Olivier CHEVALLIER
ochevall@gmail.com

RCP 216 - Analyse de Sentiment 1

RCP216 S2 2020/2021

INTRODUCTION

L'objectif de ce projet "Analyse de sentiment 1" est de mettre en place une chaîne de traitement de classement binaire (sentiment positif ou négatif) de revues de films issues du site IMDB.

Ce projet sera réalisé sur SPARK en utilisant le langage Scala pour les parties traitement des données et modèles prédictifs, et en utilisant l'outil de Data Visualisation Microsoft POWER BI pour la partie visualisation.

Dans la première partie, nous étudierons les données sources, ainsi que leur modélisation en amont de la partie prédictive.

Ensuite nous mettrons en place les traitements sur les données textuelles permettant d'obtenir une modélisation des données sources utilisables par un modèle prédictif. Pour cette partie du projet nous utiliserons principalement la bibliothèque SPARK NLP.

Dans la troisième partie, nous mettrons en place la partie prédictive en utilisant la méthode des forêts aléatoires.

Pour terminer la quatrième partie sera consacrée à la mise en place d'une visualisation de données.

L'ensemble du code Scala utilisé dans le cadre de ce projet est disponible dans le fichier *sentiment_analysis.scala.*

DONNEES SOURCES ET MODELISATION

Données Sources

Les données sources de ce projet proviennent du site http://ai.stanford.edu/~amaas/data/sentiment/, et ont été récoltées sur IMDB dans le cadre de la réalisation de la publication ci-dessous :

Learning Word Vectors for Sentiment Analysis

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (pages 142–150)

Maas, Andrew L. and Daly, Raymond E. and Pham, Peter T. and Huang, Dan and Ng, Andrew Y. and Potts, Christopher

Association for Computational Linguistics, juin 2011

Le jeu de données est constitué de 50 000 revues de films réparties équitablement en jeu d'entrainement et jeu de test. Nous utiliserons dans le cadre de ce projet la répartition fournie dans les données sources.

50 000 revues supplémentaires non classées sont également fournies, mais ne seront pas utilisées dans le cadre de ce projet d'apprentissage supervisé.

En regardant de près les fichiers sources de données, on constate les informations suivantes sur les revues de films :

- Rédigées en langue anglaise exclusivement
- Grammaire et orthographe corrects (Peu d'écriture de type SMS)

➡ Dictionnaire de lemmatisation et liste de stop-words pour langue anglaise

- Présence de caractères parasites (balises HTML
)

➡ une phase de nettoyage préalable sera à prévoir

- Les textes à analyser sont relativement longs (en moyenne 120 lemmes par revue après suppression des stop words)

➡ Limitation sur la modélisation du texte

Informations à extraire des sources

Chaque revue est contenue dans un fichier de la forme ID_NOTE.txt et se trouve dans un répertoire indiquant s'il s'agit d'un revue positive ou d'une revue négative.

- Identifiant unique de la revue : nom du fichier (sans le .txt)
- Sentiment : nom du répertoire contenant la revue
- Texte : le contenu du fichier

Modélisation

Nous avons vu quelle informations nous devons extraire des sources de données, les questions auxquelles nous devons répondre maintenant sont :

- Quelle méthode de modélisation des données textuel utiliser ?

Les textes sont assez longs, en moyenne 120 lemmes après suppression des stop-words, l'utilisation de représentation Word2Vec n'est pas adaptée. Nous ne chercherons pas ici à regrouper les textes en « concepts », l'objectif n'est pas de regrouper les textes entre eux, mais d'extraire des informations permettant aux algorithmes de ML de classer les textes en 2 classes : sentiment positif ou négatif. Les représentations de type « Analyse sémantique » ne sont pas adaptées.

Nous utiliserons afin de capter le maximum d'information une modélisation de type « Bag of Words », plus précisément une matrice « Document/Terme » contenant le nombre d'occurence de chaque terme.

- Sous quel format les données sont-elles attendues par les méthodes SPARK de ML ?

