TD Fouille de textes

Sujet : Fouille de textes avec R
Auteur(s) du sujet : J. Velcin

Préambule

Les TD de fouille de textes seront réalisés à l'aide du langage R. Il s'agit d'un langage interprété très adapté au calcul scientifique et à la fouille de données. Il possède une large collection d'outils statistiques et graphiques, relayée par une communauté très active.

Si vous souhaitez installer R sur votre ordinateur personnel, plusieurs sites web sont consacrés à ce logiciel. En particulier, le site `http://www.r-project.org` offre une description exhaustive sur le langage R et fournit les liens indispensables pour les différents téléchargements, accéder aux différentes bibliothèques de fonctions ainsi que les des documents d'aide. Des versions compilées de R sont disponibles pour Linux, Windows et Mac OS. Pour une interface graphique plus avancée (IDE), vous pouvez préférer l'utilisation de RStudio ici : `http://www.rstudio.com`.

Attention : pensez à commenter abondamment le code et à le sauvegarder régulièrement.

1 Premiers pas avec la librairie TM

Créez un premier fichier intitulé `votrenom_début.R` dans lequel vous recopierez le code R et les différents tests effectués dans les exercices qui suivent, en pensant à inclure les résultats obtenus à l'aide de commentaires.

1. Commencez par installer et charger en mémoire la librairie TM (pour *Text Mining*) fournie par R. En cas de besoin, n'hésitez pas à recourir à la documentation que vous trouverez en ligne : `http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf`.
2. Chargez grâce à la commande `readLines` un corpus textuel de votre choix. Le site du projet Gutenberg, par exemple, fournit de nombreuses œuvres en libre accès : `https://www.gutenberg.org/wiki/Main_Page` et vous pouvez y avoir facilement accès grâce à la librairie `gutenbergr`. Les exemples qui suivent se basent sur le livre "Moby Dick" d'H. Melville.
3. Vérifiez le nombre de lignes qui ont été chargées en mémoire. Si le corpus choisi s'avère trop volumineux pour un traitement en temps raisonnable sur votre machine, n'hésitez pas à ne prendre qu'une fraction du vecteur retourné par `readLines`. Comme la commande `readLines` vous retourne un vecteur de textes, il est très facile de n'utiliser qu'un sous-ensemble composé des n premières lignes (en général, entre 1000 et 5000).
4. Transformez le vecteur de texte en un corpus grâce à la commande `Corpus`. Cette commande prend en paramètre un objet de type `Source`. De nombreuses possibilités existent (`DirSource` pour des fichiers provenant d'un répertoire, `XMLSource` pour un fichier .xml, etc.). Ici, la source qui nous intéresse est `VectorSource` puisque les textes se trouvent précisément dans un vecteur.
5. Regardez ce qui se trouve dans la variable corpus puis dans les trois premiers textes grâce à la commande `inspect`. Ensuite, visualisez le texte d'un seul texte. Pour cela , n'oubliez pas que le corpus est considéré comme une liste.
6. Réaliser un premier nettoyage du corpus en utilisant la commande `tm_map`. Pour le moment, contentez-vous de passer toutes les lettres en minuscules et de retirer les signes de ponctuation. Attention : parfois, certaines opérations sont indiquées directement (cas de `removePunctuation`) alors que d'autres nécessitent de passer par la fonction `content_transformer` (cas de `tolower`).
7. Visualisez le résultat produit sur les premiers documents de votre corpus.

2 Du corpus à la matrice de données

1. Transformez le corpus en une matrice Terms x Documents avec la commande `TermDocumentMatrix`. La pondération par défaut est le nombre d'occurrences de chaque terme dans les documents (TF pour *Term Frequency*). Plus tard, vous pourrez changer cette pondération selon vos besoins.
2. Affichez quelques statistiques simples sur la matrice en regardant ce qu'il y a dans la variable R ainsi créée. Que remarquez-vous, en concordance avec ce que nous avons vu en cours ?
3. Affichez les valeurs correspond à l'un des documents de votre matrice, une fois encore à l'aide de la commande `inspect`. Pour cela, il faut utiliser une représentation compacte qui ignore les cellules vides grâce à la commande `which`. Vous devriez obtenir ce type de sortie :

```
Terms       1
  dick      1
  ebook     1
  gutenberg 1
  herman    1
  melville  1
  moby      1
  project   1
  the       2
  whale     1
```

4. Affichez les mots les plus fréquents dans le corpus à l'aide de la commande `findFreqTerms`. Commentez la liste que vous obtenez.
5. Choisissez deux mots intéressants au vu du corpus choisi (par ex., pour Moby Dick, ils peuvent être "whale" et "ahab"). Calculez les mots qui leur sont le plus souvent associés à l'aide de la commande `findAssocs`.

