Master 1 Informatique

Projet Encadré

Analyse de comportement avec Twitter

Antoine CANDA
Theo VERSCHAEVE

Version 1.0
13 décembre 2016

Table des matières

Chapitre 1

Introduction

1.1 Problématique

Le but de ce projet fut de créer une petite application capable d'estimer le sentiment d'un tweet provenant du réseau social Twitter à propos d'un sujet donné. On a définit trois classes de sentiment pour simplifier la question qui sont la classe positive, négative et neutre.

La première chose à faire pour cela est de récupérer des tweets, on dispose pour cela de l'API TWitter qui permet de récupérer des tweets à partir d'un sujet défini par un mot clef.

Une fois que l'on a récupéré un certain nombre de tweets, il va falloir les nettoyer puis les analyser selon différentes méthodes pour estimer la classe de sentiment que l'on peut associer à chacun des tweets récupérés.

1.2 Architecture de l'application

On est parti à la base sur une conception autour du pattern MVC, on y retrouve donc les trois packages controller, model et view. On y a ajouté le package tools et le package app. On a décidé d'utiliser le langage JAVA pour réaliser notre application.

Le package **app** contient seulement la classe Main de l'application.

Le package **controller** contient la classe Controller représentant le contrôleur de l'application et un sous-package action contenant les classes représentant les différentes actions possible comme sauvegarder les tweets, lancer la recherche... Le contrôleur de l'application assure la gestion des événements de synchronisation pour notamment mettre à jour l'interface et le modèle ainsi qu'assurer le lien entre les deux.

Le package **model** contient deux classes qui sont la classe Model et la classe RechercheTwitter ainsi que le package classifieurs représentant toute les classes ayant comme rôle d'attribuer un sentiment à un tweet ainsi que la classe Evaluateur permettant d'évaluer ces derniers. Le modèle représente le cœur algorithmique de l'application.

Le package **view** contient différents packages et classes ayant comme rôle la création de l'interface graphique et le rendu visuel de l'application. La vue est le cœur interactif de l'application vu que toutes les actions sont lancés à partir de la vue et le résultat y est retranscrit.

Le package **tools** contient des classes outils permettant notamment de représenter la base de tweet, un tweet (gain d'un niveau d'abstraction par rapport à la classe Status qui représente un tweet dans l'API Twitter ainsi qu'une note qui est la représentation des classes de sentiments que l'on veut attribuer.

Il y a également un dossier **ressource** qui contient les fichiers représentant les mots positifs et negatifs nécessaire à la méthode des mots clefs, la base d'apprentissage sous la forme d'un fichier CSV et des fichiers CSV qui correspondent aux tweets nettoyés après une recherche et ces même tweets annotés par le classifieur utilisé. Ces fichiers sont de la forme Tweets-MOTRECHERCHE.csv et Tweets-MOTRECHERCHE-annote.csv

Chapitre 2

Présentation des travaux

2.1 API Twitter

Il est nécessaire d'avoir accès à des tweets dans l'application que l'on veut créer, on va donc utiliser pour cela utiliser l'API Twitter. Pour pouvoir l'interroger on a utilisé la librairie *twitter4j*.

Cette librairie nous a donné accès à la classe **TwitterFactory** qui nous permet d'interroger l'API Twitter. On a également accès aux classes :

- **Query** : permettant de faire une requête selon un mot clef.
- **QueryResult** : permettant d'obtenir le résultat de la requête
- **Status** : représentant un tweet et ses méta données.

Ces classes sont utilisées afin d'obtenir les tweets associés à une recherche donnée dans la classe RechercheTwitter et plus particulièrement dans la méthode *search*.

Cette méthode renvoie une liste de tweet et non de status : chaque status devient une instance de la classe Tweet. On fixe la langue avec la méthode setLang sur "fr" et on permet de modifier le nombre de tweets récupérés par une requête selon différents paliers (25,50,75,100 et 250) avec par défaut 50 tweets.

2.2 Préparation de la base d'apprentissage

Certaines méthodes que l'on va utiliser pour annoter les tweets nécessitent l'utilisation d'une base d'apprentissage qui est nettoyée et annotée par l'utilisateur afin de servir d'exemple à notre application. Nous allons montrer comment nous avons procédé à crée la notre.

2.2.1 Nettoyage des données

Nous avons utilisé des expressions régulières pour nettoyer les tweets que l'on souhaite traiter. Nous avons pour cela utiliser les classes Pattern et Matcher de Java. Nous supprimons

en autre les hastags, arobase qui servent de référence vers un autre utilisateur, on supprime les URLs, les chiffres, les espaces supplémentaires et guillemets.

Il est fort probable qu'il soit facilement possible d'améliorer le nettoyage des données.

2.2.2 Construction de la base

Nous avons crée une classe BaseTweet qui représente notre base d'apprentissage. On l'a implémenté sous forme d'une hashmap avec en clé l'identifiant du tweet qui est sensé être unique, et en valeur le tweet lui même. La base de tweet est crée par le contrôleur à l'ouverture de l'application.

On peut décider d'agrandir la base d'apprentissage en effectuant une recherche et en sélectionnant l'onglet "Annotation manuelle", on a alors la possibilité d'attribuer pour chaque tweet une note et pour chaque tweet on a la possibilité de le conserver dans la base ou non. L'écriture sur le fichier se fait lorsqu'on clique sur le bouton sauvegarder Tweets présent sur l'application. On aurait pu également décidé de réécrire la base lors de la fermeture de la fenêtre mais on a estimé qu'il n'y avait pas forcément raison de faire cela car ce n'est pas le but premier de l'application de construire la base d'apprentissage.
J'ai crée trois bases : la première porte sur les tweets obtenus avec la recherche associée au mot "France". Le problème est qu'au final les sujets des tweets étaient fort variés. La base fait environ une centaine de tweets globalement équilibrés dans la représentation des classes (1/3).
La seconde base contient les tweets de la première ainsi que d'autres tweets cette fois ci sur de très nombreux sujets mais toujours annoté par une seule et même personne. Une fois de plus, elle est globalement équilibré dans la représentation des classes et contient environ 260 tweets.
La troisième et dernière base contient les tweets des précédentes ainsi que d'autres tweets sur d'autres sujets encore une fois mais cette fois annotés par d'autres personnes avec possiblement d'autres sensibilités ou méthodes d'annotations. Je ne sais pas spécialement quel est la représentation de chaque classe mais elle comporte plus de 550 tweets.