Les différentes méthodes SPARK de ML prennent en argument une colonne contenant les données d'apprentissage sous forme de vecteur. Il faudra donc transformer les données textuelles en un vecteur, quelque soit la représentation choisie. La variable à prédire (sentiment positif ou négatif) sera encodée sous la forme 1 pour positif et 0 pour négatif.

- Quels sont les différents jeux de données à prévoir ?

Afin de pouvoir entrainer, tester les paramètres et enfin évaluer un modèle de classement supervisé, nous aurons besoin de 2 jeux de données.

Training : pour entrainer le modèle de ML
Test : pour évaluer les résultat du modèle

Ici on utilisera la répartition fournies dans les données sources (répartition 50/50). Le répertoire /train (25K fichiers) sera utilisé pour l'apprentissage, le répertoire /test (25K fichiers également) sera utilisé pour l'évaluation des modèles.

TRAITEMENTS SUR LES DONNEES TEXTUELLES

Dans ce chapitre nous étudierons les traitements SPARK mis en place du chargement des fichiers sources jusqu'à la modélisation utilisée en source du la partie prédictive.

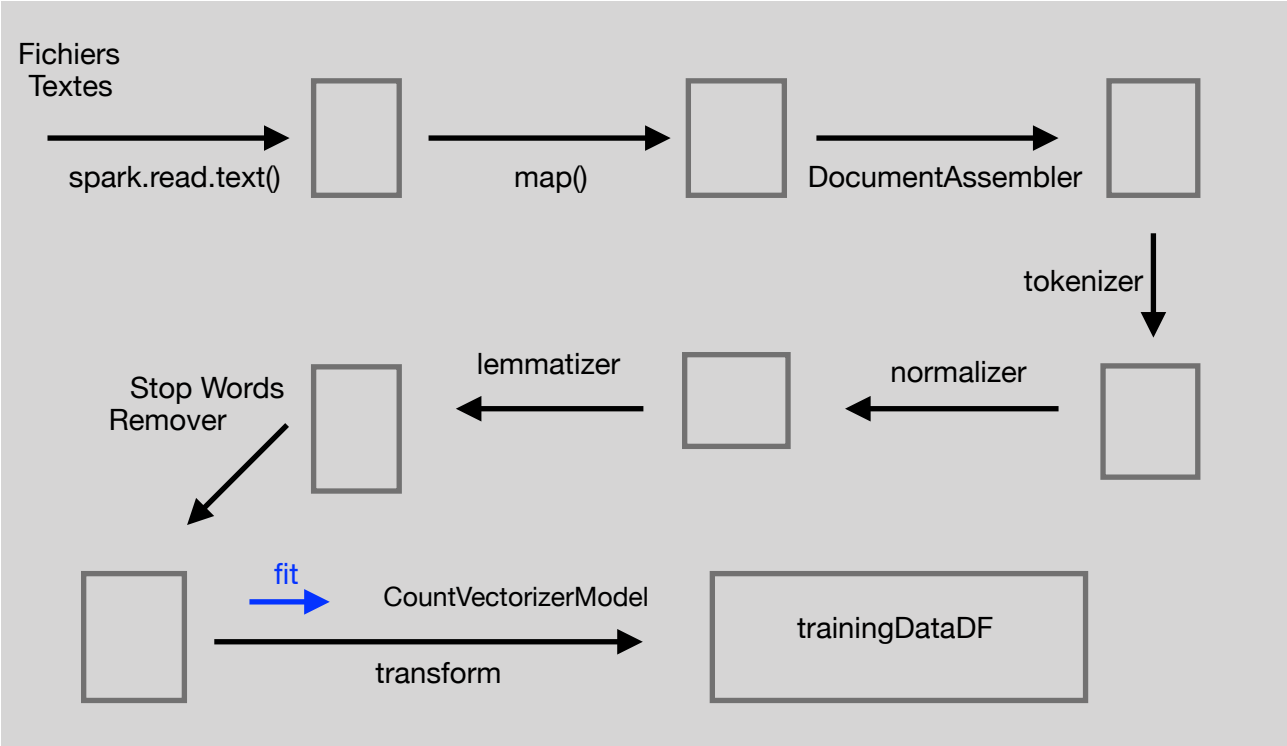


Schéma des traitements appliqués aux fichiers sources du répertoire « train »

Sauf mention contraire les différentes étapes sont indépendantes du type de jeu de données utilisé (training, test), c'est-à-dire appliquées sur l'ensemble des jeux de données.

