Projet final R

« Mini-système de recommandation »

I/ Rapport

Construction du projet :

Import des data en JSON, puis réduction aux champs voulus et échantillonnage.

Ensuite, on applique le format « tidytext » aux champs reviewText et summary pour obtenir un tableau ne contenant que les mots qui sont importants pour l'analyse.

L'analyse sentimentale vient ensuite, on fera cette analyse via un couple de lexicons (parmi Bing / NRC / AFINN / Sentiword) en vue d'obtenir un scoring.

Une fois le scoring obtenu, on construira le système de recommandation en UCBF.

1/ Choix des données

Le choix du jeu de données de base provient du site http://jmcauley.ucsd.edu/data/amazon/links.html.

Les datasets contiennent des avis (reviews) de produits provenant du site Amazon. Nous utiliserons un dataset réduit fait pour les expérimentations, ici le choix se portera sur le fichier contenant 231.780 reviews de jeux-vidéo.

Le dataset importer est au format JSON et se présente de la forme :

```
    reviewerID       asin reviewerName helpful
134037 AXM4SLU87FCT7 B002GPPPS4  Vicki Cantu    0, 0

reviewText
134037 This was a cool gMe very different then the ones I used to play but
still very enjoyable for the price I'll give it two thumbs up
       overall   summary unixReviewTime reviewTime
134037       4 Cool game     1386460800 12 8, 2013
```

Ce dataset brut contient plusieurs champs, reviewerID, asin, reviewerName, helpful, reviewText, overall, summary, unixReviewTime et reviewTime.

La seconde étape consistera à échantillonner le dataset, pour ne garder qu'un certain nombre de reviews et ainsi éviter le traitement des 231.780 reviews.

2/ Echantillonnage

Comme vu précédemment, le dataset importer contient 231.780 reviews. Pour limiter le temps de travail des machines et permettre de refaire l'expérimentation rapidement, nous allons échantillonner ce set avec une valeur arbitraire de 250 reviews.

3/ Mise en forme + « tidytext »

La troisième étape de notre travail consistera à mettre en forme le dataset pour qu'il soit plus exploitable et ne garde que les champs utiles à l'analyse sentimentale. Ici, nous avons décidé de garder les champs « reviewerName », « reviewText » et « summary ». Nous choisissons de garder le champ « summary » pour essayer de faire une analyse sentimentale dessus et ensuite la comparer avec l'analyse sentimentale faite sur le champ « reviewText » (la comparaison de ces deux champs est questionnable, en effet, plus un texte est long, plus l'analyse sentimentale sera performante. Toutefois, pour notre projet, nous voulons expérimenter cette option).

La dernière étape de pré-traitement des données sera la mise en forme finale faite via le package « tidytext ». L'utilisation du package tidytext permet d'optimiser la mise en forme des data de type texte en imposant une structure spécifique. En effet, la conversion finale permet alors d'avoir un tableau avec un « token » par ligne. Un token est une unité de texte comme un mot qui nous est utile pour une analyse. La tokenisation, procédé permettant d'obtenir un token par ligne. La tokenisation permet aussi d'obtenir d'autre format du type n-mots par ligne. La fonction utilisée est « unnest_tokens() ».

```
library(tidytext)

text_datafframe %>%
  unnest_tokens(word, reviewText)
```

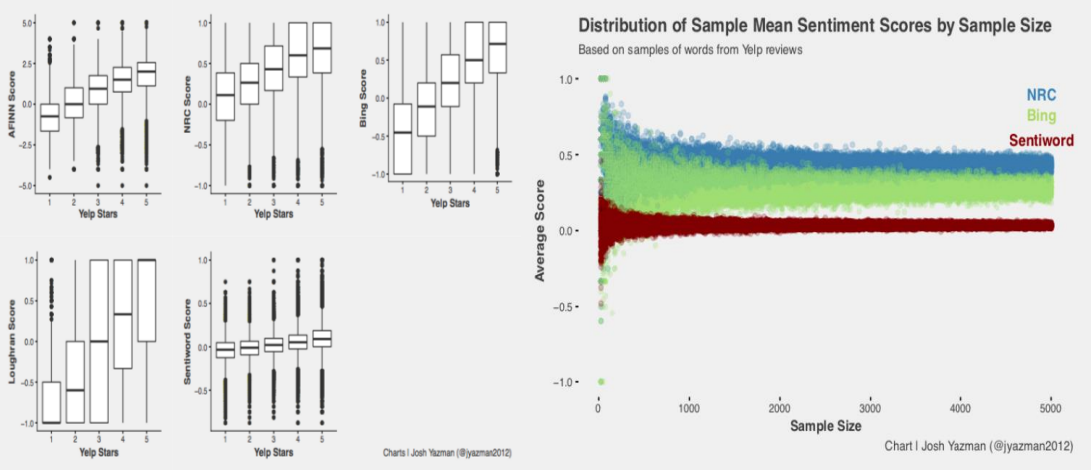
Le fait de transformer les datas sous cette forme permet un traitement facilité par la suite.