2.3 Algorithmes de Classification

Tous les classifieurs héritent de la classe abstraite classifieurs. Cette classe possède uniquement la méthode abstraite classifie qui attribue une Note (une classe de sentiment) au message du tweet.
Seule la classe Dictionnaire qui représente la méthode des mots clefs hérite directement de cette classe car il s'agit de la seule méthode qui ne requière pas l'utilisation de la base d'apprentissage.

On a donc crée une classe abstraite ClassifieurBase qui intègre la base de Tweet et sert de classe mère pour les classifieurs basés sur les méthodes des plus proches voisins (KNN) et Bayésiennes.

2.3.1 Méthode des mots clefs

La méthode par mot clé ou dictionnaire est la méthode naïve de classification, le principe est simple : on dispose d'une liste de mot positif et une de mot négatif. On sépare notre message en liste de mot et on regarde pour chaque mot si il est présent dans la liste des mots positif ou négatif. Si c'est le cas on incrémente ou on décrémente un compteur puis à la fin on regarde la valeur de ce dernier :

- La valeur est **négative** : on a eu plus de mots jugés négatifs que positif et on attribue donc la note **Négatif** au tweet
- La valeur est **nulle** : on a un équilibre entre les mots positifs et négatifs ou aucun de chaque. On attribue donc la note **Neutre**.
- La valeur est **positive** : on a plus de mots jugés positifs que négatifs et on attribue donc la note **Positif**.

Cette méthode ne peut être "efficace" que dans le cas où on est rigoureusement exhaustif sur les listes de mots négatifs et positifs, mais il faut tenir compte du fait que l'on prenne en compte que l'orthographe peut jouer un rôle avec notamment la prise en compte d'abréviation... Les performances deviendraient vite mauvaise dans ce cas.

2.3.2 Méthode des plus proches voisins (KNN)

La méthode des plus proches voisins repose sur le calcul d'une distance entre deux tweets. Il y a plusieurs façon de calculer cette distance, nous avons ici fais celle présentés à savoir le nombre total de mot entre le tweet étudié et le tweet de la base auquel on soustrait le nombre de mots en commun et on divise le tout par le nombre total de mot. C'est à dire que deux tweets identiques auront une distance de 0.5.

On décide dans notre cas de travailler sur la base de 5 voisins, on regarde dans les cinqs tweet de la base les plus proches de notre tweet quelle est la classe la plus représentée et on lui attribue cette classe.

On a crée pour cela une classe CoupleDistanceTweet qui associe la distance à un tweet.
On comprend facilement que le point important est essentiel de cette méthode est la méthode de calcul de distance.

2.3.3 Méthodes bayésiennes

Les classifieurs basés sur les méthodes bayésiennes repose sur une classe NGramme représentant le degré que l'on va associer à notre méthode, on a décidé de travailler sur les unigrammes, les bigrammes et les uni+bigrammes. Cette classe NGramme contient la méthode ConstruitNGramme qui va construire à partir du message du tweet les différents NGramme possible sur lesquels on va effectuer tout les calculs de probabilités des méthodes bayésiennes.

On a définit une classe abstraite BayesClassifieur qui contient toute les méthodes communes aux deux types de classifieurs bayésiens que l'on souhaite construire à savoir un classifieur basé sur la présence et un autre sur la fréquence. La méthode abstraite est celle qui calcule la probabilité que le message du tweet passé en paramètre ait la même note que la note passé

en paramètre. C'est cette méthode qui diffère légèrement entre les deux types de classifieurs et on l'implémente de manière séparée dans les deux classes associées.

On a la possibilité de simplifier les tweets c'est à dire de retirer tout les mots de moins de trois lettres qui correspondent notamment aux déterminants dans la langue française. On a donc un total de 12 classifieurs basés sur les méthodes bayésiennes.

2.4 Interface graphique

L'interface de l'application se décompose en deux parties principales. La première partie est la barre des onglets servant de barre menu et permettant de naviguer entre les différentes vues de l'application. Il y a 5 onglets de présents :

- Recherche : permettant d'effectuer une recherche sur un sujet et récupérer des tweets en sélectionnant le nombre voulu.
- Annotation manuelle : permettant d'annoter la liste des tweets obtenue afin de créer ou compléter une base d'apprentissage.
- Annotation automatique : permettant de noter la liste des tweets obtenue selon l'algorithme courant et afficher le diagramme de représentation des classes.
- Évaluation algorithme : permettant d'évaluer les algorithmes d'annotation pour en déterminer une qualité.
- Réglages : permettant de sélectionner l'algorithme de classification courant pour l'annotation et l'évaluation.

