Rapport de stage technique MAIN 4

Clustering de textes courts pour relier des députés

Achille Baucher
Encadrant.e.s :
Marie-Jeanne Lesot
Adrien Revault d'Allonnes

Juin - Juillet 2020

Résumé

Nous avons exploré plusieurs méthodes de représentations vectorielles de textes, de réduction de dimension et de clustering appliquées à des textes courts. Ces textes sont des lettres écrites par les députés aux ministres dans le cadre de l'assemblée nationale, et les clusters sont formés afin à déterminer des liens entre les députés en fonction de leur propension à écrire des lettres sémantiquement similaires.

Remerciements

Merci à Marie-Jeanne Lesot et Adrien Revault d'Allonnes pour leur soutien dans ce projet, leur enseignement précieux et leur attention durant tout le stage.

Contents

1 Introduction

1.1 Présentation

Ce rapport présente mon stage de quatrième année effectué au Laboratoire d'Informatique de Paris 6 (LIP6) sur le campus de Jussieu. À cause du contexte particulier lié au Covid 19, le stage s'est effectué dans la période relativement courte d'un mois et demi, et en télétravail.

J'ai cherché à travailler en laboratoire pour me familiariser avec le monde de la recherche, que j'envisage de rejoindre, pour en apprendre les bonnes pratiques et pour découvrir des concepts mathématiques intéressants, particulièrement en apprentissage artificiel. Le sujet du stage, que nous avons mis au point au cours des premiers entretiens, est la mise en œuvre d'outils de Traitement Automatique du Langage Naturel pour distinguer des clusters au sein des lettres provenant des députés de l'assemblée nationale, dans l'idée de pouvoir établir des liens entre les députés. Le choix de ces données était motivé par le fait que les avais déjà utilisées lors de mon projet industriel de cette année ([ABD+20]). J'ai présenté ce travail à mes tut.eur.rice.s et nous en avons déduit qu'il pourrait être intéressant d'approfondir certains aspects. Cela m'a aussi permis de me concentrer directement sur le fond en évitant de passer du temps sur les aspects techniques.

J'ai effectué au cours de mon stage plusieurs types de tâches propres à la recherche scientifique. La première est la recherche bibliographique, qui a consisté d'abord à trouver de la documentation sur internet avec les bons mots clés, et grâce aux conseils de mes tut.eur.rice.s . J'ai ensuite passé une grande partie de mon stage à lire les articles, en me renseignant sur les concepts inconnus, et à prendre des notes sur format numérique ou papier. Avant de commencer à coder, il a fallu trouver des bibliothèques qui implémentaient en langage Python, choisi pour sa simplicité et sa richesse, les algorithmes souhaités et apprendre à les utiliser. Je me suis principalement servi de Scikit Learn, Nltk et Scipy. L'étape d'avancement consistait ensuite à appliquer ce que j'avais appris sur les données étudiées dans un bac-à-sable sous la forme d'un notebook Jupyter. Cette forme de code très haut-niveau, malléable et graphique me permettait de réfléchir en effectuant continuellement de nouveaux tests [1]. Je pouvais ensuite regrouper les lignes dispersées pour en faire du code plus exécutable. La dernière étape a consisté à expliquer soigneusement la méthode utilisée dans ce rapport.

Nous nous retrouvions avec mes tut.eur.rice.s chaque semaine sur Skype pour discuter de mon avancement et répondre à mes questions, réfléchir à des pistes, et proposer un plan de route pour la semaine.

1.2 LIP6

Le Laboratoire d'Informatique de Paris 6 est, avec 220 chercheurs permanents et 199 doctorants, le plus gros laboratoire de recherche en informatique en France. Il regroupe plusieurs pôles de recherche répartis

[1] Le git du code : `https://github.com/AchilleBaucher/confiance_diffusion`

Député : Bertrand Sorre
Ministre : Ministre de l'intérieur
Date : 12 Juin 2018
Rubrique : Sécurité routière
Titre : Précision sur les 80 km/h

Lettre :
M. Bertrand Sorre attire l'attention de M. le ministre d'État, ministre de l'intérieur, **sur la décision prise de réduire la vitesse maximale autorisée à 80 km/h**, à compter du 1er juillet 2018, sur les routes à double sens sans séparateur central (limitée actuellement à 90 km/h). Cette disposition permettra de sauver entre 300 et 400 vies par an selon le comité des experts du conseil national de la sécurité routière dans son rapport du 29 novembre 2013. Fréquemment questionné à ce sujet dans la circonscription de La Manche dont il est l'élu, *il aimerait savoir si cette décision implique également des modifications sur les vitesses actuellement autorisées pour les professionnels de la route (transport routier, autobus) ou pour les apprentis conducteurs d'un véhicule léger, titulaires d'un permis depuis de moins de 2 ans.*

Figure 1: Exemple de lettre

dans plusieurs domaines, comme le département *Données et Apprentissage Artificiel* duquel fait partie ma tutrice Marie-Jeanne Lesot.

1.3 Données

Les lettres étudiées (voir exemple en Figure 1) sont écrites par les députés de l'assemblée nationale et sont adressées aux ministres. Elles sont assez courtes (**160** mots en moyenne) et sont au nombre de **20000** pour le mandat étudié. Elles comportent un titre généralement explicite du sujet abordé (précis), et sont classées arbitrairement dans des rubriques (il y en a environ **200**) selon leur thème (plus général). Les lettres commencent toujours par une phrase d'explication de l'objet de la lettre (en gras sur l'exemple), suivie par une partie d'argumentation et se terminent par la demande (en italique sur l'exemple), qui peut être très variée : demande de précision, de retrait d'une loi, de prise en considération d'une information ...

Les lettres ont été dépouillées de leur ponctuation, majuscules et stop-words [2] (mots communs sans importance) et mises à l'infinitif singulier. Cette normalisation a été effectuée lors de notre projet industriel, et est expliquée plus précisément dans la partie Analyse du Texte de [ABD+20].

1.4 Structure du document

L'objectif du stage est d'essayer est d'analyser et d'appliquer différentes méthodes de clustering de textes sur les données présentées dans la section 1.3. Nous commençons par présenter dans la Section

[2] Nous avons choisi la liste donnée par la librairie **stop-words** : `https://pypi.org/project/stop-words/`

2 diverses manières de représenter un document textuel sous forme de vecteur, puis dans la Section 3 quelques algorithmes de clustering utilisables avec représentations vectorielles obtenues. L'application des méthodes présentées est détaillée dans la Section 4, ainsi que les liens entre députés finalement obtenus.

2 Représentation vectorielle des textes

La première étape a consisté à mettre les documents sous des formes appropriées aux calculs de distances ou de similarités entre lettres, qui sont utilisées par les algorithmes de clustering. Les représentations les plus simples à manipuler sont les représentations vectorielles, et nous nous intéresserons donc dans cette section à la transformation de documents en vecteurs. Pour des raisons de simplicité, nous avons choisi des méthodes qui s'appuient sur le champs lexical d'une lettre, sans prendre en compte l'ordre de succession des mots. Nous supposons que le champs lexical est suffisamment révélateur du sujet abordé.

2.1 Représentations vectorielles initiales

Il est d'abord nécessaire de partir d'un matériau de base, une première représentation vectorielle de haute dimension, avant de réduire les dimensions dans la section 2.2. Ces premières matrices sont sous la forme suivante : chaque ligne représente un document et chaque colonne représente un mot.

