MASTER INFORMATIQUE

MENTION INFORMATIQUE

(2016/2017)

RAPPORT DE PROJET PJE

Analyse de Comportement avec Twitter

Réalisé par

BOUZIANE AYMANE

DEHOUCHE NABILA

Encadreurs du projet

DERBEL BILLAL

ARNAUGH

LITÉCIA

Sommaire :

1 Introduction

1.1 Problèmatique

« L’analyse de sentiments est un des nouveaux défis apparus en traitement automatique des langues avec l’avènement des réseaux sociaux sur le WEB. Profitant de la quantité d’information maintenant disponible, la recherche et l’industrie se sont mises en quête de moyens pour analyser automatiquement les opinions exprimées dans les textes».

Dans ce contexte, le but de ce projet est de développer une application permettant de récupérer des tweets à partir de l'API Twitter, puis d'en dététerminer et d’analyser les comportements, en se basant ou pas sur une base d'apprentissage.

1.2 Interface de programmation Tweeter

L'interface de programmation utilisée est celle de Twitter. Elle permet de récupérer des tweets filtrés par mots clés, ainsi que toutes les informations correspondantes, telque : l'id utilisateur, la date du tweets, l'id du tweet lui même etc.

La librairie Java utilisée est Twitter4J. Cette librairie contient les méthodes qui permettent de se connecter à l'API ainsi que diverses méthodes facilitant la récupération des informations.

2 Présentation de l'application

2.1 Généralités

Nous avons développé notre application avec Java, et nous avons utilisé Swing pour le développement de notre interface graphique.

2.1.1 Titre

Nous avons choisit “TB Analysis” comme nom pour notre application. Ce nom est une abréviation de “Twitter Behavior Analysis”, qui peut être traduit littéralement en francais par “Analyse de Comportement avec Twitter”.

2.1.2 Outil de versionning utilisé

Nous avons choisit d'utilisé Git comme outil de versionning pour sa simplicité d'utilisation par rapport à SVN.

2.2 Description de l'architecture de l'application

Pour le développement de notre application nous avons choisit le patron de conception Modèle-Vue-Contrôleur (MVC). C'est un modèle très répandu dans le développement de projets informatique, permettant de séparer l'affichage des informations, les actions de l'utilisateurs, et l'accès aux données. Il permet donc une plus grande liberté, on peut rajouter et supprimer facilement des classes et des affichages, ce qui permet une évolution plus rapide de l'application.

2.2.1 Packaging

Controler

Ce paquêt permet la gestion des évenements de synchronisation pour mettre à jour la vue et le modèle, et les synchroniser. Il analyse la requête du client et met à jour les vues en fonction de ce qui a été analysé.

Comme le controleur ne modifie pas les données, nous avons développeé notre solution en utilisant le patron de conception Observer qui permet de mettre à jour les objets “observeurs” qui sont ici les affichages quand une modification a été apporté sur un objet “observable” ici le modèle.

Model

Ce paquêt contient tous les modèles de l'application. Il gère le traitement des données et l'accès à la base d'apprentissage. Chaque modèle contient sa propre implémentation d'un algorithme de classification.

View

Ce paquêt contient tous les composants de l'interface graphique. Chaque action effectuée sur l'interface graphique est renvoyée par ce paquet au contrôleur pour qu'elle puissent être traité par le modèle.