1) CHARGEMENT ET NETTOYAGE DES FICHIERS SOURCES

Dans cette première opération nous chargeons dans un dataframe le contenu des fichiers nécessaire pour la suite du projet, c'est à dire l'identifiant de la revue, le sentiment associé ainsi que le texte brut du fichier. Dans une seconde opération, les balises HTML </BR> sont remplacées par le caractère ".".

Vérifications du chargement après l'étape de nettoyage :

```
scala> trainingCleanedDF.show(10)
+-------+---------+--------------------+
| fileid|sentiment|              review|
+-------+---------+--------------------+
| 1175_9|      pos|Match 1: Tag Team...|
| 4383_9|      pos|**Attention Spoil...|
|10044_9|      pos|Titanic directed ...|
|4076_10|      pos|By now you've pro...|
|2662_10|      pos|*!!- SPOILERS - !...|
| 1930_4|      neg|Some have praised...|
| 1929_4|      neg|Some have praised...|
|7465_10|      pos|Warning: Does con...|
|4098_10|      pos|Polish film maker...|
| 7726_1|      neg|*** Warning - thi...|
+-------+---------+--------------------+
only showing top 10 rows
```

```
scala> testCleanedDF.count()
res77: Long = 25000
scala>
trainingCleanedDF.count()
res78: Long = 25000
```

2) PREMIÈRE ÉTAPE DES TRAITEMENTS NLP : DOCUMENT ASSEMBLER

Afin de pouvoir utiliser les transformations SPARK NLP, il est nécessaire de passer par une première transformation « documentAssembler » qui va générer un dataframe contenant un type « document » utilisable par les autres transformations.

```
scala> trainingDocumentDF.printSchema()
root
 |-- fileid: string (nullable = true)
 |-- sentiment: string (nullable = true)
 |-- review: string (nullable = true)
 |-- document: array (nullable = true)
 |    |-- element: struct (containsNull = true)
 |    |    |-- annotatorType: string (nullable = true)
 |    |    |-- begin: integer (nullable = false)
 |    |    |-- end: integer (nullable = false)
 |    |    |-- result: string (nullable = true)
 |    |    |-- metadata: map (nullable = true)
 |    |    |    |-- key: string
 |    |    |    |-- value: string (valueContainsNull = true)
 |    |    |-- embeddings: array (nullable = true)
 |    |    |    |-- element: float (containsNull = false)
 |    |    |-- sentence_embeddings: array (nullable = true)
 |    |    |    |-- element: float (containsNull = false)
```

3) TOKÉNISATION ET NORMALISATION

Afin de séparer chacun des termes, nous utilisons ici l'annotateur TOKENIZER de SPARK NLP. Malgré la nécessité d'un appel à la méthode fit(), il se comporte bien comme une transformation et non comme un estimateur. Ci-dessous les 10 premiers tokens :

```
scala> trainingTokens.select(explode($"token").as("test")).select("test.*").show(10)
+-------------+-----+---+------+--------------+----------+-------------------+
|annotatorType|begin|end|result|      metadata|embeddings|sentence_embeddings|
+-------------+-----+---+------+--------------+----------+-------------------+
|        token|    0|  4| Match|[sentence -> 0]|        []|                 []|
|        token|    6|  6|     1|[sentence -> 0]|        []|                 []|
|        token|    7|  7|     :|[sentence -> 0]|        []|                 []|
|        token|    9| 11|   Tag|[sentence -> 0]|        []|                 []|
|        token|   13| 16|  Team|[sentence -> 0]|        []|                 []|
|        token|   18| 22| Table|[sentence -> 0]|        []|                 []|
|        token|   24| 28| Match|[sentence -> 0]|        []|                 []|
|        token|   30| 34| Bubba|[sentence -> 0]|        []|                 []|
|        token|   36| 38|   Ray|[sentence -> 0]|        []|                 []|
|        token|   40| 42|   and|[sentence -> 0]|        []|                 []|
+-------------+-----+---+------+--------------+----------+-------------------+
only showing top 10 rows
```