2.1.1 Term-Frequency (TF)

La représentation la plus simple pour représenter le champs lexical d'un texte est la matrice Term-Frequency, constituée d'entiers indiquant l'occurrence d'un mot dans un document. Cette représentation est toutefois assez limitée puisqu'elle accorde de l'importance à tous les mots redondants qui peuvent être banals, c'est-à-dire qui n'apportent pas d'information déterminante du sujet abordé.

2.1.2 Term Frequency - Inverse Document Frequency (TF-IDF)

Le Term Frequency - Inverse Document Frequency tente de palier le problème de la banalité des mots en réduisant leur score en fonction de leur occurence moyenne. Pour cela, on pondère le TF par un terme appelé IDF, qui augmente en fonction de la rareté du mot dans les documents.

$$TF.IDF(mot, doc) = \frac{OC(mot, doc)}{|doc|} log\left(\frac{|corpus|}{T(mot)}\right) \quad (1)$$

$OC(doc, mot)$ désigne le nombre d'apparitions (occurrence) du mot dans le texte, $|doc|$ le nombre de mots dans le document, $|corpus|$ le nombre de documents dans le corpus étudié, et $T(mot)$ le nombre de documents dans lesquels le mot apparaît. Il existe plusieurs paramétrages possibles de ce calcul, qui sont précisées dans le rapport de notre projet industriel ([ABD+20]).

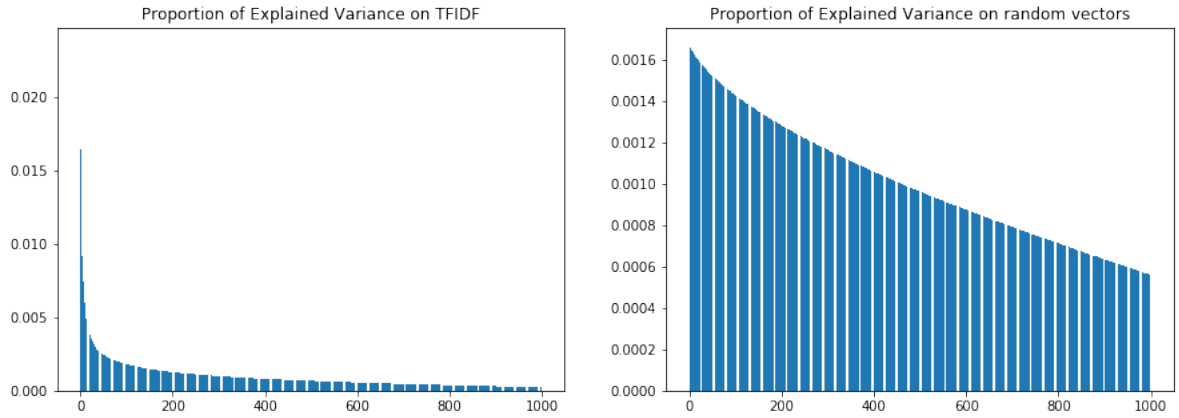


Figure 2: Variance expliquée des 1000 composantes principales

2.2 Méthodes de réduction de dimension

Les représentations vectorielles précédentes comportent un très grand nombre de dimensions. Il y a en effet une dimension par terme utilisé, un nombre qui monte jusqu'à plus de **44000** sur le corpus entier, et la majorité des vecteurs sont extrêmement creux puisque les documents utilisent en moyenne **160** mots environ. Ce nombre de dimensions est beaucoup trop élevé pour une analyse, principalement à cause d'un ensemble de phénomènes appelé malédiction de dimensionnalité, précisé dans la partie 2.5.1, mais aussi pour les difficultés computationnelles. L'enjeu des sections suivantes est de réduire drastiquement la dimension des vecteurs représentant les textes en conservant le maximum d'informations.

2.2.1 Analyse en composantes principales

L'analyse en Composante Principale, vue en cours, est un des algorithmes les plus répandus pour la réduction de dimension. Il est linéaire, efficace, et dispose de nombreuses techniques pour son interprétation et son évaluation. L'algorithme crée un nombre k de nouvelles composantes comme combinaisons linéaires des dimensions initiales, et qui expliquent mieux la manière dont les données sont réparties.

Pour choisir le nombre de composantes, nous pouvons afficher la part de variance (Figure 2) expliquée par chaque variable. On remarque qu'après les 4 premières (sachant que 4 dimensions ne sont pas suffisantes pour représenter et discriminer un corpus aussi diversifié) , la décroissance suit une tendance assez continue et ne comporte pas de gap. Ce graphique montre que des composantes principales ont bien une grande importance, même s'il faut en prendre au moins une centaine. En effet, si on compare avec un graphique similaire appliquée à des données aléatoires, la décroissance est bien plus monotone.

Nous ne pouvons pas distinguer de gap sur le graphique, et nous avons arbitrairement choisi un nombre de dimensions autour de **200**, correspondant à une explication de 20% de la variance.

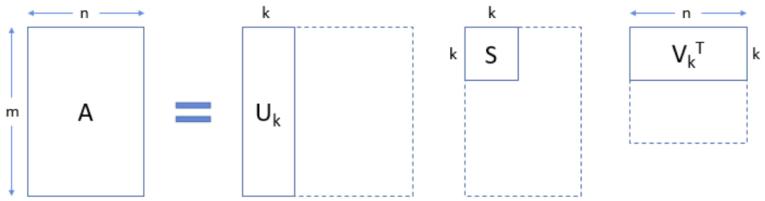


Figure 3: Décomposition en Valeurs Singulières et sélection des k vecteurs les plus importants

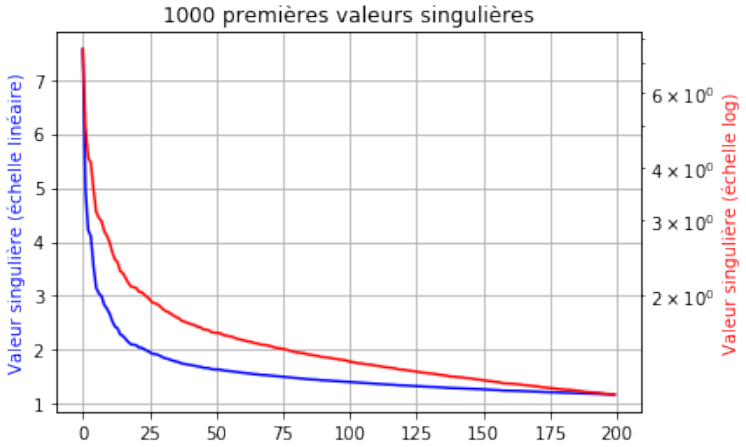


Figure 4: Valeurs singulières pour un corpus de plus de 1000 documents, comptant plus de 9000 termes

2.2.2 Analyse Sémantique Latente (LSA)

L'analyse sémantique latente [DDF+90] permet de capturer la sémantique des mots à travers leur contexte latent, appelé thème. L'algorithme fonctionne à partir d'une matrice des documents × termes ($m \times n$) qui attribue à chaque mot un score d'importance dans chaque document, comme la matrice des TF-IDF présentée en Section 2.1.2. Cette matrice est ensuite décomposée en valeurs singulières (SVD), ce qui nous renvoie trois matrices, comme illustré sur la Figure 3 : U est une présentation vectorielle des documents, S est composée des valeurs singulières, et V est une présentation vectorielle des termes. Cette décomposition permet ensuite de réduire la dimension des trois matrices en sélectionnant les $k < n$ lignes ayant les valeurs singulières les plus élevées, avec k choisi arbitrairement [3].

