Université Paris Saclay
Novembre 2016

Moteur de recherche et Topic Modeling

Professeurs : Xavier Tannier et Thomas Lavergne

Mohammed Abd Elkhalek, Sihem Abdoun, Fella Belkham, Nawel Medjkoune et Abdelhadi Temmar

Contents

1 Introduction **1**

2 Moteur de Recherche **2**
2.1 Dataset utilisé . . . 2
2.2 Indexation . . . 2
2.2.1 Méthode . . . 2
2.2.2 Limitations et améliorations possible . . . 4
2.3 Méthode utilisée pour la recherche . . . 4
2.3.1 Limitations . . . 5
2.4 Évaluation . . . 5
2.4.1 Pertinences . . . 5
2.5 Structures de données utilisées . . . 7
2.5.1 Performances . . . 7

3 Topic Modeling **11**
3.1 Latent Dirichlet allocation . . . 11
3.1.1 Pre-processing Algorithm . . . 11
3.1.2 Dirichlet distribution . . . 11
3.1.3 Graph plate notation Model . . . 12
3.1.4 Algorithm Implementation . . . 12
3.2 Présentation et Préparation des données pour le LDA . . . 14
3.2.1 Présentation des données . . . 14
3.2.2 Pré-traitement des données . . . 14
3.3 Visualisation . . . 16
3.3.1 Tendance chronologique . . . 17
3.3.2 Interprétation de topics : Word Cloud . . . 17
3.3.3 Topics Share . . . 18
3.3.4 Termite Visualization . . . 19
3.3.5 PyLDAvis . . . 20
3.4 Évaluation du topic modeling . . . 21
3.4.1 Cohérence des topics . . . 22
3.4.2 Pertinence des mots dans un topic . . . 24
3.4.3 Labellisation des topics . . . 25
3.4.4 Tests et résultats . . . 25

4 Conclusion **28**

5 Répartition des tâches **29**

References **30**

Chapter 1

Introduction

Contexte:
Avec l'expansion vertigineuse des activités sur internet par la mondialisation de la toile et l'afflux des médias, le besoin à des outils capables d'extraire de l'information à partir de grandes masses de données textuelles s'est fait ressentir. D'une part il y a les outils incontournables dont aucun (ou presque) utilisateur internet ne se passe, ce sont les moteurs de recherche. Ces derniers doivent être capables de répondre à une requête utilisateur en retournant les documents les plus pertinents qui soient. Depuis des années, plusieurs recherches tentent d'améliorer le rendu des moteurs des recherches tantôt en travaillant directement à l'amélioration de ces derniers tantôt en travaillant à des reformulations et des représentations plus appropriées des requêtes utilisateur. D'autre part il y a d'autres outils moins répandus mais dont l'utilité est plus spécifique tels que les outils qui permettent l'identification de thèmes abstraits dans des documents. Ces derniers, peuvent notamment être exploités pour la mise en œuvre de moteurs de recherche.

Objectifs
Le premier objectif définit par ce travail est la mise en œuvre d'un moteur de recherche en respectant un certain nombre de contraintes. Tout d'abord, le système ne doit pas consommer plus de 1Go de mémoire vive ni à l'indexation ni à la recherche et l'index ne doit pas dépasser 60% de la taille de la collection. Pour l'indexation, il est requis d'employer au moins deux techniques d'indexation. Le modèle vectoriel est utilisé avec la mesure de similarité cosinus. Au moins deux versions du programmes doivent être proposées: l'une s'appuyant sur l'index non normalisé, l'autre sur l'index avec stemming. Une fois le moteur de recherche au point, procéder à l'évaluation de la pertinence et des performances. Le second objectif relève d'un choix inhérent au groupe, nous avons choisi de mettre en œuvre un programme qui permet, à partir d'une collection de documents, d'en extraire des thèmes abstraits, de permettre leur visualisation et de donner une indication sur la performance du programme ce qui, nous le verrons, n'est pas une tâche triviale.

Chapter 2

Moteur de Recherche

Le but ici était de développer un moteur de recherche capable d'indexer 10 000 documents et de retourner les documents les plus similaires à une requête. Pour cela, des contraintes nous sont imposées, le programme ne doit pas utiliser plus de 1GO de mémoire RAM et le fichier index ne doit pas dépasser 60% de la taille du corpus.

Deux versions de ce moteur sont proposées, un utilisant la normalisation stemmer et un autre utilisant la normalisation tokenizer(pas de normalisation à part rendre toutes les lettres en miniscule). Les deux normaliseurs suppriment les mots vides. En effet, ici l'ordre des mots n'est pas prit en compte dans la recherche, les mots vides n'ont donc plus aucune utilité.

2.1 Dataset utilisé

Le corpus utilisé ici est celui fournit pour le projet. Un corpus contenant environ 10000 documents datant entre janvier et avril 2015.

2.2 Indexation

2.2.1 Méthode

L'index contient pour chaque mot, la liste des fichiers dans lesquels il apparaît, ainsi que son poids dans chaque fichier.

Pour obtenir un fichier index peu volumineux, il a été nécessaire de donner un identifiant aux fichiers. Ainsi, nous avons 2 fichiers, un qui contient les identifiants pour chaque fichier, et l'index contenant pour chaque mot, la liste des fichiers auquel il apparaît, ainsi que son poids dans ce fichier.

Pour donner un identifiant aux fichiers, nous générons une suite de caractères composée des lettres :

"0123456789$ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz$"

Aussi, on donne aux fichiers les plus volumineux les identifiants contenant les moins de lettres. En effet, un fichiers contenant beaucoup de mots apparaîtra plus

souvent dans l'index, cela permet d'économiser de l'espace. Avec cette méthode là, on a besoin d'au maximum 3 caractères pour identifier un fichier. En effet, avec 3 caractères on peut générer $(26*2+9)^3 = 226981$ identifiants différents. Si nous avions choisit l'identification par entier, 9000 fichiers auraient eu besoin de 4 caractères pour être identifier.
La figure 2.2.1 montre comment est organisé l'index.

Pour calculer le poids d'un mot dans un document, on utilise la méthode du tfidf qui correspond à calculer :

$$w_{t,d} = tf_{t,d} * log(\frac{n}{df_t})$$

Avec :

- $w_{t,d}$ est le poids du terme t dans le document d, relativement au corpus.
- $tf_{t,d}$ est le nombre d'occurrences du terme t dans le document d.
- n est le nombre de documents dans le corpus.
- df_t est le nombre de documents du corps contenant le terme t.

La caractéristique la plus intéressante du tfidf est la prise en compte de la rareté du mot dans le corpus. Calculer $log(\frac{n}{df_t})$ permet de donner un poids plus élevé aux mots très peu présents dans le corpus.

Les poids (tfidf) contiennent plusieurs chiffres décimales après la virgule. Pour diminuer la taille de l'index, on a décidé de ne garder que le premier chiffre après la virgule pour chaque poids. Ce choix implique une légère perte de précision mais un gain non négligeable en mémoire.

```
bizarr  3T,U8,1U,oT,1MB,0oh,0z,7b,7c,sb,iY,UE,0UV,R,0C,SF,S,uh,0
bizarrer    Q,R,S,wk,wl,oF,0k,M 7.1,7.1,7.1,21.3,21.3,7.1,7.1,7.
bizart  R,S 8.5,8.5
bizdesiziseviyoruz  12u 9.2
bizert  Du  9.2
bizertin    Du  18.4
bizet   0UV,El,Em,5B    15.6,7.8,7.8,7.8
bizon   ms,mt   8.5,8.5
bizoz   Sg  9.2
```

Figure 2.1: Extrait d'un index, sur chaque ligne on trouve un mot suivie des fichiers dans lequel il apparaît suivie du poids du mot dans chaque fichier

2.2.2 Limitations et améliorations possible

L'indexation faite ici présente beaucoup de limitations et est sujette à des améliorations possibles.

Premièrement, l'ordre des mots dans les documents est totalement perdu avec cette méthode d'indexation. Pour pouvoir garder l'ordre des mots, il faudrait utiliser des n-grammes, ce qui augmente la taille de notre index de façon polynomiale.

Un autre point important, est la perte en précision en raison du choix que nous avons fait de tronquer les tfidfs.

La mise à jour de l'index implique la réécriture de l'index car celui-ci est enregistré dans l'ordre alphabétique des mots.

