
**Traitement Automatique de la Langue**

# Compte rendu des TMEs

*Étudiants :*
Célina HANOUTI
Rym KACI

Mai 2020

# Table des matières

# 1 Détection d'auteur : Chirac/Mitterrand

Dans cette partie, l'objectif est de créer un modèle capable à partir d'extraits de discours de Chirac et de Mitterand, de prédire l'orateur.

## 1.1 Description des données

Les données sont des phrases extraites des discours de Chirac et de Mitterand. Une base de 57 413 extraits labélisés est fournie ainsi que des données de tests : 27162 entrées non labélisées.

Sur la base d'apprentissage les classes ne sont pas équiprobables. Les entrées labélisées Mitterand représentent $13,1\%$ des données contre $86,9\%$ pour les extraits de Chirac.

On remarque également que la base est préalablement parsée, les dates sont toutes remplacées par $<date>$ de même pour les noms de personnes.

## 1.2 Métrique d'évaluation

Les données étant déséquilibrées, une évaluation basée sur le taux de bonne classification n'est pas pertinente. En effet un classifieur à priori (classe majoritaire) sera jugé comme très bon avec un taux de classification très élevé sur les données d'apprentissage. Ce modèle sera évidemment très mauvais sur des données de test différemment distribuées. On optera donc dans cette partie pour une évaluation basée sur le score f1 pour la classe minoritaire. Le score f1 étant une moyenne harmonique de la précision et du rappel.

$$F1 = \frac{2}{prcision^{-1} + rappel^{-1}} = 2.\frac{prcision.rappel}{prcision + rappel}$$

## 1.3 Pré-traitement des données

Une grande partie de la tâche d'apprentissage en TAL repose sur les différents pré-traitements que l'on peut opérer sur le texte. Nous détaillerons dans cette partie les différentes modifications apportées à la base.

### 1.3.1 Représentation des données

Nous optons pour une représentation en sac de mots avec un codage fréquentiel. Autrement dit les colonnes de la base d'apprentissage représentent les mots du vocabulaire, et pour chaque extrait nous stockons le nombre d'occurences de chaque mot.

Le stockage fréquentiel donne les meilleurs résultats. En effet, un stockage en TF-IDF peut sembler peu adéquat étant donné la tâche. Pour chaque discours il serait pertinent pour identifier l'orateur de déterminer les mots les plus fréquents. Le TF-IDF donnerait moins de poids à ces derniers.

### 1.3.2 Caractéristiques des données

Une fois le format de représentation établi, un certain nombre de choix relatifs à la structure des mots est testé :

- **Le nombre d'apparitions minimum d'un mot dans différents documents** afin de juger si un attribut est pertinent pour la classification. Par exemple si un mot apparait dans un seul document de la base d'apprentissage, il sera très discriminant pour prédire un orateur (sur-apprentissage), alors qu'il peut seulement s'agir d'un mot rare. Nous pouvons ici observer le nombre de mots gardés en fonction du nombre d'occurences minimum. Pour éviter de tomber dans du sous-apprentissage nous choisirons de garder un mot s'il apparait dans au moins 2 extraits.

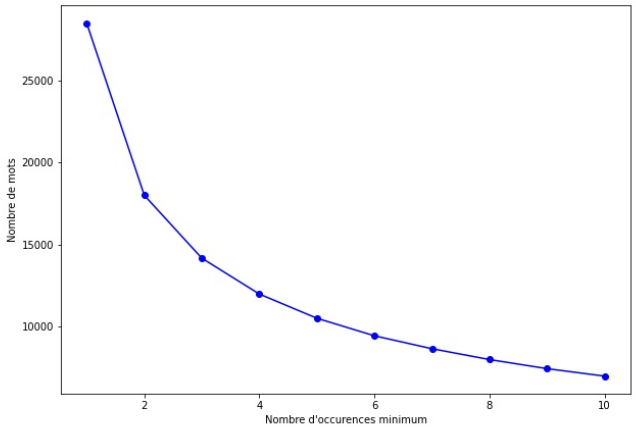


FIGURE 1 – Nombre de mots gardés en fonction du nombre minimum d'apparitions dans la base

- **Stemmatisation/Lemmatisation** : La stemmatisation consiste à ne garder que la racine de chaque mot sans se soucier si le token racine appartient au vocabulaire de la langue. La lemmatisation cependant rapporte les mots à leur forme canonique (en enlevant préfixes et suffixes) le mot résultant est le nom racine. Nous testerons l'impact des deux approches sur nos modèles.
- **Stopwords** : Il est commun en TAL de se débarrasser des mots très fréquents dans les documents pour ne pas brouiller l'apprentissage. Dans ce cas là cependant il peut être intéressant de garder certaines locutions qui peuvent être propres à un orateur.
- **N-Grams** : Il est parfois intéressant d'en plus de s'intéresser aux mots indépendemment (unigrammes), de considérer également les combinaisons de mots deux à deux (bigrammes). Des patterns d'expressions par exemple pour un président peuvent ainsi être détectés.
- Les tokens contenant des chiffres ne sont pas pertinents pour l'apprentissage, nous les ignorerons. Cependant un orateur peut se démarquer car il s'exprime beaucoup avec des chiffres. Afin de ne pas perdre cette information qui pourrait être intéressante, nous récupérons (par une expression régulière) pour chaque extrait le nombre de chiffres dans un attribut *nb_ digits*.
  Ex : « ...plutôt que de penser aux quelques 100000 chômeurs de moins par

