2 exercices sous R + 1 exercice sous Python durant cette séance.

A. Accès aux tweets + Expressions régulières

Nous travaillons sous R pour cet exercice.

Préambule : Nous souhaitons travailler à partir des données récoltées directement sur **Twitter** via l'API dédiée, relatifs au thème « #ol ». Jusqu'en 2022, je demandais aux étudiants de se créer un compte développeur préalablement à la séance. Avec leur « access token », ils effectuaient les requêtes directement « on-line ». Cela n'a plus été possible en février 2023. Aucun étudiant n'a réussi à se créer un compte exploitable. Disposant moi-même d'un compte ancien, j'ai réussi à effectuer une ultime requête de 5000 tweets, que j'ai sauvegardé dans un fichier RDS (https://stt4230.rbind.io/manipulation_donnees/lecture_ecriture_r/). Nous travaillerons à partir de ces données.

Pour la petite histoire (vraiment petite, mesquine même je dirais), mon compte est désactivé depuis.

1. Chargez les tweets stockés dans le fichier « **5000_tweets_ol_du_04-02-2023.rds** » (readRDS)
2. Affichez le type de la structure (list).
3. Affichez l'ensemble des tweets.
4. Affichez le premier tweet. Affichez ses propriétés (cf. doc. twitteR, page 21, status-class), à savoir : le texte, le nom de l'auteur, l'ID du tweet, etc. Combien de fois le message a-t-il été retwitté ? (0)
5. Refaites la même opération pour message n°25. Combien de fois a-t-il été retwitté ? (11)
6. Installez et chargez le package « twitter » (https://cran.r-project.org/package=twitteR).
7. Mettez les tweets dans une structure data frame (twListToDF) et sauvegardez le dans un fichier texte **tweets.txt** (séparateur tabulation). Ouvrez le fichier dans Excel pour vérifier que vous disposez bien de l'ensemble des informations.
8. **Revenons sous R**. Affichez la liste des champs du data frame.
9. Quel est le message le plus mis en favori ? (n°4510)
10. Affichez la liste des pseudos des auteurs. Comptabilisez le nombre de messages de chaque auteur. Quels sont les auteurs les plus prolifiques ? (RobinFr90405216 : 227 ; OL__Plus : 215 ; etc.)
11. Récupérez l'ensemble des messages (textes) dans un vecteur msg et affichez-le.
12. Combien et quels sont les textes des tweets qui ont été mis en favoris au moins une fois ? (915) Qui ont été retweeté au moins une fois ? (3282) Quels sont les textes des tweets qui sont en réalité des retweets ? (2763)
13. Affichez la liste des messages où le terme « lyon » (en minuscule ou en majuscule) apparaît au moins une fois (voir l'utilisation de grep) (1403)
14. Affichez la liste des messages où apparaît (grep + voir du côté des expressions régulières [support sur le drive]) :

a. Un chiffre (3561)

b. Le symbole @ indiquant les pseudos des auteurs (3148)

15. Extraire des messages ces pseudos, d'abord en se limitant aux 8 premiers caractères (voir peut-être l'utilisation de regexpr combinée à regmatches) (1888)
16. Extraire des messages ces pseudos, en entier, en vous appuyant sur les délimiteurs (ex. espace, deux points, etc.) (3106)

B. Analyse des sentiments via les thésaurus de polarités

Nous travaillons sous R pour cet exercice.

Le fichier « **sentiment_tweets.txt** » contient une collection de **10403 tweets**, étiquetés selon le sentiment qu'ils véhiculent : **0 - opinion négative, 1 - opinion neutre, 2 - opinion positive**. Ouvrez le fichier dans un éditeur de texte dans un premier temps pour en apprécier la structure.

CHARGEMENT ET PREPARATION DU CORPUS

1. Chargez le fichier des tweets (read.table). Attention : la première correspond à l'en-tête des colonnes (header), le caractère tabulation fait office de séparateur de colonnes (sep), l'encodage est UTF-8 (encoding), et il n'est pas nécessaire de convertir les colonnes (par ex. caractères vers factor) à la volée (as.is).
2. Affichez les caractéristiques du data frame (str). Quel est le nombre de messages ? (nrow) (10403).
3. Affichez la variable « sentiment », comptabilisez l'occurrence des opinions dans le corpus (0 : 1633, 1 : 4911, 2 : 3859)
4. Recodez la variable « sentiment » en {'negative', 'neutral', 'positive'}. Effectuez de nouveau le décompte, les résultats doivent être cohérents avec celles de la précédente question (negative : 1633, neutral : 4911, positive : 3859).
5. Affichez les 6 premiers messages (head).

```
print(head(tweets$message))
[1] "Gas by my house hit $3.39!!!! I m going to Chapel Hill on Sat. :)"
[2] "Theo Walcott is still shit, watch Rafa and Johnny deal with him on Saturday."
[3] "its not that I m a GSP fan, i just hate Nick Diaz. can t wait for february."
[4] "Iranian general says Israel s Iron Dome can t deal with their missiles (keep
talking like that and we may end up finding out)"
[5] "Tehran, Mon Amour: Obama Tried to Establish Ties with the Mullahs
http://pjmedia.com/tatler/2012/10/28/tehran-mon-amour-obama-tried-to-establish-
ties-with-the-mullahs/ … via @PJMedia_com No Barack Obama - Vote Mitt Romney"
[6] "I sat through this whole movie just for Harry and Ron at christmas. ohlawd"
```