La fusion d'index n'est pas utilisée et n'est pas utile ici, si le corpus grandit il sera nécessaire de répartir l'index sur plusieurs fichiers différents.

Le choix d'identifier un fichier par une chaîne de caractères à la place d'un entier, implique une utilisation de la mémoire RAM plus grande. En effet, l'entête d'un String occupe plus d'espace mémoire que l'entête d'un Integer. Par conséquent une HashMap<String,String> consommera plus de mémoire RAM qu'une HashMap<String, Integer>. Ce choix a tout de même été privilégié pour réduire la taille de l'index sur le disque dur.

Malgré toutes ces limitations, l'indexation respecte toutes les contraintes imposées comme nous le verrons dans la section 2.5.1.

2.3 Méthode utilisée pour la recherche

Le moteur de recherche utilise le modèle vectoriel pour représenter une requête et un document. Par conséquent pour mesurer la similarité entre une requête et un document, la similarité par cosinus est utilisée :

$$sim(\vec{Q}, \vec{D}) = \frac{\vec{Q} \bullet \vec{D}}{|\vec{Q}| \times |\vec{D}|} = \frac{\sum_{i=1}^{n} w_{i,Q} \times w_{i,D}}{\sqrt{\sum w_{i,Q}^2} \times \sqrt{\sum w_{i,D}^2}}$$

Avec $\vec{Q}$ représentant la requête et $\vec{D}$ le document.

On représente la requête avec la moyenne des tfidfs de chaque mot de la requête. On utilise cette méthode pour essayer de discerner les mots les plus importants dans la requête. Ainsi, si par exemple l'utilisateur recherche "Bon vélo" et qu'il existe aucun document dans le corpus qui contient les deux mots mais juste des fichiers qui contiennent soit "Bon", soit "vélo". Le programme saura qu'il faut privilégier le mot ayant le tfidf moyen le plus élevé (ici ça sera sûrement "vélo").

L'avantage de la similarité cosinus est qu'elle normalise suivant la longueur des vecteurs(et donc la somme des poids des mots dans un document). Cela

permet d'avoir une mesure de distance qui n'est pas discriminante envers les fichiers contenant peu de mots. De plus, cette distance se situe obligatoirement entre 0 et 1. Ce qui signifie que l'on peut avoir une idée de la pertinence d'un document juste en regardant son score, sans avoir besoin de comparer avec les autres scores.

Pour retourner les documents les plus pertinents, le programme suit les étapes suivantes :

- Lecture de l'index pour récupérer les documents qui contiennent au moins un des mots de la requête
- Calcule de la distance de similarité entre la requête et chacun de ces documents
- Trie des documents par ordre de similarité
- Renvoie les 100 documents les plus pertinents

2.3.1 Limitations

Un défaut majeur de la similarité par cosinus est qu'il suffit que tout les mots de la requête soient présents dans le document pour que la distance par cosinus soit très proche de 1.
Limitation du tfidf (ne prend pas en compte la position des mots). La recherche ne prend donc pas en compte l'ordre des mots dans la requête de l'utilisateur.

Aussi, prendre la moyenne des tfidfs de chaque mot pour représenter la requête n'est pas toujours la meilleure solution. En effet, les poids calculés sur le corpus, ne sont pas forcément représentatifs de la volonté de l'utilisateur. On pourrait aussi demander à l'utilisateur de trier les mots de sa requête par ordre d'importance, mais cela rendrait la recherche pénible pour ce dernier.

Ce moteur présente un gros désavantage, aucune interface graphique n'est implémentée. Le programme renvoie juste une liste de documents. Ce n'était pas le but du projet, mais il est impérative pour un moteur de recherche d'avoir un interface agréable pour qu'il soit utilisable.

2.4 Évaluation

2.4.1 Pertinences

Le moteur de recherche a été testé sur les 10 requêtes suivantes :

- Charlie Hebdo
- volcan
- playoffs NBA
- accidents d'avion
- laïcité
- élections législatives
- Sepp Blatter
- budget de la défense
- Galaxy S6
- Kurdes

Pour juger de la pertinence des résultats retournés, un fichier est généré à la fin de chaque requête. Il est de la forme suivante :

```
Requete renvoyée en 262ms

20150120_d07e7d3653cf4ba7001f670c18026345.txt : 0.9999903895900052
    charlie : 5 fois  pourcentage : 0.004591368%
    hebdo : 4 fois  pourcentage : 0.0036730946%
20150120_9f37b1bd9f9eafb31d30269f72c8a5ff.txt : 0.9999903895900052
    charlie : 5 fois  pourcentage : 0.013227513%
    hebdo : 4 fois  pourcentage : 0.01058201%
20150112_026d2ed0eb31f99e17224275edd9d30b.txt : 0.9999903895900052
    charlie : 5 fois  pourcentage : 0.006038647%
    hebdo : 4 fois  pourcentage : 0.004830918%
20150108_a22682c1ca6630a2083bc16555ba9946.txt : 0.9999903895900052
    charlie : 5 fois  pourcentage : 0.041322313%
    hebdo : 4 fois  pourcentage : 0.03305785%
```

Figure 2.2: Extrait fichier générée après recherche avec normalisation tokenizer

```
Requete renvoyée en 197ms

20150113_eb676ee8433a6998b95bcc24f2f8122d.txt : 0.9999999741100307
    charl : 8 fois  pourcentage : 0.031496063%
    hebdo : 7 fois  pourcentage : 0.027559055%
20150114_ad5ab58e40489e96bd8141c4b44f92e3.txt : 0.9999999741100307
    charl : 8 fois  pourcentage : 0.0075901328%
    hebdo : 7 fois  pourcentage : 0.006641366%
20150108_d1e685fc8dc3666f765586846c549679.txt : 0.9999999741100307
    charl : 8 fois  pourcentage : 0.012048192%
    hebdo : 7 fois  pourcentage : 0.010542168%
20150123_9cbaf460f86bfc0cf8a65d5f230331a1.txt : 0.9999876189040134
    charl : 9 fois  pourcentage : 0.035856575%
    hebdo : 8 fois  pourcentage : 0.03187251%
```

Figure 2.3: Extrait fichier générée après recherche avec normalisation stemmer

Il permet de savoir pour chaque fichier retourné, le nombre de fois qu'est présent chaque mots de la requête ainsi que son pourcentage par rapport au doc-

ument. Les fichiers sont listés dans l'ordre décroissant des tfidf. Ainsi, juste en regardant le fichier, nous sommes capables de juger approximativement la pertinence des résultats retournés par le moteur de recherche. Nous avons donc pour chaque requête regardé ce fichier et observé si l'ordre des documents retournés a un sens. Et c'est parfaitement le cas, on retrouve toujours à la fin de cette liste les documents contenant très peu de fois les mots recherchés et en haut de la liste l'inverse. On remarque aussi qu'il y a beaucoup d'égalités entre les scores, ce qui va compliqué la tâche à l'utilisateur pour trouver ce qu'il recherche.

Cette méthode d'évaluation est très peu précise car l'avis de l'utilisateur n'est à aucun moment pris en compte. Elle ne permet pas de donner un score (comme par exemple un taux de réussite), car nous ne savons pas ce que l'utilisateur recherche réellement.

La meilleure façon de juger la pertinence reste de demander directement à l'utilisateur si il a trouvé ce qu'il recherche pour chaque requête et ainsi pouvoir calculer un taux de réussite.

2.5 Structures de données utilisées

La structure de données la plus utilisée durant ce projet est sans aucun doute, la hashmap. En effet, il est souvent nécessaire de stocker en mémoire différentes associations (comme par exemple le tfidf de chaque mot pour chaque fichier). Une hashmap permet de faire ça de manière efficace et simple. La TreeMap est ici utilisée que lors de la création de l'index, dans le but d'avoir notre index dans l'ordre alphabétique des mots. Cela simplifiera la tache si l'on souhaite faire de la fusion d'index dans le future.

Bien que solution de facilité ici, utiliser des hashmap ou treemap pour indexer un grand volume de données n'est pas la meilleur solution. En effet, une HashMap prend beaucoup de place en mémoire en raison des entêtes ajoutées à chaque entrée. En effet, cette structure de données à besoin de 48 octets pour l'entête et 40 octets d'entêtes pour chaque entrées. On à donc à la fin une grande majorité de la mémoire RAM qui est occupée par les entêtes. Utiliser deux tableaux à la place diminuerait considérablement l'espace mémoire. Néanmoins, étant donné la taille de notre corpus, on a privilégier l'écriture d'un code simple et facile à lire.