2.4.1 Copie d'écran

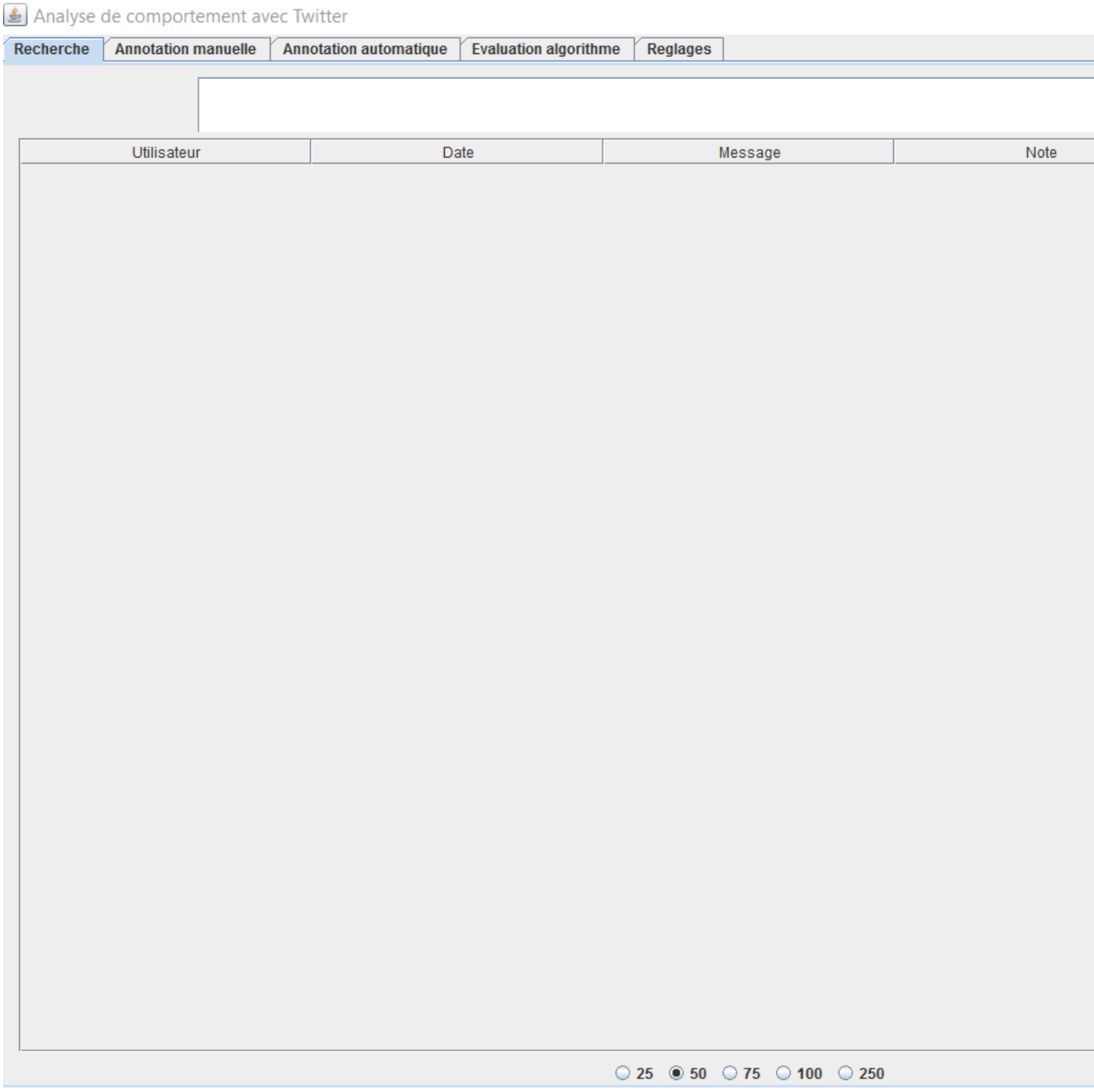

FIGURE 2.1 – Écran d'accueil de l'application

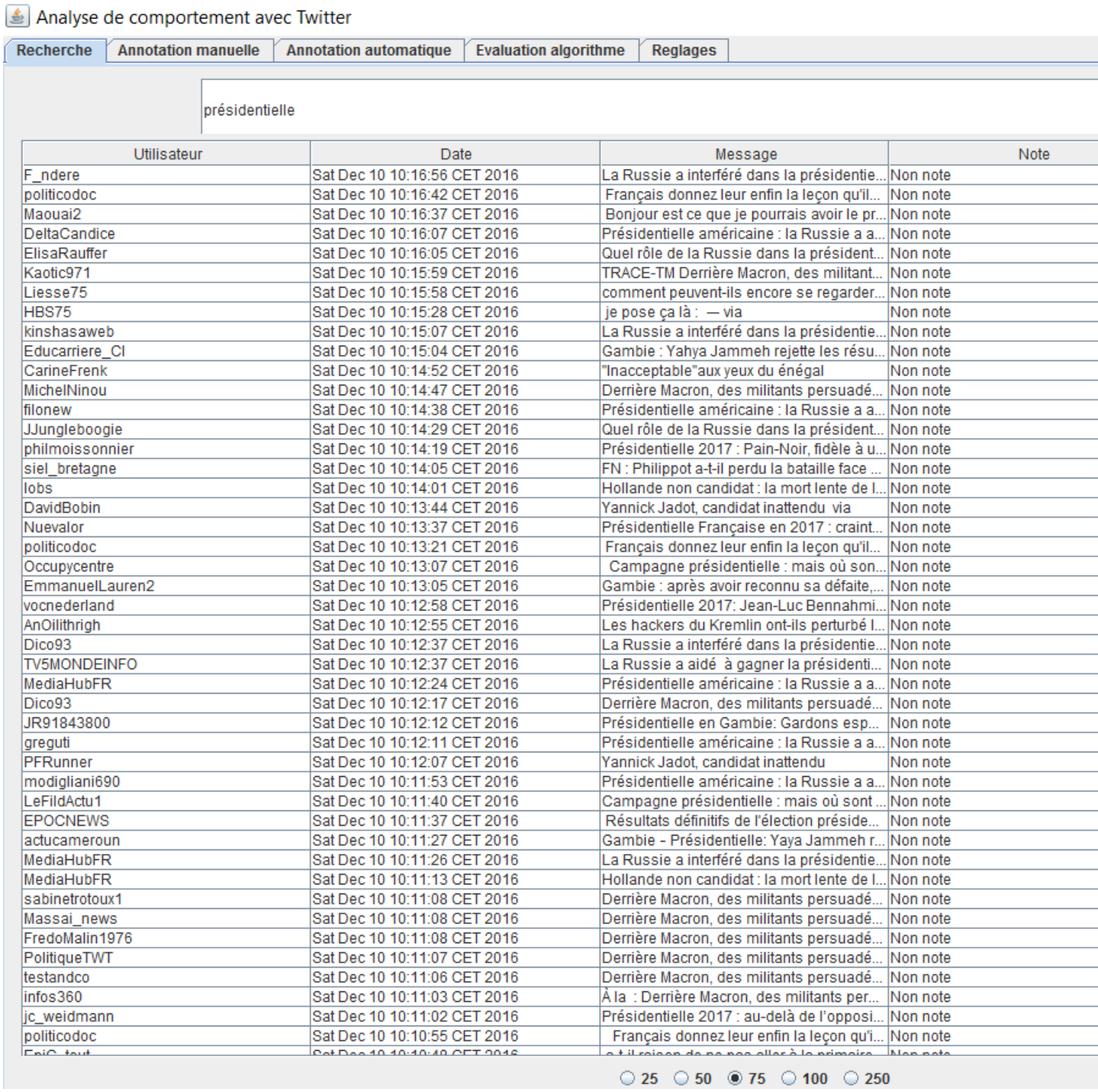