4/ Analyse de sentiment + scoring

L'analyse de sentiment est le procédé nous permettant de savoir si un texte dégage un ressenti plutôt positif ou négatif. La méthode la plus logique serait alors de décomposer le texte selon les mots utiles et d'analyser ces mots un par un pour en dégager le ressenti total. De ce point de vue, la décomposition via la méthode tidytext est appropriée. Vient ensuite la question de savoir comment un mot est compté positivement ou négativement. Le package tidytext contient quatre lexiques de sentiments, à savoir Bing, NRC, AFINN et Sentiword, chacun ayant des spécificités propres.

Ces lexiques sont basés sur des mots uniques auxquels on affecte des scores pour des émotions positives ou négatives et possiblement des émotions (haine, joie par exemple). Par exemple, le lexique AFINN assigne aux mots un score allant de -5 à 5, un score négatif étant caractérisé par une émotion négative.

Pour faire notre choix du couple de lexique utilisés, nous allons analyser deux graphiques :



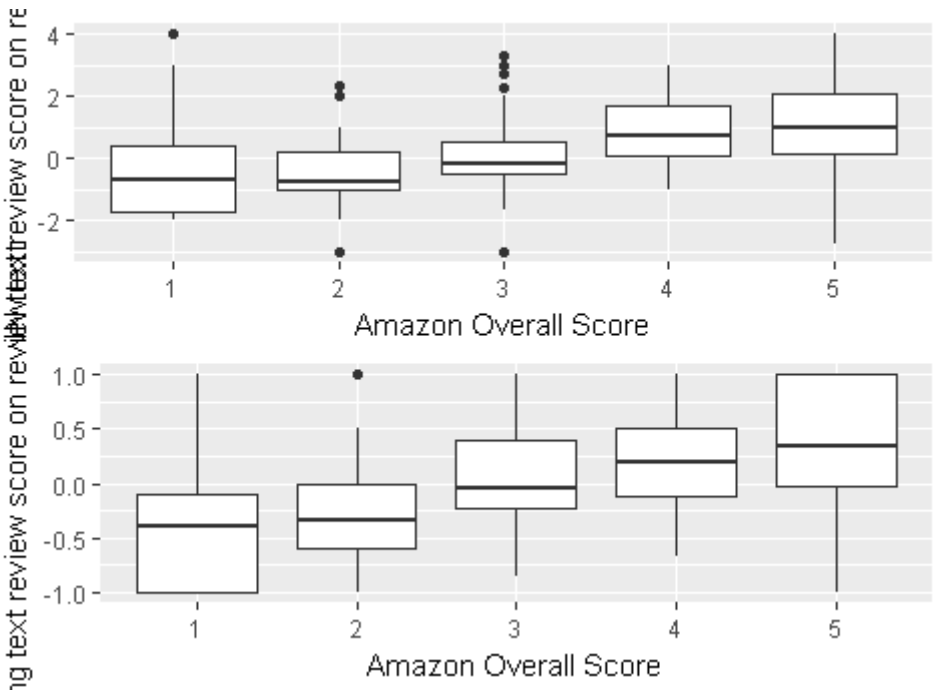
Le graphique à gauche représente le score global attribué par un lexique en fonction d'une review Yelp (allant de 1 à 5 étoiles). Le résultat logique attendu, serait que le lexique dégage un ressenti positif pour une review ayant cinq étoiles et un ressenti négatif voir très légèrement positif pour un review ayant une seule étoile. On peut noter que AFINN et NRC sont quelque peu similaire dans la distribution mais que AFINN présente moins d'écart par rapport au score médian pour chaque étoile. Pour Bing, on note une nette séparation des

scores médians entre chaque étoile malgré des écarts plus prononcés. Le lexique Sentiword ne présente que peu d'écart pour chaque score médian et le ressenti reste quasiment le même pour chaque étoile.

