T. D. n° 3
Text mining

Résumé

Ce document est le TD n° 3 du module Modèle pour la datascience. Il reprend rapidement des éléments du cours et propose une mise en pratique interactive des techniques de text mining.

1 Identifier des sujets d'intérêts

FIGURE 1 – Un changement d'algorithme impactant

source :`http://blog.ourchurch.com`

Dans le machine learning et le traitement du langage naturel, le topic modeling est un type de modélisation statistique pour découvrir les «sujets» abstraits qui se produisent dans une collection de documents. Le topic modeling est un outil de text mining fréquemment utilisé pour la découverte de structures sémantiques cachées dans un corps de texte. Intuitivement, étant donné qu'un document est sur un sujet particulier, on pourrait attendre que des mots particuliers apparaissent dans le document plus ou moins fréquemment : "chien" et "os" apparaîtront plus souvent dans les documents sur les chiens, "chat" et "miaou" apparaîtront dans les documents sur les chats, et "le" et "est" apparaîtront également dans les deux. Un document concerne généralement plusieurs sujets dans des proportions différentes ; ainsi, dans un document qui est à 10% sur les chats et 90% sur les chiens, il y aurait probablement environ 9 fois plus de mots de chien que de mots de chat. Les "topics" produits par des algorithmes de topic modeling sont des classes de mots similaires. Une modélisation mathématique, permet l'examen d'un ensemble de documents, et associe chaque mots d'un document, une probabilité d'être dans un topic. Supposons

que vous ayez un ensemble de documents. Vous avez fixé un certain nombre de sujets K à découvrir, et que vous souhaitez utiliser L.D.A. pour apprendre la représentation des sujets de chaque documents et les mots associés à chaque sujets. Pour chaque document, on attribue au hasard chaque mots de ce document à l'un des sujets de K. Notez que pour l'initialisation, on initialise un document à un topic. Lors du processus itératif, pour chaque document d on sélectionne chaque mot w dans d. Et pour chaque topic t, l'algorithme calcule deux choses :

— $P(Topic_t|Document_d)$ = la probabilité d'avoir un topic t dans un document d.

— $P(Mot_w|Topic_t)$ = la probabilité d'avoir un mot w dans un topic t.

Pour affecter un document à un topic t, l'algorithme choisi les topics associés aux mots les plus représentés dans le document (grâce à la loi de Diriclet). Le document appartient aux différents topics avec différentes probabilités (en général on retient le topic avec la meilleure probabilité). [1]

Vous allez étudier les liens d'articles proposés à un utilisateur à travers la plateforme `https://news.google.fr/`

Le document *urls.csv* contient l'url et le texte contenu dans un article proposé à travers un lien. Il y a 446 liens proposés. La plateforme *Google news* étant personnalisée, on cherche à synthétiser le contenu proposé à l'utilisateur.

Commencez par charger le fichier *urls.csv* dans un dataframe. Supprimez les urls où le texte de l'article n'a pas pu être extrait.

```
DataURL <- read.csv("C:/users/claey/Documents/cour/My TD/TD 7/urls.
    csv", sep=",")
df <- as.data.frame(DataURL)
dfUrl <- df[!(is.na(df$Text) | df$Text==""), ]
dfUrl <- dfUrl[! dfUrl$Text=="0", ]
```

1.1 Corpus

Vous allez transformer la liste des textes extraits en un objet *corpus* grâce à la fonction *Corpus()* du package *tm*.

```
# prepare corpus
###############
library(tm)
library(ggplot2)
library(lsa)
library(scatterplot3d)
library(SnowballC)

docs <- Corpus(VectorSource(dfUrl$Text[1:nrow(dfUrl)]))
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, tolower)
```

1. L'échantillonneur de Gibbs est une méthode populaire pour les analyses M.C.M.C. (Markov chain Monte-Carlo). Il constitue une bonne manière d'échantillonner les distributions combinées de plusieurs variables. En se basant sur la notion d'échantillonnage, à travers une distribution combinée, on échantillonne de manière répétitive en prenant en compte les résultats précédents

```
docs <- tm_map(docs, removeWords, stopwords("french"))
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, PlainTextDocument)
dtm <- DocumentTermMatrix(docs)
tdm <- TermDocumentMatrix(docs)
```

À vous !

a) Chercher l'utilité de la fonction *VectorSource*.

b) Justifiez l'application de cette fonction au dataframe.

c) Listez les étapes que nous avons appliqué à notre corpus pour nettoyer le texte.