FIGURE 2.2 – Écran présentant l'affichage des tweets issues d'une recherche

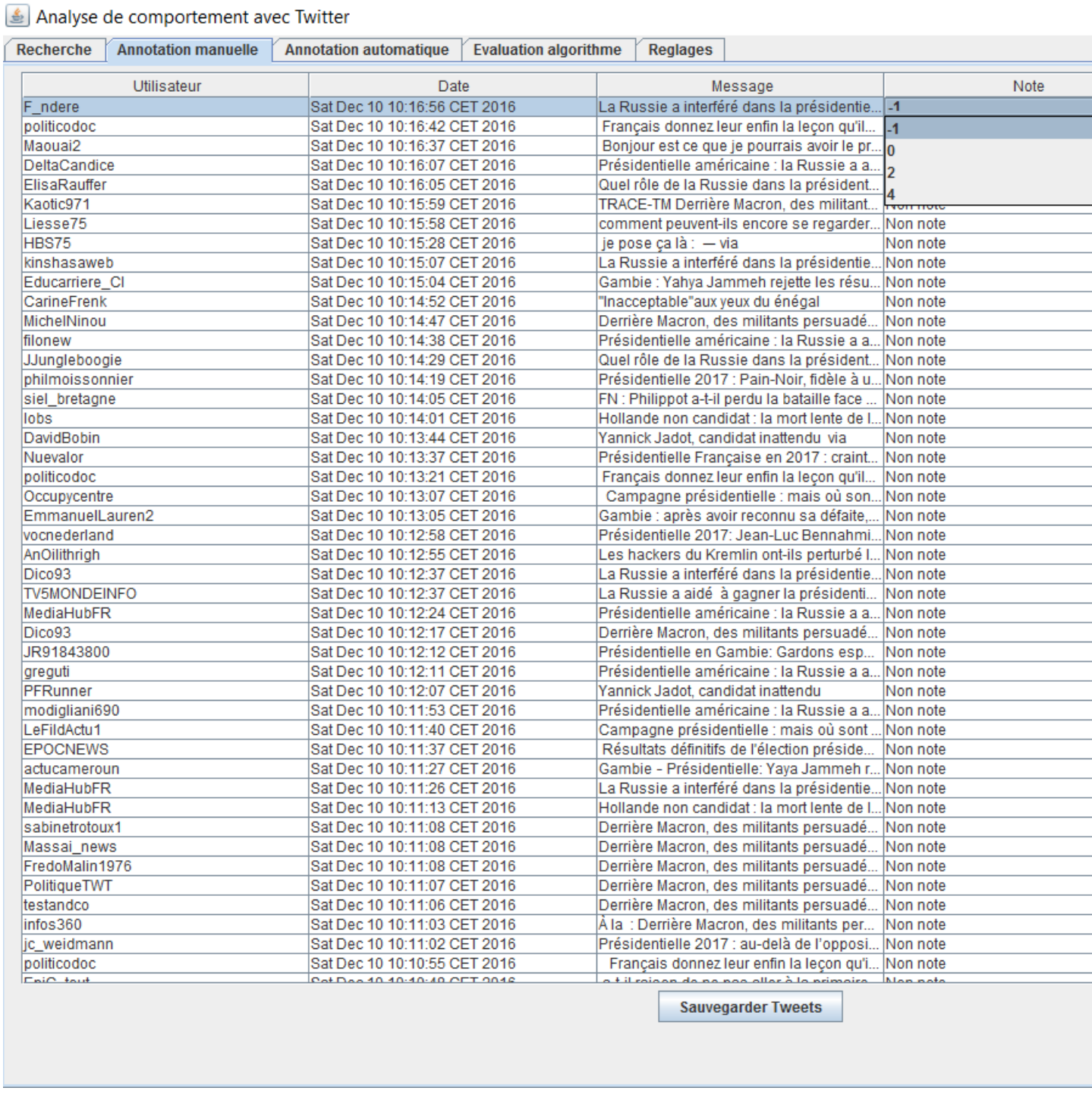

FIGURE 2.3 – Écran présentant le choix de la note d'un tweet pour la base