On obtient alors U_k et V_k qui font correspondre respectivement les documents et les termes à des vecteurs de dimension k. La décomposition en valeurs singulières permet de sélectionner les termes les plus importants pour représenter un thème et la matrice U_k fait correspondre à chaque document un vecteur attribuant une valeur à chacun des k thèmes. On peut alors manipuler ce vecteur pour calculer des distances entre les documents, car il a l'avantage d'être beaucoup moins creux et d'avoir potentiellement pris en compte la sémantique latente des termes.

[3] Précisions en http://blog.onyme.com/stats-semantique-lsa/ sur la SVD pour le LSA

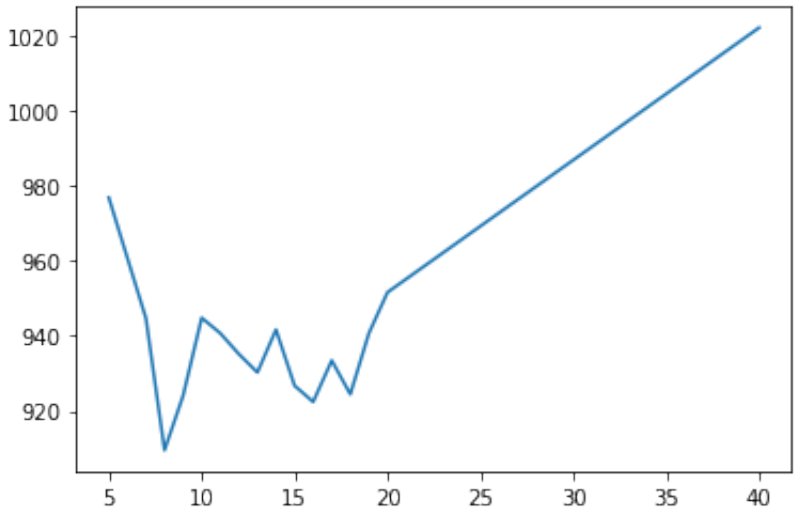


Figure 5: Perplexité du modèle LDA en fonction du nombre de topics choisis

Nous avons appliqué cette méthode au corpus présenté en section 1.3 [4], en nous aidant de la documentation de Scikit Learn. Celle-ci indique de meilleurs résultats avec un TF-IDF sublinéaire, qui pénalise les grands scores de Term-Frequency, en remplaçant TF par $1 + log(TF)$. Nous avons choisi arbitrairement après observation de l'évolution des valeurs singulières sur la Figure 4 un k autour de **50**, au delà de quoi les valeurs singulières décroissent beaucoup plus lentement.

Les défauts de cette méthode sont que la SVD a une complexité de $O(m^2n)$, et que les liens identifiés sont conditionnés par la linéarité de la SVD.

2.2.3 Allocation de Dirichlet Latente

L'Allocation de Dirichlet Latente (LDA), initiée par [BNJ01], s'appuie sur la matrice des TF présentée en 2.1.1 . On commence par choisir la valeur de k, qui représente concrètement le nombre de thèmes pouvant être abordés. L'algorithme attribue à chaque document un vecteur de dimension k, représentant à quel point le document correspond à chacun des k thème générés. Ces vecteurs sont représentés par des distributions de probabilités sur les thèmes. Les thèmes sont représentés par des distributions de probabilité pour chaque mot. Pour choisir le paramètre du nombre de thèmes, une méthode populaire [5] est de mesurer l'évolution de la perplexité ou de la cohérence 2.4 en fonction du nombre de topics, et de choisir la valeur qui minimise ou maximise cette mesure. Nous avons choisi comme dans l'article [BNJ01] la perplexité qui est plus intuitive et déjà implémentée. Nous avons appliqué l'algorithme et calculé la perplexité sur le corpus présenté dans la section 1.3 [6]. On voit sur le graphique 5 que le premier creux est à **7** dimensions, mais comme ce nombre n'est pas suffisant pour établir des clusters, nous avons choisi **18**, qui est le dernier creux.

[4] Code inspiré de `https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analysis/`

[5] `https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0`

[6] Nous nous sommes appuyé sur cette page pour le code : `https://medium.com/mlreview/topic-modeling-with-scikit-learn-e80d33668730`

2.2.4 Factorisation en Matrices non Négatives (NMF)

Cette méthode [XLG03] décompose la matrice des TFIDF ($m \times n$), dont chaque terme est positif, en deux matrices $W(m \times k)$ et $H(k \times n)$ aussi positives, avec k fixé arbitrairement. W est donc une représentation vectorielle réduite des documents et H une représentation vectorielle des termes. L'algorithme NMF peut s'implémenter de plusieurs manières, selon le choix du calcul de l'erreur de décomposition [7] . Nous avons appliqué à nos données l'implémentation de Scikit Learn, dont les précisions sont dans la documentation (User Guide) [8] [9] .

Pour le choix du nombre de dimensions, les méthodes pour le déterminer automatiquement sont assez complexes [YZO10], et nous avons donc simplement remarqué qu'en fixant un nombre de dimensions assez élevé, comme **100**, on obtenait des résultats plus pertinents qu'en dimension faible pour la reconnaissance des thèmes. Il est même proposé dans [XLG03], dans un cadre de clustering, de prendre le même nombre de dimensions que de clusters souhaités, et d'attribuer les clusters avec l'indice de la valeur maximale de la représentation vectorielle d'un document.

2.3 Représentation visuelle

Pour nous aider à évaluer les caractéristiques des représentations vectorielles présentées dans la partie précédente, nous avons essayé de représenter visuellement les vecteurs. Cela nous permet de savoir si les représentations vectorielles trouvées distinguent bien des groupes séparés, et d'avoir une idée de la qualité des clustering effectués en observant la répartition des clusters sur le graphique.

2.3.1 ACP

Nous avons commencé par représenter visuellement les points des représentations vectorielles obtenues à l'aide d'une analyse en composantes principales en 2 dimensions. Comme on peut le voir sur la figure 6, elle peut être efficace pour distinguer certains groupes de points bien séparés des autres, mais la plupart des points sont trop enchevêtrés au centre pour pouvoir repérer des groupes particuliers.

En effet, l'ACP est linéaire et n'est pas capable de prendre en compte des géométries plus particulières, contrairement aux t-SNE [MH08] et UMAP (voir section 2.3.2).

[7] `https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-explo-nmf.pdf`
[8] User Guide`https://scikit-learn.org/stable/modules/decomposition.html#nmf`
[9] Code inspiré de `https://medium.com/mlreview/topic-modeling-with-scikit-learn-e80d33668730`

