission du tweet.

Sous chaque tweet il y a 4 boutons radios (Négatif, Neutre, Positif, Aucun).

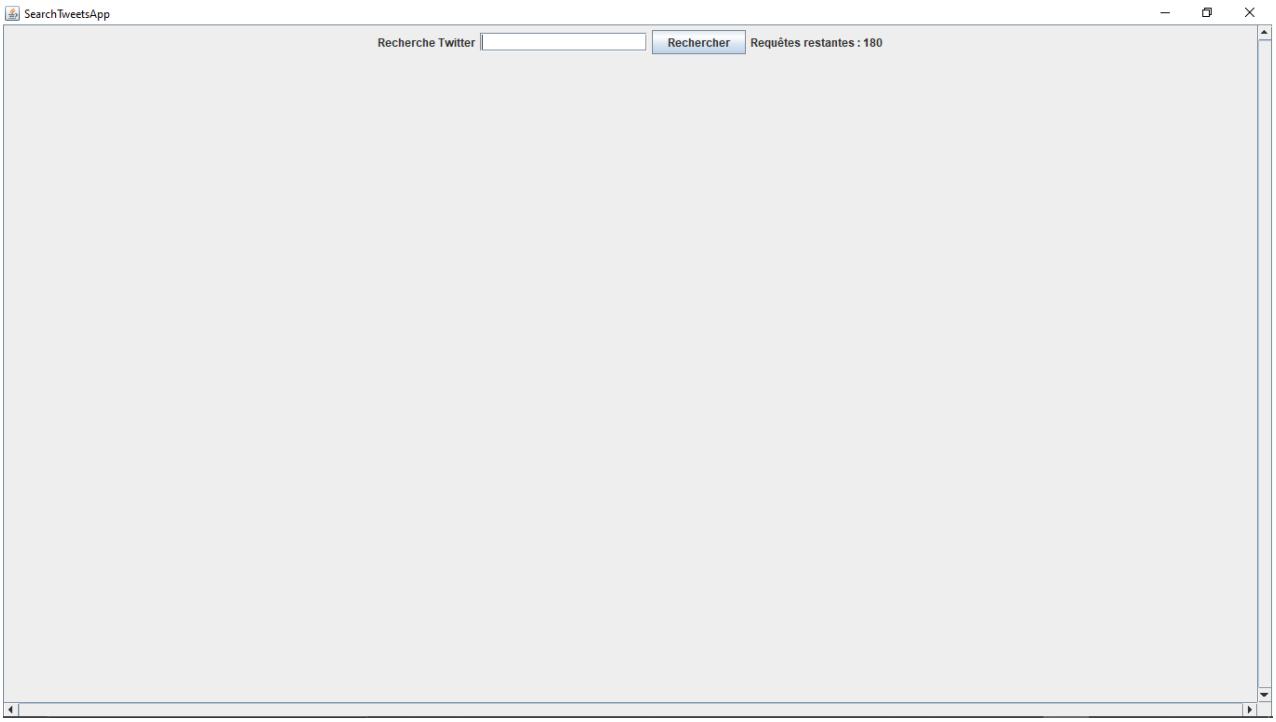
Un seul d'entre eux peut être coché.

Puis, en défilant tout en bas il y a les boutons permettant de lancer les différentes classifications. Ces boutons enregistrent la classification dans un fichier CSV à part et

compte le nombre de tweets appartenant à chaque catégorie pour pouvoir ensuite afficher le diagramme en camembert.