2.5.1 Performances

Mémoire RAM consommée

Pour évaluer, la mémoire RAM consommée par le programme, l'outil JConsole a été utilisé.

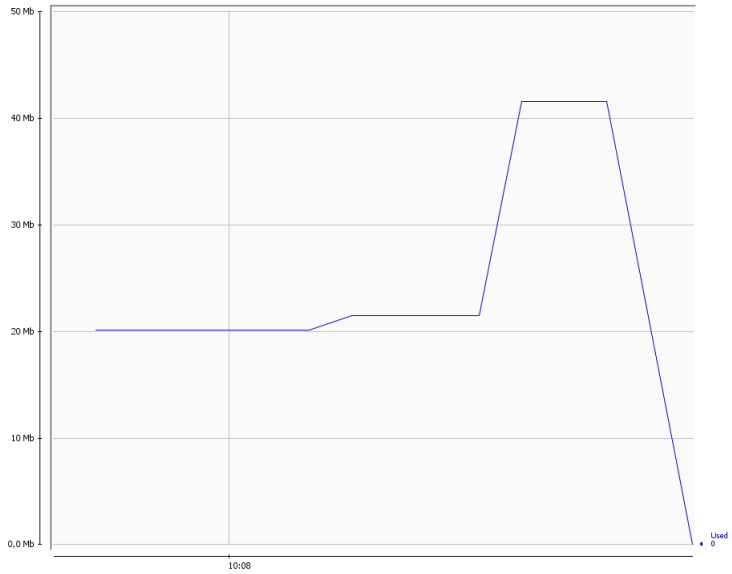

Figure 2.4: Evolution de la mémoire consommée lors de l'utilisation du moteur de recherche avec l'index stemmer

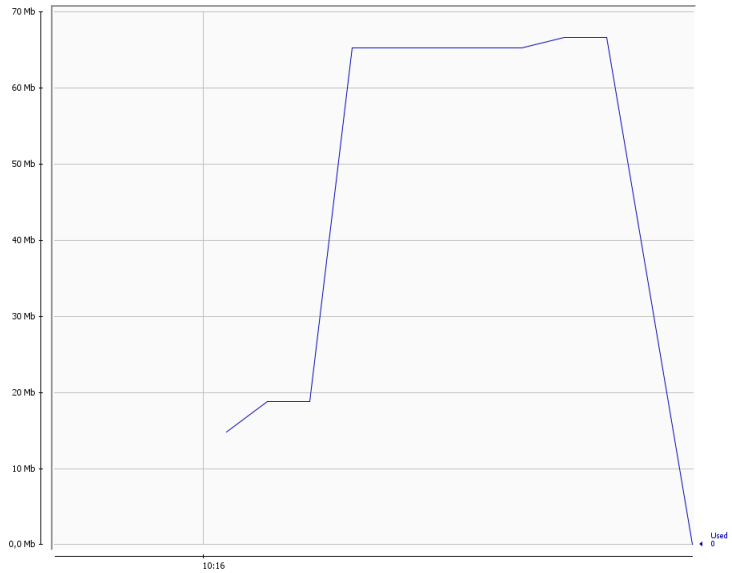

Figure 2.5: Evolution de la mémoire consommée lors de l'utilisation du moteur de recherche avec l'index tokenizer

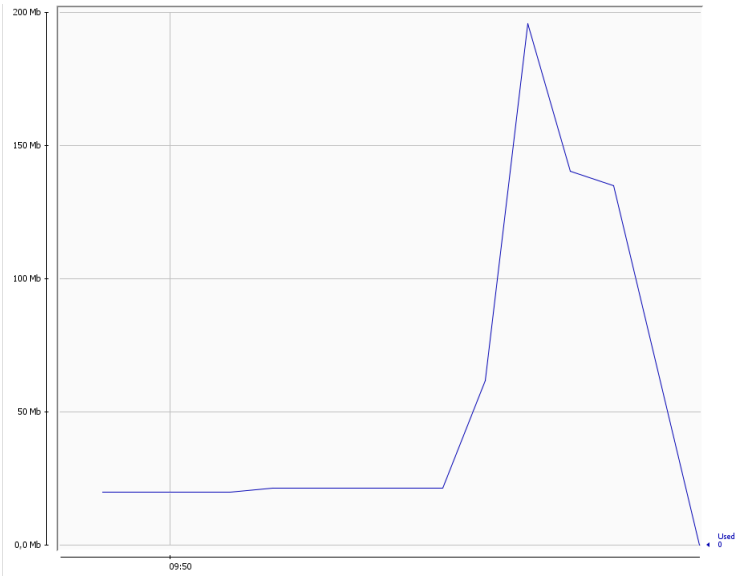

Figure 2.6: Evolution de la mémoire consommée lors de l'indexation du corpus en utilisant le normaliseur stemmer

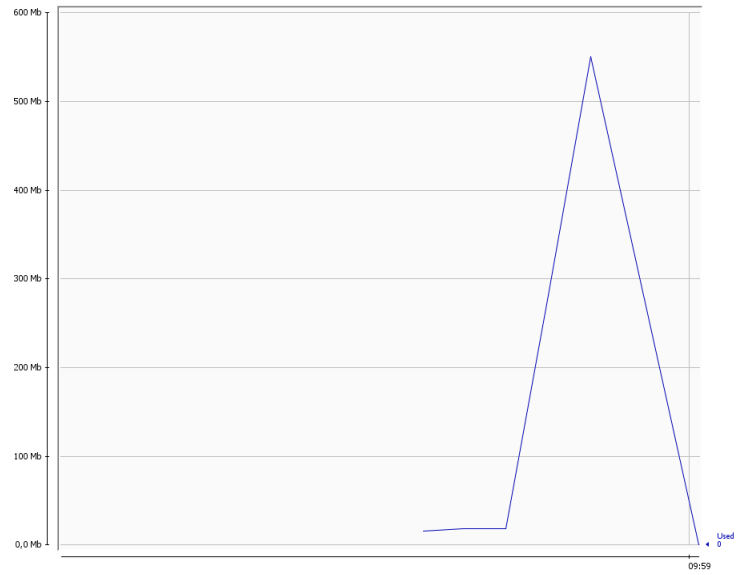

Figure 2.7: Evolution de la mémoire consommée lors de l'indexation du corpus en utilisant le normaliseur tokenizer

On remarque sur les figures si haut qu'utiliser une normalisation de type tokenizer augmente la mémoire consommée(comparée à la normalisation de type stemming), aussi bien par le programme d'indexation que par le moteur de recherche. En effet, avec une normalisation basique(juste séparer les mots et les

mettre en minuscule) on doit stocker beaucoup plus de mots que si l'on considère que la base du mot.

D'autres part, on atteint jamais les 1 GO de RAM. La contrainte du projet est donc respectée. Pour pouvoir respecter cette contrainte, le code a dû être réécrit à maintes reprises en faisant attention à ne pas stocker deux fois une même information, et de ne pas stocker l'information inutilement. Par exemple, lors du calcul des tfidf, on a décidé de ne pas les stocker en mémoire mais des les écrire directement dans l'index.

Temps d'exécution

Indexation identifiants fichiers	Indexation Corpus Steemer	Indexation Corpus Tokenizer
1840 ms	10612 ms	11037 ms

Figure 2.8: Temps d'indexation en ms

Moteur de recherche Steemer	Moteur de recherche Tokcenizer
270 ms	290 ms

Figure 2.9: Temps de recherche en ms

Tailles de l'index sur le disque dur

Corpus	Fichier d'indexation des identifiants	Index stemmer	Index tokenizer
31967 ko	795Ko	14 365Ko	16 595Ko

a

Figure 2.10: Taille des différents fichiers d'indexation(comparaison avec la taille du corpus)

On remarque que les deux index ne dépassent pas 60% de la taille du corpus. La contrainte est donc respectée.

Chapter 3

Topic Modeling

3.1 Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is a generative statistical text model[Blei et al., 2003], that allows sets of observable variables to be explained by Latent(unobservable) variables, where in the texts, the words are the observables variables and topics are the latent variables. This algorithm aims to discovers the topics of the corpus or trying to find the related other corpus with this corpus.