1.2 Topic modeling

Vous allez utiliser la fonction *LDA()* avec un échantillonnage de Gibbs. L'échantillonnage de Gibbs demande de rentrer un certains nombres de paramètres, notamment le nombre d'itération. Le calcule via la fonction *LDA()* prendra quelques minutes.

```
> library(topicmodels)

#Set parameters for Gibbs sampling
> burnin <- 2000
> iter <- 500
> thin <- 500
> seed <-list(2003,5,63,100001,765)
> nstart <- 5
> best <- TRUE

#Number of topics
> k <- 5

#Run LDA using Gibbs sampling
> ldaOut <-LDA(dtm,k, method="Gibbs", control=list(nstart=nstart,
    seed = seed, best=best, burnin = burnin, iter = iter, thin=thin)
    )

#write out results
#docs to topics
> ldaOut.topics <- as.matrix(topics(ldaOut))
> ldaOut.topics
             [,1]
character(0)    5
character(0)    5
character(0)    5
....
character(0)    4
character(0)    4
character(0)    3
character(0)    4
```

```
#top 10 terms in each topic
> ldaOut.terms <- as.matrix(terms(ldaOut,10))
> ldaOut.terms

      Topic 1       Topic 2      Topic 3    Topic 4    Topic 5
 [1,] "plus"        "septembre"  "cest"     "plus"     "france"
 [2,] "compte"      "france"     "plus"     "google"   "harkis"
 [3,] "france"      "deux"       "premi re" "savoir"    "vues"
 [4,] "aussi"       "croissance" "tr s"     "tous"     "hollande"
 [5,] "cest"        "fois"       "bien"     "toute"    "pr sident"
 [6,] "deux"        "grand"      "deux"     "site"     "droite"
 [7,] "commentaire" "the"        "peu"      "tout"     "fran ais"
 [8,] "millions"    "ans"        "coup"     "services" "fait"
 [9,] "paris"       "entre"      "fait"     "jour"     "septembre"
[10,] "voir"        "premier"    "place"    "comme"    "selon"

topicProbabilities <- as.data.frame(ldaOut@gamma)
> topicProbabilities
             V1           V2          V3           V4           V5
1   0.132812500 0.078125000 0.11875000 0.137500000 0.532812500
2   0.089843750 0.054687500 0.04687500 0.070312500 0.738281250
3   0.107382550 0.087248322 0.12080537 0.080536913 0.604026846
4   0.019230769 0.628959276 0.02036199 0.014705882 0.316742081
...
262 0.196078431 0.196078431 0.19607843 0.215686275 0.196078431
263 0.166666667 0.107843137 0.13725490 0.470588235 0.117647059
264 0.196078431 0.196078431 0.21568627 0.196078431 0.196078431
265 0.196078431 0.196078431 0.19607843 0.215686275 0.196078431
>
```

À vous !

a) Quel est l'avantage d'utiliser l'échantillonnage de Gibbs ?

b) Combien de classes de topic avez-vous paramétré ?

c) Chercher la signification de la loi de Dirichlet. Pourquoi est-elle utile dans la recherche de topic ?

d) Justifiez le paramétrage de la variable *best*.

e) Commentez les résultats.

f) Cherchez à quoi correspond l'attribut *gamma* de votre fonction *LDA()*.

g) Réajustez la variable X dans votre dataframe *dfUrl* (elle doit lister les liens dans l'ordre croissant).

h) A quel topic appartient le 1er lien ? Le 4ième ? Le dernier ? Vérifiez dans la base de données et commentez.

i) Faites varier le nombre de groupes.

1.3 Représentation matricielle

Le corpus de documents transformé en une matrice *dtm*, est une matrice triple. Cette matrice décrit la fréquence des termes qui apparaissent dans une collection de documents. Dans une matrice de documents-termes, les lignes correspondent aux documents de la collection et les colonnes correspondent à des termes. Il existe différents systèmes pour ajouter le poids ou la probabilité pour chaque termes. La technique d'*tf-idf* (term frequency–inverse document frequency) retourne une valeur statistique qui reflète l'importance d'un mot dans un document, pour une collection ou un corpus. Cette valeur est souvent utilisée comme un facteur de pondération dans la recherche d'information et de text mining. La valeur augmente proportionnellement avec nombre de fois où un mot apparaît dans le document, mais est compensé par la fréquence du mot dans le corpus, ce qui contribue à tenir compte du fait que certains mots apparaissent plus fréquemment.