Afin de supprimer les caractères de ponctuation et transformer l'ensemble des tokens en minuscules, nous faisons appel ensuite à l'annotateur NORMALIZER.

4) LEMMATISATION

Pour la lemmatisation des tokens précédemment extraits, nous utiliserons l'annotateur SPARK NLP LEMMATIZER, pour lequel on doit fournir un dictionnaire de lemmes en langue anglaise. Le dictionnaire utilisé ici est le fichier proposé par défaut sur le site de SPARK NLP *AntBNC_lemmas_ver_001.txt.*

5) SUPPRESSION DES STOP WORDS

N'ayant pu faire fonctionner l'annotateur de suppression des stop words de SPARK NLP, nous utiliserons ici la transformation StopWordRemover de SPARK ML, après avoir transformer le type de la colonne contenant les lemmes.

La liste des stop word est celle par défaut de la transformation.

Vérification du résultat sur les données de test et de training :

```
scala>
testRemovedDF.select(explode($"cleanLemmas")).show(20)
+----------+
|       col|
+----------+
|      sign|
|      lose|
|   highway|
|       say|
|     major|
|   spoiler|
|     ahead|
|   already|
|      know|
|     didnt|
|     since|
|     great|
|      deal|
|    people|
|apparently|
|       get|
|     point|
|     movie|
|        id|
|      like|
+----------+
only showing top 20 rows
```

```
scala>
trainingRemovedDF.select(explode($"cleanLemmas")).show(20)
+--------+
|     col|
+--------+
|   match|
|       1|
|     tag|
|    team|
|   table|
|   match|
|   bubba|
|     ray|
|   spike|
|  dudley|
|      vs|
|   eddie|
|guerrero|
|   chris|
|  benoit|
|   bubba|
|     ray|
|   spike|
|  dudley|
|   start|
+--------+
only showing top 20 rows
```

6) MODELISATION SOUS FORME DE VECTEUR

Afin d'obtenir la modélisation définie au chapitre précédent, les tokens jusqu'à présent modélisés sous forme de tableau de chaînes de caractères vont être transformés en matrice « Document - Terme », et représentés par un vecteur pour chaque document.

Pour cela nous allons utiliser l'estimateur CountVectorizer dont le corpus sera extrait du jeu de données d'entrainement, en 4 étapes :

- Sélection des colonnes nécessaires à la suite du projet
- Encodage du sentiment sur (0,1)
- Entrainement du CountVectorizerModel sur la base du jeu de training obtenu

minDF : on rejette les termes apparaissant moins de 20 fois (réduction du bruit)
maxDF : on rejette les termes apparaissant dans plus de 50% des documents
vocabSize : limitée à 100000 (les 100000 termes les plus fréquents) pour des raisons de performance