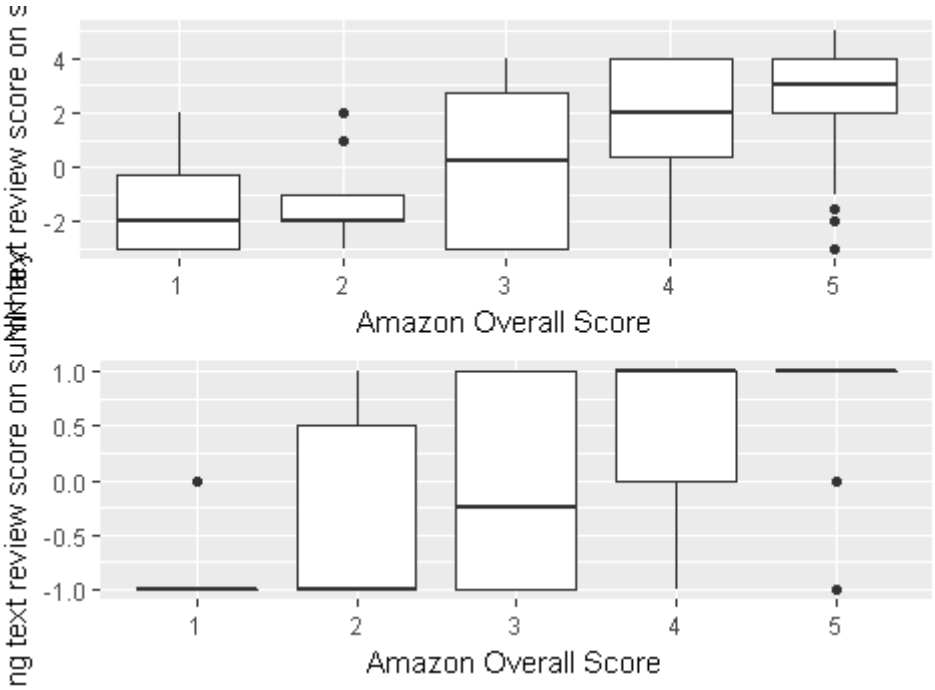
Le deuxième graphique montre le score moyen en fonction du nombre de mots dans un échantillon pour trois lexiques différents (NRC, Bing et Sentiword). La tendance générale est que plus il y a de mots, plus le score convergera vers le ressenti moyen sans variation.

Nous avons donc choisi les lexiques AFINN et Bing pour notre analyse sentimentale.

4.1 Résultats de l'analyse (AFINN & Bing)



Analyse sentimentale appliquée au « reviewText »



Analyse sentimentale appliquée au « summary »

En observant les deux graphiques de résultats (pour une analyse sur la partie « reviewtext » et « summary »), nous voyons que les résultats dégagent la même tendance. Toutefois, l'analyse sur la partie « summary » présente des écarts plus importants et donc une plus grande incertitude. De plus, pour le scoring Bing, les deux extrêmes sont trop catégorique.

5/ Système de recommandation

Le système de recommandation sera fait via la méthode UCBF, User Base Collaborative Filtering. L'idée primaire de cette méthode est que, si des utilisateurs ont eu des centres d'intérêts commun par le passé, il est probable que par la suite, ils aient des similarités sur leurs choix.

Concrètement, si deux utilisateurs ont mis des notes similaires sur un même produit, on peut alors raisonnablement prédire que si l'un achète un autre produit, l'autre utilisateur aura potentiellement de l'intérêt pour ce produit !

Ici, nous allons donc séparer le dataset en deux, une partie pour entraîner le modèle et l'autre utiliser pour la prédiction. Le rapport des deux set (train & predict), nous donneras l'erreur de prédiction.

