Master INFORMATIQUE

Mention INFORMATIQUE

Rapport de ProJet Encadré

Analyse de Comportements avec Twitter

Antonin Carette
Alexandre Verkyndt

Promo 2014/2015

Table des matières

Introduction

0.1 Problèmatique

Le but de ce projet était de réaliser une application permettant d'analyser les comportements de tweets contenus dans l'application Twitter (en particulier, l'analyse de sentiments), via l'utilisation de son interface de programmation.
L'analyse de comportements sera étudié via différents algorithmes de classification (ou modèle), utilisant ou non une base d'apprentissage : le modèle basique (basé sur des dictionnaires), le modèle KNN et enfin le modèle Bayesien.

0.2 Interface de programmation Twitter

Pour ce projet, nous allons utiliser l'interface de programmation [1] de Twitter.
Cette API nous permettra de récupérer les tweets voulus, sur un sujet donné, ainsi que toutes les informations suivantes : l'ID du tweet, l'émetteur, s'il s'agit d'un tweet original ou s'il s'agit d'un *retweet* [2].
Aussi, nous allons utiliser cette interface via la librairie Java *Twitter4J*, dans sa version stable. En effet, cette librairie (à jour en fonction des mises à jour de l'interface de programmation) Java nous permettra d'accéder facilement aux tweets recueillis ainsi qu'aux informations du tweet, via des classes et des méthodes Java programmées.

1. Une interface de programmation est aussi appelée "API".
2. Un *retweet* est un tweet partagé - il n'est alors en aucun cas original.

Chapitre 1

Présentation du logiciel

1.1 Généralités

Ce programme a été écrit avec Java et Swing, pour la partie Interface Utilisateur. Suite à l'utilisation de fonctions avancées telles que le *switch* sur une chaîne de caractères, ou le *cast* des *JComboBox*[1], ce programme n'est compatible qu'avec une machine virtuelle Java de version 7 ou plus.

1.1.1 Titre et logo

Notre logiciel se nomme *BehAnTweet*[2]. Le nom du logiciel se réfère à la phrase anglaise "Behavior Analysis of a Tweet", littéralement "Analyse de Comportement d'un Tweet", qui est le sujet de notre projet encadré.
Le logo du logiciel est, quant à lui, une référence à *Bahamut*[3], déposé sous licence libre.

1.1.2 Logiciels de versionning utilisés

Git a été utilisé comme logiciel de versionning principal pour la gestion du projet, dû à sa simplicité d'emploi par rapport à *svn* mais aussi à l'utilisation faite de ce logiciel depuis plusieurs mois sur d'autres projets et à l'application web interactive *Github*[4].
Svn a lui été utilisé pour transmettre la version principale du logiciel sur les serveurs de l'Université[5], toutes les 3 semaines, afin de pouvoir visualiser l'avancement du

1. Liste d'options graphique, en Java.
2. La documentation développeur du logiciel est présente dans le répertoire ***doc***, à partir de la racine du projet.
3. Un poisson (ou serpent géant), présent dans la mythologie Arabe.
4. `https://github.com/WebTogz/BehAnTweet`
5. **Serveur** : svn-etu.fil.univ-lille1.fr ; **Projet** : pje14-15-carette.

projet régulièrement.
Une utilisation poussée de *Git* a été effectuée : gestion des commits en binôme, commits de chaque petite partie de code, création et gestion de plusieurs branches, merge avec la branche *origin*, etc...

1.2 Description de l'architecture de l'application

Nous avons utilisé principalement le modèle **MVC** (pour Modèle - Vue - Contrôleur) pour notre projet.
Ce modèle, très connu pour le développement logiciel, est destiné à répondre aux besoins des applications interactives, en séparant les problématiques liées aux différents composants au sein de leur architecture respective.
Aussi, le gros bénéfice de ce patron est qu'il permet facilement l'évolution de l'application par la suite, via l'ajout de classes dans les packages spécifiques suivants : **Controler**, **Model**, **View**.

1.2.1 Packaging

Controler

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle, et les synchroniser. Il analyse la requête du client et se contentera alors de renvoyer la vue correspondant à la demande, en fonction du modèle appelé. Aussi, il est important de savoir que le contrôleur n'effectue aucun traitement, et ne modifie aucunement les données !
Il était ainsi nécessaire d'implémenter le patron **Observable/Observeur**, afin de pouvoir un Observer (objet qui observe) qu'une modification a été apporté sur un Observable (objet observé par un Observer).