La fréquence d'un terme :
Le choix le plus simple est d'utiliser la fréquence brute d'un terme dans un document, à savoir le nombre de fois que ce terme t se produit dans le document d. La fréquence brute d'un terme dans un document, divisée par la fréquence brute maximale parmi touts les termes dans le document.

FIGURE 2 – Lois probabilistes associées à la fréquence d'un terme

Variants of TF weight

weighting scheme	TF weight
binary	$0, 1$
raw frequency	$f_{t,d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

source :wikipedia

La fréquence inverse d'un document : La fréquence inverse d'un document est une mesure de la quantité d'informations que fournit le mot (si le mots est commun ou rare dans tous les documents). Elle correspond à la fraction inverse logarithmique du nombre total de documents divisé par le nombre de documents contenant le mot.

FIGURE 3 – Lois probabilistes associées à la fréquence inverse d'un document

Variants of IDF weight

weighting scheme	IDF weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t}$
inverse document frequency smooth	$\log(1 + \frac{N}{n_t})$
inverse document frequency max	$\log\left(1 + \frac{\max_{\{t' \in d\}} n_{t'}}{n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

source :wikipedia

Vous allez chercher les termes associés au mots "*françois*". Vous allez également observer les mots fréquents dans le corpus de documents. On décide d'observer si la distribution suit le test de Zipf et la loi Heaps.

```
> findAssocs(dtm, "fran ois", corlimit=0.7)
$fran ois
primaire hollande familles
    0.81     0.75     0.71

> # termes en colonne
> matrice_docs_termes <- (dtm)
> # top 50 des mots fr quents
> findFreqTerms(matrice_docs_termes, 50)
  [1] "abonn s"        "afin"           "ainsi"
  [4] "alors"          "ann e"          "ans"
  [7] "apple"          "apr s"          "article"
 [10] "articles"       "aucun"          "aussi"
 [13] "autre"          "autres"         "avant"
 [16] "avoir"          "ballon"         "beaucoup"
 [19] "belfort"        "bien"           "bon"
 [22] "bonne"          "campagne"       "candidat"
 [25] "car"            "cas"            "cause"
 [28] "centre"         "certains"       "cest"
 [31] "ceux"           "chaque"         "chef"
 [34] "chez"           "clinton"        "comme"
 [37] "commentaire"    "compte"         "conditions"
 [40] "contenu"        "contenus"       "contre"
 [43] "c  t  "           "coup"           "cour"
 [46] "croissance"     " d       "          "dautres"
 [49] "d bat"           "d but"          "d  j  "
 [52] "depuis"         "dernier"        "d  s"
```

```
 [55] "deux"          "devant"         "dimanche"
 [58] "dire"          "direct"         "dit"
 [61] "doit"          "donc"           "donn es"
 [64] "dont"          "droit"          "droite"
 [67] "dun"           "dune"           " galement "
 [70] "elles"         "encore"         "enfants"
 [73] "entre"         " quipe "         "etatsunis"
 [76] " tre "          "exemple"        "face"
 [79] "facebook"      "faire"          "fait"
 [82] "faut"          "figaro"         "fin"
 [85] "fois"          "football"       "fran ais"
 [88] "france"        "fran ois"        "gauche"
 [91] "google"        "gouvernement"   "gr ce"
 [94] "grand"         "groupe"         "guerre"
 [97] "harkis"        "heures"         "hollande"
[100] "homme"         "informations"   "iphone"
[103] "jamais"        "jeunes"         "joseph"
[106] "jour"          "journal"        "journ e"
[109] "jours"         "lancien"        "letat"
[112] "lexpress"      "lieu"           "ligne"
[115] "ligue"         "lillois"        "lire"
[118] "loi"           "lors"           "lundi"
[121] "macron"        "maire"          "match"
[124] "meilleur"      "merci"          "millions"
[127] "ministre"      "minutes"        "mis"
[130] "mise"          "moins"          "mois"
[133] "moment"        "monde"          "nest"
[136] "new"           "nice"           "nicolas"
[139] "nom"           "non"            "notamment"
[142] "nouveau"       "nouvelle"       "paris"
[145] "part"          "partie"         "passe"
[148] "pays"          "pendant"        "personnes"
[151] "petit"         "peu"            "peut"
[154] "place"         "plein"          "plus"
[157] "plusieurs"     "point"          "police"
[160] "politique"     "pouvez"         "pouvoir"
[163] "premier"       "premi re"        "pr s"
[166] "pr sident"      "pr sidentielle" "primaire"
[169] "prix"          "public"         "quand"
[172] "quatre"        "quelques"       "quil"
[175] "rabbin"        "r agir"          "recherche"
[178] "r publique"     "r seaux"         "reste"
[181] "retour"        "rien"           "samedi"
[184] "sant "          "sarkozy"         "savoir"
[187] "s curit "       "selon"           "semaine"
[190] "septembre"     "services"       "sest"
[193] "site"          "sitruk"         "soir"
[196] "sous"          "temps"          "texte"
[199] "the"           "toujours"       "tour"
[202] "tous"          "tout"           "toute"
[205] "toutes"        "travail"        "tr s"
[208] "trois"         "trop"           "trump"
[211] "twitter"       "vendredi"       "vers"
[214] "version"       "verts"          "veut"
```

```
[217] "vid o"          "vie"            "ville"
[220] "voir"           "voiture"        "vue"
[223] "vues"
```

```
> inspect(removeSparseTerms(matrice_docs_termes, 0.4))
<<DocumentTermMatrix (documents: 265, terms: 1)>>
Non-/sparse entries: 180/85
Sparsity           : 32%
Maximal term length: 4
Weighting          : term frequency (tf)

              Terms
Docs           plus
  character(0)    9
  character(0)    0
  character(0)    0
...
  character(0)    6
  character(0)    0
  character(0)    0

> Zipf_plot(matrice_docs_termes)
(Intercept)           x
  9.5920106  -0.9925186
> Heaps_plot(matrice_docs_termes)
(Intercept)           x
  1.2710717   0.7513148
```