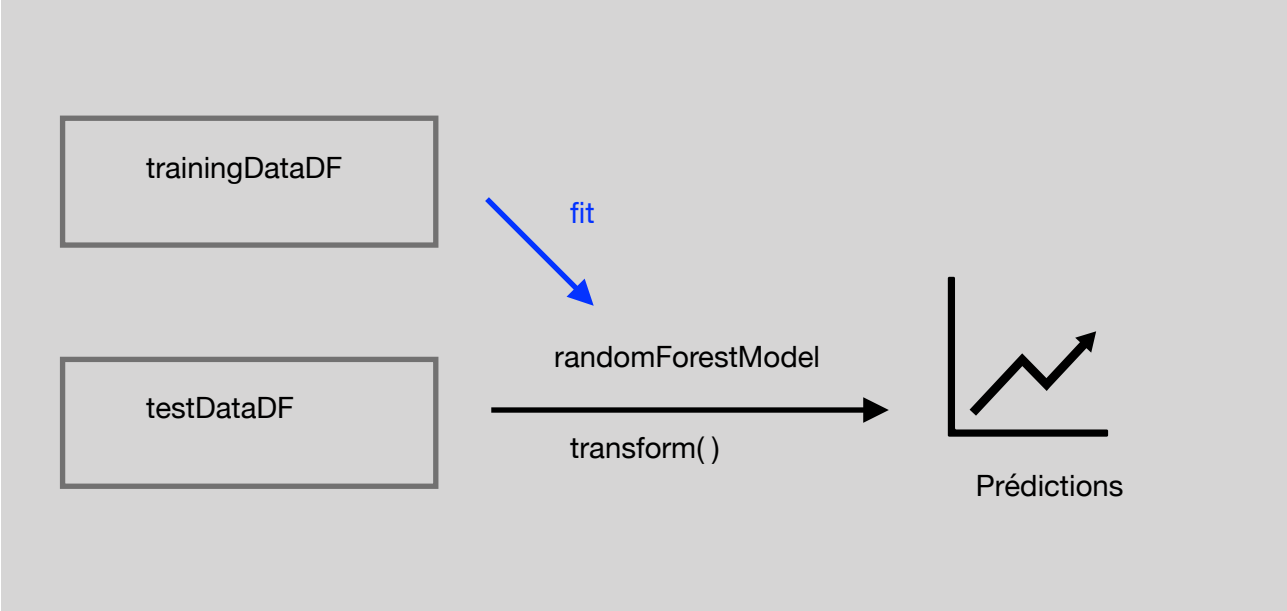
- Application de la transformation sur les 2 jeux de données

On obtient alors 2 dataframes que l'on mettra en cache, et qui seront utilisés dans le prochain chapitre (trainingDataDF, testDataDF)

MODELES DE PREDICTION

Dans cette partie du projet nous allons mettre en place la partie prédictive basée sur les données issues du chapitre précédent. J'ai ici choisi d'utiliser la méthode des « forêts aléatoire » étudiée lors du cours STA211. Cette méthode est adaptée au classement binaire, et limite les risques de sur-apprentissage.

A noter qu'il existe dans la bibliothèque SPARK NLP des pipelines pré-définis d'analyse de sentiment basés sur un corpus dédié, que nous n'utiliserons pas ici car ce n'est pas le sujet de ce projet.



HYPER-PARAMÈTRES DES FORÊTS ALÉATOIRE

Ci-dessous les hyper-paramètres disponibles pour l'implémentation des forêts aléatoires dans SPARK :

Nombre d'arbre
Profondeur maximale des arbres
Mesure d'impureté

ENTRAINEMENT D'UN PREMIER MODÈLE

Entrainons un premier modèle avec des valeurs suivantes :

Nombre d'arbre : 20
Profondeur maximale des arbres : 3
Mesure d'impureté : Gini

```
val randomForestClassifier = new RandomForestClassifier()
  .setImpurity("gini")
  .setMaxDepth(3)
  .setNumTrees(20)
  .setFeatureSubsetStrategy("auto")

// Entrainement du modèle
val randomForestModel = randomForestClassifier.fit(trainingDataDF)

// Application au jeu de test
val predictionDF = randomForestModel.transform(testDataDF)
```

Evaluation du résultat :

```
scala> val accuracy = evaluator.evaluate(predictionDF)
accuracy: Double = 0.8019003936
```

RECHERCHE DES MEILLEURS PARAMÈTRES PAR VALIDATION CROISÉE

Afin d'améliorer ce résultat, nous allons rechercher la meilleure combinaison d'hyper-paramètres pour ce modèle par validation croisée (K-Fold avec k = 5).

Afin de limiter le temps de calcul, nous n'évaluerons que les 2 paramètres suivants :

Nombre d'arbre	:	10,20,30
Profondeur maximale des arbres	:	4, 6, 8

Cela fait donc 45 modèles à calculer.