Model

Le modèle est le coeur algorithmique de l'application - il gère le traitement des données, la mise à jour des vues mais aussi les interactions avec la base de données. Le package Model contiendra tous les modèles de l'application - chaque modèle (à part la base Model.java) contiendra sa propre implémentation d'un algorithme de classification (Dictionnaire, KNN, Bayes).

View

La vue est le coeur de l'intéraction avec l'application.
Elle détient plusieurs tâches à effectuer :

— la première est de présenter les résultats renvoyés par le modèle,
— la seconde est de recevoir l'action de l'utilisateur - via la manipulation de boutons, objets cliquables, etc...

Toutes ces tâches, une fois finalisées, seront renvoyées au Contrôleur de l'application afin de pouvoir être traitées par le modèle.

resources

Le répertoire resources/ contient tous les fichiers ressources nécessaires à l'application tels que :

— positive.txt : le dictionnaire de mots-clefs positifs,
— negative.txt : le dictionnaire de mots-clefs négatifs,
— tweets.csv : le fichier contenant tous les tweets non-nettoyés, issus d'une recherche,
— tweets_clean.csv : le fichier contenant tous les tweets nettoyés des identifiants, date, *hashtags* et *retweets*,
— base_apprentissage.cv : le fichier CSV constituant notre base d'apprentissage.

1.3 Interface graphique

Nous avons pensé avant tout à une utilisation simple du logiciel, reposant sur une seule fenêtre. Cette fenêtre contiendra tous les paramètres pouvant être manipulés par l'utilisateur (le nombre de tweets à recueillir, l'algorithme utilisé et ses spécifications), une barre de recherche concernant un sujet donné, et enfin un grand espace dédié à la liste des tweets recueillis et de leur classification par l'algorithme choisi.

1.3.1 Capture d'écran

Vous trouverez ci-dessous l'écran principal de l'application. Il est constitué en 4 grandes parties, pouvant être divisées chacune à leur tour.