6. Chargez la librairie « tm ». Transformez les messages en « Corpus » (cf. notre corrigé de la séance 2 de text mining, le fichier « **reuters.r** ») (Corpus et VectorSource).
7. Nettoyez (tm_map) les documents en appliquant successivement les opérations suivantes :

a. Retirer la ponctuation.

b. Passer la casse en minuscule.

c. Retirer les nombres.

d. Retirer les espaces en trop.

8. L'outil a dû afficher des « warnings » ! Vérifiez le nombre de documents que comporte le corpus par sécurité (10403 toujours, ouf !).
9. Affichez de nouveau les 6 premiers messages.

```
[1] "gas by my house hit i m going to chapel hill on sat"
[2] "theo walcott is still shit watch rafa and johnny deal with him on saturday"
[3] "its not that i m a gsp fan i just hate nick diaz can t wait for february"
[4] "iranian general says israel s iron dome can t deal with their missiles keep talking
like that and we may end up finding out"
[5]  "tehran  mon  amour  obama  tried  to  establish  ties  with  the  mullahs
httppjmediacomtatlertehranmonamourobamatriedtoestablishtieswiththemullahs … via pjmediacom
no barack obama vote mitt romney"
[6] "i sat through this whole movie just for harry and ron at christmas ohlawd"
```

Analyse des sentiments avec « SentimentAnalysis »

10. Installez puis chargez le package « **SentimentAnalysis** » (https://cran.r-project.org/web/packages/SentimentAnalysis/vignettes/SentimentAnalysis.html).
11. Il propose plusieurs dictionnaires de polarités (cf. « Built-in dictionaries »). Nous décidons de nous focaliser sur QDAP qui est le plus généraliste. Chargez le thésaurus en question (loadDictionaryQDAP). Affichez les propriétés associées (attributes), puis la liste des mots correspondant à une opinion positive, ainsi que négative.
12. Analysez les sentiments de notre corpus (analyzeSentiment). Que remarquez-vous dans les « warnings » renvoyés par la commande ? Affichez les premières lignes du tableau de données obtenu.
13. Convertissez les scores « SentimentQDAP » en polarités associées aux documents (convertToDirection).
14. Combien de valeurs obtenez-vous dans le vecteur de prédiction (10403, sinon on est mal). Calculez la fréquence des prédictions (negative : 1591, neutral : 3037, positive : 5775).
15. Confrontez ces prédictions avec les polarités observées des documents via une matrice de confusion. Calculez le taux de reconnaissance (accuracy = 0.499...).
16. Quel serait le taux de reconnaissance si on affecte systématiquement les tweets à la classe majoritaire ? (0.4720). Le résultat avec « SentimentAnalysis » est-il probant ? (pas vraiment).

Analyse des sentiments avec « TidyText »

17. Nous nous proposons de mettre à l'épreuve le package « **tidytext** » - associé à l'univers « tidyverse » - cette fois-ci (https://www.tidytextmining.com/). Installez et chargez la librairie.
18. Ici également, plusieurs dictionnaires sont proposés (cf. « The sentiments dataset »). Chargez le thésaurus « **bing** » (get_sentiments). Affichez les 20 premiers termes (head) (2-faces, abnormal, abolish, abominable, etc.).