-rapport au début janvier <date>, je ne pense pas d'abord aux 2300000 chômeurs de trop ? » $nb\_digits = 13$

- **Attributs additionnels** : En plus du nombre de chiffres par extrait, nous avons également pensé à certains attributs qui pourraient guider la classification. Par exemple la taille moyenne des mots avant stemmatisation, un président peut avoir recours à des mots plus long que l'autre. Pour aller plus loin sur ce point, on réalise un marquage (POS TAG) sur les mots, et on comptabilisera en moyenne le nombre de noms, de verbes, d'adverbes ... Certains attributs seront plus pertinents que d'autres.

## 1.4 Modèles et optimisation de paramètres

Les algorithmes d'apprentissage utilisés dans la suite des expérimentations sont : Naive Bayes Multinomial, SVM avec un kernel linéaire et Régression Logistique. Nous faisons le choix de ces classifieurs pour leur simplicité et leur efficacité sur des données avec un grand nombre d'attributs.

### 1.4.1 Naive Bayes Multinomial

De part sa simplicité, ce classifieur sert souvent de modèle de référence (benchmark) en TAL. Il fait l'hypothèse *naïve* que les mots sont indépendants Pour chaque mot une probabilité à posteriori d'appartenir à une classe est attribué. La prédiction de la classe se fait par la suite par maximum de vraisemblance.

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, P(y). \prod_{i=1}^{n} P(x_i/y)$$

Un des problèmes qui peut se poser sur un classifieur naive-bayes est : si un mot n'est jamais observé dans une classe, la vraissemblance de ce mot sur cette classe sera égale à 0 ($P(mot/y) = 0$. Une telle probabilité annulerait la vraissemblance de tout l'extrait (produit par 0). Pour palier à ce probleme on applique un lissage dit lissage de LaPlace qui consiste à ajouter 1 à toutes les occurences de mot dans les extraits. En pratique la valeur ajoutée est souvent plus petite que 1. On effectue ici une série de tests pour obtenir une valeur de lissage (hyperparamètre $\alpha$) adéquate à notre problème (figure 2). La valeur utilisée pour la suite des expérimentation est 0.2.

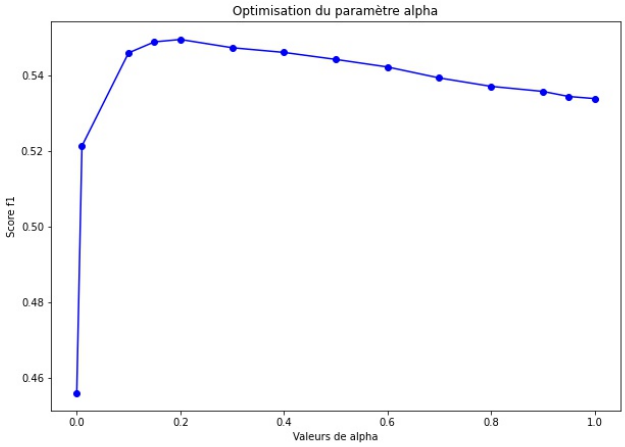


FIGURE 2 – Optimisation de l'hyper-paramètre alpha (lissage de LaPlace

### 1.4.2 SVM Linéaire et Régression Logistique

Sur les algorithmes SVM et régression Logistique l'un des paramètres important à optimiser est l'intensité de la régularisation. La régularisation consiste à pénaliser la complexité du modèle pour éviter de tomber dans du sur-apprentissage. Les courbes de la figure 3 montrent l'influence de l'hyper-paramètre C sur les performances des classifieurs. On retiendra pour la suite des expérimentation une valeur de $C = 0.4$ pour le SVM linéaire et $C = 1$ pour la régression logistique.