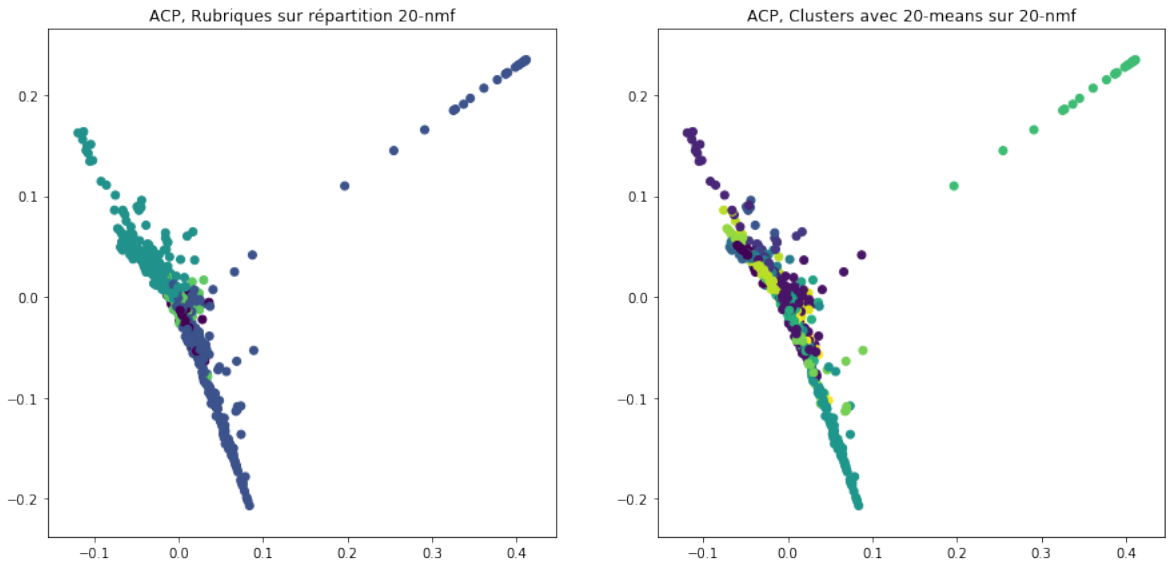


Figure 6: Représentation des documents en deux dimensions grâce à une Analyse en Composantes Principales, appliquée à des vecteurs NMF de dimension 20. Les couleurs correspondent à gauche aux rubriques présentées dans la Section 1.3 et à droite aux clusters trouvés par l'algorithme des K-means avec K=20

2.3.2 Uniform Manifold Approximation for Projection and Representation (UMAP)

Cette méthode récente [MHSG18] est souvent utilisée comme une amélioration du T-SNE, en réduisant en deux dimensions (ou plus) un ensemble de vecteurs pour pouvoir les représenter sur un graphique. Son principal intérêt est de faire en sorte que la distance entre les points soient assez bien représentés sur le graphique réduit, en prenant en compte des structures non linéaires. Les résultats sont que les graphiques séparent très nettement les clusters trouvés avec diverses techniques, comme on peut le voir sur la figure 7. La UMAP est radicalement plus lisible que le PCA en figure 6, et souvent plus pertinente et bien plus rapide que le t-SNE [10], d'où son utilisation régulière dans notre étude. Nous avons utilisé l'implémentation disponible en [11].

2.4 Perplexité

La perplexité est une mesure d'évaluation intrinsèque (qui ne fait pas appel à une tâche et comparaison extérieure) adaptée aux modèles de langages. Son principal avantage est donc de ne pas nécessiter d'étiquettes pour l'évaluation des représentations vectorielles des documents. Elle mesure l'écart sur le corpus entre la probabilité estimée d'apparition d'un mot et son apparition réelle. Elle est algébriquement équivalente à l'inverse géométrique de la vraisemblance de chaque mot selon le modèle de langage mesuré ([BNJ01]). Elle doit donc décroître en avec la vraisemblance des données. Elle a été créée pour le modèle LDA, mais peut être étendue. On calcule d'abord la probabilité d'un document comme le produit des

[10] `https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668`
[11] `https://umap-learn.readthedocs.io/en/latest/`

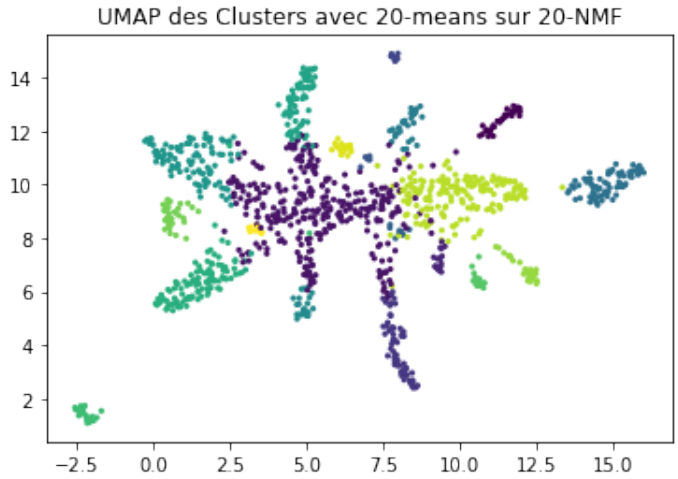


Figure 7: Représentation UMAP des vecteurs NMF en dimension 20, les couleurs sont les clusters trouvés avec k-means, k=20

probabilités des mots selon le thème associé au document :

$$p(w) = \prod_{n=1}^{N} \sum_{z=1}^{Z} p(w_n|z)p(z|w)$$

avec N le nombre de mots du document, Z l'ensemble des topics et d le document. On en déduit la perplexité du modèle de langage entier :

$$perplexity = exp\left(-\frac{\sum_{d=1}^{M} log(p(w_d))}{\sum_{d=1}^{M} N_d}\right)$$

avec M le nombre de documents, w_d le document numéro d, $p(w_d)$ la probabilité du document selon le modèle, et N_d le nombre de mots dans le document d.

La cohérence d'un modèle de langage est une autre mesure d'évaluation intrinsèque, et indique le contraire de la perplexité[12].

2.5 Distances

Une fois la représentation vectorielle obtenue, il faut choisir une distance avec laquelle on manipulera les données. Nous avons étudié le comportement (concentration, répartition) des différentes mesures de distances pour chacune des représentations vectorielles trouvées dans la Section 4.2.

[12] https://towardsdatascience.com/evaluate-topic-model-in-python-latent-dirichlet-allocation-lda-7d57484bb5d0

2.5.1 Malédiction de la dimensionnalité

La "malédiction de la dimensionnalité" désigne une série de problèmes qui affectent la manipulation des mesures de similarités, de distances, etc., des données en (très) grande dimension. Ces problèmes sont une des raisons aux réductions de dimension présentées dans la section 2.2, mais peuvent toujours concerner les données réduites, selon les mesures utilisées pour comparer les points. Les explications à propos des trois principales caractéristiques de ce phénomène sont inspirées de [Fal19].

- **Espaces creux (sparsity)** : Plus on augmente la dimension, plus la densité des données diminue. On peut illustrer ce phénomène par le fait que le nombre de points contenus dans une boule de même rayon en distance euclidienne diminue très vite lorsqu'on augmente la dimension.
- **Concentration des distances** : Les distances euclidiennes deviennent très concentrées autour d'une même valeur, ce qui rend plus difficile les distinctions.
- **Hubness** : Un petit nombre de points deviennent des Hub, i.e. ils font partie des plus proches voisins d'énormément de points, tandis que la grande majorité font partie des plus proches voisins de très peu de points.

2.5.2 Distance euclidienne

La distance euclidienne est la plus classique pour mesurer une distance entre deux points. Cependant, comme les vecteurs utilisés dans notre étude représentent un ensemble de thèmes, il semble étrange de mesurer des différences de position : si en effet une lettre est plus longue qu'une autre, ces coefficients seront plus élevés, et une distance sera donc mesurée entre ces deux lettres tandis qu'elles abordaient peut-être le même sujet. C'est pourquoi nous avons préféré l'éviter lorsqu'il a été possible de le faire, mais son utilisation étaient tout de même nécessaire pour les algorithmes des k-means et x-means qui reposent dessus.

2.5.3 Distance du cosinus

La similarité cosinus représente la similarité de direction entre 2 vecteurs et se calcule comme :

$$\frac{\langle A.B \rangle}{||A||||B||}$$

La distance du cosinus est égale à $1-$ la similarité.

Se concentrer sur les directions nous a semblé assez pertinent car on veut mesurer à quel point des documents sont orientés vers les mêmes thèmes (qui sont les coefficients des vecteurs).