19. « tidytext » a besoin que les données soient présentées d'une manière particulière pour être traité, sous la forme d'un « tibble » (une extension du data frame du package « dplyr ») à 2 colonnes avec le numéro de document et le texte associé. Effectuez la transformation de manière à obtenir quelque chose qui ressemble à ceci :

```
head(dfMsg)
# A tibble: 6 x 2
      numero document
      <int> <chr>
1      1 "gas by my house hit i m going to chapel hill on sat "
2      2 "theo walcott is still shit watch rafa and johnny deal with him on saturd…
3      3 "its not that i m a gsp fan i just hate nick diaz can t wait for february"
4      4 "iranian general says israel s iron dome can t deal with their missiles k…
5      5 "tehran mon amour obama tried to establish ties with the mullahs httppjme…
6      6 "i sat through this whole movie just for harry and ron at christmas ohlaw…
```

20. Une seconde transformation est nécessaire pour que l'on puisse procéder à l'analyse des sentiments (unnest_tokens) (cf. « Sentiment analysis with inner join », et aussi : https://rdrr.io/cran/tidytext/man/unnest_tokens.html). Affichez les 20 premières lignes de la structure obtenue. Qu'observez-vous ? Est-ce que les informations sur le corpus sont préservées ?

```
print(head(tbMsg,20))
# A tibble: 20 x 2
   numero mot
    <int> <chr>
 1      1 gas
 2      1 by
 3      1 my
 4      1 house
 5      1 hit
 6      1 i
 7      1 m
 8      1 going
 9      1 to
10      1 chapel
11      1 hill
12      1 on
13      1 sat
14      2 theo
15      2 walcott
16      2 is
17      2 still
18      2 shit
19      2 watch
20      2 rafa
```

21. Appliquez le thésaurus « bing » à ce corpus (inner_join). Affichez les 10 premières lignes du nouveau data frame obtenu. Comment comprenez-vous cette nouvelle structure ?

```
print(head(res,10))
# A tibble: 10 x 3
   numero mot        sentiment
    <int> <chr>      <chr>
 1      2 shit       negative
 2      3 hate       negative
 3      4 like       positive
 4      7 successful positive
 5      7 tough      positive
 6      8 stresses   negative
 7      9 happy      positive
 8     10 winning    positive
```

```
 9     13 best       positive
10     13 excited    positive
```

22. Affichez le document n°7 et faites le parallèle avec la sortie ci-dessus ("with j davlar th main rivals are team poland hopefully we an make it a successful end to a tough week of training tomorrow"). Que constatez-vous ? (*l'outil n'exploite que 2 termes dans le document !!!*)
23. Pour obtenir la polarité des documents, il faut consolider les sentiments associés aux termes qui les composent. Appliquez les règles suivantes :
    a. Si le nombre de termes positifs est supérieur aux négatifs, la polarité du document est « positive ».
    b. L'inverse si les termes « négatifs » sont plus nombreux.
    c. On associe au document la polarité « neutral » si termes positifs et négatifs sont en nombres égaux.
24. Combien de document sont ainsi étiquetés (6486). Quelle est la distribution de fréquences des étiquettes attribuées ? (negative : 1858, neutral : 790, positive :3838).
25. Un certain nombre de documents ne sont pas étiquetés avec ce dispositif (10403 – 6486 = 3917) parce qu'ils ne comportent aucun terme connoté (c'est le cas par exemple des documents n°1, 5, 6, etc.). Pour ceux-là, attribuez arbitrairement l'étiquette « neutral ». Affichez de nouveau la distribution des prédictions (negative : 1858, neutral : 4707, positive : 3838). Ce résultat est-il cohérent avec le précédent ? (oui, seul « neutral » a été modifié)
26. Calculez la matrice de confusion entre les étiquettes observées et attribuées avec « **tidytext** ». Quel est le taux de reconnaissance ? (0.572). Que faut-il en penser ? (c'est nettement mieux que le précédent package…)

ANALYSE DES SENTIMENTS PAR APPRENTISSAGE SUPERVISE

27. Nous souhaitons passer par l'apprentissage supervisé pour catégoriser les opinions. Pour réduire le dictionnaire des termes : retirez du corpus les mots-vides (stopwords), puis effectuez une racinisation (stemming) des termes (cf. le corrigé de la séance 2 de Text Mining – « reuters.r »). Affichez les premières lignes du corpus.

```
print(head(msg$content))
[1] "gas hous hit m go chapel hill sat"
[2] "theo walcott still shit watch rafa johnni deal saturday"
[3] "m gsp fan just hate nick diaz can t wait februari"
[4] "iranian general say israel s iron dome can t deal missil keep talk like
may end find"
[5] "tehran mon amour obama …"
[6] "sat whole movi just harri ron christma ohlawd"
```

28. Scindez aléatoirement les données en **6403** observations pour l'apprentissage et **4000** pour le test (Remarque : mettez set.seed(1) pour que nous ayons des subdivisions identiques).
29. Créez un objet Corpus avec les documents en apprentissage (Corpus, VectorSource).