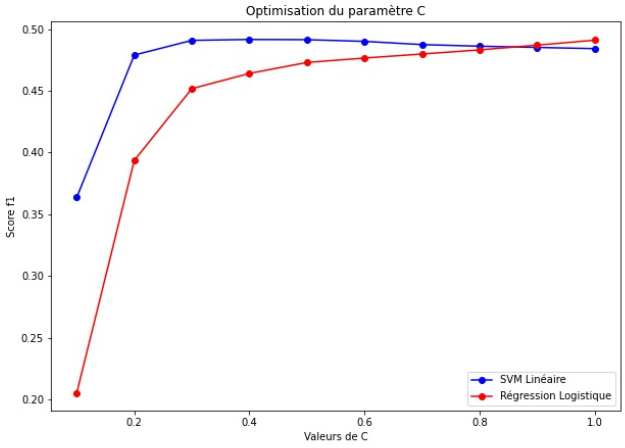


FIGURE 3 – Optimisation de l'hyper-paramètre C

## 1.5 Expérimentations

### 1.5.1 Post-traitement des données

Comme spécifié dans le travail de Romain Ayres [1] on peut remarquer que les données sont des extraits de discours fragmentés. Chaque entrée de la base représente une phrase les labels C ou M sont donc plus susceptibles de se retrouver dans des blocs contigus. En supposant que les données de tests répondent à la même logique, nous appliquant une fonction de lissage sur nos données.
Ex : prédictions avant lissage : C C C M C C M M M C M.
prédictions après lissage : C C C C C C M M M M M.

### 1.5.2 Séléction du modèle

Une fois les hyper-paramètres optimisés et les étapes de pré-traitement et post-traitement établies, nous procédons à une série de tests (en cross validation sur 5 folds) afin de sélectionner notre modèle.

| **Smoothing** | **Stopwords** | **Additional Features** | **NGrams** | **Naive Bayes** | **Logistic Regression** | **SVM** |
|---|---|---|---|---|---|---|
| NO | NO | NO | NO | 53% | 48% | 48% |
| NO | NO | NO | YES | 56% | 52% | 51% |
| NO | NO | YES | NO | 53% | 48% | 48% |
| NO | NO | YES | YES | 56% | 51% | 51% |
| NO | YES | NO | NO | 54% | 49% | 49% |
| NO | YES | NO | YES | 57% | 54% | 53% |
| NO | YES | YES | NO | 54% | 49% | 49% |
| NO | YES | YES | YES | 57% | 54% | 53% |
| YES | NO | NO | NO | 67% | 47% | 55% |
| YES | NO | NO | YES | 73% | 51% | 61% |
| YES | NO | YES | NO | 67% | 47% | 56% |
| YES | NO | YES | YES | 73% | 51% | 61% |
| YES | YES | NO | NO | 69% | 48% | 57% |
| YES | YES | NO | YES | 75% | 55% | 64% |
| YES | YES | YES | NO | 70% | 49% | 58% |
| **YES** | **YES** | **YES** | **YES** | **75%** | **55%** | **59%** |

1. http ://webia.lip6.fr/ guigue/wikihomepage/uploads/Course/Ayres2013.pdf

# 2 Analyse de sentiments : Movie reviews

Dans cette partie, l'objectif est de prédire la polarité d'un ensemble de revues de films en utilisant les données textuelles contenues dans ces dernières. Afin d'extraire des features pertinentes qui serviront à l'apprentissage de nos modèles, un pré-processing des données brutes et une modélisation à l'aide d'un ensemble de méthodes (Bag of Words, modélisation N-Gram, modélisation TF-IDF ..etc) sont indispensables.

Nous considérons ici deux classifieurs uniquement : SVM avec un kernel linéaire et Regression Logistique. Nous optimisons les hyper-paramètres des modèles à l'aide d'une validation croisée de 5 folds.

Les classes étant équilibrées, nous utilisons le taux de bonnes classifications comme indicateur de performances.

## 2.1 Expériences réalisées

Notre compagne d'expérience consiste principalement à analyser l'impact du pré-traitement, de la modélisation des données et des hyper-paramètres sur les performances des classifieurs. La suppression des nombres et de la ponctuation (mis à part le point d'exclamation) a été appliquée systématiquement.

### 2.1.1 Version 1 : Sans pré-traitement, Modélisation Bag of Words et Unigrammes

Dans un premier temps, on considère une version dans laquelle on garde tout le vocabulaire présent dans l'ensemble d'apprentissage, y compris les stop-words. L'hyper-paramètre C,qui représente l'intensité de la régularisation, est un hyper-paramètre important à optimiser, en effet, avoir plus de features que de données causerait un éventuel sur-apprentissage. On observe sur la figure 4 que les performances en validation croisée des deux classifieurs diminuent lorsque C augmente, notons que l'intensité de la régularisation est inversement proportionnelle à la valeur de C. Dans la suite de l'analyse, on considérera à chaque fois une régularisation assez forte. La figure 5 donne les 10 mots les plus discriminants (positivement et négativement) pour chaque classifieur. On peut déduire que les résultats sont relativement cohérents : des avis contenant des mots comme « excellent », « perfectly » et « memorable » seront considérés comme positifs et ceux contenant des mots comme « worst », « awful » et « boring » seront considérés comme négatifs.

