Rapport PJE

Twitter Sentiment Analyzer

Julien Leclercq & Jean-Frédéric Durand

Lien vers le Github

Sommaire

Présentation du projet

I / Problématique

Objectif

L'objectif de ce PJE est de manière globale d'analyser l'attitude d'un tweet, en fonction de son contenu. Étant dans la région Nord, Pas-de-Calais et proche de la Belgique, nous avons estimé que culturellement parlant faire une analyse autours du sujet qu'est la bière et les frites nous permettrait de révéler l'attitude d'une majorité de tweets "locaux". Notre objectif est aussi de comparer les différents algorithmes de classification.

Notion de classification

Lors de ce PJE la classification s'est faite selon la norme suivante. Lors de la récupération des tweets on index un nombre à -1. Une fois annoté il se verra affecté un nombre : 0, 1, 2 si le tweet est respectivement négatif, neutre ou positif. Par la suite ce classement nous permettra de catégoriser les tweets à visée positive, négative ou neutre par l'intermédiaire des différents algorithmes proposés.

Marge d'erreur

La classification automatique n'est cependant pas parfaite. En effet il est très difficile de classifier certains tweets tel que les tweets à connotation sarcastique par exemple; ou encore l'ironie étant donné que ces types de tweets sont sémantiquement très proche des tweets ayant une visée au premier degré.

II / Architecture de l'application

Paquetage

Le projet réalisé a été développé en Ruby on rails avec les paquets suivant :

- Base de donnée : SQLITE3
- HTML : Slim
- CSS : Bootstrap
- API : Twitter (for ruby on rails)

Modèle de conception

Pour ce projet nous avons opté pour un modèle *MVC* natif et représentatif de l'environnement Ruby. Il sera ainsi facile de retrouver les principales classes dans le modèle. Les classes sont : - TweetSearch : Classe de la fonction recherche - KeywordsAnalysis : Classe de la fonction de classification par mots-clés - KNN : Classe de la fonction de classification par Knn - Bayes : Classe de la fonction de classification par Bayes

Pour le cas de l'enregistrement en base et de la lecture, nous fonctionnant en *ActiveRecord*. Ceci nous permet de faire beaucoup de requête et de les modifier par la suite simplement.

Détails des travaux effectués

I / API Twitter

Premièrement, on a configuré le client de l'API twitter dans un contrôleur d'initialisation : app/controllers/concerns/application_concern.rb.

Deuxièmement, à l'aide du tweet_search_controller.rb nous effectuons une requête d'envoi de recherche de tweet via un paramètre de recherche. Ce paramètre de recherche peut être un mot

clé ou un pseudo. Il est à noter que nous avons défini la langue lors de l'envoi de la requête, nous ne récupérons donc que les tweets en français.

II / Base d'apprentissage

Nettoyage des données

Le nettoyage des tweets s'effectue dans le repertoire : app/models/tweets.rb. On y retrouve les principales fonctions de suppression qui sont :

- remove_url : Supprime un lien url fournit dans un tweet
- remove_rt : Supprime les caractères "RT" lors d'un retweet
- remove_arobase : Supprime les @ d'un tweet
- remove_hashtag : Supprime les # d'un tweet
- remove_quotes : Supprime les guillemets d'un tweet

S'ajoute à cela quelques transformations :

- transform_price : Transforme un prix en euro par X
- different_smiley : Suppression des smiley positif et négatif lorsqu'ils sont regroupés dans un même tweets

Construction de la base

La construction de la base est visible depuis le répertoire db/migrate du projet. Nous y définissons la création de la table de stockage d'un tweet selon les critères suivants :

- tweet_id : l'identifiant twitter du tweet
- twittos : la personne qui a publié le tweet
- text : le corps du tweet
- annotation : l'entier défini pour la classification
- created_at : date de création dans la base
- updated_at : date de mise a jour dans la base
- hand_annoted : un boolean qui renvoie vrai si le tweet à été annoté à la main

A noter que nous avons du augmenter la taille des entier par défaut de sqlite3 à 20.En effet le tweet_id à actuellement une longueur de 18 caractères.

De plus nous avons défini les les données de dates (created_at,updated_at) comme facultatives.

III / Algorithmes de classifications

Mots-clés

Nous avons récupéré les fichiers positif.txt et negatif.txt que vous trouverez dans lib/tasks sous les noms positive_word.rb et negative_word.rb contenant la liste des mots-clés mise en forme pour Ruby.

KNN

Bayes

IV / Interface graphique

mettre des copies d'écrans et décrire l'utilisation

V / Conclusion

Ce projet nous a permis de découvrir une approche du "machine learning" et du "big data". En effet nous avons pu par l'intermédiaire des différents algorithmes, mettre en oeuvre une classification. De plus nous avons découvert la limite de ces algorithmes, notamment dûe à la complexité du langage qu'est le français.

Glossaire

MVC : Model-View-Controller est un patron d'architecture logicielle permettant de répondre aux besoins d'une application interactive. Les fonctions sont réparties en trois parties :

- un Modèle de données permettant de traiter des données provenant de la base
- une Vue représentant l'interface de utilisateur
- un Contrôleur permettant de faire la liaison entre le Modèle et la vue

ActiveRecord : Encapsulage des données dans une classe pour la lecture ou l'écriture de données en base. Lorsque l'on met à jour un objet, son objet en base aussi. De plus la classe implémente des accesseurs pour chaque attribut.