2.3 Interface graphique

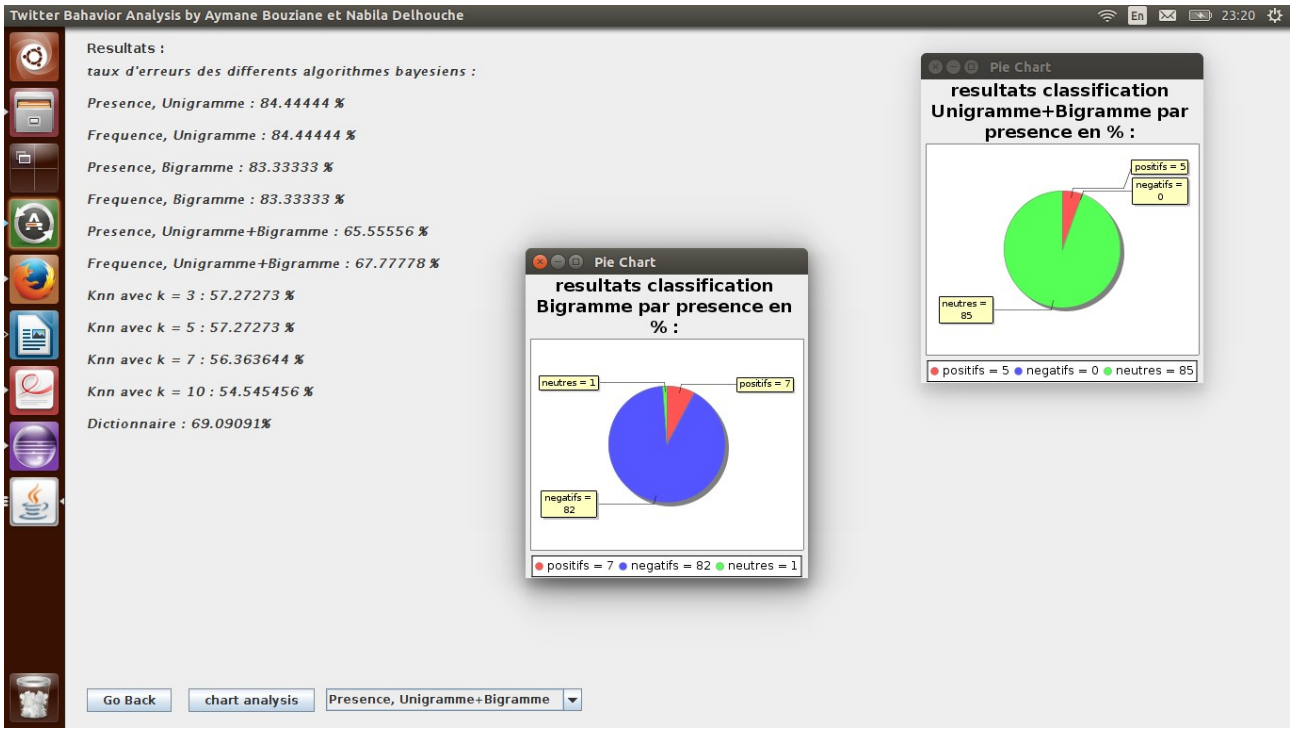
Notre interface graphique contient deux écrans. Le premier écran permet d'effectuer la recherche et d'afficher les tweets annotés grâce un algorithme de classification choisit. Il contient une barre de recherche, une barre d'options pour choisir un algoritme, et finalement une barre qui permet d'ajouter tous les tweets sur l'écran à la base d'apprentissage, et de nettoyer cette base. Cette dernière barre permet aussi de lancer une analyse de l'éfficacité des différents algorithmes, et de visualiser la dernière analyse effectuée.

Le deuxième écran affiche le résultat de l'analyse de l'efficacité des différents algorithme. Cette écran permet aussi de lancer une analyse graphique pour l'algorithme choisit.

2.3.1 Capture d'écrans

Écran principal :

Écran d'Analyse :



2.3.2 Manuel d'utilisation

Comment effectuer une recherche ?

Choisissez dans la barre d'options un algorithme, tapez un mot clé dans la barre de recherche et appuyez sur “search”. La liste des tweets retrouvés et déjà classifiés avant l'affichage sont alors visible sur l'écran principal. Vous pouvez modifier mannuellement la classe d'un tweet si vous vous rendez compte qu'il est mal classifié.

Comment choisir un algorithme ?

Dans la barre des options, choisir d'abord le modèle de classification. Si vous choisissez le modèle Knn, assurez vous que vous avez choisit le bon le nombre de “k” avant de lancer votre recherche. Si vous choisissez le modèle Bayes, assurer vous aussi que vous avez choisis le bon algorithme car il y en a plusiseurs.

Comment classifier mannuellement un tweet ?

Après une recherche si vous voyez un tweet mal classifié, vous pouvez corriger la classification mannuellement, en choisissant la bonne parmis celle proposées à gauche du tweet. Quand vous corrigez tous les tweets, vous pouvez les rajouter tous à votre base d'apprentissage en un seul clique.

Comment rajouter des tweets annotés à la base d'apprentissage ?

Après avoir corrigé la classification de tous les tweets, vous pouvez les rajoutés à votre base d'apprentissage en cliquand sur le boutton “add”.

Comment nettoyer votre base d'apprentissage ?

Le boutton clean permet de nettoyer votre base d'apprentissage et créer une autre base contenant les même tweets que la première mais néttoyés. Cette dernière est prête à être utilisé pour la classification automatique.

Comment vérifier l'efficacité des algorithmes utilisées dans le développement de cette application ?

Cliquez sur le boutton “new Analysis”, l'analyse des différents algorithmes est alors démarré et le temps avant de voir les résultats de l'analyse dépend de la taille d'apprentissage. Une fois l'analyse est fini, vous pouvez analyser graphiquement chaque algorithme tout seul, en le choisissant dans le menu déroulant en bas de l'écran des résultas.

Comment visualiser la dernière analyse ?

Il suffit de cliquer sur le boutton “last Analysis” sur le premier écran . Vous retrouverez la dernière analyse que vous avez faite.