3 Clustering des textes

Le clustering consiste à diviser un ensemble de données en plusieurs groupes. Dans notre étude, l'objectif est que chaque groupe représente un sujet, assez précis, et donc que les lettres qui abordent le même sujet, et elles seules, soient réunies au sein d'un même groupe. Nous avons essayé plusieurs algorithmes de clustering pour sélectionner ensuite le plus pertinent pour nos données. Nous avons donc eu aussi besoin de disposer d'outils de mesure de la qualité d'un clustering.

3.1 Évaluer une partition et trouver le bon nombre de clusters

Comme nous ne disposons pas d'étiquettes pour connaître la pertinence d'une partition obtenue, nous avons calculé des indicateurs de qualité *intrinsèques*. [GCLL10] en propose une certaine quantité adaptée au clustering de documents, mais nous n'avons pas eu le temps de les essayer tous et nous avons pris les critères les pus courants, et qui étaient déjà implémentés.

3.1.1 Inertie intra-classe

L'inertie intra-classe se calcule comme la somme sur chaque cluster des distances de tous les individus du cluster.

$$W = \sum_{r \in R} \sum_{i \in I_r} ||x_i - c_r||^2$$

Avec W l'inertie, R l'ensemble des clusters trouvés, I_r les individus du cluster r, c_r le centre de gravité du cluster r, et x_i l'individu i.

Plus un cluster est homogène, c'est à dire constitué de points proches entre eux selon la distance choisie, plus son inertie sera faible. Il est surtout intéressant d'utiliser la méthode ELBOW, c'est à dire d'observer l'évolution de l'inertie augmentant au fur et à mesure le nombre de clusters, pour identifier le point de bascule, comme sur le graphique 8.

3.1.2 Critère de Davies-Bouldin

Le critère de Davies-Bouldin [DB79] veut mesurer la capacité des clusters à être bien distincts. Pour cela, on calcule sur chaque cluster la somme du maximum pour tous les autres clusters de la somme de la distance moyenne avec leur centre des deux clusters et la distance entre les centres de deux clusters.

$$S_{DB} = \frac{1}{|R|} \sum_{r \in R} \max_{r' \neq r} \left(\frac{\bar{\delta_r} + \bar{\delta_{r'}}}{d(c_r, c_{r'})} \right)$$

avec

$$\bar{\delta_r} = \frac{1}{|r|} \sum_{i \in I_r} d(x_i, c_r)$$

la distance moyenne des points d'un cluster à son centre.

Plus l'indice est petit, plus la classification est bonne. Cela signifie en effet que la distance des points avec leur centre réduit alors que celle entre les clusters augmente.

3.1.3 Critère de Calinski-Harabasz

L'e critère de Calinski-Harabasz [CH74] tente de mesurer la capacité des clusters à être moins dispersés en leur intérieur qu'entre eux, en calculant le rapport entre la variance inter-groupes et des variance intra-groupes [13].

La variance inter-groupes est la somme des distances entre le centres des clusters et le centre global, pondéré par le nombre de points de chaque cluster. Elle mesure la dispersion des clusters.

$$V = \sum_{r \in R} |r| \|c_r - c\|$$

avec c le centre global.

La variance intra-groupes est la somme des distances des points d'un cluster avec le centre de ce cluster, divisé par le nombre de points de ce cluster. Elle mesure la dispersion à l'intérieur d'un cluster.

$$W_r = \sum_{r \in R} \frac{1}{\|I_r\|} \sum_{i \in I_r} \|x_i - c_r\|$$

Pour obtenir le critère de Calinski-Harabasz, on divise la variance inter-groupes par la somme des variances intra-groupes. Le tout est pondéré par le rapport entre la différence entre le nombre de points et le nombre de clusters désirés et le nombre de clusters moins 1.

$$S_{CH} = \frac{(n - |R|)V}{(|R| - 1) \sum_{r \in R} W_r}$$

Plus l'indice est grand, meilleure est la classification, car cela signifie que les clusters sont plus dispersés entre eux et que les points d'un clusters sont moins dispersés. Pour choisir le bon nombre de clusters, on essaie donc de choisir celui qui maximise ce critère, comme sur le graphique 8.

[13] Les calculs sont tirés de `https://scikit-learn.org/stable/modules/clustering.html`

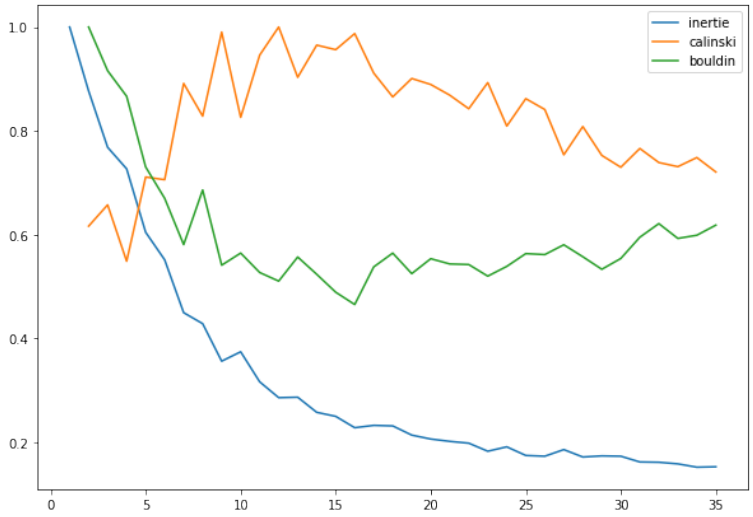


Figure 8: Évolution des 3 indices présentés sur les clusters obtenus par k-means en fonction de k

3.1.4 Trouver le bon nombre de clusters

En affichant l'évolution des indices d'évaluation des partitionnements en fonction du nombre de clusters demandés sur la figure 8, on peut tenter de choisir le point de bascule. Il se situerait ici autour de **16**, qui correspond au creux global de l'indice de Davies-Bouldin et à un pic pour celui de Kalinski-Harabasz.

3.2 K-means

Les K-means sont un algorithme de référence pour le clustering principalement en raison de son efficacité et sa rapidité. Il présente le défaut d'avoir à imposer au départ le nombre k de clusters, mais nous avons la possibilité d'effectuer un grand nombre de partitions avec différents k pour ensuite choisir le bon à l'aide de méthode d'évaluation comme vu en 3.1.4. Le principal problème est qu'il ne peut s'effectuer que sur une distance euclidienne. Des extension de l'algorithme comme le kernel k-means [DGK04] existent pour remédier à ce problème, mais nous n'avons pas eu le temps de nous y pencher.

3.3 X-means

L'amélioration apportée aux k-means par l'algorithme des X-means [PM02] est d'obtenir un partitionnement sans choisir au préalable le nombre de clusters, et plus rapidement que l'exécution des k-means pour chaque valeur de k possible. L'idée est qu'à chaque étape, l'algorithme choisit de diviser ou non chaque cluster en deux autres en utilisant le critère BIC, jusqu-à ce qu'aucune division ne devienne avantageuse. Le Bayesian Information Criterium (BIC) est un score de pénalité statistique qui permet de mesurer la perte d'information du modèle.

Nous avons utilisé les X-means avec la bibiliothèque Python PyClustering [14].