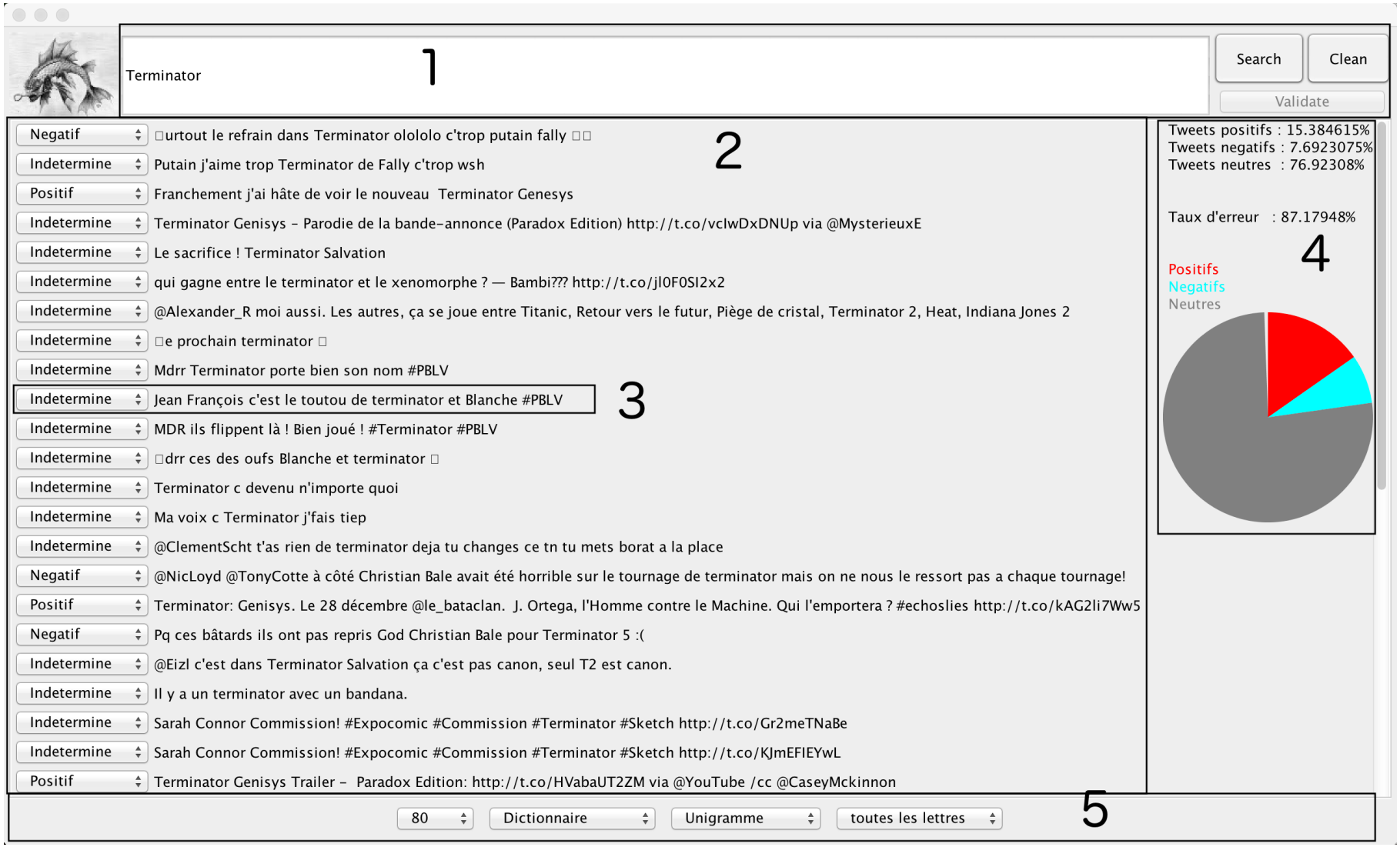
— **encart de recherche (1)** : contient la barre de recherche de tweet, le bouton de recherche "Search", le bouton de nettoyage du fichier temporaire tweet_clean.csv "Clean", et le bouton de validation "Validate" [6],
— **encart de visualisation (2)** : visualisation des tweets [7] recueillis,
— **encart statistiques (4)** : visualisation des statistiques concernant la dernière recherche effectuée - elle pourra s'effectuer par visualisation des pourcen-

6. Le bouton "Validate" permet la sauvegarde des tweets recueillis après vérification des comportements, dans un fichier temporaire.

7. Un exemple de tweet peut être visualisé en **3** - nous pouvons ainsi voir la classification obtenue du tweet par le classifieur utilisé (ici le classifieur par mot-clef), ainsi que le corps du tweet sur la droite.

tages de tweets classifiés, ou alors par visualisation du camembert, donnant une réponse visuelle immédiate quant à l'ensemble des tweets classifiés (juste en dessous de sa légende),

— **encart de paramètres (5)** : ce sont les paramètres ajustables de l'application - cet encart contient la modification du nombre de tweets à recueillir, la méthode de classification à utiliser ainsi que les paramètres Bayèsiens à prendre en compte.