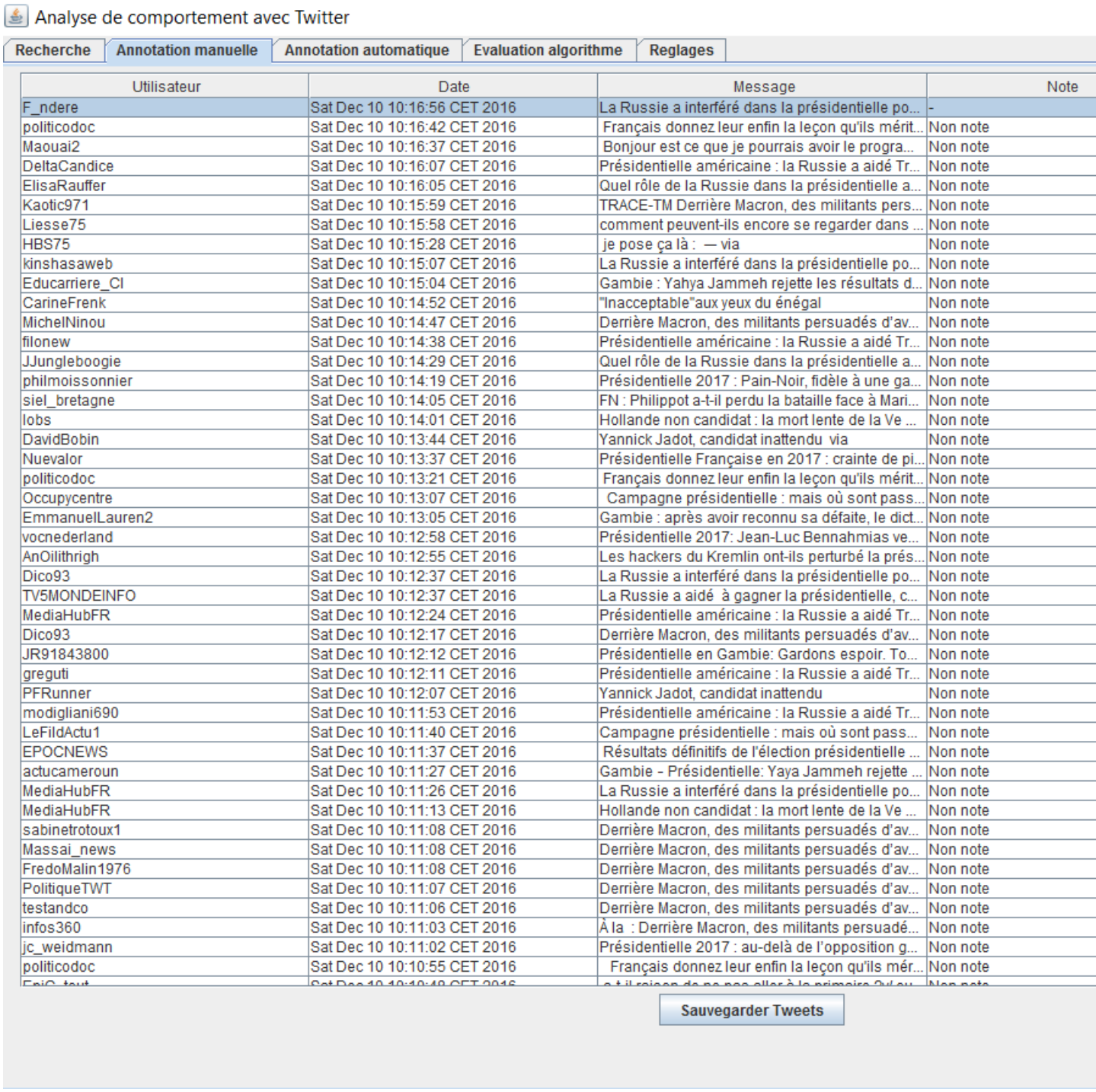

FIGURE 2.4 – Écran présentant le choix de conservation d'un tweet dans la base

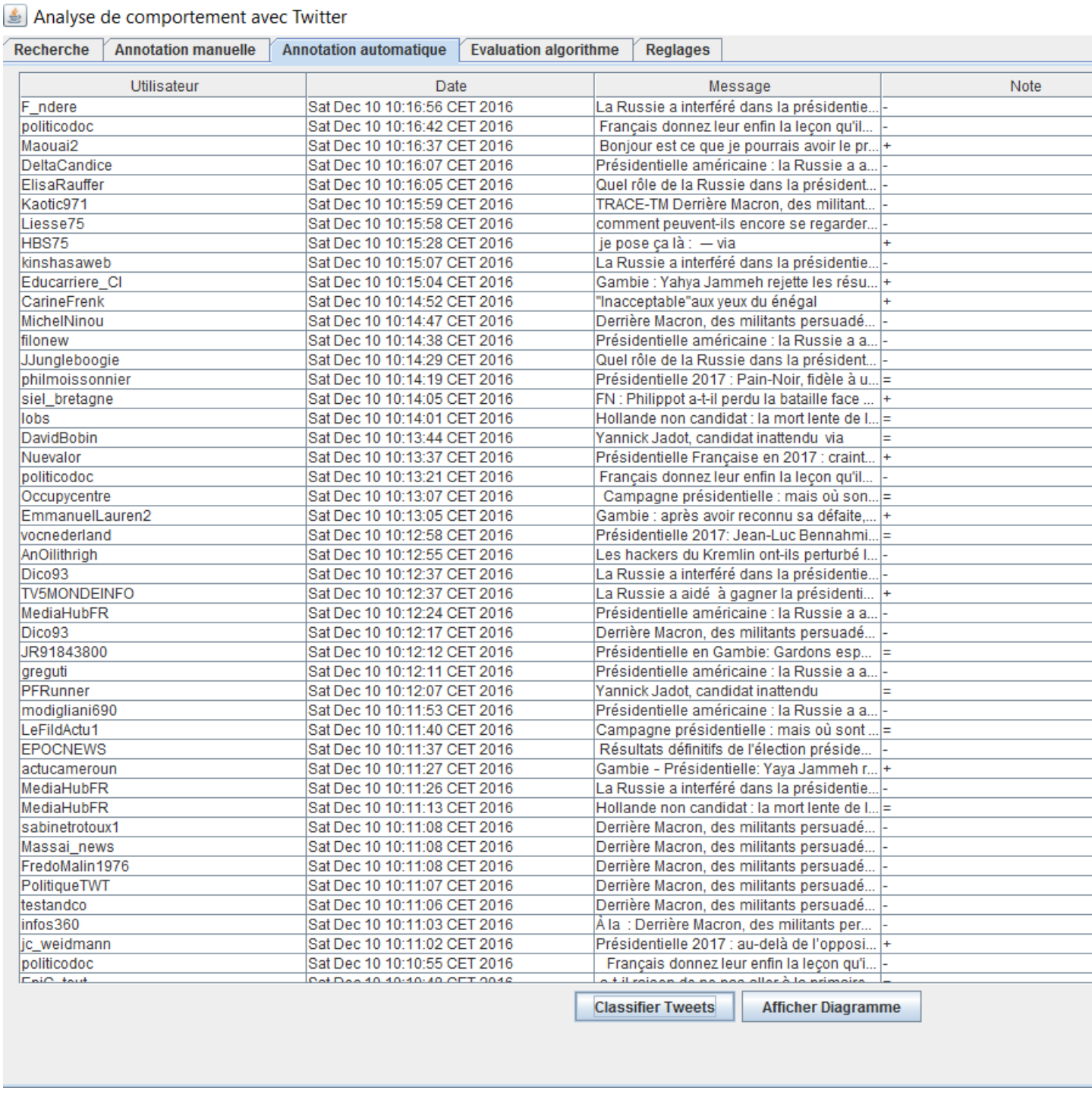

