Compte Rendu: Approches de Sentiment Analysis sur des tweets à propos de EM

Mohammed Tadjine & Enzo Bonnal

Extraction

Notre code utilisé pour l'extraction est basé sur l'API twitter4j et est contenu dans le package de notre projet `com.enzobnl.annotweet.extraction`

Annotation

Nous avons annoté les tweets chacun de notre côté et nous avons fait un concensus automatique basé sur des règles.

Techno utilisée

Spark ML en Scala

Tokenisation

C'est la partie sur laquelle nous avons investi le plus de temps.
Voici un apperçu des étapes mises en place à partir d'un exemple:

Original	“Macron il a une bonne testas,on dirait un Makroudh! Mais lui,il a #rien de bon:(#aie https://t.co/FiOiho7”
smileys	“Macron il a une bonne testas,on dirait un Makroudh! Mais lui,il a #rien de bon **sadsmiley** #aie https://t.co/FiOiho7”
URLs	“Macron il a une bonne testas,on dirait un Makroudh! Mais lui,il a #rien de bon sadsmiley #aie **http FiOiho7**“
Ponctuation	“Macron il a une bonne testas on dirait un Makroudh **!** Mais lui il a #rien de bon sadsmiley #aie http FiOiho7”
Split	[“Macron”, “il”, “a”, “une”, “bonne”, “testas”, “on”, “dirait”, “un”, “Makroudh”, “!”, “Mais”, “lui”, “il”, “a”, “#rien”, “de”, “bon”, “sadsmiley”, “#aie”, “http”, “FiOiho7”]
Filtre “Mais”	[“lui”, “il”, “a”, “#rien”, “de”, “bon”, “sadsmiley”, “#aie”, “http”, “FiOiho7”]
Fillers	[“lui”, “il”, “a”, “#rien”, “bon”, “sadsmiley”, “#aie”, “http”, “FiOiho7”]
Hashtags	[“lui”, “il”, “a”, **“rien”**, “bon”, “sadsmiley”, “#aie”, “http”, “FiOiho7”, **“#rien”**]
Paires de mots	[“luiil”, “ila”, “arien”, “rienbon”, “bonsadsmiley”, “lui”, “il”, “a”, “rien”, “bon”, “sadsmiley”, “#aie”, “http”, “FiOiho7”, “#rien”]

Les paires de mots sont surtout efficace lorsque la suite du système (vectorisation et/ou classification) ne permettent pas de prendre en compte le contexte des mots, par exemple TF-IDF + Regression Logistique.

Chaque étape a été selectionnée car elle apportait une amélioration d’accuracy, cependant nous pourrions passer plus de temps à essayer de comprendre l’influence de chacune des étapes sur les résultats pour les affiner : Quelles sont les erreurs qu’elles ôtent ? quelles sont les erreurs qu’elles ajoutent ?

Vectorisation

Pour la vectorisation, nous étions dans un premier temps parti sur un **TF-IDF**, puis nous avons augmenté drastiquement nos performances de 7 points en passant sur du **word2vec**.

Classification

Pour le choix de l'algorithme de classification, nous avons testé (ordonné du meilleur au moins bon):

1. Gradient Boosting Tree (meilleur)
2. Régression Logistique
3. Khiops (implémentation propriétaire d'Orange de l'approche MODL)
4. Random Forest

Nous n'avons pas pu tuner notre GBT autant que nous l'aurions voulu (seul les le nombre d'itération et le learning rate ont été optimisés).

Méthodologie de selection de modèles

Que ce soit pour des modifications de tokenisation, de solution de vectorisation ou de classification, nous avons évalué nos modèles globaux à chaque étape à l'aide de cross-validations (nombre de blocks = 10).

Résultats

Sur nos cross-validation nous obtenons une accuracy de **68%±2%** avec la tokenisation précédemment présentée + Word2Vec + GBT.
Mentions spéciale au temps passé à la tokenisation qui nous a permis de gagner entre 2 et 7 points d'accuracy, selon la vectorisation et la classification qui suis, l'amélioration la plus importante étant constatée lorsque nous enchainions avec du TF-IDF + Regression Logistique.