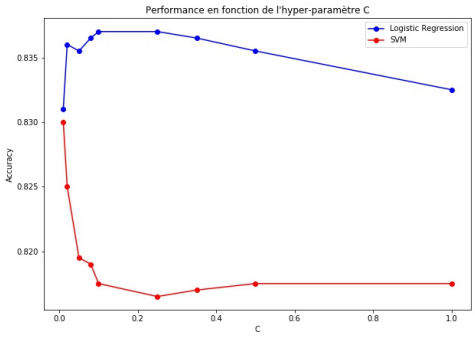


FIGURE 4 – Performances en fonction de l'hyper-paramètre de régularisation

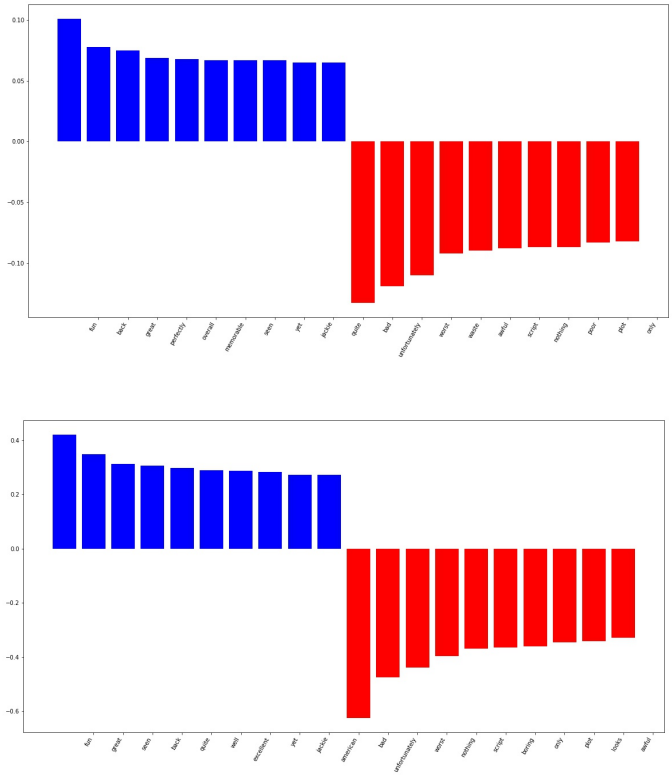

FIGURE 5 – 10 most positive and 10 most negative words

### 2.1.2 Version 2 : Suppression des stop-words et stemmatisation

Dans cette version, nous supprimons les stop-words et nous réduisons chaque mot à sa racine (stemmatisation). Ces transformations permettront de réduire

considérablement le vocabulaire. Comme dans la version précédente, Les performances diminuent lorsque la régularisation devient faible. L'interprétation des poids est cependant moins parlante ici.

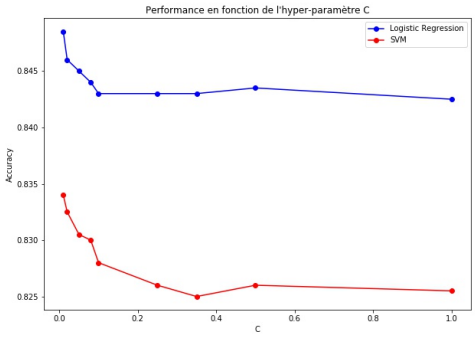


FIGURE 6 – Performances en fonction de l'hyper-paramètre de régularisation

| | SVM | Logistic Regression |
|---|---|---|
| Most positive words | « Enjoy », « Fun »,« Matrix », « Overal »,« Great »,« Surpris », « Thank »,« Sometim », « Carri »,« Excel » | « Fun »,« Enjoy »,« Great », « Surpris »,« Well »,« Excel », « Quit »,« Natur », « Overal »,« Entertain » |
| Most negative words | « Wast »,« Unfortun », « Bad »,« Suppos »,« Worst », « Noth »,« Bore »,« Poor », « Attempt »,« Potenti » | « Bad »,« Wast »,« Suppos », « Bore »,« Unfortun »,« Noth », « Worst »,« Attempt », « Poor »,« Stupid » |

### 2.1.3 Version 3 : Bigrammes et suppression des stop-words

Ici, on considère un dictionnaire contenant uniquement des Bigrammes. Cette piste est intéressante, à titre d'exemple, si on prend le bigramme « pretty bad » (plutôt mauvais), ce dernier est sensé coïncider avec un sentiment négatif alors que l'unigramme « pretty » coïnciderait plutôt à un avis positif. Les Bigrammes sont aussi très utiles lorsqu'on a des entités nommées (eg. New York, San Francisco). Parmi les termes les plus discriminants, on retrouve des titres de films ayant majoritairement des avis positifs sur IMDb (eg. Pulp Fiction, Private Ryan (Saving Private Ryan), Big Lebowski (The Big Lebowski) et Star Wars) et des titres de films ayant beaucoup d'avis négatifs (eg. Five Minutes). Ces deux modèles semblent faire du sur-apprentissage malgré une régularisation forte, ce qui les rend donc mauvais en terme de généralisation. Il serait cependant intéressant de considérer les Bigrammes et les Unigrammes en même temps.