FIGURE 2.5 – Écran pour la classification automatique

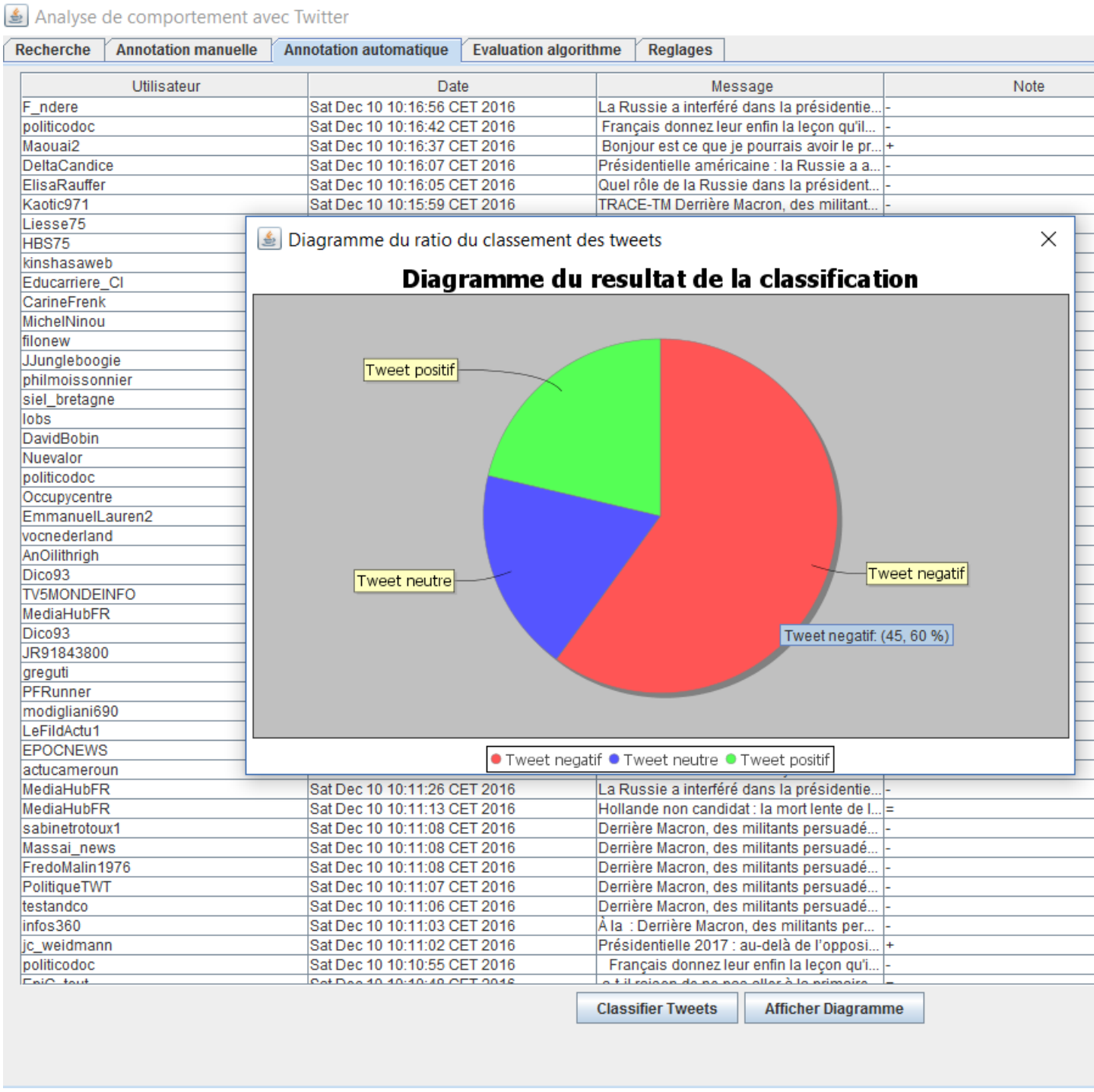

FIGURE 2.6 – Écran avec l'affichage du diagramme présentant le résultat de la classification