3.1.1 Pre-processing Algorithm

LDA's performance depends on the vocabulary, so it is good to do parsing for all documents of the corpus and remove all words which cannot explore topic and can be found many times such as (and , or ,.... etc) where this preprocessing will increase the performance of LDA.

3.1.2 Dirichlet distribution

It is a continuous multivariate probability distributions ,generalization for Beta distribution[Kotz et al., 2005] but for multivariate . This distribution mainly used as prior distribution of categorical variable such as text model. Its density function follows the next density function.

$$f(x_1, \cdots, x_K; \alpha_1, \cdots, \alpha_K) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{3.1}$$

X1,.... Xk Xi $\in$ (1,N) $\sum x_i = 1$ i =1 , α is called concentration parameters where $\alpha i > 0$ and β (α) is calculated using Gamma function as in the equation :

$$\mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}, \qquad \boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_K). \tag{3.2}$$

$$x_i = \frac{\alpha_i - 1}{\alpha_0 - K}, \qquad \alpha_i > 1 \tag{3.3}$$

X=(x1,x2,x3,..Xk) Dir (α)

3.1.3 Graph plate notation Model

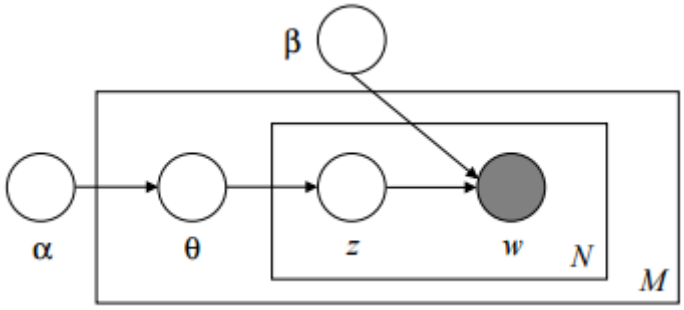

Figure 3.1: Graph plate notation Model

The figure 3.1 is a graphical model that illustrates LDA, where it has some assumption as in the following: M is the number of the documents.
α is the parameter of the Dirichlet prior on the per-document topic distributions.
β is the parameter of the Dirichlet prior on the per-topic word distribution.
these parameters have to be estimated to maximize the (marginal) log likelihood of the data using EM(Estimation maximization) algorithm. In a symmetric Dirichlet distribution, high alpha means that all your documents contain most topics (as opposed to documents containing a few or just a single topic). High beta means that all topics contain most of the words in the corpus. LDA works towards optimizing topic assignments based on these parameters through the iterative process.
θi is the topic distribution for document i
k: is the dimensionality of the Dirichlet distribution is assumed to be known and fixed.
zij is the topic for the jth word in document i, and
wij is the specific word.

3.1.4 Algorithm Implementation

This algorithm consists of the following five steps:

Step 1: Estimate (α, β) estimate their values as discussed in the previous section.

Step 2: define the number of the topics , it can be given by an informed estimate (e.g. results from a previous analysis), or calculated through the trial-and-error. In estimating or trying different values of the number of the topics , the number of the topics can be selected either when the level of interpretability is reached , or the one yielding the highest statistical certainty (i.e. log likelihood).

Step 3: Intial assignment of the topic for each word Assign in initial topic for each word to a initial topic based on a prior Dirichlet distribution (α, β). This can lead to if a word appears more than one time , each instance of this word may be assigned to different topic.

Step 4: Iterative step where for each word has to be checked to update its topic assignment.Note that the topic assignment can be changed based on:

- Finding the probability of occurrence each topic in the current document of the current word.
- Finding the probability of occurrence each topic for this word in the current document and the others documents.
- Weighing the values or multiplying these probabilities of each topic and the maximum multiplication is the new class for this word.

Step 5: Check on convergence of the words topics check on if topics per words' are not converged then go to step 49 , otherwise go to step 6

Step 6: calculate the distribution matrix. It is (V X K) matrix where V is the number of the vocabularies of this corpus and K is the number of the topics for this corpus. Each cell wij represents the number of assigning topic j for word i divided by (V x K) where the summation of this matrix cells has to give 1. Summation of each row is P(occurrence word i in the corpus) . and summation of each column is P(occurrence topic j in the corpus).

Finally from this matrix , we can built this graph which represents the probability of occurrence each topic in the corpus. and it is found that for example Red topic has the highest probability and this can help to determine the main topic about the corpus.

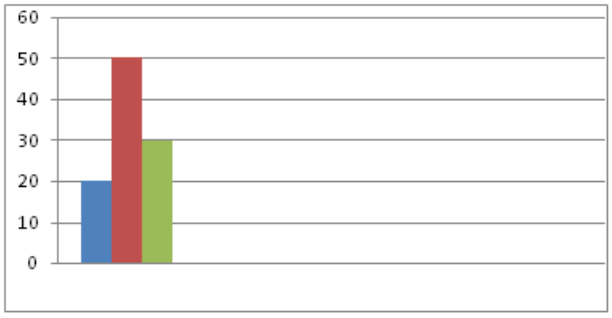

Figure 3.2: Visualisation of distribution matrix over the corpus

3.2 Présentation et Préparation des données pour le LDA

3.2.1 Présentation des données

Pour mettre en œuvre ce projet, nous disposons d'un ensemble d'articles issus de la collection WEB-NEWS composé de 1,55 million de documents de la presse francophone. Ces articles sont parus entre Mars 2013 et Septembre 2015. Étant donné que le corpus est trop volumineux pour se charger en mémoire; vue les limitations matérielles et le manque de puissance calculatoire de nos machines, nous prenons un sous-ensemble de cette collection. Nous prenons donc aléatoirement pour chaque année 5000 documents. Nous sélectionnons ces documents de façon aléatoire afin d'être certains de diversifier les thèmes au fil des mois et des jours. Nous n'avons pas hésité à tronquer le corpus pour deux raisons principales: La première est qu'il aurait été impossible de faire les tests nécessaires en utilisant un aussi grand volume de données et en terminant le projet à temps. La seconde raison est la découverte d'une fonction "update" dont est doté l'algorithme LDA du package Gensim que nous avons utilisé et qui permet d'enrichir le LDA avec plus de données quand c'est nécessaire.

3.2.2 Pré-traitement des données

Dans toute application mettant en jeux de grandes quantités de données textuelles, il est primordial de passer par une étape de préparation et de nettoyage des données afin de les adapter au mieux à l'usage qu'on veut en faire. La complexité et la nature de ces pré-traitements dépend fortement de la langue utilisée. Étant donné que la langue que nous utilisons est le français, voici la liste de pré-traitements que nous proposons pour le Topic-Modeling:

- Détection de la langue pour exclure les articles écrits dans une autre langue que le français (Librairie: "langdetect")
- Lemmatisation du texte, pour ne garder que la forme non fléchi des mots. (Librairie: pattern)
- Stemming du texte pour ne garder que la racine des mots. (Librairie: snowball)
- Uniformisation de la casse pour les caractères; consiste à transformer tous les caractères majuscules en caractères minuscules.
- Suppression de la ponctuation, des nombres et des caractères spéciaux.
- Suppression des mots vides (stopwords) récupérés depuis le lien: `http://members.unine.ch/jacques.savoy/clef/index.html`

- Suppression des mots très peu fréquents (dont l'occurrence dans un document est inférieure à 2).
- Suppression des documents vides et des documents trop petits (moins de 5 mots par article).

À noter que la lemmatisation et le stemming font objet d'une utilisation exclusive entre eux. C'est-à-dire que quand nous utilisons la lemmatisation nous n'utilisons pas de stemming et vise versa. Le stemming n'a au final fait dans notre implémentation que l'objet d'un test. En effet, nous l'avons vite abandonné pour la lemmatisation car il s'avère moins pratique que cette dernière surtout pour la visualisation.

Les traitements tels que la lemmatisation, le stemming, l'uniformisation de la casse sont très importants et visent à réduire le nombre de formes surfaciques pour les mots diminuant ainsi la sparsité des données. Les autres traitements visent surtout à éliminer ce qui est inutile et peut porter préjudice aux résultats de l'algorithme en faussant les calculs.