3.4 Clustering Ascendant Hiérarchique (HAC)

Le Clustering Ascendant Hiérarchique est un algorithme de clustering très utilisé, qui débute en attribuant à chaque point un cluster différent, puis fusionne deux clusters à chaque itération, jusqu'à une condition à déterminer. Il présente le grand avantage de pouvoir considérer n'importe quelle mesure de similarité entre les points, contrairement aux k-means. Comme clustering est hiérarchique, il est plus aisé de choisir le nombre de clusters voulu, en stoppant l'algorithme lorsque le partitionnement paraît correct. La complexité en temps et en espace est cependant plus longue que les k-means. À chaque étape, les deux clusters à fusionner sont ceux qui minimisent une certaine mesure de dissimilarité entre clusters. Il existe principalement trois version de cette mesure de différence entre deux clusters, qu'on appelle chaînage :

- **Complete** : La distance maximale qu'on peut trouver entre deux points de chaque cluster $\underset{x_1 \in R_1, x_2 \in R_2}{max} d(x_1, x_2)$
- **Average** : La distance moyenne entre les points de chaque cluster $\frac{1}{|R_1||R_2|} \underset{x_1 \in R_1, x_2 \in R_2}{\sum} d(x_1, x_2)$
- **Single** : La distance minimale qu'on peut trouver entre deux points de chaque cluster $\underset{x_1 \in R_1, x_2 \in R_2}{min} d(x_1, x_2)$

Avec d la dissimilarité entre deux points et R_1 l'ensemble des points du cluster 1.

Nous avons essayé et comparé ces trois méthodes nos représentations vectorielles, pour une distance euclidienne et une distance du cosinus. Un exemple des dendrograms tronqués obtenus pour la représentation NMF est donné sur la Figure 9.

On s'aperçoit sur cette figure la forme caractéristique de peigne prise par le linkage *single*, qui ajoute chaque point au fur et à mesure à un gros cluster, car sa grande taille lui donne plus de chance de contenir des points proches des nouveaux, et donc de minimiser le critère de fusion. Ce linkage ne permet donc pas de bien distinguer les clusters ici. On remarque aussi que cette forme en peigne est aussi présente lorsqu'on prend la distance euclidienne, ce qui est explicable par la hubness et l'aspect creux des espaces en grande dimension, vu dans la partie 2.5.

Les dendrogram complete et average sur la distance du cosinus ont l'air de distinguer beaucoup plus de clusters de taille raisonnable. De cette façon, nous avons pu choisir pour chaque représentation une méthode de Clustering Hiérarchique adaptée.

Pour le nombre de clusters, nous pouvons soit le fixer au départ, soit déterminer un seuil de distance à partir duquel l'algorithme arrête les fusions. L'affichage des distances en fonction des itérations de fusion, sur la figure 10, peut nous aider à choisir cette valeur.

[14] `https://pyclustering.github.io/docs/0.9.3/html/d2/d8b/namespacepyclustering_1_1cluster_1_1xmeans.html`

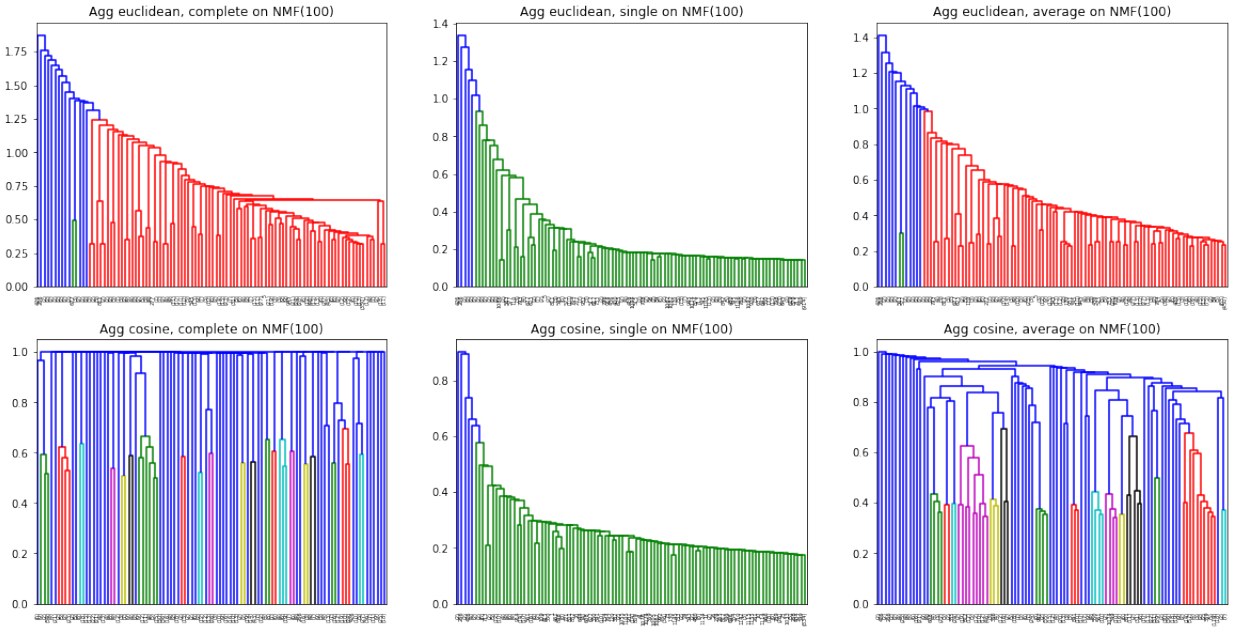


Figure 9: Comparaison de dendrogrammes obtenus les distances euclidienne et du cosinus, et les chaînages Complete, Single et Average sur une représentation NMF en 100 dimensions

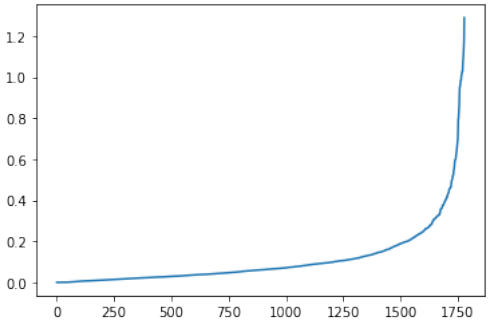


Figure 10: Évolution des distances en fonction des itérations d'un clustering hiérarchique sur une représentation LSA en 20 dimensions

4 Outils et méthodes mises en œuvre

4.1 Protocole expérimental

Données

Pour des raisons de capacité et de temps, nous avons choisi de réduire le corpus de documents à **10** rubriques choisies aléatoirement, et qui forment un corpus d'environ **2000** lettres.

Mise sous forme vectorielle

Nous avons commencé par obtenir les TF et TFIDF du corpus réduit. Puis, pour chaque représentation vectorielle réduite présentée dans la Section 2, nous avons mis les documents sous la forme étudiée, en faisant varier le nombre de dimensions. Nous avons choisi pour chaque représentation vectorielle un nombre de dimension adapté grâce aux critères de qualité disponibles.

Observations

Pour chacune des représentations vectorielles ainsi obtenues, nous avons effectué des observations sur le comportement des distances des lettres dans la Section 4.2 et sur la cohérence des thèmes identifiés par l'algorithme de réduction de dimension dans la Section 4.3.

Clustering

Nous avons ensuite appliqué dans la section 7.3 chacun des algorithmes de clustering présentés dans la section 3 à chacune des représentations vectorielles, en analysant les résultats à l'aide d'indices de qualité et de graphiques.

Liens entre députés

Nous avons finalement choisi les algorithmes de réduction de dimension et de clustering qui nous ont paru les plus pertinents pour créer dans la section 4.4 un graphe de députés dont les liens sont calculés à l'aide des clusters de lettres identifiés.