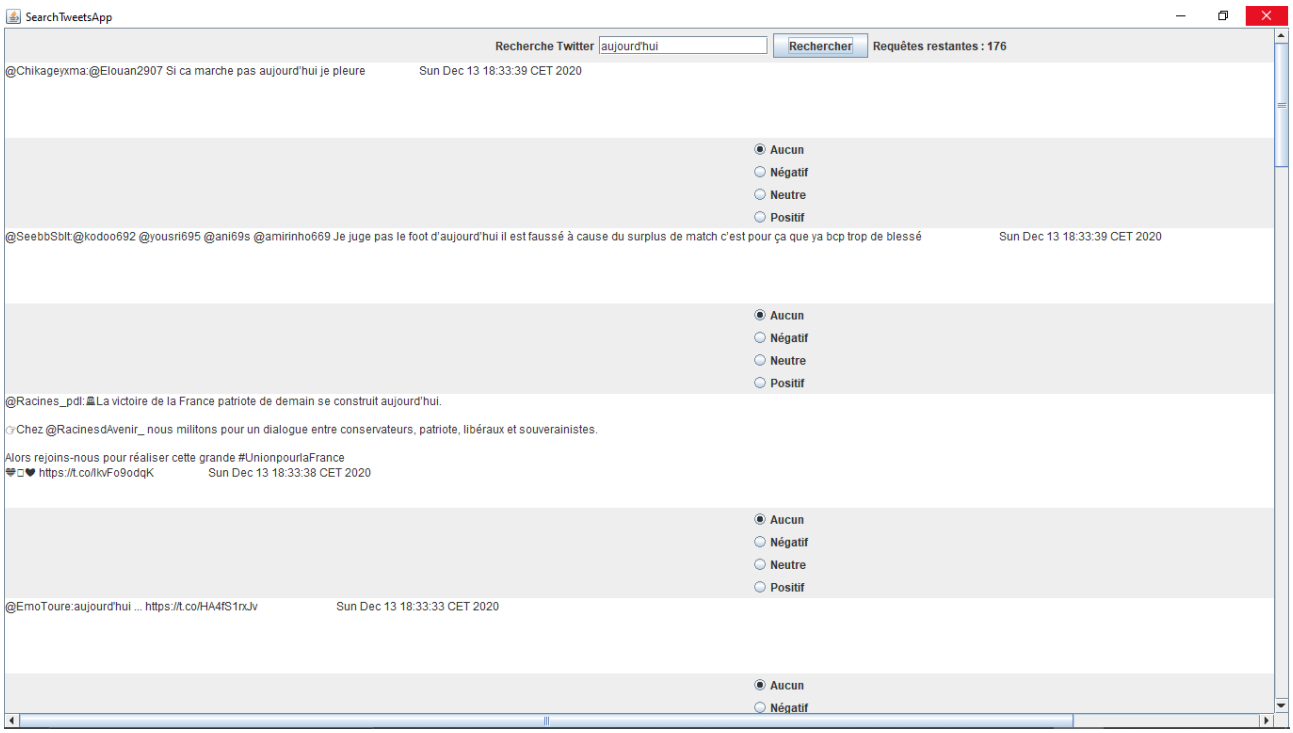
Enfin le dernier bouton, le bouton « Ajouter à la base » permet d'enregistrer les notations faites à la main dans la base d'apprentissage « requests.csv ».