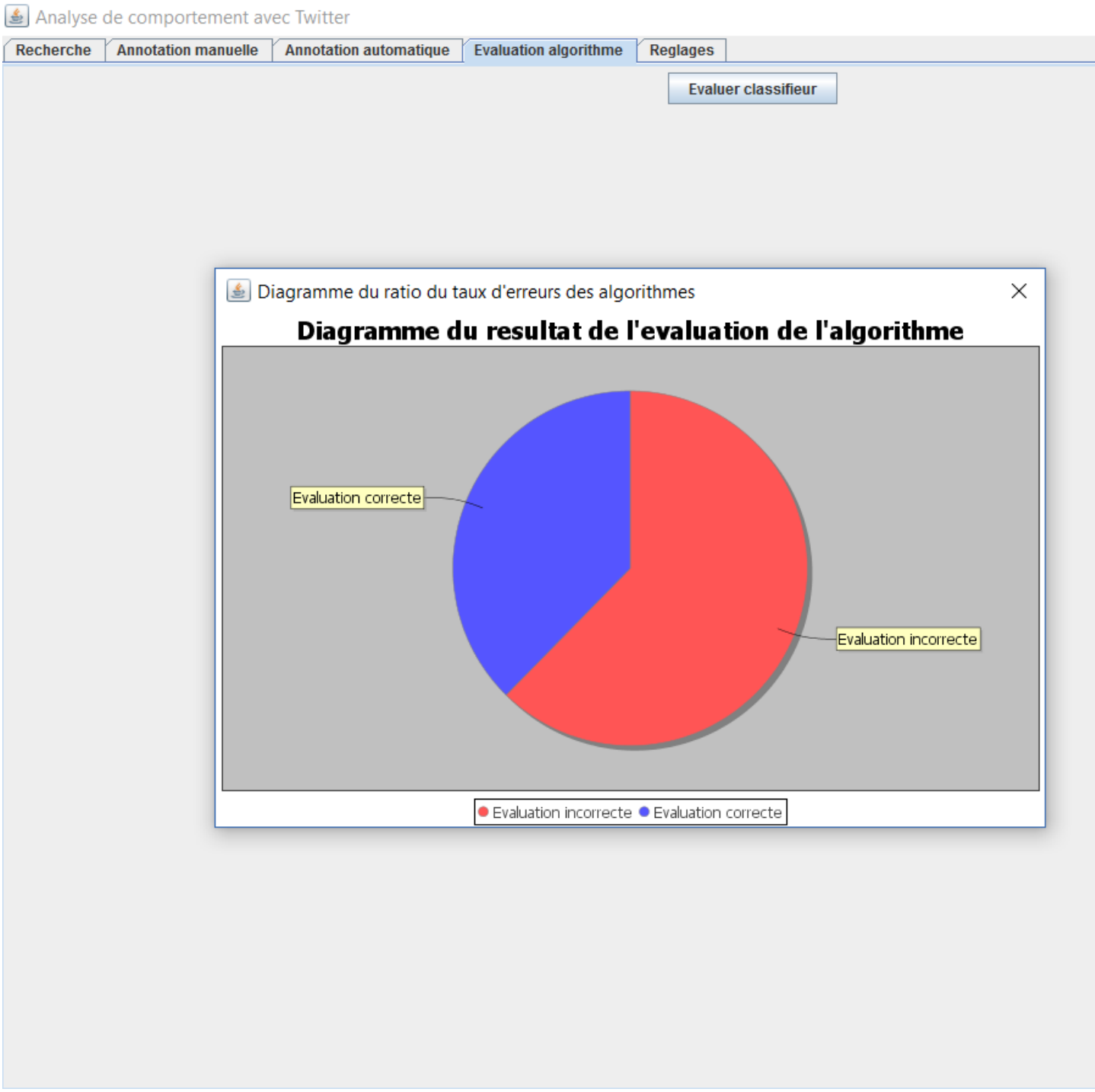

FIGURE 2.7 – Écran d'évaluation des algorithmes de classification

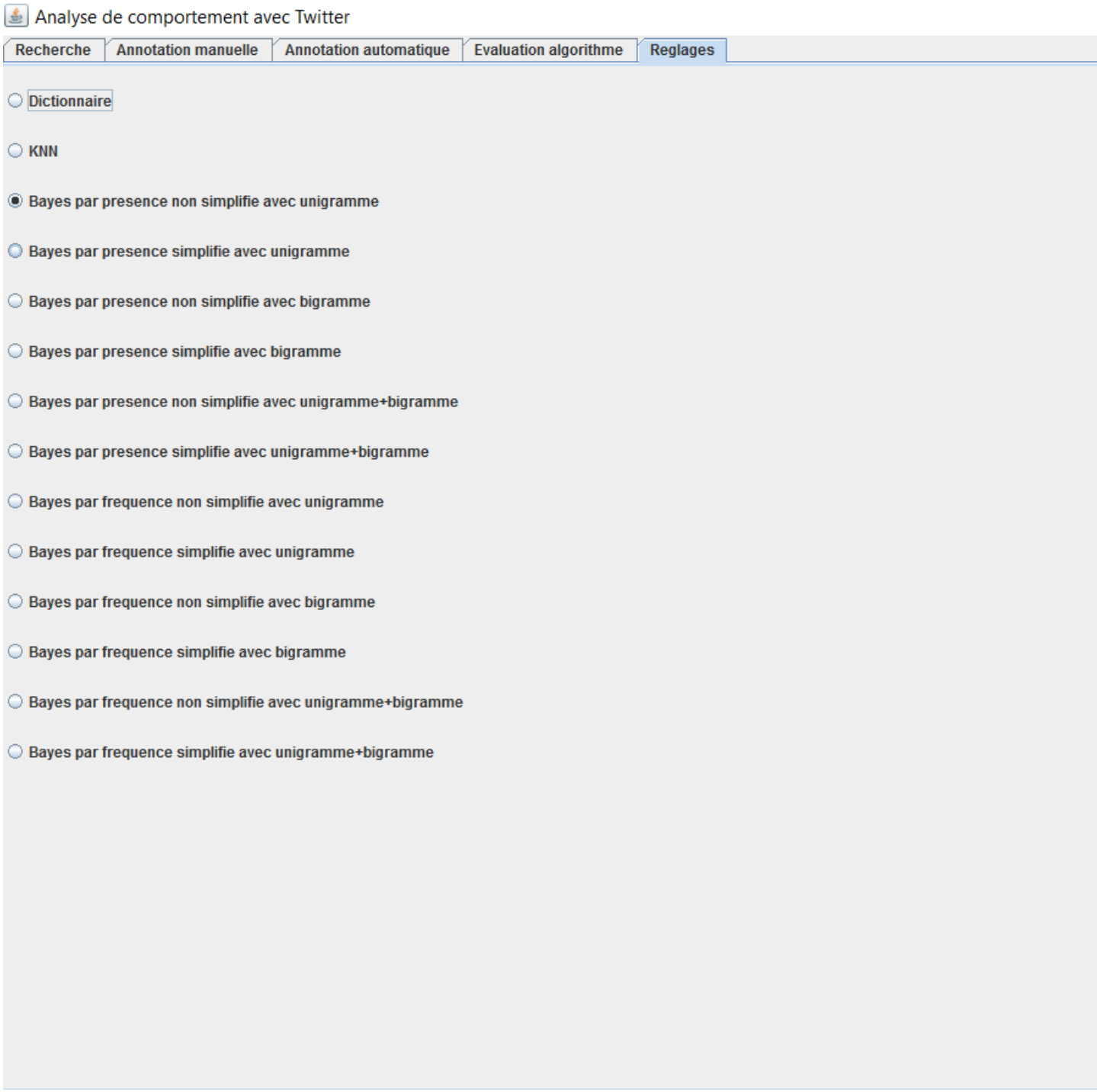

FIGURE 2.8 – Écran du choix de classificateur

2.4.2 Manuel d'utilisation

L'application nécessite d'avoir au moins Java 7 pour fonctionner.

L'application est fournie sous forme d'un fichier JAR avec un dossier ressource contenant

les différents fichiers essentiels au bon fonctionnement de l'application.

L'exécution s'effectue avec la commande suivante :

java -jar analyseTweet.jar

L'onglet Recherche permet de sélectionner le nombre de tweets que l'on veut lors de la recherche que l'on effectue entre 25, 50, 75, 100 et 250 tweets. Il permet également de renseigner un mot clé dans l'encart supérieur et de lancer la recherche avec le bouton "Rechercher".
Le résultat de la recherche va s'afficher dans la partie centrale de l'écran.

Le second et troisième onglet nécessite que l'on ait effectuée une recherche pour avoir un affichage et une interaction avec l'utilisateur.

Le second onglet qui se nomme "Annotation manuelle" va permettre à l'utilisateur de créer sa base d'apprentissage en choisissant pour chacun des tweets que l'on souhaite annoté une note et la valeur booléenne true dans la colonne "Conserver dans la base d'apprentissage".

Il suffira une fois l'annotation des tweets terminées de cliquer sur le bouton "Sauvegarder Tweets".

Le troisième volet se nomme "Annotation automatique" et il permet d'automatiser l'annotation des tweets. Le classifieur de base est le classifieur basé sur la méthode Bayes Présence sur unigramme avec simplification des tweets.

Il est possible de changer à volonté le classifieur utilisé dans l'onglet Reglages. Il n'est pas obligatoire de faire une nouvelle recherche pour essayer un autre classifieur ou pour l'évaluer.