30. Elaborez la matrice documents-termes à partir de ce corpus d'apprentissage (DocumentTermMatrix). Affichez ses caractéristiques (inspect).
31. Transformez l'objet en matrice standard (as.matrix). Quels en sont les dimensions ? (6403 documents, 14729 descripteurs).
32. Affichez le dictionnaire du corpus d'apprentissage (colnames de la matrice).
33. Transformez la matrice en data frame et adjoignez-lui la variable cible « sentiment » (uniquement pour les individus de l'échantillon d'apprentissage bien sûr). Affichez la distribution des classes pour cet échantillon.
34. Nous cherchons à expliquer la catégorie de sentiments à partir des mots qui composent le document **à partir de la méthode de machine learning de votre choix**. Construisez un modèle prédictif sur les données d'apprentissage.
35. Affichez les caractéristiques du modèle. Y a-t-il des choses à remarquer ?
36. Créez la matrice documents-termes pour le corpus de test, en utilisant le dictionnaire issu du corpus d'apprentissage (cf. par ex. https://stackoverflow.com/questions/44643961/use-documenttermmatrix-in-r-with-dictionary-parameter). Quelles sont les dimensions de la matrice obtenue (3000, même nombre de colonnes que pour la matrice en apprentissage).
37. Vérifiez que les dictionnaires des matrices documents-termes en apprentissage et en test sont bien identiques (oui, sinon il y a un sérieux problème).
38. Appliquez le modèle prédictif sur l'échantillon test, calculez la matrice de confusion et le taux de reconnaissance (0.58925 avec un arbre de décision « rpart » en ce qui me concerne ; les variables importantes étant : good, love, excit, great, happi, best, ...).

Conclusion ? Que penser de cette expérimentation sous R ? (*construire un modèle prédictif « from scratch » n'est pas si mal finalement, tout dépend de la qualité des packages en face...***)**

C. Analyse des sentiments par apprentissage sous Python

Nous travaillons sous Python pour cet exercice.

Le fichier « **sentiment_tweets.txt** » contient une collection de **10403 tweets**, étiquetés selon le sentiment qu'ils véhiculent : **0 opinion négative, 1 opinion neutre, 2 opinion positive**.

Remarque : Ce tutoriel devrait vous aider dans votre démarche : http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#5733803713887666235 (Text Mining : Catégorisation de SMS sous Python)

IMPORTATION ET PREPARATION DES DONNEES

1. Importer le fichier « **sentiment_tweets.txt** » en utilisant la bibliothèque Pandas (http://pandas.pydata.org/). Stockez les données dans une structure que vous appellerez **tweets**.
2. Affichez le type de l'objet tweets.

3. Affichez les caractéristiques de l'objet tweets (columns, dtypes, describe()).
4. Combien y a-t-il de lignes de colonnes dans le data frame ? (shape) (10403, 2)
5. Quel est le nombre de tweets négatifs, neutres et positifs ? (1633, 4911, 3859)
6. Créez un sous-ensemble de données où ne figureraient que les tweets négatifs et positifs. Vous appellerez l'objet **tweets2**. Combien de lignes observe-t-on dans tweets2 ? (5492) (Voir http://pandas.pydata.org/pandas-docs/stable/indexing.html)
7. Recodez la colonne « sentiment » de manière à ce que les codes des tweets positifs soient égaux à 1 (celui des tweets négatifs étant 0 toujours). Vérifiez votre recodage en comptabilisant les tweets de chaque catégorie (0 : 1633, 1 : 3859).

Apprentissage et evaluation

8. Subdivisez les données en corpus d'apprentissage et de test avec respectivement **3492** et **2000** observations (random_state=1). Effectuez un échantillonnage stratifié (stratify) sur la variable cible et vérifiez que vous avez bien (à peu près) la même proportion de positifs et négatifs dans les deux sous-échantillons.
9. Nous souhaitons travailler dans la configuration suivante pour la prédiction :
   9.1. Pas de nettoyage préalable du texte (vraiment pour simplifier)
   9.2. Pondération binaire, retrait des stopwords (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html) [taille de la matrice obtenue : (3492, 11258)].
   9.3. SVM linéaire (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) avec les paramètres par défaut (random_state = 0).
10. Construisez le modèle prédictif sur le corpus d'apprentissage, évaluez-le sur le corpus de test. Quelle valeur de F1-Score obtenez-vous ? (0.827).

Hugging Face

11. Et ça donne quoi si on utilisait un modèle pré-entraîné sur des grands corpus ? (cf. Hugging Face / Transformers : https://www.youtube.com/watch?v=Seqxhwv7I6U)

Nettoyage elabore du corpus

12. Enfin, si on procède à un décrit dans le tutoriel suivant (le code source Python est à disposition) : http://tutoriels-data-science.blogspot.com/p/tutoriels-en-francais.html#2729040876256946384 [(Vidéo) Natural Language Processing avec Keras]. Est-ce que les deux approches concurrentes ci-dessus sont améliorées ?