Terminator
1
Search
Clean
Validate
Negatif □urtout le refrain dans Terminator ololololo c'trop putain fally □□
Indetermine Putain j'aime trop Terminator de Fally c'trop wsh
Positif Franchement j'ai hâte de voir le nouveau Terminator Genesys
Indetermine Terminator Genisys - Parodie de la bande-annonce (Paradox Edition) http://t.co/vcIwDxDNUp via @MysterieuxE
Indetermine Le sacrifice ! Terminator Salvation
Indetermine qui gagne entre le terminator et le xenomorphe ? — Bambi??? http://t.co/jl0F0SI2x2
Indetermine @Alexander_R moi aussi. Les autres, ça se joue entre Titanic, Retour vers le futur, Piège de cristal, Terminator 2, Heat, Indiana Jones 2
Indetermine □e prochain terminator □
Indetermine Mdrr Terminator porte bien son nom #PBLV
Indetermine Jean François c'est le toutou de terminator et Blanche #PBLV
Indetermine MDR ils flippent là ! Bien joué ! #Terminator #PBLV
Indetermine □drr ces des oufs Blanche et terminator □
Indetermine Terminator c devenu n'importe quoi
Indetermine Ma voix c Terminator j'fais tiep
Indetermine @ClementScht t'as rien de terminator deja tu changes ce tn tu mets borat a la place
Negatif @NicLoyd @TonyCotte à côté Christian Bale avait été horrible sur le tournage de terminator mais on ne nous le ressort pas a chaque tournage!
Positif Terminator: Genisys. Le 28 décembre @le_bataclan. J. Ortega, l'Homme contre le Machine. Qui l'emportera ? #echoslies http://t.co/kAG2li7Ww5
Negatif Pq ces bâtards ils ont pas repris God Christian Bale pour Terminator 5 :(
Indetermine @Eizl c'est dans Terminator Salvation ça c'est pas canon, seul T2 est canon.
Indetermine Il y a un terminator avec un bandana.
Indetermine Sarah Connor Commission! #Expocomic #Commission #Terminator #Sketch http://t.co/Gr2meTNaBe
Indetermine Sarah Connor Commission! #Expocomic #Commission #Terminator #Sketch http://t.co/KJmEFIEYwL
Positif Terminator Genisys Trailer - Paradox Edition: http://t.co/HVabaUT2ZM via @YouTube /cc @CaseyMckinnon
2
3
Tweets positifs : 15.384615%
Tweets negatifs : 7.6923075%
Tweets neutres : 76.92308%
Taux d'erreur : 87.17948%
4
Positifs
Negatifs
Neutres
80
Dictionnaire
Unigramme
toutes les lettres
5

1.3.2 Manuel d'utilisation

Comment modifier les paramètres réseaux de l'application ?

Ouvrez le fichier *twitter4j.properties*.
Recherchez les deux lignes situées sous le commentaire : //PROXY.

Si vous vous trouvez sur l'Université Lille1, veuillez entrer les propriétés proxy :
http.proxyHost=cache-etu.univ-lille1.fr
http.proxyPort=3128

Si vous ne vous trouvez pas sur sur l'Université Lille1, vous pouvez désactiver les propriétés proxy en commentant les lignes précédentes.

Comment modifier le nombre de tweets recueillis pour une requête ?

Dans l'encart des paramètres, cliquez sur le bouton "20" - cela signifie que le nombre de tweets maximal affiché est de 20.
Cliquez alors sur la nouvelle valeur que vous voulez prendre en compte.

Comment modifier la méthode de classification pour une requête ?

Dans l'encart des paramètres, cliquez sur le bouton "Dictionnaire", et choisissez alors votre méthode de classification.
Si vous choisissez la méthode Bayes Fréquence, choisissez alors le nombre de grammes que vous voulez utiliser, et enfin le "type" de mots que vous voulez comparer - avec le nombre de lettres par mot.

Comment effectuer une requête ?

Vous pouvez effectuer une recherche en tapant votre requête dans la barre de recherche de tweet. Après cela, vous pouvez choisir dans l'encart de paramètres le nombre de tweets maximal à recueillir pour cette requête, ainsi que la méthode de classification demandée.
Après le choix des paramètres, cliquez sur le bouton "Search" afin de lancer votre requête, via *Twitter4J* et l'API Twitter. Vous obtiendrez alors dans l'encart de visualisation, quelques secondes après, un ensemble de tweets recueillis et en lien à cette recherche.

Comment modifier la classification d'un ou de plusieurs résultat(s) de requêtes ?

Après une requête, vous pouvez remarquer dans l'encart de visualisation que, via la méthode de classification choisie, une classification est déjà établie - à gauche du tweet.
Si la classification d'un tweet ne vous plaît pas, vous pouvez la modifier en cliquant sur la classification du tweet. Vous pouvez alors choisir la classification qui vous plaît en cliquant sur celle choisie.
Vous pourrez alors constater que votre demande a bien été prise en compte, via le changement de la classification dans l'encart de visualisation.

Comment enregistrer les résultats d'une requête dans la base d'apprentissage ?

Une fois les classifications de chaque tweet analysées, vous pouvez enregistrer celles-ci dans un fichier CSV temporaire (tweet_clean.csv), en cliquant sur le bouton "Validate", présent dans l'encart de recherche. Une fois cliqué, le bouton devient gris - il ne vous ai plus possible de valider une seconde fois ces mêmes classifications.
Pour enregistrer les tweets recueillis dans la base d'apprentissage, il suffit juste d'ouvrir le fichier base_apprentissage.csv, et de copier/coller tous les tweets que vous voulez enregistrer dans celle-ci.
L'avantage de cette solution est de permettre à l'utilisateur de choisir exactement le ou les tweet(s) qu'il veut sauvegarder, et le ou les tweet(s) qu'il veut ignorer.