| | SVM | Logistic Regression |
|---|---|---|
| Most positive words | « one best », « pulp fiction »,<br>« ive seen »,« truman show »,<br>« works well »,« even better »,<br>« private ryan »,« dark city »,<br>« worth seeing »,« big lebowski » | « one best »,« pulp fiction »,<br>« truman show »,« ive seen »,<br>« star wars »,« private ryan »,<br>« even better »,« dark city »,<br>« works well »,« big lebowski » |
| Most negative words | « bad movie », « waste time »,<br>« one worst », « looks like »<br>« much better »,« everyone else »,<br>« bad guy »,« even worse »,<br>« doesnt work »,« die hard » | « bad movie », « waste time »,<br>« one worst », « bad guy »,<br>« looks like »,« much better »,<br>« even worse », « batman robin »,<br>« everyone else »,« five minutes » |

### 2.1.4 Version 4 : Codage binaire, Unigrammes & Bigrammes et suppression des stop-words

Dans cette version nous considérons une représentation binaire, autrement dit, un codage présentiel, qui est sans doute le codage le plus simple et le plus efficace. En effet, nous obtenons des performances relativement bonnes ( 86% de bonne classification) comparé aux autres codages (cf. tableau comparatif en fin de section). Semblables aux résultats obtenus par Romain Ayres [2], « Hilarious » et « Memorable » sont les mots les plus positifs, « Bad » et « Worst » sont les mots les plus négatifs.


FIGURE 7 – Word Cloud des mots les plus discriminants pour chaque classifieur

### 2.1.5 Influence de la taille du dictionnaire

Nous nous intéressons ici à l'impact de la taille du vocabulaire considéré sur les performances des classifieurs. En observant la figure 8, On peut conclure qu'un nombre de tokens entre 20000 et 40000 est plutôt raisonnable. Avec un vocabulaire plus petit, un sous-apprentissage des données peut avoir lieu. Cependant, on remarque que lorsqu'on considère plus de 40000 tokens, cela n'a pas réellement d'impact sur les performances.

2. http ://webia.lip6.fr/ guigue/wikihomepage/uploads/Course/Ayres2013.pdf

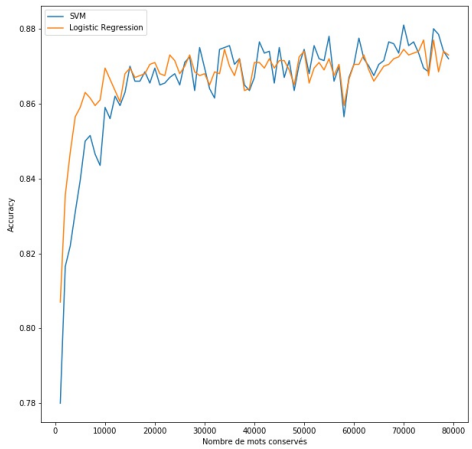


FIGURE 8 – Influence de la taille du vocabulaire sur les performances des deux classifieurs

### 2.1.6 Résumé : Tableau récapitulatif des résultats

D'autres expériences ont été effectuées, notamment pour observer l'influence du codage (Codage fréquentiel, codage binaire et TF-IDF). On remarque que le codage binaire donne de meilleurs résultats que les deux autres. Contrairement aux résultats obtenus par Romain Ayres [3], le codage TF-IDF est le moins bon des trois. On notera que nous avons choisi de conserver les majuscules car en observant les données, on remarque que les utilisateurs ont tendance à affirmer leur satisfaction/mécontentement vis-à-vis d'un film en écrivant en majuscule. La stemmatisation permet d'augmenter légèrement les performances dans le cas d'un codage fréquentiel et d'un codage TF-IDF, mais pas dans le cas d'un codage binaire.

3. http ://webia.lip6.fr/ guigue/wikihomepage/uploads/Course/Ayres2013.pdf

| **Codage** | **Stemming** | **Stop-words** | **SVM** | **Logistic Regression** |
|---|---|---|---|---|
| TF (Unigrams) | NO | NO | 83% | 84% |
| TF (Unigrams) | NO | YES | 84% | 84% |
| TF (Unigrams) | YES | YES | 85.4% | 85.1% |
| TF (Bigrams) | NO | NO | 85.8% | 86.4% |
| Binaire (Unigrams) | NO | NO | 86.7% | 86.5% |
| Binaire (Unigrams) | YES | NO | 85.2% | 83% |
| Binaire (Unigrams & Bigrams) | NO | YES | 88% | 87.4% |
| TF-IDF (Unigrams & Bigrams) | NO | NO | 78.2% | 77.5% |
| TF-IDF (Unigrams & Bigrams) | NO | YES | 78% | 77% |
| TF-IDF (Unigrams & Bigrams) | YES | NO | 80% | 79.4% |

# 3 Partie Bonus

## 3.1 Topic Modeling avec LDA

Le Topic Modeling -modélisation de sujet- consiste à générer un modèle probabiliste capable d'attribuer un thème à un document.