Définition de la grille :

```
val paramGrid = new ParamGridBuilder()
  .addGrid(randomForestClassifier.numTrees, Array(10, 20, 30))
  .addGrid(randomForestClassifier.maxDepth, Array(4, 6, 8))
  .build()
```

Définition de la validation croisée :

```
val cv = new CrossValidator()
  .setEstimator(randomForestClassifier)
  .setEvaluator(evaluator)
  .setEstimatorParamMaps(paramGrid)
  .setNumFolds(5)
```

Malheureusement il ne sera pas possible d'aller au terme de cette validation croisée, d'explorer les résultats et d'analyser les métriques obtenues, le poste de travail utilisé n'ayant pas les performances nécessaires pour réaliser cette étape, notamment en ce qui concerne la mémoire (erreur *java.lang.OutOfMemoryError: Java heap space* systématique).

VISUALISATION

Dans ce chapitre nous allons nous intéresser aux tokens obtenus après l'étape de suppression des stop-words dans le jeu de données d'entrainement.

Afin d'obtenir une idée des termes les plus fréquents par sentiment, nous allons extraire le nombre de fois qu'un token est présent dans les fichiers sources, et représenter les 75 plus fréquents dans une visualisation de type "Word Cloud" pour chaque sentiment.

Les informations extraites seront ensuite importées dans le logiciel de Data visualisation Microsoft Power BI pour générer la visualisation.

EXTRACTION DES TOP75

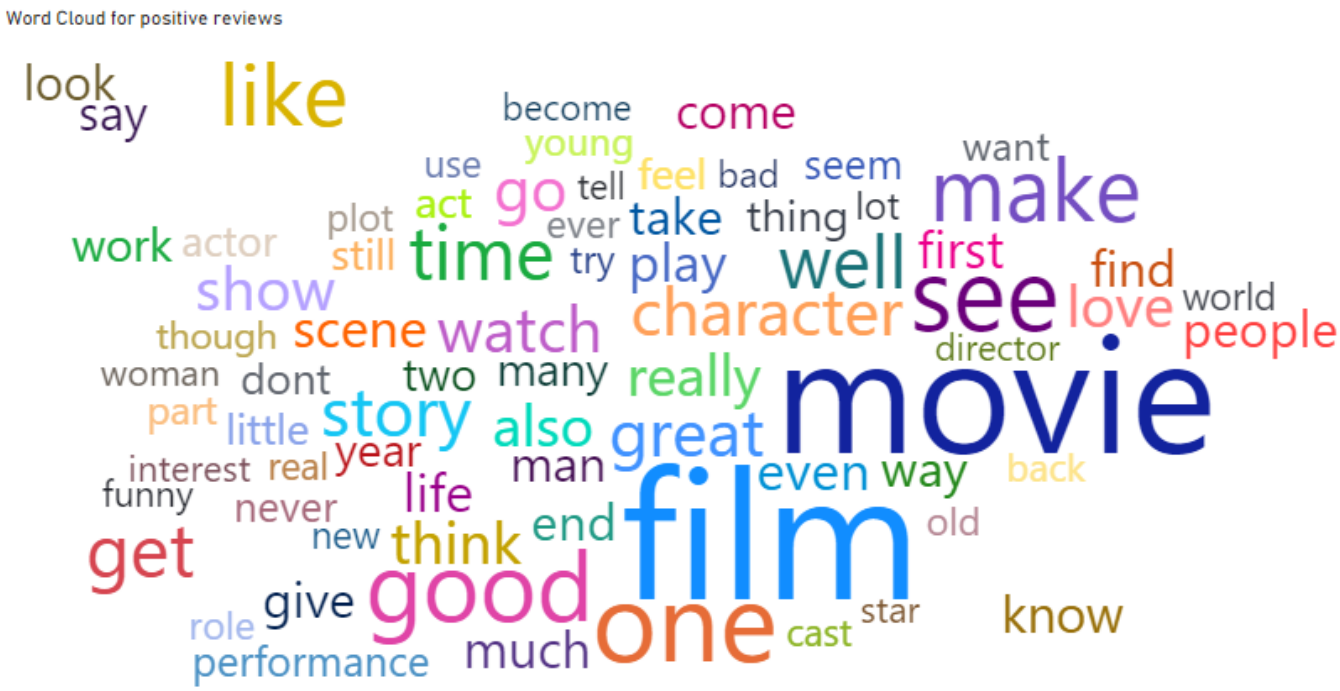
```
// Dataframe des tokens pour les sentiments positifs

val posTokens = trainingRemovedDF.where("sentiment == 
'pos'").select(explode($"cleanLemmas"))


// Extraction du top 75 pour les sentiments positifs

posTokens.groupBy("col").count().filter($"count" >= 2).sort($"count".desc).show(75)


// Dataframe des tokens pour les sentiments négatifs

val negTokens = trainingRemovedDF.where("sentiment == 
'neg'").select(explode($"cleanLemmas"))


// Extraction du top 75 pour les sentiments négatifs

negTokens.groupBy("col").count().filter($"count" >= 2).sort($"count".desc).show(75)
```