ATTENTION : si vous voulez enregistrer les tweets nettoyés dans le fichier tweet_clean.csv sans tenir compte des résultats précédents sauvés dans ce même fichier, il sera nécessaire de cliquer sur le bouton "Clean" - qui nettoiera le fichier tweet_clean.csv - avant de cliquer sur "Validate".

Chapitre 2

Algorithmes de classification

Pour l'analyse de tweets, divers algorithmes de classification ont été utilisés.
Voici la liste des algorithmes de classification utilisés.

2.1 Dictionnaire

La méthode **Dictionnaire** (ou encore par **mot-clef**) a été l'algorithme le plus facile à programmer.
En effet, il consiste dans un premier temps à obtenir une liste de mots positifs et négatifs. Chaque tweet recueilli sera découpé en mots, et chaque mot sera confondu avec les deux listes - aussi, le tweet partira avec un score entier nul. Si un des mots du tweet est reconnu dans la liste de mots positifs, on diminuera son score - à contrario, si ce mot est reconnu dans la liste de mots négatifs, on augmentera son score. Ainsi, chaque mot du tweet est un mot-clef à reconnaître.

Finalement, on obtiendra pour chaque tweet un score, somme des scores de chaque mot appartenant à celui-ci. Ce score décidera de la classification auquel appartient le tweet :

- positif, si le score est négatif,
- indéterminé, si le score est nul,
- négatif, si le score est positif.

2.2 KNN

La méthode **KNN** [1], est une méthode d'apprentissage automatique basée sur des exemples contenus dans une base d'apprentissage. On cherchera ici à trouver les k plus proches tweets du tweet analysé, contenus dans la base d'apprentissage du

1. Aussi appelé méthode des k plus proches voisins.

projet [2].

Pour chaque tweet, on aura un tableau de k objets, contenant un couple (distance, tweet). On range ainsi les k premiers tweets de la base d'apprentissage dans le tableau énoncé précédemment, les distances étant calculées entre le tweet à classifier et le tweet de la base d'apprentissage selon la formule suivante : **((nombreMotsTotal) - nombreMotsCommuns) / nombreMotsTotal**.
Pour les (k+1) à n [3] tweets, on calculera à chaque fois la distance entre le tweet courant et le tweet à classifier. Si cette distance est inférieure à la plus grande distance contenue dans le tableau des k objets, on remplacera alors cet objet par la nouvelle distance ainsi que le tweet courant associé à celle-ci.

Une fois cette étape faite, il nous restera à définir le comportement du tweet à classifier - pour cela, on comptera combien il y a de tweets positifs issus de la base d'apprentissage, de même pour les tweets négatifs et indéterminés. Le score le plus élevé parmi les 3 sera l'avis décisif pour le tweet à classifier.

2.3 Bayés

La classification Bayésienne s'appuie, quant à elle, sur une base d'apprentissage préétablie sur un sujet donné.

Basée sur les probabilités conditionnelles et la règle de Bayés, elle calculera les probabilités pour un tweet d'appartenir à la classe des tweets positifs, négatifs ou encore indéterminés. Ainsi, la probabilité la plus forte décidera de la classe a laquelle sera attribué le tweet.
Pour cette classification, nous pourront distinguer deux sous-méthodes de classification : *Classification par Présence* et *Classification par Fréquence*.

2.3.1 Classification par Présence

Afin de déterminer les probabilités d'un tweet donné au sein d'une des trois classes possibles, la classification par présence applique la règle de Bayes : $P(c|t) = P(t|c).P(c)/P(t)$, où c est la classe d'un tweet (Positif, Négatif ou Indétermine), et t le tweet à classifier.
Dans cette sous-méthode de la classification Bayésienne, la probabilité se détermine par la présence ou non de chacun des mots du tweet testé au sein des tweets conservés

2. Cette base d'apprentissage consiste en la sauvegarde de tweets déjà analysés pour le comportement.
3. n étant le nombre de tweets total contenus dans la base d'apprentissage.

dans la base d'apprentissage.