- Copies d'écrans



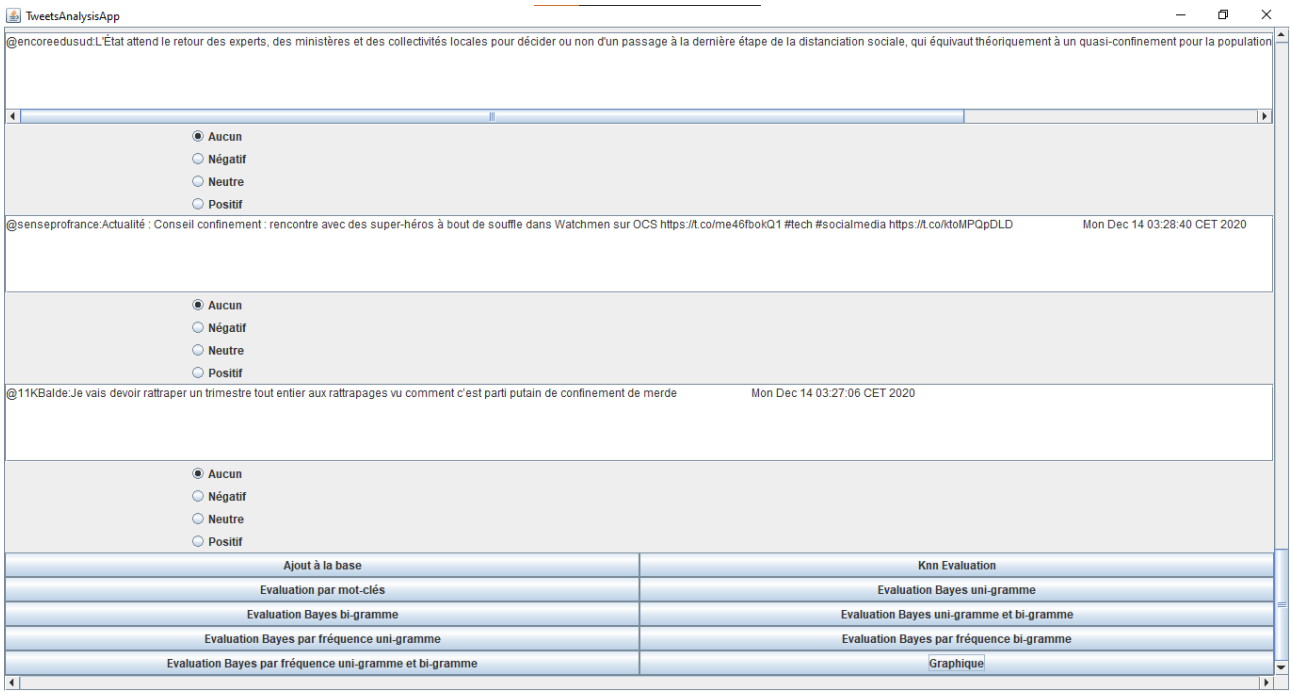
Écran au lancement de l'application.

On peut uniquement écrire une recherche et la lancer.