Il est important de noter également qu'ayant utilisé la librairie "Pattern" pour la lemmatisation du français. Le programme produit d'abord les POS-TAGS qui sont utilisés pour produire les lemmes, ceci conduit alors à l'augmentation du temps de calcul et de l'espace mémoire utilisé par rapport à l'utilisation d'un simple Stemming. Mais étant donné que l'exactitude des outputs de la librairie n'est pas de 100% mais plutôt une exactitude de par exemple 92% pour la singularisation des mots au pluriel, 95% pour la forme non fléchi d'un adjectif...etc (d'après les données présentes sur leur site internet[1]). Ceci correspond à avoir une à deux erreurs par phrase et affecte ainsi d'une façon non négligeable les résultats du LDA. Soit par exemple la phrase: "nous n'avons pas de contraintes, nous pouvons marcher ici car nous sommes libres" le lemmatiseur de la librairie pattern retourne pour cette même phrases les lemmes suivants: "nou ne avoir pas de contraint , nou pouvoir marcher ici car nou sommer libre". Même si certains sont correctes, il est évident que d'autres posent un énorme problème tel que le pronom "Nous" qui démuni de son "S" ne pourra même plus être filtré par le traitement des mots vides, et étant assez redondant faussera le contenu des topics. Cependant inverser les deux traitements ne serait pas non plus une sage décision étant donné que la librairie "Pattern" pour lemmatiser a besoin de produire les POS-TAGS et par conséquent la suppression des mots vides pourrait considérablement affecté les résultats du POS-TAGGER et de façon transitive ceux du lemmatiseur.

La suppression des mots peu fréquents est faite par rapport au document et non au corpus entier en nous basant sur l'hypothèse que si un mot n'apparaît que très peu de fois dans un document c'est qu'il est peu probable qu'il fasse partie de l'un de ces principaux topics.

[1] http://www.clips.ua.ac.be/pages/pattern-fr

Le filtrage des autres données qui peuvent être inutiles telles que les adresses mail, les liens d'un site internet...etc est assuré par le fait que une fois les caractères spéciaux supprimés les différents mots constituant la donnée se retrouvent concaténés et forment un seul mot dont le nombre d'occurrences dans le document devient donc très petit et est alors supprimé par le traitement qui permet de supprimer les mots peu redondants d'un document.

Pour la suppression des documents en anglais, nous avons utilisé une fonction de la librairie "langdetect" issue de l'api de Google pour la détection des langues. Cette fonction retourne pour un texte donné son langage le plus probable et le plus dominant, ainsi en supprimant les documents qui ne sont pas indiqués comme étant "français" par la fonction, nous ne supprimons pas chaque document contenant des mots en anglais sans distinction (1 ou plusieurs) mais uniquement ceux qui contiennent majoritairement des termes anglais.

Ayant programmé notre solution en Python 2.7, qui par défaut utilise l'encodage ASCII. Nous avons pallié aux problèmes d'encodage en ayant tout simplement recours aux fonctions python permettant d'indiquer que l'encodage de telle ou telle donnée doit être en "UTF-8".

D'autres erreurs sur le corpus traité sont à relever mais inhérentes au corpus de départ qui contenait des erreurs de nettoyages: des articles vides, des adresses de sites web écrites sous format "httpslashslah..." au lieu de "http://"...etc.

Le texte pré-traité produit à l'issu de toutes ces étapes est donné en entrée à l'algorithme LDA qui va en extraire les différents topics pour chaque document. Nous avons choisi d'utiliser l'implémentation de LDA présente dans la librairie Gensim[2]. Gensim est une librairie open source de Python spécialement destinée au traitement de grandes quantités de texte. Elle fournit les outils nécessaires principalement au topic modeling et modèle vectoriel. Son code est écrit en utilisant Numpy et scipy ce qui garantit une certaine performance. Des exemples de topics obtenus sont présentés dans la section 3.4 de ce rapport portant sur l'évaluation du topic modeling.

3.3 Visualisation

Le Topic Modeling nous fournis un nouveau moyen d'explorer un corpus, en effet, au lieu de nous intéresser aux mots et leur fréquences, nous nous intéressons à ce corpus en terme de «topic». Nous avons vu dans la section précédente comment les algorithmes du Topic Modeling extraient des structure thématiques dans le corpus et représente chaque documents comme un ensemble de thèmes. Seulement, le LDA est un outils statistique, qui nous retourne des distribution qu'il faudra, à leur tour, explorer pour en comprendre le résultats, d'où l'utilité de la visualisation.

[2]https://radimrehurek.com/gensim/

En visualisant les résultats du LDA, on cherche en premier lieu à comprendre le sens ou interprétation de de chaque topic (1), quelle est la fréquences d'un topic dans le corpus et comment il évolue dans le temps (2). Pour comprendre un modèle en entier, il ne suffit pas de se focaliser sur un topic mais plutôt l'ensemble des topics, cela pour découvrir comment les topics sont liés entre eux (3), mais aussi comment les documents le sont (4) [Sievert and Shirley, 2014].

Pour répondre à ces questions, de nombreuses techniques nous permettent de visualiser les résultats du LDA, certaines de ces techniques ne sont pas dédiées aux Topic Modeling, mais elles sont plutôt dans une catégorie plus générale pour de techniques de visualisation des données, d'autres par contre le sont. Dans ce qui suit, nous allons aborder ces deux types de visualisation pour explorer nos topics et juger de leurs cohérences.

3.3.1 Tendance chronologique

Notre corpus étant divisé en un ensemble de documents répartis en années, mois et jours, nous cherchons dans cette partie à voir comment chacun de nos topics évoluent dans le temps i.e. sont-il plus ou moins fréquents dans les documents. Pour ce faire et répondre à la question, nous utilisons la distribution des documents-topics. À des fins comparatives, nous avons choisi de procéder de deux manières:

- On considère que chaque document est représenté par un topic, dont la profitabilité est la plus élevée et on incrémente l'apparition de ce topic pour la période correspondante au document.
- Nous fixons un seuil, pour chaque probabilité «document-topic» supérieur à ce seuil, on considère que le topic représente le document et on incrémente son apparition. Dans ce cas, un document peu être représenté par plusieurs topics.
- Nous visualisons l'évolution de ces tendances soit par années ou par mois.

Pour cette visualisation nous utilisons les bibliothèques Matplotlib et Numpy de Python.

3.3.2 Interprétation de topics : Word Cloud

Un autre moyen d'interpréter un topic serait de l'explorer à travers les mots qui le composent ainsi que leurs probabilités. Pour cela nous créons un Word Cloud qui est une représentation graphique des fréquences de mot, que nous remplaçons par les probabilités associées à chaque top N mot représentant un topic. Pour l'implémentation du word cloud, nous avons utilisé la bibliothèque WordCloud

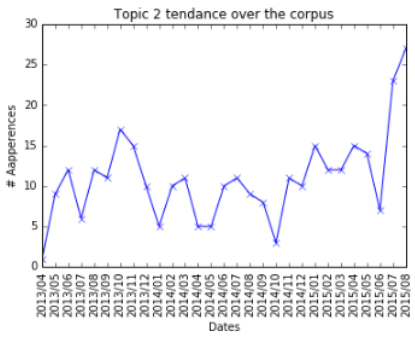

Figure 3.3: Tendance chronologique du Topic 2 par mois

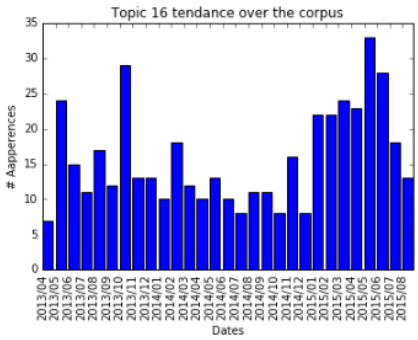

Figure 3.4: Tendance chronologique du Topic 16 par mois

de Python. Pour cette visualisation nous utilisons les bibliothèque Matplotlib et Numpy de Python.

Figure 3.5: Word Cloud

3.3.3 Topics Share

Topic shares est un autre moyen d'explorer nos topics ou plutôt de voir comment ils sont répartis sur le corpus. Le principe est de considérer chabraque document comme étant une barre qu'on segmente en utilisant la matrice de distributions documents-topic que retourne le LDA. Du coup; chaque segment pressente la proportion du topic dans le document en question. Le choix des documents pour cette visualisation se fait soit de manière aléatoire ou ciblée. Pour le premier cas, l'utilisateur n'a aucune maîtrise sur la comparaison obtenue à la fin, le deuxième nécessite une connaissance préalable des documents contenus dans le corpus. Les topics shares sont implémentés soit sous forme de stacked bar ou de heatmaps en utilisant la bibliothèque matplotlib de python.