3 Algorithmes de classification

3.1 Dictionnaire

La méthode dictionnaire a été l'algorithme le plus facile à programmer. Il consiste à

découper le tweet en une liste de mots, pour chaque mot on vérifie s'il existe dans une base de mots positifs, s'il existe on incrémente un compteur de mots positifs dans le tweets, s'il existe dans la base de mots négatifs on incrémente le compteur des mots négatifs, sinon on incrémente le compteur des tweets neutre. À la fin de cette opération on associe au tweet : la classe “positif” si le nombre de mots positifs est le plus grand, la classe “negatif” si le nombre de mots negatifs est le plus grand , sinon on lui associe la classe “neutre”.

3.2 Knn

Cet algorithme est basé sur des exemples contenus dans la base d'apprentissage. On cherchera à trouver les k tweets les plus proches du tweets sur lequel on travail.

Principe en détail :

On récupère d'abors les k premiers tweets de la base. On trie ces cas k premiers tweets par distance du tweet sur lequel on travail. C'est à dire du moins proche au plus proche du tweets à classifier. Appelons le tweet le plus proche tpp. Puis pour le reste de la base d'apprentissage : pour chaque tweet ti , s'il est encore plus proche que tpp, ti prend la place de tpp, et ainsi de suite.

On obtient à la fin la liste des K tweets les plus proche du tweet à classifier. Ainsi dans cette liste des K tweets les plus proches, si la classe “positif” est la plus présente, alors le tweet à classifier prend la classe “positif”, si la classe “negatif” est la plus présente, alors le tweet à classifier prend la classe “negatif”, et la classe “neutre” sinon .

3.3 Bayes

La classification Bayesienne, s'appuie sur une base d'apprentissage pré-établie sur un sujet donné. Elle est basée sur les probabilitées conditionnelles et la règle de Bayès, elle calcule la probabilité pour un tweet d'appartenir à la classe “positif”, “negatif”, ou “neutre”. Ainsi la probabilité la plus forte décidera de la classe du tweet.

Remarque : il faut distinguer deux types de classification bayesienne, la classification par présence, et la classification par fréquence.

3.3.1 Classification par présence

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 et t le tweet à classifier. Cette méthode de classification se base sur la présence de chacun des mots du tweet dans les tweets de la base d'apprentissage.

3.3.2 Classification par fréquence

Cette classification ne change pas beaucoup de la classification par présence hormis le fait qu'elle prend en compte un autre facteur qui est le nombre d'occurence d'un mot du tweet à classifier dans l'ensemble des tweets de la base d'apprentissage.

3.3.3 N-Gramme

En plus des deux méthode citées ci avant à savoir la classification par présence et la classification par fréquence, on peut apporter à notre développement une option de calcul qui consiste à considérer non les mots d'un tweet à classifier un par un, mais également des mots composés.

Uni-Gramme

Cette méthode est celle utilisée par défaut par la classification Bayesienne. Elle consiste à considérer un mot à la fois. Par exemple : “le sujet de pje2 est interessant”, il y a 6 uni-grammes : “le”, “sujet”, “de”, “pje2”, “est”, “interessant” .

L'inconvénient de cette méthode est qu'elle ne reflète pas le contexte dans lequel sont employés les mots.

Bi-Gramme

Cette méthode considère deux mot par deux mot dans un tweet à classifier. Par exemle si on prend l'exemple dans le paragraphe précédent, on obtient 3 bi-grammes : “le sujet”, “de pje2”, “est interessant”.

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

Unigramme+Bigramme

On peut employer une méthode qui serait plus efficace, et qui est de combiner les deux méthodes précédentes. Ainsi pour le même exemple on obtient une combinaison comme la suivante :

“le”, “sujet”, “de”, “pje2”, “est”, “le sujet”, “de pje2”, “est interessant”.

4 Résultats de la Classification

4.1 Dictionnaire

Cet algorithme, de sa naivité, est malheureusement peu fiable, car il ne se base pas sur une base d'apprentissage. Il a toujours besoin d'un dictionnaire riche, et des fois il peut ne pas trouver des mots dans le dictionnaire, ce qui par la suite classifie les tweets comme neutres .

Tweets Positifs	Tweets Négatifs	Tweets Neutres	Taux d'erreur
40%	49%	9%	50%

Nous pouvons observer dans les résultats de notre analyse que la méthode dictionnaire a un taux d'erreur de 50%. Une classification optimal d'un tweet est satisfaite si tous les mots du tweet existent dans la base. La méthode dictionnaire peut s'avérer très longue si la base des mots est importante.

4.2 Knn

Cet algorithme se base se base sur une base d'apprentissage et tous les tweets dans cette base sont annotés manuellement.