3 Premières manipulations de la matrice

1. Chargez la matrice Termes x Documents dans une variable R standard afin de procéder aux manipulations qui suivent. Pour cela, vous aurez besoin de la commande `as.matrix`.
2. Vous pouvez à présent visualiser le texte de votre choix mais sans passer par la commande `inspect`. Essayez-le, toujours en passant par la commande `which` pour éviter d'afficher les cellules vides.
3. Affichez le vocabulaire, ou au moins les premiers termes de la matrice, à l'aide de la commande `rownames`.
4. En sommant les lignes de la matrice avec la commande `rowSums`, vous pouvez calculer facilement le nombre d'occurrences total pour chaque mot du vocabulaire. Triez cette valeur afin de pouvoir sauvegarder la liste des termes par ordre décroissant de leur importance.
5. Construisez l'histogramme des 50 termes de fréquence la plus élevée à l'aide d'un `barplot`. Pour information, l'option `las=2` permet d'afficher les étiquettes de l'axe des x de manière verticale.

4 Prétraitements supplémentaires

1. Supprimez les nombres et les mots-outils du corpus, puis relancez le calcul et l'affichage des 50 mots les plus fréquents. Que constatez-vous ?
2. Profitez-en pour supprimer suffisamment de termes du vocabulaire pour conserver entre 1000 et 5000 mots. Pour ce faire, vous utiliserez la commande `removeSparseTerms`.
3. Observez la différence que cela fait en terme de proportion de cellules vides avant et après ces traitements.

4. Afficher une nouvelle fois les termes les plus fréquents mais, cette fois, en utilisant un nuage de mots grâce à la librairie `wordcloud` de R. Limitez-vous aux premiers termes de la liste, par exemple 100 ou 200. N'hésitez pas à une utiliser une palette de couleur grâce à l'option `colors` et la librairie `RColorBrewer`.

5 Mon premier moteur de recherche (préparation)

Créez un nouveau script `votrenom_moteur.R` qui contiendra votre code R ainsi que vos différents tests, en pensant à inclure les résultats obtenus à l'aide de commentaires.

1. Réinitialisez toute la procédure, mais en incluant cette fois-ci la racinisation (*stemming*) et on utilisant la pondération TFxIDF. La racinisation nécessite l'emploi de la librairie `SnowballC` avec l'argument `stemDocument` pour `tm_map`. La pondération peut être précisée dans la commande `TermDocumentMatrix` grâce au paramètre `control = list(weighting=weightTfIdf)`.
2. Vérifiez la taille du vocabulaire obtenu à la suite de ces prétraitements. Cela correspond-il à ce que vous avez vu durant le cours ?
3. Affichez les termes les plus fréquents et commentez l'effet de la racinisation sur votre vocabulaire.
4. Créez une fonction R intitulée `printdoc` dont l'objectif est d'afficher le vecteur correspondant à un ensemble (vecteur) de textes $[d_{f(1)}, d_{f(2)} \dots d_{f(p)}]$ issus de la matrice Termes x Documents m (matrice dans le format natif R). Attention : l'idée est d'obtenir un affichage présentant uniquement les termes qui sont présents dans le texte, comme par exemple :

```
> printdoc(melville.data, c(1))
           Docs
Terms             1
  dick      1.0093411
  ebook     1.2931023
  gutenberg 1.1847606
  herman    1.5540351
  melvill   1.5540351
  mobi      1.0093411
  project   0.9664801
  whale     0.4975050
```

5. Créez une fonction R intitulée `printdoc_raw` dont l'objectif est cette fois d'afficher un ensemble de textes dans leur format original. Pour le même document que précédemment, cela donnerait :

```
> printdoc_raw(corpus.original, c(1))
The Project Gutenberg EBook of Moby Dick; or The Whale, by Herman Melville
```

6 Mon premier moteur de recherche (formuler une requête)

A présent, l'objectif consiste à développer une fonction R pour trier les documents en fonction d'une requête composée de mots-clefs.

1. Ecrivez la fonction `cosine` qui calcule la mesure du cosinus entre deux vecteurs. Pour rappel, le cosinus se calcule ainsi à partir de deux vecteurs x et y :

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x}.\vec{y}}{||\vec{x}||_2 \times ||\vec{y}||_2}$$

   $\vec{x}.\vec{y}$ is the dot product between $\vec{x}$ and $\vec{y}$ and $||\vec{x}||$ stands for the L2 norm : $||\vec{x}||_2 = \sqrt{\sum_i x_i^2}$. Testez cette fonction sur des vecteurs issus de votre matrice de données.
2. Ecrivez la fonction `query2vector` qui construit le vecteur à partir de la matrice de données m et l'ensemble des mots q entrés par l'utilisateur. Pour cela, initialisez avec des 0 un vecteur de longueur égale au nombre de termes dans le vocabulaire, puis attribuez la valeur 1 aux dimensions correspondant aux termes de q.

3. Ecrivez la fonction `run_query` qui retourne la liste triée des documents (ou plutôt de leur index dans la matrice) en fonction de leur similarité du cosinus avec l'ensemble des mots q entrés par l'utilisateur. Cette fonction utilisera les deux fonctions que vous avez écrites dans les exercices précédents.
4. Testez votre fonction requête avec plusieurs entrées possibles (notamment en faisant varier le nombre de mots) et commentez les résultats.