Les méthodes de visualisation cité ci-dessus, sont plus ou moins adaptées pour explorer un topic en particulier. Bien qu'il soit possible de les exploiter pour visualiser ou comparer plusieurs topics/ documents à la fois, la visualisation aura des limites. Par exemple, bien que la méthode du Topics Share peut être informative, elle devient inefficace lorsque le nombre de documents ou de topics augmente. Il existe des méthodes de visualisation plus appropriées et plus efficaces pour explorer les topics individuellement et l'ensemble du modèle à la fois, nous les citons ci-dessous.

3.3.4 Termite Visualization

Termite visualization est un outil permettant de visualiser les résultats de LDA et d'avaluer les topics obtenus, en considerant un topic à la fois ou l'ensemble des topics.
Dans ce systeme, deux mesures sont utilisés pour calculer l'utilité d'un terme pour interpreter un topic: Distinctivity (distinctivité) et Saliency (importance).
Ces deux mesures capturent à quel point un terme w est informatif dans un topic en calculant la distinctivité comme étant la divergence de KullbackLiebler entre la distribution d'un topic sachant le terme $P(T|w)$ et la distribution marginale $P(T)$.

$$distincivness(w) = \sum_{T} P(T|w) * log\frac{P(T|w)}{P(T)}$$

La saliency du terme est obtenue en ponderant la distinctivité par la fréquence globale du terme w.

$$saliency(w) = P(w) * distincivness(w)$$

La saliency est tilisée comme un seuil pour determiner les termes inclus dans la visualisation (entre 10 et 250 termes affichés).
Le deuxième principe utilisé dans Termite est une méthode de sériation ou ordonnancement des mots informatifs pour capturer les differences entre les topics.
La combinaison de ces deux principes permet à Termite de fournir une classification cohérente des topics, detecter les "junk" topics et identifier les topics qui se chevauchent d'où une évaluation et comparaison plus rapide entre les sujets [Chuang et al., 2012a].
Inconvénient que présente la visualisation avec Termite est le fait qu'elle inclue uniquement les termes ayant une grande Saliency, donc elle tend à fournir une vue globale du modèle. Même si les topics sont interprétés chacun indépendamment des autres, cette interprétation doit se faire seulement avec les termes affichés. Du coup, un utilisateur n'a pas la possibilité d'explorer en profondeur chaque topic individuellement en visualisant un ensemble potentiellement différent de termes

pour chaque sujet [Sievert and Shirley, 2014].Pour y remédier et fournir une visualisation plus flexible à l'utilisateur, LDAvis propose une nouvelle métrique paramétrable que nous pressentons ci-dessous.

3.3.5 PyLDAvis

LDAvis est un systeme de visualization interactif des topics obtenus avec LDA. Il permet une inspection approfondie des relations topic-term du LDA tout en fournissant un aperçu globale des topics à travers leurs pertinance et similitudes les un par rapport aux autres dans un espace compact.
D'abord, LDAvis introduit une nouvelle metrique: la Relevence (pertinence) qui permet de determiner les termes les plus pertinants dans l'interpretation d'un topic et les ordonner selon cette pertinence.
On défini la pertinance d'un terme w à un topic k par la formule suivante:

$$r(w, k|\lambda) = \lambda log(\psi kw) + (1 - \lambda) log(\frac{\psi kw}{Pw})$$

Avec:

- ψkw : la probabilité d'un terme pour un topic.
- pw : la probabilite marginale d'une trme w.
- Le parametre λ est compris entre 0 et 1. Il determine le poid attribué à un terme w pour un sujet k. E mettant $\lambda = 1$ nous retrouvons les termes dans l'ordre qui leur a été attribué selon leur probabilité thématique. Par contre avec le parametre $\lambda = 0$ l'ordre des termes selon leur nouveau poids qu'on peut considéré comme poid soulevé.

Cette formule favorise les termes qui sont propre à chaque topic, i.e. les termes qui apparaissent avec une grande fréquence dans un topic mais avec de petite fréquence dans le reste des topics du modèle.
La fléxibilité de LDAvis se reflète au fait que le parametre λ peut être réajuster par l'utilisateur, donc l'utilisateur peut décider de quel poids affecté aux termes.
Ce que propose LDAvis dans la visualization des topics se résume dans les quatre points suivants:

- Une meilleure interprétation des topics en affichant ses mots les plus pertinants grace à la mesure de "Relevence".
- Visualiser la pertinence des topics: dans LDAvis, les topics sont représentés par des cercles, la taille de chacun d'eux reflète la pertinence du topic en question dans tout le corpus.
- Différence inter-topic : cette différence est calculé en utilisant une distance de divergence de Jensen-Shanon.

- Il est également possible de manipuler les termes considérés comme pertinents pour visualiser leur distribution conditionnelle dans tous les topics. Cette distribution est visualisée en modifiant les zones des cercles thématiques (cercle représentant le topic) de façon à ce qu'elles soient proportionnelles aux fréquences spécifiques au terme sélectionné dans le corpus.

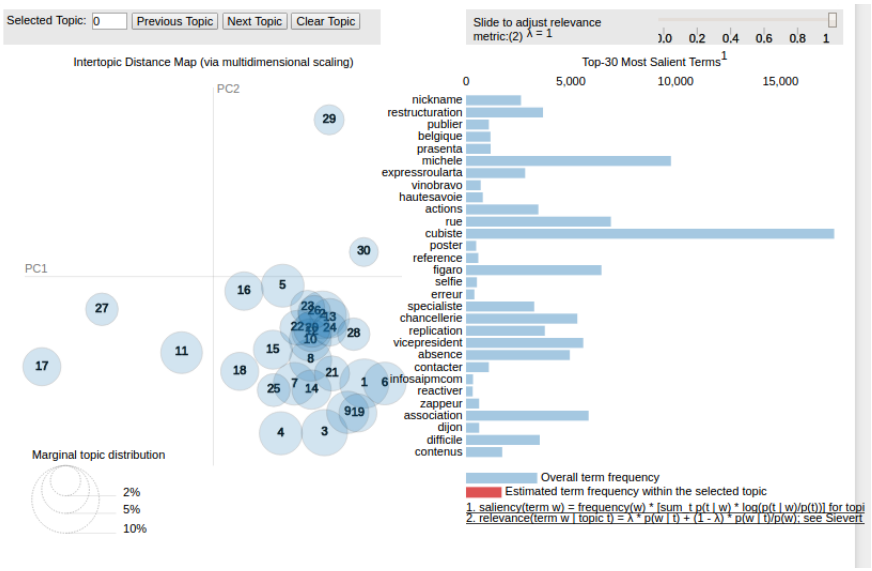

Figure 3.6: Résultats avec LDAvis

Pour résumé, les différentes méthodes de visualisation présentées dans cette partie permettent de répondre aux questions que nous nous somme initialement posés:

- (1) L'interprétation des topic peut se faire en utilisant les WordCloud, Termite et LDAvis.
- (2) L'évolution des topics peut être visualisé avec les graphes des tendances, il est à noté que cela est uniquement possible dans notre cas puisque nos article sont organisés dans dossiers de la forme "corpus/année/mois/jour/$document_i$".
- (3) La perception des liens entre les topics peut être visualisé avec LDAvis et Termite.
- (4) Enfin, le lien thématique entre les documents peut être visualisé grâce à Tpics Share.

3.4 Évaluation du topic modeling

Dans les sections précédentes, nous avons vu que le LDA est une technique populaire qui permet de découvrir les topics cachés dans un corpus de documents.

Le Topic Modeling apprend des topics, représentées par un ensemble de mots, de manière non supervisée sur un ensemble de documents non étiquetées. Cependant, même si les topics appris par le LDA peuvent être utiles (par exemple pour la recherche thématique), leur interprétation (comme étant un ensemble de mots) peut être difficile car les topics n'ont pas de thèmes particuliers. Un des problèmes engendré par LDA est la présence de topics incohérents appelées "junk topics" dont les mots n'ont pas de signification visible quand ils sont mis ensemble. Les résultats doivent souvent être analysées par des experts humains afin de determiner si le topic est valide ou bien est considéré comme "junk topic". Beaucoup de travaux ont essayé de simuler le jugement humain afin d'évaluer le topic modeling en se basant sur des mesures probabilistes. Dans cette partie nous allons expliquer et mettre en œuvre quelques techniques d'évaluation des topics. Nous commencerons par présenter des mesures de cohérence des topics se basant sur l'information mutuelle et la co-occurrences des mots. Les mots d'un topic étant classés par ordre de fréquence, nous ensuite expliqué comment changer la structure d'un topic pour faire apparaître les mots qui distinguent ce topic en premier. Enfin, nous proposerons deux méthodes pour labelliser les topics, une qui prend le mot le plus distinguant, et l'autre dont l'information mutuelle est la plus importante. Nous conclurons par des comparaisons.

