Rapport PJE

Twitter Sentiment Analyzer

Julien Leclercq & Jean-Frédéric Durand

Lien vers le Github Lien vers l'Application

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)

D'autres paquets ont été utilisés pour nous simplifier la réalisation de l'interface graphique. Ces paquets sont listés dans le "Gemfile" à la racine du projet.

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 gestion des recherches de tweets - TweetAnalysis : Classe de gestion des analyses de classification - KeywordsAnalysis : Classe de la fonction de classification par mots-clés - KnnAnalysis : 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,un pseudo ou encore un groupe de mots. 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. La méthode "create" nous permet quant-à elle d'effectuer la requête via l'API et d'enregistrer ces données en base. Il est également important de souligner que nous avons défini un critère de recherche complémentaire : Le nombre de tweets à enregistrer par recherche. En effet si nous ne limitons pas le nombre de tweet comprenant le mot clés, celle-ci peut s'avérer très longue et lourde à rentrer en base.

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

Table de Tweets

La construction de la base est visible depuis le répertoire db/schema.rb 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
- tweet_search_id : l'identifiant de la recherche qui nous permet de trier les tweets par recherche.

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.

Table de recherches

Une deuxième table nous permet de sauvegarder les recherches précédemment effectuées. Cette table s'appelle tweet_searches et contient :

- full_text : Le nom de la recherche entré
- created_at : date de création dans la base
- updated_at : date de mise a jour dans la base

L'objectif de cette table de recherche est premièrement de sauvegarder nos recherches pour y accéder plus tard, mais surtout de nous permettre de mettre à jour les recherches par la suite.

Table d'analyses

Une troisième table nous permet de sauvegarder les analyses effectués sur les recherches. Cette table s'appelle tweet_analyses et est contitué des attributs suivants :

- tweet_search_id : l'identifant de recherche des tweets.
- type : le type d'analyse effectué
- neutral_tweets : nombre de tweets neutres
- positive_tweets : nombre de tweets positif
- negative_tweets : nombre de tweets négatif
- last_performed : Date de dernière analyse

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.

L'algorithme est simple on va effectuer une boucle sur tous les Tweets de la base en recherchant les mots clés du tableau. Lorsqu'un mot clé est trouvé on augmente sa cardinalité s'il est positif et on l'a baisse s'il est négatif.

Nous avons fait une petite analyse des tweets par mots-clés et avons remarqué que pas mal d'entres eux comportaient des émoticones "spéciales". Nous avons donc ajouté à la liste ceux que nous avons retrouvé la correspondance UTF-8.

Symbole	Code	UTF-8	Nom	Annotation
👍	1F44D	F0 9F 91 8D	THUMBS UP SIGN	Positif
💋	1F48B	F0 9F 92 8B	KISS MARK	Positif
🎉	1F389	F0 9F 8E 89	PARTY POPPER	Positif
🍻	1F37B	F0 9F 8D BB	CLINKING BEER MUGS	Positif
👌	1F44C	F0 9F 91 8C	OK HAND SIGN	Positif
😍	1F60D	F0 9F 98 8D	SMILING FACE WITH HEART-SHAPED EYES	Positif

Symbole	Code	UTF-8	Nom	Annotation
	1F602	F0 9F 98 82	FACE WITH TEARS OF JOY	Positif
	2764	E2 9D A4	HEAVY BLACK HEART	Positif
	1F4AF	F0 9F 92 AF	HUNDRED POINTS SYMBOL	Positif
	1F609	F0 9F 98 89	WINKING FACE	Positif
	1F607	F0 9F 98 87	SMILING FACE WITH HALO	Positif
	1F60F	F0 9F 98 8F	SMIRKING FACE	Positif
	1F628	F0 9F 98 A8	FEARFUL FACE	Negatif
	1F612	F0 9F 98 92	UNAMUSED FACE	Negatif
	1F601	F0 9F 98 81	GRINNING FACE WITH SMILING EYES	Positif

KNN

KnnAnalysis nécéssite d'avoir annoté au préalable au minimum 20 tweets.

Distance

La fonction définie permet de calculer la distance entre deux tweets. Pour cela on sépare chaque mots des tweets et on les comptabilise dans le résultat. Cette fonction renvoie la distance entre deux tweets.

Plus proche voisin

L'étape suivante est de trouver des voisins. On va donc appliquer notre fonctions distance à tous les tweets annoté dans un premier temps, puis trier les voisins par ordre de proximité à l'aide de sous fonctions. Les plus proches voisins de ces tweets vont être déplacés dans des catégories négatifs, neutres et positifs.

Base de traitement

Maintenant que notre base d'apprentissage est construite. On va répéter l'opération pour les tweets non annotés en se basant sur les tweets déjà annotés et classés. A savoir : calcul des distances entres les tweets. Trie des distances pour trouver les plus proches voisins, puis attribution des catégories pour ces tweets.

Bayes

Non fonctionnel.

IV / Interface graphique

Vue de la page d'accueil

Utilisation : Entrer une recherche dans le premier champ, un nombre de tweets maximum a enregistrer en base dans le deuxième champs.

Vue de la page des précédentes recherches

Utilisation : Toutes les recherches sont enregistrées dans cette page. Il est alors possible de cliquer sur une recherche pour y afficher la liste des tweets et les annoter à la main. Egalement possible d'afficher les différentes analyses ou d'exporter en CSV

Vue d'une analyse

Utilisation : On retrouve ici les différents diagrammes de statistiques. D'une part un diagramme cammembert en pourcentage et d'autre part un diagramme baton en comptabilisant les tweets positifs, négatifs et neutres. Des détails sont également disponibles.

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.