PRESENTATION DE LA VISUALISATION "WORD CLOUD"

Afin de garder une certaines lisibilité, la visualisation sera basée sur le top 75. Pour chaque mot la taille représente le nombre d'occurence du token dans les données sources.

Les couleurs affichées dans les visualisation n'ont malheureusement qu'un but esthétique, la visualisation "Word Cloud" dans Power BI ne permettant pas d'assigner une information à la couleur affichée. Il aurait pu être interessant de mettre sur la même visualisation les termes fréquents pour les sentiments positifs et négatifs en les affichant dans 2 couleurs différentes.

TOP 75 POUR LES REVUES POSITIVES



L'on peut voir sur la visualisation les éléments suivants :

- 2 termes sur-représentés qui n'apportent ici aucune information : movie et film.
- Quelques termes positifs assez bien représentés : good, like, well, love, great, funny
- La présence du terme négatif bad
- Beaucoup de termes qui ne semblent pas apporter d'information quant au sentiment de la revue

TOP 75 POUR LES REVUES NÉGATIVES

L'on constate ici les éléments suivants :

- Les termes sur-représentés movie et film sont présents ici également
- La présence du terme négatif bad bien représenté
- La présence de termes positifs assez bien représentés, avec surtout le terme good très présent

Ce travail de visualisation de données a été réalisé en fin de projet, après réalisation des parties traitements des données et mise en place de la partie prédictive. Nous avons pu voir ici qu'il s'agit là d'une erreur.

Ces 2 visualisations remettent en cause la modélisation choisie et définie lors du premier chapitre. En effet, la bonne représentation de termes positifs dans les revues négatives laissent à penser qu'il aurait fallu utiliser une modélisation permettant la prise en compte de la négation (n-gram au lieu de bag of words par exemple).

On aurait également pu rajouter les termes sur-représentés film et movie à la liste des stop words, car ils n'apportent ici aucune information.

CONCLUSION

Nous avons pu voir durant la réalisation de ce projet qu'en utilisant des méthodes "classiques", c'est à dire une modélisation des données de type "Bag of Words" associée à une méthode d'apprentissage supervisé, nous avons obtenu 80% de prévisions correctes sur l'échantillon de test. Il n'a pas été possible pour des raisons de performance d'aller au bout de la phase de validation croisée, qui aurait permis d'améliorer ce résultat.

Il aurait été interessant, mais malheureusement impossible par manque de temps, de pouvoir aller plus loin et de comparer ce résultat aux méthodes optimisées pour l'analyse de sentiment comme par exemple l'approche utilisée par SPARK NLP basée sur un corpus spécifique.

On a pu également voir l'importance de l'utilisation de la visualisation comme outil et non pas seulement pour afficher les résultats du projet. Son utilisation durant la phase de décision concernant la modélisation nous aurait permis de partir dans une direction différente, et probablement plus efficace.