II/ Code source R

```
## Install Package


## install.packages(c("dplyr",
"jsonlite","recommenderlab","data.table","tidytext","tidyr","stringr","grid
Extra","ggplot2"))

library(data.table)
library(tidyr)
library(tidytext)
library(jsonlite)
library(dplyr)
library(recommenderlab)
library(stringr)
library(gridExtra)
library(ggplot2)


## Import DataSet


dataset_auto <- stream_in(file(fichier_dataset))
datasample <- dataset_auto[sample(nrow(dataset_auto), 250), ]

## Tokenization appliquée à reviewText
datasample$reviewText <- as.character(datasample$reviewText)
tidy_auto <- datasample %>%
  select(reviewerID,asin, reviewText, overall) %>%
  unnest_tokens(word, reviewText) %>%
  filter(!word %in% stop_words$word, str_detect(word, "^[a-z']+$"))

## Tokenization appliquée summary
datasample$reviewText <- as.character(datasample$summary)
tidy_auto_summary <- datasample %>%
  select(reviewerID,asin, summary, overall) %>%
  unnest_tokens(word, summary) %>%
  filter(!word %in% stop_words$word, str_detect(word, "^[a-z']+$"))


## AFINN
afinn <- sentiments %>%
  filter(lexicon == 'AFINN') %>%
  select(word, afinn = score)

## BING
bing <- sentiments %>%
  filter(lexicon == 'bing') %>%
  mutate(bing = ifelse(sentiment == 'positive',1,-1)) %>%
  select(word, bing)

scores_revues <- tidy_auto %>%
  left_join(afinn, by = 'word') %>%
```

```
  left_join(bing, by = 'word')

sommaire_scores_revues <- scores_revues %>%
  group_by(reviewerID, overall) %>%
  summarise(afinn_score = round(mean(afinn, na.rm = T),3),
            bing_score = round(mean(bing, na.rm = T),3))

## Summary
scores_revues_summary <- tidy_auto_summary %>%
  left_join(afinn, by = 'word') %>%
  left_join(bing, by = 'word')

sommaire_scores_revues_summary <- scores_revues_summary %>%
  group_by(reviewerID, overall) %>%
  summarise(afinn_score_summary = round(mean(afinn, na.rm = T),3),
            bing_score_summary = round(mean(bing, na.rm = T),3))


## Result Display

afinn_graphique <- ggplot(sommaire_scores_revues, aes(x =
as.character(overall), y = afinn_score))+
  geom_boxplot()+
  labs(x = 'Amazon Overall Score',
       y = 'AFINN text review score')

bing_graphique <- ggplot(sommaire_scores_revues, aes(x =
as.character(overall), y = bing_score))+
  geom_boxplot()+
  labs(x = 'Amazon Overall Score',
       y = 'Bing text review score')

grid.arrange(afinn_graphique, bing_graphique, nrow = 2)

## Result Display Summary

afinn_graphique_summary <- ggplot(sommaire_scores_revues_summary, aes(x =
as.character(overall), y = afinn_score_summary))+
  geom_boxplot()+
  labs(x = 'Amazon Overall Score',
       y = 'AFINN summary score')

bing_graphique_summary <- ggplot(sommaire_scores_revues_summary, aes(x =
as.character(overall), y = bing_score_summary))+
  geom_boxplot()+
  labs(x = 'Amazon Overall Score',
       y = 'Bing summary score')

grid.arrange(afinn_graphique_summary, bing_graphique_summary, nrow = 2)


## Recommandation System

datasample_rrmatrix <- as(datasample,"realRatingMatrix")

Rec.model <- Recommender(datasample_rrmatrix[1:250], method = "UBCF")

eval <- evaluationScheme(datasample_rrmatrix[1:75], method="split",
train=0.9, given=15)
```

```
rec_ubcf <- Recommender(getData(eval, "train"), "UBCF")

predict_ubcf <- predict(Rec_ubcf, getData(eval, "known"), type="ratings")

error_ubcf <- calcPredictionAccuracy(predict_ubcf, getData(eval,
"unknown"))
```

III/ GitHub

https://github.com/TomRac3r/R_Final_Project