4.2 Comparaisons des distances

Afin de sélectionner la bonne mesure de distance et la bonne représentation vectorielle, nous avons réalisé plusieurs graphiques. La Figure 12 représente la densité de toutes les distances entre les points représentés avec une NMF. On voit que le grapqhique de la distance du cosinus est plus satisfaisant car il y a un petit pic de lettres très similaires et un grand pic de lettres très différentes, ce qui semble cohérent avec une situation dans laquelle les lettres sont réparties en petits groupes abordant un sujet similaire. Nous avons ensuite réalisé sur la Figure 13 une heatmap des distances pour des lettres arbitrairement choisies, dont nous savons à l'avance si le sujet est similaire ou non. On peut ainsi voir si les distances conviennent bien à une différence de sujet. Enfin, la Figure 14 contient plusieurs graphiques représentant pour des lettres aléatoirement choisies la répartition des distances avec toutes les autres lettres. Cela permet de savoir si on a ou non dans la plupart des cas quelques lettres très proches et beaucoup de très éloignées.

Nous avons produit ces graphiques avec les deux mesures de distances et toutes les représentations vectorielles existantes pour aider à choisir la méthode la plus pertinente. Il s'est avéré que la distance cosinus donnait toujours des résultats plus satisfaisants. Pour les représentations vectorielles, les TF, TF-IDF et LSA n'ont pas donné de très bons résultats sur la répartition des distances tandis NMF et LSA avaient une heatmap avec des contrastes moins marqués. La LDA s'est révélée bonne sur les deux graphiques.

4.3 Thèmes identifiés par les représentations vectorielles

Pour évaluer la qualité de la représentation vectorielle, nous avons choisi d'afficher les caractéristiques des dimensions trouvées par les différents algorithmes. Si on retrouve dans une même dimension des mots qui paraissent avoir un sens à se retrouver ensemble, alors on peut accorder une certaines confiance à cette représentation. On peut observer sur le tableau 15 pour chaque type de représentation les 10 mots les plus importants (avec la plus grande valeur) dans 5 des topics identifiés.

Même si les mots d'une même dimension ne semblent pas complètement sans rapport, c'est à dire qu'il peut exister un sujet dans lequel ils sont employés ensemble, il est assez difficile de savoir s'ils sont représentatifs ou non. Il est aussi difficile de distinguer une représentation plus pertinentes qu'une autre.

4.4 Graphe de députés

À partir des clusters, nous avons déduit des liens entre les députés les plus proches en terme d'envoi de lettres sur le même sujet. Plusieurs approches, qui dépendent de l'information que nous voulons capturer, sont possibles pour déterminer une mesure de la proximité entre deux députés. Nous avons ici choisi une méthode très simple et représenté le principe suivant : deux députés sont proches si ils envoient beaucoup de lettres proches, c'est à dire qui sont classées dans le même cluster. Le calcul effectué pour

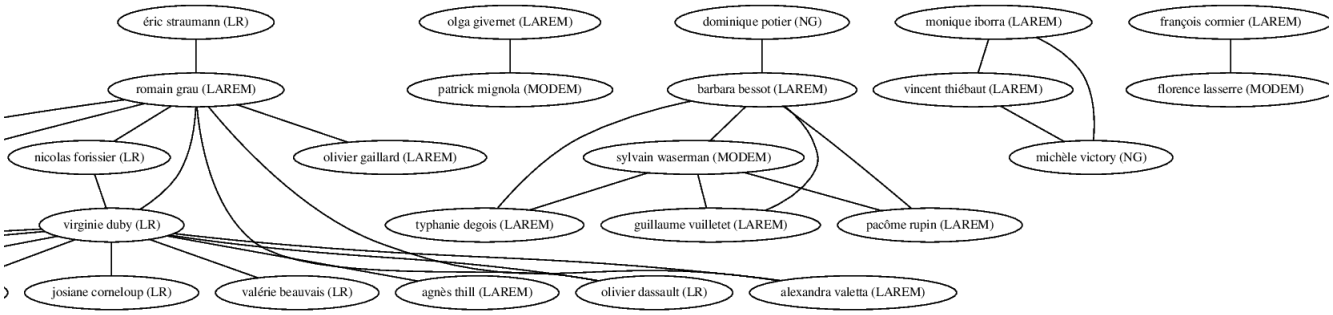


Figure 11: Graphe représentant les députés liés à condition qu'ils aient un seuil supérieur à **0.015**

un score de proximité est donc le nombre de fois où des lettres des députés sont dans le même cluster, divisé par la somme de cette valeur avec tous les autres députés :

$$Q(a_1, a_2) = \sum_{i \in I(a_1)} \sum_{j \in I(a_2)} C(i) = C(j)$$

$$score(a_1, a_2) = \frac{Q(a_1, a_2)}{\sum_{a \in A} Q(a_1, a)}$$

avec Q le score de quantité, a_1 et a_2 les deux députés auteurs, $I(a)$ l'ensemble de lettres de a, $C(i)$ l'indice du cluster i, et A l'ensemble des auteurs.

La normalisation a l'avantage d'éviter que les individus écrivant beaucoup de lettres aient un lien de valeur élevée avec tout le monde, et donne un score compris entre 0 et 1. Le problème est que cette valeur n'est pas symétrique.

En affichant le graphe des députés ayant un score plus élevé qu'un seuil choisi arbitrairement à **0.015**, on obtient un graphe de députés dont une partie est montrée sur la Figure 11.

5 Impacts sociétaux

Le travail effectué pendant ce stage rejoint, en terme d'impacts sur la société, les travaux qui visent à appliquer de nouvelles technologies de l'information à des organes démocratiques. Les aspects positifs, intentionnels, qui sont la mise à disposition d'information utiles aux citoyens, ont été présentés dans la Section *Impacts sur la société* de notre rapport de projet industriel ([ABD+20]). Cependant, l'utilisation massive ou bien à mauvais escient de ces technologies pourraient aussi vulnérabiliser notre structure démocratique. Le fonctionnement de l'assemblée nationale, que ce soient dans la forme du déroulement d'une séance ou dans les modes de régulations des groupes d'intérêts, a été pensé dans un contexte technologique précis. Bousculer ce contexte avec de nouvelles méthodes rendant disponibles des informations autrefois difficiles à obtenir, pourraient déstabiliser les mécanismes démocratiques. Nous avons pu voir pendant ce stage des algorithmes capables de regrouper toutes les lettres parlant du même sujet, et de faire automatiquement des liens entre des députés. Des technologies comme l'analyse de sentiments pourraient aussi permettre d'identifier des députés favorables à certains arguments, ou sensibles à certaines causes. Si ces travaux peuvent être utiles à tous les citoyens, il est aussi possible que des groupes d'intérêts s'emparent les premiers de cette mine d'information, qui faciliterait leur travail d'influence en ciblant mieux les députés et aux meilleurs moments. Les groupes d'intérêts disposeraient ainsi d'un atout contre lequel l'assemblée nationale et les législations seraient plus lentes à réagir. Ainsi, même si notre étude a été effectuée sans aucune dépendance à une entreprise, le développement de ce type de recherche pourrait être utilisé à mauvais escient, et il faudrait alors que des organes comme l'assemblée nationale s'adapte au contexte technologique pour rester fonctionnels.

6 Conclusion et perspectives

Nous avons pu analyser et essayer une bonne diversité de méthodes pour la représentation vectorielle en faible dimension de textes courts, et pour le clustering de ces textes. S'il nous a manqué de temps pour nous aventurer dans des méthodes plus élaborées comme celles citées plus haut, les résultats obtenus pour permettre d'identifier des groupes sont déjà beaucoup plus précis pour distinguer des sujets que la classification en rubriques arbitrées par l'administration.