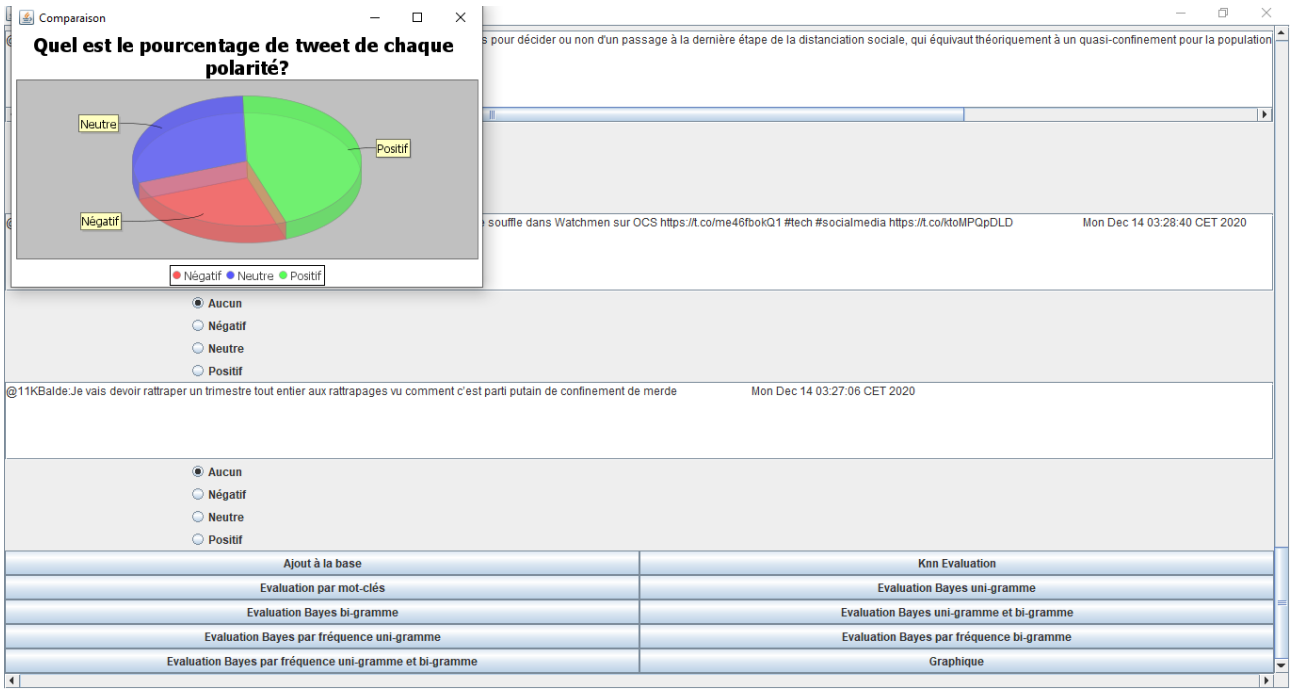


Écran après une recherche

Boutons d'annotations sous chaque tweet.



Tous les boutons sont situés en bas de l'interface.



Affichage du camembert permettant de se rendre compte des tendances après avoir appuyé sur un bouton d'évaluation suivi d'un appui sur le bouton « Graphique ».

Manuel d'utilisation

Pour lancer le projet il faut se placer à la racine et taper les commandes suivantes :

- mvn package
- mv target/pje….jar ./
- java -jar pje….jar

La commande mvn javadoc:javadoc permet de générer la javadoc

Une fois la fenêtre de recherche à l'écran, il faut taper la recherche que l'on veut effectuer dans le champ réservé et appuyer sur le bouton recherche à gauche du nombre de requêtes restantes.

Les tweets et les boutons permettant de les annoter apparaissent.

À partir de là on peut ajouter les tweets à la base d'apprentissage en les annotant manuellement au préalable ou simplement analyser les sentiments qui s'en dégagent.

Pour ce faire il suffit de cliquer sur un type d'évaluation. Ce clic déclenche la création d'un fichier qui contiendra tous les tweets et leur classe associée calculée grâce au type d'évaluation choisie ainsi que la possibilité de visualiser en cliquant sur le bouton « Graphique » de visualiser dans un diagramme camembert la répartition des sentiments liés aux tweets de la recherche.

Résultat de la classification

Les algorithmes de classification par mots-clés et KNN apparaissent comme les moins performants avec des taux d'erreurs supérieur à 60 %.

Au contraire, les algorithmes de classification bayésienne se sont montrés particulièrement performant avec des taux d'erreurs jusqu'à inférieur à 20%.

En effet, la classification bayésienne avec les uni-grammes nous donne un taux d'erreurs de 20 % pour la présence et 25% pour la fréquence.

La deuxième meilleure composition est celle avec la fréquence et les bi-grammes avec un taux d'erreurs de 34 %.

L'union des uni-grammes et bi-grammes montre un taux d'erreur d'environ 50%.

Enfin la pire classification bayésienne est celle composée de la présence de bi-grammes qui donne un taux d'erreurs de 74 %.

Les meilleurs classifieurs implémentés sont donc les BayesClassifier. Parmi eux, ce sont les BayesClassifier basés sur les uni-grammes qui s'avèrent être les plus efficaces. Il nous semble tout de même intéressant de revoir l'équilibre de la base d'apprentissage et sa taille pour pouvoir peut-être obtenir des résultats plus précis.

Conclusion

L’application nous permet de détecter de plusieurs manières distinctes les différents avis pour un sujet précis sur Twitter. Nous avons rencontré plusieurs de difficultés durant l'élaboration de cette application : tout d'abord dans sa simple réalisation en essayant de rendre l’interface agréable et cohérente. Mais surtout dans l'écriture des différents algorithmes de classification qui restent compliqués à valider et dont les résultats sont difficilement justifiables.

Cependant, la conception de cette application a été une expérience plutôt agréable pour nous. Elle nous a en effet permis de développer une application complète et utile, basée sur des algorithmes complexes et en totale autonomie.

Nous sommes également conscients que l'application est entièrement perfectible dans les détails. Une idée de fonctionnalité à ajouter serait par exemple de pouvoir manipuler plusieurs bases d'apprentissages distinctes et de choisir l'une d’entre elles pour tester rapidement les classifieurs sur des échantillons différents.