### 3.1.1 Applications

Les applications du topic modeling peuvent être diverses. En histoire on peut identifier des évènements majeurs en analysant des textes par année. Les librairies en ligne peuvent utiliser le topic modeling pour proposer aux lecteurs des livres similaires à leurs lectures passées. Les médias d'informations peuvent regrouper des articles traitant de sujets similaires rapidement. Une application également intéressante peut être le clustering d'images où les images sont traitées comme du texte.

### 3.1.2 Latent Dirichlet Allocation

LDA est un modèle génératif probabiliste utilisé en topic modeling. L'intuition est que chaque document est représenté par une distribution de Dirichlet de thèmes, et que chaque thème est représenté par une distribution de mots-clés. L'algorithme prend en entrée le nombre de thèmes par lequel on veut clusteriser nos documents, au début des distributions aléatoires sont attribuées, elles sont ensuite modifiées à chaque itération jusqu'à convergence en une distribution thèmes/mots-clés acceptable.

La bibliothèque gensim propose un package complet pour LDA (plusieurs implémentations, outils d'évaluations, outils de visualisation...)

### 3.1.3 Entraînement d'un modèle LDA

**Données :** Les données que nous avons utilisées pour expérimenter le topic modeling avec LDA sont les données du dataset *newsgroup* regroupant 11 000 exemples regroupés en 20 thèmes.

La qualité d'un modèle dépend entre autres de l'étape de pré-traitement des données. Nous avons procédé à un nettoyage de la base avec des expressions régulières (suppression de la ponctuation, des adresses emails...), suppression de stopwords, tokenization, lemmatization et n-grams.

Le modèle LDA prend en entrée :
- Un dictionnaire du vocabulaire du corpus dans lequel à chaque mot est associé un identifiant.
- Le corpus sous forme de listes de tuples (word_id,word_frequency), où chaque liste représente une entrée de la base.
- Le nombre de thèmes selon lequel on souhaiterait séparer les documents.
- On peut également détérminer un certain nombre d'hyper-paramètres tels que le nombre de documents à utiliser dans l'apprentissage, la fréquence de mise à jour du modèle ou le nombre de tours d'entrainement.

Le modèle retourne une liste de $n$ thèmes et pour chaque thème la liste des mots clés par ordre d'importance. On peut alors inférer un nom à chaque thème en fonction des mots détectés.

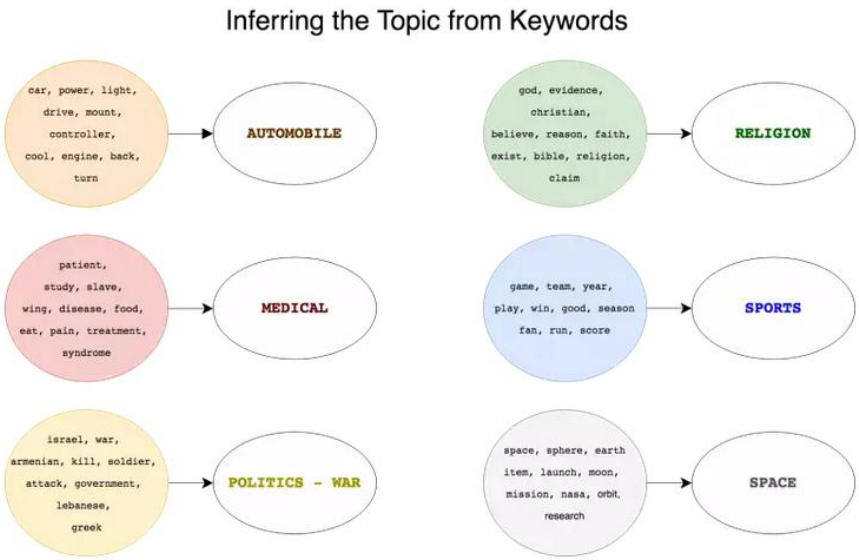


FIGURE 9 – Inférence de thèmes à partir des mots-clés [4]

Une des implémentations les plus efficaces proposées est le package MALLET topic model qui inclus une implémention très rapide et évolutive de l'échantillonnage de Gibbs, des méthodes efficaces d'optimisation d'hyper-paramètres et des outils pour l'inférence de thèmes sur de nouveaux documents à partir du modèle.