Pour lancer la classification, il faut avoir effectuer une recherche et ensuite cliquer sur le bouton "Classifier Tweets". Dans la colonne note apparaîtra pour chaque tweet l'une des classes suivantes :

— + pour désigner un tweet noté positivement

— = pour désigner un tweet noté neutre

— - pour désigner un tweet noté négativement

Il sera ensuite possible de créer un diagramme représentant les pourcentages des notes obtenues en cliquant sur le bouton "Afficher diagramme".

Il reste l'onglet "Évaluation algorithme" qui se compose d'un bouton "Évaluer classifieur" qui affiche le taux de succès de chaque algorithme lors de l'évaluation sur la base d'apprentissage.

Chapitre 3

Résultats obtenus avec les différentes méthodes et analyse

Les résultats ne sont malheureusement pas très bon dans la globalité.

En effet, quelques soit la méthode utilisée, je ne passe pas sous la barre des 50% de taux d'erreurs, et la méthode naïve (dictionnaire) propose des résultats équivalents ou presque à ceux basés sur la méthode bayésienne quand je fais l'évaluation sur la base d'apprentissage.

Il est important de souligner le point suivant : les nombres avancés sont les résultats les plus favorables obtenus lors des différents tests (différentes bases, bases avec sujet mélangés ou non...), et ils ont été obtenus avec une base d'environ 550 tweets sur un nombre de sujet importants allant de la politique, au couleur, à l'hiver, au matin, examens, disney... et que les résultats moyens étaient souvent de l'ordre de +/- 5%. La génération des ensembles étant pseudo-aléatoire, je prenais une moyenne sur X calculs, généralement $10 < X < 100$ selon la taille de la base...

La méthode dictionnaire affiche un taux d'erreur environ égale à 60%.
La méthode KNN affiche un taux d'erreur environ égale à 70%.
Les méthodes bayésiennes basées sur la présence et les unigrammes affichent un taux d'erreur inférieur à 60%.
Les méthodes bayésiennes basées sur la présence et les bigrammes affichent un taux d'erreur égale ou très légèrement supérieur à 60%.
Les méthodes bayésiennes basées sur la présence et le couple uni+bi gramme affichent un taux d'erreur compris entre 55 et 60%.
Les méthodes bayésiennes basées sur la fréquence et les unigrammes affichent un taux d'erreur égale à environ 55%.
Les méthodes bayésiennes basées sur la fréquence et les bigrammes affichent un taux d'erreur égale ou très légèrement supérieur à 60%.
Les méthodes bayésiennes basées sur la fréquence et le couple uni+bi gramme affichent un taux d'erreur compris entre 55 et 60%.

On remarque qu'il y a une équivalence entre les résultats obtenues et le type de n-gramme utilisé.

Le problème que je note est que les résultats de l'évaluation sur la base d'apprentissage ne se retrouve absolument pas dans les résultats lors de la notation d'une liste de tweets récupérés. En effet, la méthode Dictionnaire se révèle beaucoup moins efficace avec prêt de 80% des tweets classés neutre la plupart du temps contre des résultats bien plus faible avec les algorithmes bayésiens (souvent inférieur ou égal à 30%) **sauf** pour les algorithmes basés sur les bigrammes simplifiés approchant d'un score proche ou égal à 100% en neutre.

J'en déduis que j'ai probablement soit une erreur dans mes algorithmes bayésiens ou dans mon évaluateur, soit j'ai une base d'apprentissage qui est biaisée/problématique.

Les tests ont été effectués sur différentes base d'apprentissage : avec plus ou moins de tweets, sur différents sujets ou un sujet unique et les résultats étaient relativement équivalents.

Une théorie avancée pour expliquer le résultat est la difficulté de noté un tweet qui peut dans certains cas se rapprocher de la méthode naïve (un peu plus positif que négatif...) ou la prise en compte selon le sujet d'une sensibilité "variable" notamment par rapport à l'ironie, une sensibilité politique... au contexte d'un tweet par rapport à des termes fortement connotés tel que terrorisme, attentat, mort qui parfois avec un mot placés ailleurs dans la phrase change totalement le contexte et que l'on ne retrouve pas dans la notation... Les tweets de la base étant sélectionnés parfois en fonction de ces caractéristiques il n'est pas surprenant que l'on perde l'algorithme quand on évalue sur ces mêmes données. Cette idée se renforce quand je compare l'annotation proposés sur une liste aléatoire et celle que j'aurais mis dans le cas d'une annotation dans prêt de 2 cas sur 3.

Chapitre 4

Conclusions

Cette UE fut l'occasion d'améliorer mes compétences en matière de développement notamment avec une utilisation plus importante de librairies externes et la création d'une interface dédiée à l'application, ainsi que dans la conception que l'on souhaitait un minimum correcte : on sait que dans les détails il y a beaucoup de points à revoir et retravailler mais on a manqué de temps pour le faire. Ce fut également l'occasion de rencontrer pour la première fois le design pattern MVC et d'essayer de s'y familiariser.

L'écriture des algorithmes de classification fut aussi parfois un défi dans la compréhension de ces derniers et surtout dans leur validation qui est compliquée à faire/justifier. J'aurais aimé faire des tests unitaires et d'intégrations de l'application mais je n'ai pas eu le temps de pouvoir le faire malheureusement. Et les résultats semblent peu cohérent dans certains cas.

Mais malgré tout cela, il est important de souligner que cette UE fut finalement très enrichissante et m'a donné l'envie d'approfondir ce domaine d'application avec en autre le choix d'un PJI portant sur l'apprentissage profond et les réseaux de neurones conjointement avec Arthur d'Azémar.

L'application bien que fonctionnelle reste très perfectible sur de nombreux points, il serait également possible de très facilement l'améliorer. Une liste rapide d'amélioration pourrait être l'ajout d'une classe abstraite distance avec la possibilité d'ajouter des nouvelles distances en tant que plugin pour la méthode KNN. La possibilité d'utiliser plusieurs bases d'apprentissage et donc un système de choix de base, la possibilité de mieux gérer une base avec en autre une vue dédiée et la possibilité de supprimer des tweets. Il est possible d'améliorer le nettoyage de tweet, d'avoir une recherche plus complète en changeant d'API Twitter, éventuellement d'avoir plus de méthodes bayésiennes (trigramme ?). L'interface pourrait être revu totalement pour être plus minimaliste (tout réunir sur une seule fenêtre par exemple)...