3.4.1 Cohérence des topics

La cohérence d'un topic exprime le degré de similarité sémantique entre les mots de ce topic. Un topic cohérent exprime généralement une même idée, un même thème, auquel la majorité des mots du topics réfèrent. La mesure de cohérence est un score donné à un topic qui permet de classer les topics selon leur cohérence. Cette mesure est souvent calculée en sommant la similarité distributionnelle des paires de mots dans un topic:

$$Coherence = \sum_{(w_i,w_j)\in V} score(w_i, w_j, \epsilon)$$

où V est l'ensemble des mots du topic et ϵ un facteur de lissage qui garantie que le score soit un nombre réel [Stevens et al., 2012].

L'intuition derrière ce modèle basé sur la similarité des mots deux à deux vient du fait que la cohérence d'un ensemble de mots implique que tous les mots de l'ensemble soient liées [Pleple, 2013].

Les mesures que nous allons utilisé reposent sur l'information mutuelle ponctuelle.

Information mutuelle

Dans la théorie de l'information, l'information mutuelle de deux variables aléatoires X et Y est une quantité qui mesure la dépendance statistique de ces deux

variables. Elle mesure la quantité d'information apportée en moyenne par un terme X sur un terme Y [Schütze, 2008].

Pointwise Mutual Information (PMI) L'information mutuelle ponctuelles mesure l'association entre deux réalisations de variables X et Y (notés x et y), contrairement à l'information mutuelle qui mesure une moyenne sur toutes les réalisations. L'information mutuelle ponctuelle entre deux variables x et y est calculée par:

$$pmi(x, y) = log \frac{P(x, y) + \epsilon}{P(x).P(y)}$$

Où $P(x, y)$ est la probabilité jointe des deux variables et $P(x), P(y)$ les probabilités marginales.

Normalized Pointwise Mutual Information (NPMI) La PMI peut être normalisée , ce qui donne la formule suivante [Bouma, 2009]:

$$npmi(x, y) = log \frac{pmi(x, y)}{-log(P(x, y))}$$

Mesure UCI

La mesure UCI a été introduite dans [Newman et al., 2010]. Elle se base sur la PMI entre deux mots du topic, où les probabilités conjointes sont calculées avec une fenêtre glissante de taille 10 (deux mots sont co-occurrents s'ils sont au plus 10 termes loin l'un de l'autre). Une moyenne arithmétique est effectuée

$$UCI = \frac{2}{N.(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N-1} PMI(w_i, w_j)$$

Les probabilités utilisées pour le calcul de la PMI sont estimées en comptant les occurrences des mots dans l'ensemble des documents du corpus et en divisant par la taille du corpus. L'auteur propose d'utiliser un corpus externe pour calculer la PMI afin de ne pas renforcer le bruit présent dans le corpus d'apprentissage. Un exemple de corpus utilisé est les articles de Wikipédia.

Mesure UNPMI

La mesure UNPMI introduite par [Aletras and Stevenson, 2013] est une version améliorée de la mesure UCI où l'information mutuelle ponctuelle normalisée (NPMI) est utilisée au lieu de l'information mutuelle ponctuelle (PMI)

$$UNPMI = \frac{2}{N.(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N-1} NPMI(w_i, w_j)$$

Mesure Umass

La mesure Umass est une mesure de cohérence asymétrique proposée par [Mimno et al., 2011] basée sur les probabilités conditionnelles. La formule est exprimée comme suit:

$$Umass = \frac{2}{N.(N-1)} \sum_{i=2}^{N-1} \sum_{j=1}^{i-1} log \frac{P(w_i, w_j) + \epsilon}{P(w_j)}$$

Les mots étant ordonnées par ordre de fréquence, l'idée ici est que l'occurrence d'un mot doit être confirmée par tout mot du même topic qui le précède dans le classement des fréquences. Autrement dit, la probabilité qun document contienne le mot d'ordre t doit être plus grand si le document contient déjà les mots d'ordre $t-i$ du même topic. Pour cela, le logarithme de la probabilité conditionnelle est calculée pour chaque mot précédent le mot t. La Umass mesure à quel point un mot fréquent déclenche un mot plus rare dans le topic.

3.4.2 Pertinence des mots dans un topic

Le résultat du LDA est une liste de topics où dans chaque topic les mots sont ordonnées par fréquence dans le topic. Souvent, une telle représentation n'est pas très descriptive et distinctive d'un topic. Un mot qui est fréquent dans plusieurs topics ne donnera aucune information qui puisse nous aider à interpréter le topic. Nous cherchons à présent une nouvelle représentation d'un topic où ce ne sont plus les mots les plus fréquents qui représentent le topic mais les mots les plus pertinents.

[Chuang et al., 2012b] proposent une méthode pour trouver les mots les plus informatifs du corpus en calculent l'importance d'un mot dans tout le corpus. Ce score n'est pas intéressant dans la mesure où il trouve les mots pertinents dans tout le corpus, et non pas pour un topic spécifique, ce qui peut tronquer les résultats et marginaliser des mots importants pour certains topics.

[Pleple, 2013] propose de calculer la pertinence des mots par rapport à un topic. On considère la probabilité conditionnelle du mot w sachant le topic k: $P(w/k)$ divisé par l'exponentiel de l'entropie du mot w

$$Relevance(w,k) = \frac{P(w/k)}{e^{H_w}}$$

$$H_w = \sum_k P(k/w) log P(k/w)$$

L'entropie d'un mot w peut être vue comme la dispersion de la distribution du mot w Nous estimons la probabilité $P(k/w)$ en appliquant la formule de Bayes où nous obtenons:

$$P(k/w) = p(w/k)p(k) = p(w/k)\sum_{d} p(k/d)p(d) = p(w/k)\sum_{d} p(k/d)N_d$$

Où N_d est la taille du document d, $p(k/d)$ est la probabilité du topic k dans le documents d et $p(w/k)$ est la fréquence du mot w dans le topic k

3.4.3 Labellisation des topics

Les résultats du topic modeling sont des ensembles de mots représentant des topics. Cependant le LDA ne donne pas d'étiquettes aux topics générés. Cette tâche est souvent laissé à une expertise humaine. Plusieurs travaux ont proposé des méthodes pour labelliser les topics de façon automatique. Dans ce qui suit, et en se basant sur ce qui a été utilisé précédemment nous proposons deux méthodes de labellisation:

Mot le plus pertinent : Après avoir transformé la représentation d'un topic des mots les plus fréquents aux mots les plus pertinents, nous proposons de prendre le premier mot de la liste en tant que label.

PMI la plus élévée : L'idée ici est de simuler le jugement humain qui dit que le mot le plus représentative d'un ensemble de mots est celui qui est lié à tous les autres mots. Ceci revient donc à calculer la PMI. Ici la PMI sera calculée pour chaque mot avec tous les autres mots du topic et une moyenne est faite. Le mot avec la plus grande moyenne sera choisi comme label.

3.4.4 Tests et résultats

Plan de test

Nous effectuons notre évaluation du topic modeling de la manière suivante. D'abord nous testons les mesures de cohérences sur les topics définis par les N mots les plus fréquents, puis nous testons ces mêmes mesures sur les topics définis par les N mots les plus pertinents afin de comparer les deux représentations. Nous donnons les résultats de labellisation pour quelques topics puis en s'aidant des mesures de cohérences nous choisissons un nombre de topics adéquats pour exécuter LDA.

