Traitement de textes pour la classification binaire supervisée

Chargement des données

On va commencer par charger les données nécessaires. Pour cela on se rendra à la page :

http://www.cs.cornell.edu/people/pabo/movie-review-data/

Il faut alors télécharger les fichiers utiles à notre étude. La base que l'on va utiliser dans la suite se nomme "Sentiment polarity datasets" version v1.0. Cette base de données contient des textes anglophones de critiques de films. Ces critiques sont réparties dans deux bases de données : l'une pour les critiques positives, et l'autre pour les critiques négatives.

Dans ce TP on utilisera les packages `textir`, `stringr` et `plyr`. Si vous ne disposez pas des droits (non-administrateur) pour installer ces packages vous pouvez consulter l'un des sites suivants selon votre système d'exploitation :

- Sous linux : http://linuxishbell.wordpress.com/2010/12/10/install-r-package-without-root-accesson-linux/
- Sous windows : http://lamages.blogspot.fr/2012/04/installing-r-packages-without-admin.html

1. Charger les données du TP avec les commandes suivantes :

```
#positive part
posText <- read.delim(file='polarityData/rt-polaritydata/rt-polarity-pos.txt',
                      header=FALSE, stringsAsFactors=FALSE)
posText <- posText$V1
posText <- unlist(lapply(posText, function(x) { str_split(x, "\n") }))

#negative part
negText <- read.delim(file='polarityData/rt-polaritydata/rt-polarity-neg.txt',
                      header=FALSE, stringsAsFactors=FALSE)
negText <- negText$V1
negText <- unlist(lapply(negText, function(x) { str_split(x, "\n") }))
```

2. Tester sur la base de données initiales, et déterminer les sources d'erreurs. Tester sur les données :

   http://josephsalmon.eu/enseignement/Telecom/MDI343/rt-polarity_clean.neg
   http://josephsalmon.eu/enseignement/Telecom/MDI343/rt-polarity_clean.pos

3. Déterminer le rôle des fonction `lapply` et `str_split`

Chargement du dictionnaire

Pour essayer de faire de la prédiction sur cette base, on va utiliser un dictionnaire qui classe les mots selon leur degré de positivité. Le dictionnaire est disponible à l'adresse suivante :

http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010

4. Charger les données de AFINN avec les commandes suivantes :

```
#load up word polarity list and format it
afinn_list <- read.delim(file='AFINN/AFINN-111.txt',
                         header=FALSE, stringsAsFactors=FALSE)
names(afinn_list) <- c('word', 'score')
afinn_list$word <- tolower(afinn_list$word)
```

5. Que fait la fonction `tolower` ?
6. Ne garder que 4 niveaux (négatif, très négatif, positif et très positif) en fusionnant les score -5 et -4 ensemble, les -3,-2,-1 ensemble, les 3,2,1 ensemble, et enfin 4 et 5 ensemble. On utilisera notamment l'opérateur logique `|`.
7. Utiliser la fonction Naive Bayes (classifieur naïf de Bayes) du package `e1071` pour prédire les classes. On pourra consulter l'article suivant pour comprendre le fonctionnement de cette dernière fonction :

   https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Document_Classification

8. On extraira 100 lignes positives et 100 lignes négatives de la base de données initiale sur laquelle on testera la performance du classifieur naïf de Bayes.

Le TF-IDF (Term Frequency-Inverse Document Frequency)

On peut définir une statistique TF-IDF (fréquence d'un terme- fréquence inverse dans un document) résumant l'influence d'un mot contenu dans un document. On part du principe qu'ici on dispose d'une matrice dont le terme général donne la fréquence des mots utilisés dans un texte (ici on prend l'exemple du sénat américain, avec 529 "auteurs" différents, *e.g.,* Spencer Bachus ou Barack Obama) parmi les éléments d'un dictionnaire (ici formé d'une liste de 1000 expressions, *e.g.,* health.care.poor ou supreme.court.rejected). Plus d'indications sont données sur le TF-IDF à l'adresse suivante : http://fr.wikipedia.org/wiki/TF-IDF. Le point important est que cette statistique pondère les termes les plus utilisés en tempérant ce nombre par l'inverse de la fréquence du terme dans le document

Dans ce qui suit D est le dictionnaire que l'on considère, t un terme d'intérêt et d est un document considéré.

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}} \qquad \text{(Fréquence normalisée)}$$

$$\mathrm{idf}(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \qquad \text{(Fréquence inverse de document)}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D) \qquad \text{(TF-IDF)}$$

Ici $|D| = 1000$, t varie de 1 à 529 et d de 1 à 1000.

9. Charger le package `textir`.
10. Analyser le type de donnée de `congress109Counts`. Comprendre ce que produit le code suivant :

```
data(congress109)
sort(sdev(tfidf(congress109Counts)), decreasing=TRUE)[1:20]
```

Des éléments supplémentaires en lien avec ce TP peuvent être trouvés sur la page d'A. Bromberg :

http://andybromberg.com/sentiment-analysis/