2.3.2 Classification par Fréquence

La classification par fréquence conserve la même approche que celle par présence cependant elle prend en compte un autre facteur qui est le nombre d'occurence d'un mot du tweet à classifié dans ceux contenus dans la base d'apprentissage.

2.3.3 *N*-Gramme

En plus des deux sous méthodes citées plus haut, on peut apporter une option de calcul permettant de considérer non pas les mots un par un, mais également les combinaisons de mots.

Uni-gramme

Les uni-grammes consistent à la version par défaut de la classification. C'est à dire, ne considérer qu'un mot à la fois.
Par exemple la phrase : *L'oiseau rejoint son nid* , contient 4 uni-grammes : *L'oiseau, rejoint, son, nid.*

Cette solution permet facilement d'obtenir des résultats, mais ne reflète peas le contexte dans lequel sont employés chacun des mots.

Bi-gramme

Les bi-grammes permettent de considérer des expressions ou autres associations de mots dans une phrase.
Par exemple la phrase : *Cette poésie est un chef d'oeuvre*, contient 5 bi-grammes : *(Cette, poésie), (poésie, est), (est, un), (un, chef), (chef, d'oeuvre).*

Cette solution fournit des résultats se rapprochant plus du contexte dans lequel ils sont employés.

Uni-bi-Gramme

Le compromis entre les deux solutions est d'encore les employer ensemble afin d'obtenir des résultats plus probants.
Par exemple la phrase : *Cette poésie rejoint son nid*, contient 5 uni-grammes : *Cette, poésie, rejoint, son, nid*
et 4 Bi-grammes : *(Cette, poésie), (poésie, rejoint), (rejoint, son), (son,nid).*

Chapitre 3

Résultats de la classification

Nous travaillerons sur un ensemble d'apprentissage basé sur le mot-clef "**Terminator**", en prenant compte seulement des mots de plus de 3 lettres.
La base d'apprentissage, elle, contient 173 tweets classifiés.

3.1 Dictionnaire

Cet algorithme, facile à implémenter, est malheureusement peu fiable mais aussi assez lent. En effet, il s'appuie sur deux dictionnaires : un dictionnaire positif et un négatif - cet algorithme ne s'appuie donc pas sur une base d'apprentissage. Surtout, il ne comptabilise pas les mots qui ne sont pas présents dans les deux dictionnaires, ce qui est n'est pas évident pour la conjugaison d'un verbe par exemple, ou alors lors d'une malencontreuse faute d'orthographe dans le mot...
Aussi, l'algorithme est lent car, à chaque tweet, il doit regarder si chaque mot issu du dictionnaire positif est présent dans le tweet à analyser - de même pour le dictionnaire négatif.

Tweets positifs	**Tweets négatifs**	**Tweets indéterminés**	**Taux d'erreur**
15%	10%	75%	85%

Nous pouvons remarquer ici que la classification est majoritairement indéterminée (75%) - en effet, il peut être difficile de trouver le mot exact dans le dictionnaire, et pouvoir ainsi classifier le tweet en fonction de la somme des scores de chaque mot. La classification positive et négative est presque équivalente - 15% de tweets positifs et 10% de tweets négatifs.

3.2 KNN

L'algorithme implémenté pour KNN se base sur une base d'apprentissage - à savoir un fichier CSV contenant tous les tweets sauvés et dont la classification de chaque tweet a été vérifié.
Pour que la classification se fasse de façon optimale, il faut une base d'apprentissage rigoureuse mais aussi de taille très grande. De plus, il aurai fallu choisir k de façon optimale aussi, ce qui n'a pas été effectué dans le cadre de ce projet.

Tweets positifs	**Tweets négatifs**	**Tweets indéterminés**	**Taux d'erreur**
25%	55%	20%	80%

En résultat de l'exploitation des résultats, nous pouvons observer que nous obtenons une majorité de tweets négatifs (55%), un ordre de tweets positifs de l'ordre de 1/4 (25%) ainsi que 20% de tweets indéterminés, le tout pour un taux d'erreur de l'ordre de 80%.
Nous pouvons observer que cet algorithme KNN a travaillé pour classifier les tweets recueillis, et ne les a pas classer en majorité en indéterminé.