Pour une classification efficace, il faut disposer d'une base d'apprentissage de bonne qualité, c'est à dire bien riche en terme de sujets, contenant autant de tweets positifs que négatifs aussi autant que neutres. Et pour une classification optimale, il faut choisir la bonne valeur de “k” ce qui n'a pas été fait dans le cadre de ce projet.

Pour k = 3 :

Tweets Positifs	Tweets Négatifs	Tweets Neutres	Taux d'erreur
65%	24%	0%	57%

Pour k = 10 :

Tweets Positifs	Tweets Négatifs	Tweets Neutres	Taux d'erreur
50%	39%	0%	54%

Nous pouvons observer d'après les résultats ci dessus que l'algorithme knn classifie mieux les tweets que la méthode dictionnaire. Le nombre de tweets neutres est quasiment nul. On remarque aussi que plus la valeur de k augmente plus le taux d'erreur diminue. Donc plus la valeur de k augment plus l'algorithme est optimal.

4.3 Bayes

voici les résultats de notre analyse. Notre classifieur bayesien utilise une base d'apprentissage à part autour du thème “election presidentielle”.

Algorithmes	Tweets Positifs	Tweets Negatifs	Tweets Neutres	Taux d'erreurs
Presence, Unigramme	14%	70%	6%	84%
Frequence, Unigramme	14%	70%	6%	84%
Presence, Bigramme	7%	82%	1%	83%
Frequence, Bigramme	80%	1%	9%	83%
Presence, Unigramme+Bigramme	5%	0%	85%	65%
Frequence, Unigramme+Bigramme	0%	8%	82%	67%

Nous pouvons remarqué que la méthode de classification bayesienne la plus efficace est celle combinant la classification unigramme et la classification bigramme avec un taux d'erreur autour de 65% contre 83% et 84% pour les autres méthodes de classification. Mais si on regarde en détail les taux de tweets positifs et négatifs dans la classification bigramme par exemple, le taux de tweets positif est de 85% quand on classifie par fréquence et 7% quand on classifie par présence, et le taux de tweets négatifs est de 82% quand on classifie par présence et 1% quand on classifie par fréquence. Alors que dans la classification unigramme+bigramme les taux de tweets positifs et négatifs sont très faible (entre 0% et 10%) et on voit que le taux des tweets neutres est le plus élévé soit autour de 65%.

Après avoir observé ces résultats on peut dire que dans notre cas la classification bigramme est la plus efficace même si la classification unigramme+bigramme a un taux d'erreur moins important car cette dernière classification a classifié la plus part des tweets comme neutres ce qui nous laisse remettre en question la crédibilité de son travail.

En principe le classifieur unigramme+bigramme est censé être le plus efficace, car en

l'utilisant on se rapproche plus du contexte dans lequel un mot est utilisé. Mais dans notre cas, son comportement (beaucoup de tweets neutres) doit être du à la mauvaise qualité de notre base d'apprentissage.

5 Conclusion

Conclusion sur la classification des tweets

Suite aux résultats données des différentes classifications, nous pouvons désormais conclure que la méthode dictionnaire est la moins susceptible d'être utilisée pour la classification des tweets.

La différence entre la méthode Knn et la méthode dictionnaire est que, en utilisant la méthode Knn, on peut toujours garder la même base, et il suffit seulement d'augmenter la valeur de k afin d'optimiser la classification, et celà n'augmentent pas le temps du processus, contrairement à la méthode dictionnaire qui elle, pour optimiser la classification, a besoin de toujours de rajouter des mots au dictionnaire afin de l'enrichir ce qui le rend très grand ce qui rend donc le processus de classification très lent.

Nous pourrons donc choisir entre la classification Knn et la classification Bayesienne, bien que d'autres conditions doivent être respectées pour une classification optimale, comme une base d'apprentissage riche, de taille importante, et qui contient autant de tweets positifs que négatifs que neutres.

Si la classification Bayesienne est choisit pour une requête, il faut choisir la méthode la plus précise et la plus efficace à savoir la méthode par présence, par fréquence , uni-gramme , bi-gramme ou la combinaison des deux. D'après les résultats observés précédemment, la méthode Bi-gramme serait la plus efficace, si on utilise la base d'apprentissage utilisée pour ce projet.

Conclusion générale

Ce projet nous a permis d'approffondir plusieurs notions en développement, notamment l'utilisation du patron MVC ainsi que le patron Observateur que nous avons vu seulement en théorie durant le cours de COO en L3, mais qu'on a pu mettre en pratique durant ce projet.

Ce projet nous a permis de nous introduire à la notion de “machine learning” ou “apprentissage automatique” qui est un champ d'étude de l'intelligence artificielle, un domaine auquel nous nous intéréssant beaucoup.