FIGURE 4 – Test de Zipf

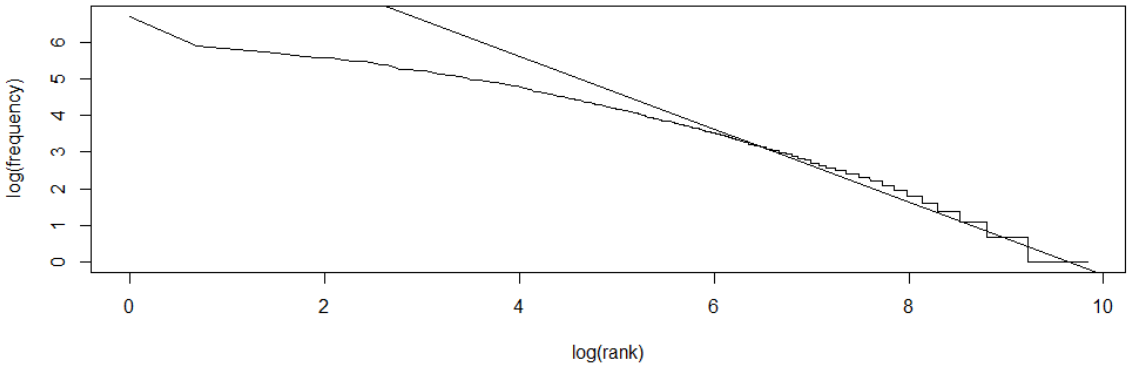
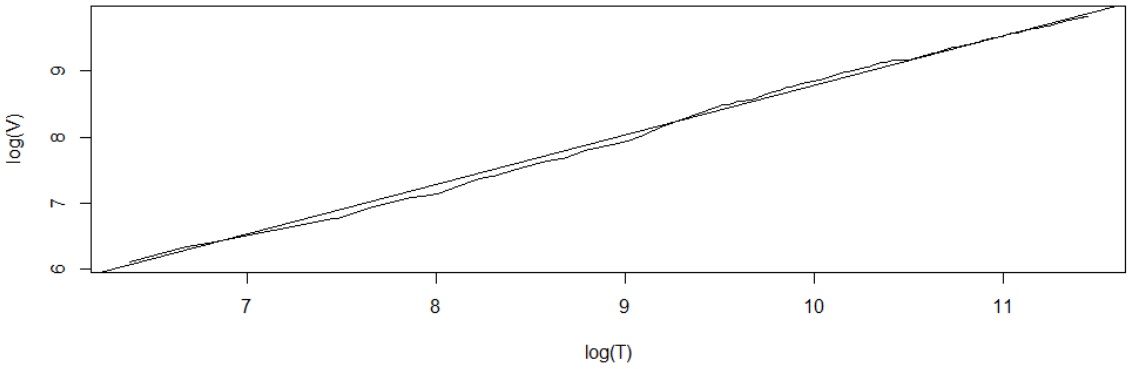