3.3 Bayés

Voici un tableau permettant de visualiser le taux d'erreur d'algorithmes de classification bayésienne, à savoir **Présence uni-gramme**, **bi-gramme** et **uni+bi-gramme**, mais aussi **Fréquence uni-gramme**, **bi-gramme** et **uni+bi-gramme**.

Algorithme	**Tweets pos.**	**Tweets nég.**	**Tweets indét.**	**Taux d'erreur**
Prés., uni	21.05%	31.57%	47.37%	82.46%
Prés., bi	0%	36.84%	63.16%	**36.84**%
Prés., uni + bi	0%	78.95%	21.05%	78.95%
Fréq., uni	35%	0%	65%	**41.67**%
Fréq., bi	10%	0%	90%	**33.33**%
Fréq., uni + bi	20%	0%	80%	83.33%

Nous pouvons remarquer que la classification Bayésienne par présence bi-gramme est la plus précise (taux d'erreur de 36.84% - aucun tweet positif mais 36.84% de tweets négatifs et 63.16% de tweets indéterminés) - à contrario, l'uni-gramme est celle ayant le plus grand pourcentage d'erreur (82.46% en taux d'erreur - aucun tweet positif non plus, 78.95% de tweets négatifs et 21.05% de tweets indéterminés). Quant à l'unicité uni-gramme/bi-gramme, elle obtient un résultat compris entre les deux les précédents (78.95%).
Nous pouvons avoir une incompréhension lors de la vue des résultats quant à la clas-

sification Bayésienne par Fréquence - en effet, les résultats sont assez proches les uns des autres, sauf pour l'uni-gramme/bi-gramme (83.33%) qui a un taux d'erreur bien plus élevé que l'uni-gramme (42.67%) et le bi-gramme (33.33%) indépendants l'un de l'autre. Ainsi, on pourra douter du bon calcul du taux d'erreur pour cette dernière. L'uni-gramme et le bi-gramme sont, à contrario, plus précis que ceux concernant la classification Bayésienne par Présence, et ne détient par de tweets négatifs (35% de tweets positifs pour la Fréquence uni-gramme, 65% de tweets indéterminés pour la Fréquence uni-gramme | 10% de tweets positifs pour la Fréquence bi-gramme, 90% de tweets indéterminés pour la Fréquence bi-gramme) - classés donc permis les tweets positifs ou indéterminés.

Chapitre 4

Conclusion

Conclusion concernant la classification

Suite aux résultats donnés dans le chapitre 3, nous pouvons d'ores-et-déjà conclure que la méthode du Dictionnaire est nettement la moins susceptible d'être utilisée pour classifier des tweets.
Nous pourrons alors choisir entre KNN et Bayès, afin de pouvoir classer par apprentissage automatique les tweets recueillis pour chaque requête. Aussi, il sera nécessaire d'utiliser une des méthodes la plus précise - à savoir ici l'une des méthodes Bayésienne suivantes : Présence bi-gramme, Fréquence uni-gramme ou encore Fréquence bi-gramme (les taux d'erreurs sont surlignés dans le tableau).

Conclusion logicielle et personnelle

Ce projet nous a permis d'utiliser et de parfaire notre utilisation de Java 7 (notamment de l'API Swing), mais aussi d'utiliser les design pattern objets MVC (pour Modèle-Vue-Contrôleur), et Observable/Observeur [1]. L'utilisation des deux logiciels de versionning nous a permit de nous familiariser avec le développement participatif sur un même projet ; de nombreuses problématiques se sont manifestées, et nous ont permit d'acquérir une certaine expérience.
Pour terminer, ce projet nous a permis d'en savoir un peu plus quant à l'analyse de texte et des différentes méthodes de classification employées pour de l'analyse comportementale, ce qui nous a beaucoup intéressé.

1. Voir le chapitre 1.2 : Packaging pour plus d'informations.

Bibliographie - Webographie

— *PJE : Analyse de Comportements avec Twitter - Partie classification*
Laetitia Jourdan
`http://www.lifl.fr/~jourdan/PJE/PJE_vl2014.pdf`

— *PJE : Analyse de Comportements avec Twitter - Classification supervisée*
Arnaud Liefooghe
`http://www.fil.univ-lille1.fr/~liefooghe/PJE/bayes-cours.pdf`

— *k-nearest neighbors algorithm*
`http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`