IUT D'ORLÉANS **2ème année**
Département Informatique
Module Aide à la décision

TP5 : Fouille de Textes avec R

Le TP va s'organiser autour de l'exploration d'un petit corpus de textes d'actualité. Ce corpus `actu.zip` est disponible sur Célène : téléchargez-le, décompressez l'archive dans votre répertoire de travail et prenez connaissance des thématiques couvertes.

1 Présentation et installation de la librairie `tm` (Text Mining)

La librairie `tm` prend un charge une palette de structures de données et d'outils qui permettent de réaliser une chaîne de traitements complète pour l'analyse d'un corpus de textes, telle que celle étudiée en cours. En particulier cette librairie offre une structure de donnée `Corpus` pour stocker un ensemble de textes et y accéder aisément. On notera également la fonction générique `tm_map(c,f)` qui, à l'instar des fonctions `apply()`, permet d'appliquer une fonction `f` à chacun des textes d'un corpus `c`.
Les fonctions spécifiques proposées par `tm` aident au pré-traitement (normalisation) des textes, comme par exemple les fonctions `removeWords`, `removePunctuation`, et `stemDocument` qui, assurent respectivement : la suppression d'une liste de mots, la suppression des ponctuation et le *stemming* (racinisation) des mots d'un document.
D'autres fonctions très utiles, applicables cette fois sur le corpus entier, permettent d'extraire les mots fréquents (`findFreqTerms`), de construire une matrice *documents* × *mots* (`DocumentTermMatrix`) ou encore de pondérer cette matrice par exemple en utilisant le *tfidf* vu en cours (`weightTfIdf`).

Rendez-vous sur le site du CRAN (*The Comprehensive R Archive Network*) et recherchez le paquet `tm` dans la liste des paquets disponibles. Observez les dépendances avec d'autres paquets (e.g. NLP, slam, etc.), téléchargez les sources (.tar.gz) des paquets nécessaires et installez-les dans votre session R via la commande `install.packages("xxx.tar.gz")`.

2 Préparation du corpus

1. Appeler la librairie `tm` puis utiliser la fonction `Corpus()` afin de charger le corpus `actu` dans une variable `actu`.

2. Que contiennent les variables `actu`, `actu[1]` et `actu[[1]]`? Visualiser le premier document du corpus `actu`.

3. Lire l'aide, repérez la librairie de provenance puis tester les fonctions `tolower`, `removePunctuation`, `removeNumbers`, `removeWords` et `stemDocument` sur le premier document, sans en stocker les résultats pour le moment, .

4. Observer la liste (prédéfinie) des "mots outils" proposée pour le français via la fonction `stopwords("french")`.

5. Proposer un enchaînement de traitements permettant de réaliser un nettoyage du corpus, écrire une fonction dédiée puis appliquer ce traitement au corpus `actu`:

3 Vectorisation des documents

1. Lire l'aide, la librairie de provenance puis tester la fonction `DocumentTermMatrix` sur le corpus nettoyé. Sauvegarder le résultat dans une variable `mat` et en observer la structure et le contenu général : combien de lignes et colonnes? Combien de valeurs nulles? Nature de ces valeurs?

2. Afficher la liste des termes utilisés comme descripteurs de la matrice `mat`.

3. Utiliser la fonction `findFreqTerms` pour afficher la liste des termes qui apparaissent au moins 20 fois dans le corpus (matrice `mat`).

4. Construire un vecteur de taille 50 tel que la $i$ème composante du vecteur contient le nombre de mots qui apparaissent au moins i fois dans le corpus (matrice `mat`). Tracer la courbe de la taille du vocabulaire relativement au nombre d'apparitions.

5. Stocker dans un vecteur `vocab` les termes qui apparaissent au moins 20 fois dans le corpus. Ensuite créer une nouvelle matrice `mat20` n'utilisant que les mots de `vocab` comme descripteurs.

4 Classification de documents

1. À partir de `mat20`, créer une "vraie" matrice R `M` avec 15 lignes et le nombre de colonne correspondant à la taille du vocabulaire `vocab` plus une colonne contenant des étiquettes de classe.

2. Tester le classifieur plus proche voisins sur ce jeu de données.

3. Visualiser la matrice des distances euclidiennes entre chaque paire de documents.

4. Recommencer les étapes précédentes en considérant d'autres pondérations (binaires ou *tfIdf*) des mots dans les documents.

5. Prendre un nouveau texte de votre choix (par copier/coller sur le web), construire sa représentation sur le vocabulaire `vocab`, et prédire sa classe d'après le classifieur plus proche voisins.

6. Envisager d'autres manières de mesurer la similarité entre documents (e.g. distance du cosinus)