Cohérence des topics définis par fréquence

Nous calculons la cohérence UNPMI et UMASS sur les résultats du LDA. Les tableaux suivants montrent les résultats où chaque topic est représenté par 10 mots. Dans chaque tableau nous présentons les deux premiers topics les plus cohérents et les deux derniers moins cohérents

-0.93	a commentaire france faire pouvoir no devoir an soumettre politique
-1.044	commentaire faire soumettre abu an signaler vou france nou desole
-1.49	pouvoir faire an nou homme devoir pays ministre contenu vouloir'
-1.94	pouvoir france faire nouvel president million monde francais observateur louison

Table 3.1: Cohérence UNPMI pour les deux topics les plus cohérents et les deux topics les moins cohérents

1.066	commentaire faire soumettre abu an signaler vou france nou desole
0.46	a, commentaire, france, faire, pouvoir, no, devoir, an, soumettre, politique
- 0.35	faire pouvoir an devoir dernier euro grand jour france million
-0.40	faire an pouvoir devoir francais nou aller ministre match

Table 3.2: Cohérence UMASS pour les deux topics les plus cohérents et les deux topics les moins cohérents

Nous remarquons d'abord que les deux mesures donnent plus ou moins les mêmes résultats. D'autre part, il est évident qil est difficile d'interpréter les résultats, plusieurs mots sont fréquents dans des topics différent ce qui donne l'impression que c'est le même topic.

Cohérence des topics définis par pertinence

Nous réordonnons les mots du topics de façon à ce qils soient représentés par les mots les plus pertinents. Cette fois nous n'utilisons pas la mesure UMASS car elle dépend sur la fréquence des mots.

-0.074	retweets favoris retweet parent favori lepoint faire an pouvoir nou
-1.24	salaire er nucleaire euro faire pouvoir an france million milliard
-2.99	faire pouvoir devoir an annee ministre gouvernement nou a
-3.02	faire pouvoir devoir europeen an ump ministre fn paris americain

Table 3.3: Cohérence UNPMI pour les deux topics les plus cohérents et les deux topics les moins cohérents

Nous remarquons que les topics à présents sont plus parlant, on constate de plus en plus de liens entre les mots et moins de répétition.

Choix du nombre de topics

Le LDA est une technique non supervisées pour la découverte de topics dans un corpus. Compte tenu de ce fait, le nombre de topics générés est un paramètre en entrée. Le choix de ce nombre est une tâche difficile car elle ne peut pas être estimée de manière précise. Nous utilisons les métriques d'évaluation présentées

précédemment pour choisir le nombre de topics qui donne le plus de cohérence et d'interprétation. Nous montrons dans le tableau suivant les topics les plus cohérents en exécutant le LDA trois fois avec 30, 50 et 100 topics

30	- retweets favoris retweet parent favori lepoint faire an pouvoir nou - salaire er nucleaire euro faire pouvoir an france million milliard - louison observateur km pouvoir france faire nouvel president million monde
50	- uber vtc chauffeur taxi transport faire pouvoir samedi service dimanche - retweets favoris retweet favori lepoint http pouvoir faire aller - mandela trimestre afrique faire president pays etat pouvoir politique france
100	- agatha mandela michael cancer sein accident an place faire devoir - argentin asile argentine bresil ata fonds faire a pouvoir devoir - sncf migrant voiture ata train policiers faire points ministrean'

Table 3.4: Interprétation des topics selon le nombre de topics générés par LDA

Nous remarquons que pour ce dataset, un nombre de topics entre 50 et 100 donne les meilleurs cohérences.

Labellisation des topics

Pour labelliser les topics, nous prenons quelques topics des résultats précédents et donnons le label selon les deux méthodes présentés précédemment

Topic	**Label 1 Pertinence**	**Label 2 PMI**
uber vtc chauffeur taxi transport faire pouvoir samedi service dimanche	uber	uber
retweets favoris retweet favori lepoint http pouvoir faire aller	retweets	retweets
mandela trimestre afrique faire president pays etat pouvoir politique france	mandela	mandela

Chapter 4

Conclusion

Pour conclure, on peut dire que nous avons atteint nos objectifs de départ. Le moteur de recherche a été implémenté et respecte les contraintes mais il est sujet à quelques d'améliorations possibles. Il est nécessaire d'implémenter une belle interface graphique pour que le moteur soit réellement utilisable. Aussi, bien que respectant les contraintes du projets, en utilisant d'autres structures de données le programme pourrait être beaucoup plus performant et consommer moins de mémoire RAM. Nous avons aussi pu faire un vrai travail de recherche sur la partie Topic Modeling. Nous avons réussi à faire du Topic Modeling sur notre corpus, trouver différentes façon de visualiser ces topics pour pouvoir en tirer le plus d'informations possibles mais aussi définir différentes métriques d'évaluations pour juger de la pertinences des topics. Il s'avère également que pour le moteur de recherche ainsi que pour le topic modeling l'étape de nettoyage et pré-traitements des données est cruciale pour les résultats. Comme tout travail le notre n'est pas exhaustif et plusieurs améliorations peut y être apportées que nous résumons en ces quelques points:
Tout d'abord, comme il a été expliqué dans les précédentes sections concernant le topic modeling, nous n'avons pas réussi à exécuter notre code sur l'ensemble du corpus, ceci nous permet donc de formuler une première perspective d'amélioration. Une idée qui nous tentait particulièrement est la visualisation des tendances historiques des articles en utilisant un LDA dynamique. En effet, ceci nécessite l'utilisation d'un LDA dit dynamique qui permet d'analyser l'évolution des articles au fil du temps. La performance du LDA actuel repose notamment sur le bon ajustement de ces paramètres (tel que le nombre de topics à générer, le nombre de passages sur le corpus pour training... etc), il serait donc fortement intéressant de se pencher sur l'automatisation de cet ajustement. Par ailleurs, les métriques de cohérence ont été utilisés intrinsèquement, ce qui a pu favorisé le bruit dans les estimations de probabilités. Une amélioration serait d'utiliser un corpus externe (comme Wikipedia) pour estimer les probabilités de d'apparition des mots.

Chapter 5

Répartition des tâches

Le bon déroulement de tout projet de groupe dépend fortement de la bonne répartition des tâches parmi les membres participants. Étant au nombre de 5 nous avons donc décomposé le projet en 5 tâches principales que nous avons réparti ainsi:

- Réalisation du moteur de recherche: Abdelhadi Temmar
- Pré-traitement des données, application du LDA et préparation des input et des output du LDA pour la visualisation et l'évaluation: Sihem Abdoun
- Étude/explication du fonctionnement de l'algorithme LDA: Mohamed Abdelkhalek
- Mise en œuvre de la visualisation des Topics: Fella Belkham
- Étude et Implémentation de méthodes d'évaluation: Nawel Medjkoune

Concernant la rédaction du rapport chacun a participé en rédigeant les détails inhérents à la tâche qui lui a été affectée. Les parties communes ont fait objet d'une participation collective.

References

[Aletras and Stevenson, 2013] Aletras, N. and Stevenson, M. (2013). Evaluating topic coherence using distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)–Long Papers*, pages 13–22.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

[Bouma, 2009] Bouma, G. (2009). Normalized (pointwise) mutual information in collocation extraction. *Proceedings of GSCL*, pages 31–40.

[Chuang et al., 2012a] Chuang, J., Manning, C. D., and Heer, J. (2012a). Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 74–77. ACM.

[Chuang et al., 2012b] Chuang, J., Manning, C. D., and Heer, J. (2012b). Termite: Visualization techniques for assessing textual topic models. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, pages 74–77. ACM.

[Kotz et al., 2005] Kotz, S., Balakrishnan, N., and Johnson, N. L. (2005). *Dirichlet and Inverted Dirichlet Distributions*, pages 485–527. John Wiley and Sons, Inc.

[Mimno et al., 2011] Mimno, D., Wallach, H. M., Talley, E., Leenders, M., and McCallum, A. (2011). Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272. Association for Computational Linguistics.

[Newman et al., 2010] Newman, D., Lau, J. H., Grieser, K., and Baldwin, T. (2010). Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics.

[Pleple, 2013] Pleple, Q. (2013). *Interactive Topic Modeling.* PhD thesis, UNIVERSITY OF CALIFORNIA, SAN DIEGO.

[Schütze, 2008] Schütze, H. (2008). Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference.*

[Sievert and Shirley, 2014] Sievert, C. and Shirley, K. E. (2014). Ldavis: A method for visualizing and interpreting topics. In *Proceedings of the workshop on interactive language learning, visualization, and interfaces*, pages 63–70.

[Stevens et al., 2012] Stevens, K., Kegelmeyer, P., Andrzejewski, D., and Buttler, D. (2012). Exploring topic coherence over many models and many topics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 952–961, Stroudsburg, PA, USA. Association for Computational Linguistics.