### 3.1.4 Évaluation d'un modèle LDA

Pour évaluer les modèles LDA on trouve dans la littérature la métrique de perplexité du modèle et le score de cohérence. La perplexité reflète le degré de surprise d'un modèle face à de nouvelles données, formellement c'est la log-vraissemblance normalisée sur l'ensemble de test. Le score de cohérence quand à lui mesure le degré de similarité sémantique entre les mots-clés les plus importants assignés à un thème. Cette mesure aide à distinguer les sujets qui sont véritablement sémantiquement interprétables et les thèmes qui sont justes des artefacts d'inférences statistiques.
**Artefact d'inférences statistiques :** use, also, system, may, number, new, high, work, will, need.
**Thème sémantiquement interprétable :** team, game, year, play, win, hit, fan, goal, run, score.

Le score de cohérence étant prouvé plus efficace en pratique on se basera sur cela.

4. https ://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

### 3.1.5 Détermination du nombre optimal de thèmes pour un modèle LDA

En pratique pour optimiser l'hyper-paramètre *num_ topics* du modèle LDA, on teste différentes valeurs et on étudie l'évolution du score de cohérence. Le nombre de thèmes qui signe la fin de l'évolution rapide du score de cohérence fournit généralement des thèmes significativement interprétables. Prendre un nombre de thèmes très grand (meilleur score de cohérence) peut isoler des sous-thèmes. Un des indicateurs permettant de détecter que le nombre de thèmes est trop important est le fait de trouver des mots qui reviennent sur plusieurs thèmes.

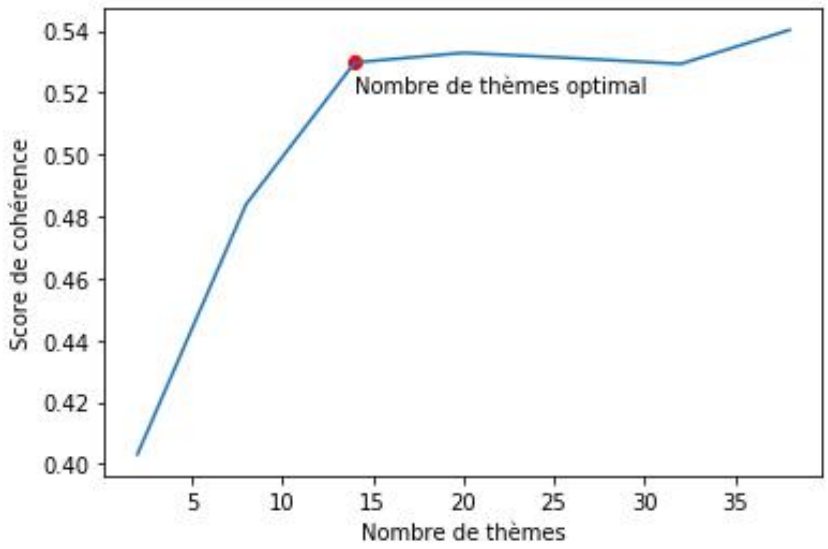


FIGURE 10 – Détermination du nombre de thèmes optimal

### 3.1.6 Inférences à partir d'un modèle LDA

Une fois le modèle établi, une des principales applications est de prédire le thème d'un nouveau document. Pour ce faire on sélectionne le thème ayant le plus grand pourcentage de contribution dans le document.

Parfois les mots clés ne suffisent pas à déterminer un thème. Dans ces cas là on peut trouver le document qui a le plus contribué à la sélection du thème, la lecture de ce document peut alors guider le choix du sujet.

### 3.1.7 Visualisation

pyLDAvis est outil de visualisation interactif permettant d'observer la répartition des thèmes ainsi que de nombreuses statistiques sur les mots-clés et les documents.