FIGURE 5 – Comparaison avec la loi de Heaps



À vous !

a) Comment la fonction *findAssocs()* à trouvé les mots associés à "*françois*" ?

b) Utilisez *findAssocs()* mais en changant la valeur de *corlimit* à 0,5. Commentez.

c) Qu'avez-vous modifié en utilisant la fonction *removeSparseTerms()* ? Justifiez la modification de la matrice *matrice_ docs_ termes*.

d) Cherchez la signification du test Zipf. Commentez la figure 4.

e) Cherchez la signification de la loi Heaps. Commentez la figure 5.

f) Justifiez la modélisation des probabilités à travers une distribution de Diriclet pour l'allocation de Dirichlet latente (L.D.A.).

1.4 Cluster et nuage de mots

On décide de regarder la proximité des topics. On utilise la fonction *hclust()* que vous connaissez déjà. Le package *wordcloud* vous permet de représenter graphiquement les mots les plus fréquents dans votre corpus.

```
> ######################Cluster###################"
> mtd4.TfIdf <- (DocumentTermMatrix(docs, control=list(weighting=
    weightTfIdf)))
> dim(mtd4.TfIdf)
[1]   265 18629
>
> dist4 <- dist(t(topicProbabilities), method="euclidean")
> dist4
          V1        V2        V3        V4
V2 4.737062
V3 4.604693 4.320373
V4 4.760172 4.663028 4.544935
V5 4.821895 4.695225 4.798466 5.076432
> hc4 <- hclust(dist4, method="ward.D2")
> plot(hc4)
```

```
> library(wordcloud)
> # Wordclouds
> wordcloud(docs,
+           scale=c(5,0.1), rot.per=0.35,
+           min.freq=5, max.words=50, use.r.layout=FALSE,
+           colors= brewer.pal(8,"Spectral")
+ )
```

FIGURE 6 – C.A.H. sur les topics

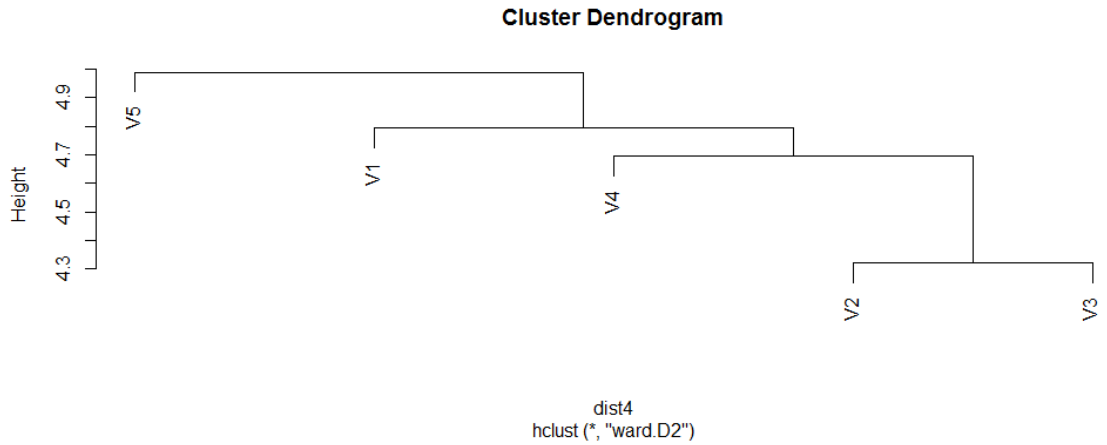


FIGURE 7 – Nuage de mots du corpus docs

À vous !

a) Expliquez comment à été faite la classification ascendante hiérarchique sur votre matrice *topicProbabilities*.

b) Quels topics sont proches ?

c) Supprimez les termes "plus" et "dune" de votre corpus *docs*.

d) Affichez le nuages avec le corpus *docs* modifié.

e) Sachant que l'extraction à été faite entre le 19 et le 25 Septembre 2016, proposez le jour où l'extraction à sûrement eu lieu. Argumentez votre proposition.