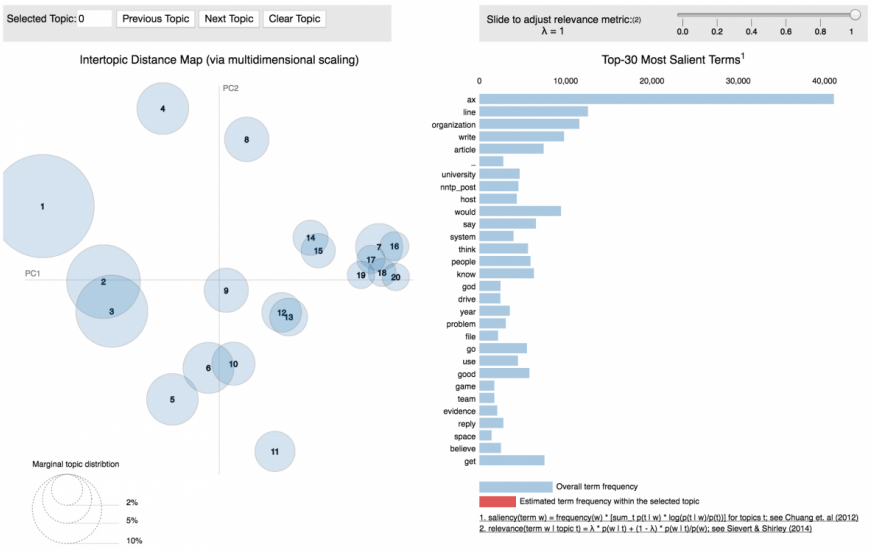


FIGURE 11 – Visualisation des thèmes du modèle LDA

## 3.2 Word embeddings : Word2Vec

### 3.2.1 Principe

Les méthodes Word2Vec associent une représentation latente à chaque mot de tel sorte que deux mots qui apparaissent dans des contextes similaires auront justement une forte similarité. Jusqu'à présent, nous avons modélisé les mots à l'aide d'une représentation « sparse » (eg. BoW), cette dernière a plusieurs inconvénients dont le fossé sémantique, ces méthodes viennent pallier ce problème en permettant de capturer notamment la synonymie des mots.

Word2Vec est composé de deux modèles de langues : Skip-Gram, qui consiste à prédire un mot en connaissant le contexte de ce dernier et CBOW (Continuous Bag of Word) qui lui, prend en entrée un contexte et prédit un mot susceptible d'appartenir à ce contexte.

Il y a deux paramètres importants à prendre en considération : Le contexte, qui est une fenêtre de mots dont la taille est définie au préalable et la taille de notre espace latent.

En entraînant le modèle sur la base de données IMDb (bien plus large que celle utilisée précédemment pour la tâche de classification de sentiments), on peut notamment faire des soustractions et des additions de représentations de mots mais également résoudre des analogies :

$$"bad" + ("awesome" - "good") = "awful"$$
$$"actor" + ("woman" - "man") = "actress"$$
$$"boy" + ("woman" - "man") = "girl"$$

### 3.2.2 Visualisation : t-SNE

La visualisation peut être très intéressante notamment pour pouvoir interpréter les relations entre les vecteurs de représentation. Cependant, ces derniers peuvent avoir un nombre considérable de variables, il est donc indispensable d'effectuer une réduction de dimension. t-SNE est parmi les méthodes de réduction de dimension les plus utilisées en machine learning. Elle consiste à minimiser la divergence de Kullback-Leibler entre une distribution qui mesure la similarité entre deux points dans l'espace d'origine et une distribution qui elle, mesure la similarité de deux points dans l'espace de visualisation.

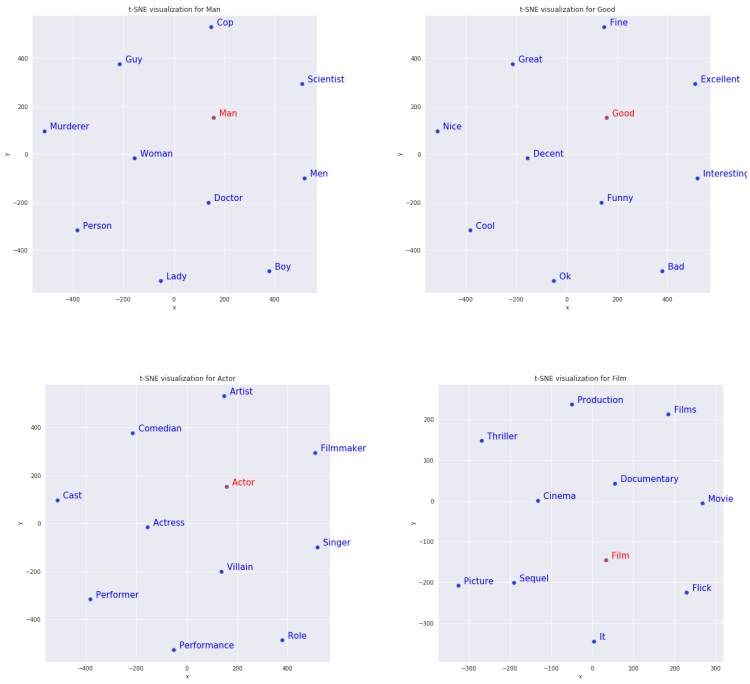


FIGURE 12 – Exemples de visualisations avec t-SNE

# Conclusion

Nous avons à travers ce projet abordé différentes thèmatiques en TAL à savoir : la classification sur des données non équilibrées, l'analyse de sentiments, le topic modeling et les représentations de données (BOW et Word2Vec). Dans un premier temps ce projet aura permis la prise en main des principaux outils de text mining sous Python (nltk, gensim). Nous avons également compris l'importance du pré-traitement de données sur les problématiques en TAL et comment un bon pré-traitement pouvait considérablement améliorer les performances d'un modèle.