Beaucoup d'autres méthodes auraient pu être utilisées pour afin cette étude, mais nous avons préféré faire aboutir les plus simples plutôt que de toutes de les analyser en détail et de les implémenter, devant la courte période du stage. Le Deep Clustering [HSDD19] en est un exemple, mais nous en avons rencontré d'autres au cours de nos recherches, comme la méthode BERT ([DCLT19]), une autre approche Deep très performante pour une grande quantité de tâches, qui fait appel au concept d'attention ([VSP+17]). Le Clustering Ensemble ([WW07], [SG03]), dont l'idée est de combiner un grand nombre de partitions faibles, obtenues à l'aide de méthodes différentes et peu discriminantes, pour en déduire une partition forte, qui aurait pris en compte une grande diversité d'informations. Le spectral clustering, résumé et comparée aux kernel k-means dans [DGK04], est une méthode de clustering qui s'appuie sur la matrice des affinités. Celle-ci peut être calculée avec n'importe quelle mesure de similarité, ce qui nous aurait permis d'utiliser la distance cosinus. Beaucoup d'autres critères d'évaluation de modèles de langage auraient pu être utilisés (voir [Des17]), et nous aurions aimé voir la perplexité implémentée pour chacun

des modèles essayés.

Ce stage a largement répondu à mes attentes car j'ai pu y apprendre énormément de choses tant sur le Traitement Automatique du Langage Naturel et le Clustering que sur les aspects pratiques de la recherche.

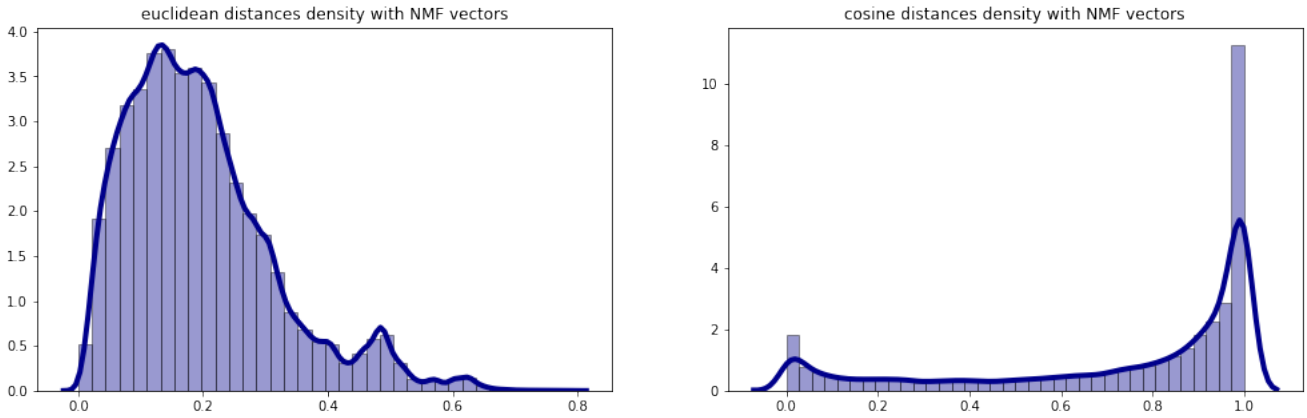


Figure 12: Répartition des distances sur une représentation NMF

7 Annexes

7.1 Distances

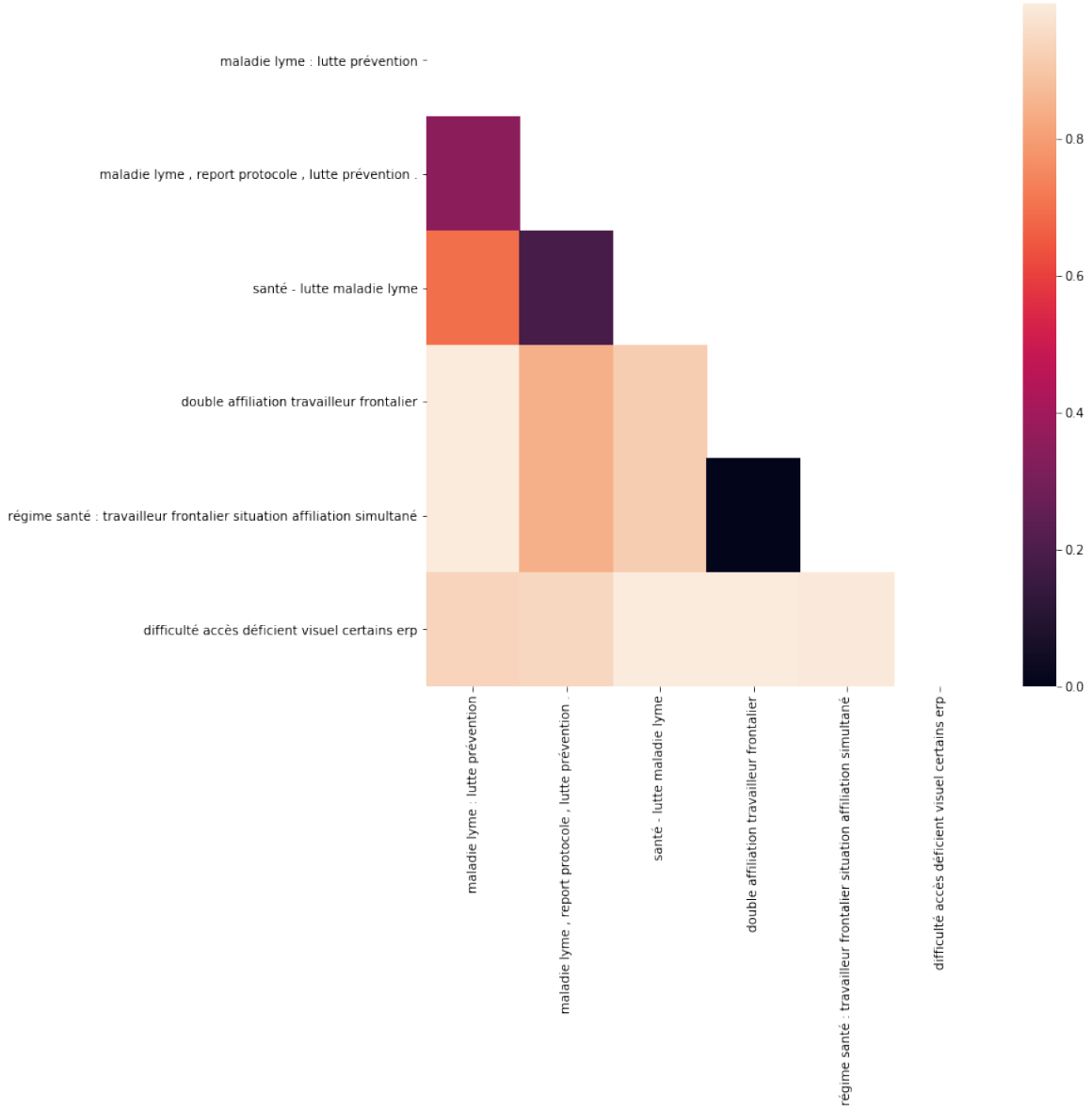


Figure 13: Heatmap des distances du cosinus sur une représentation LDA pour quelques lettres choisies. Le calcul des distances a été efficace car les cases foncées (resp. claires) désignent bien des lettres sémantiquement proches (resp. éloignées)

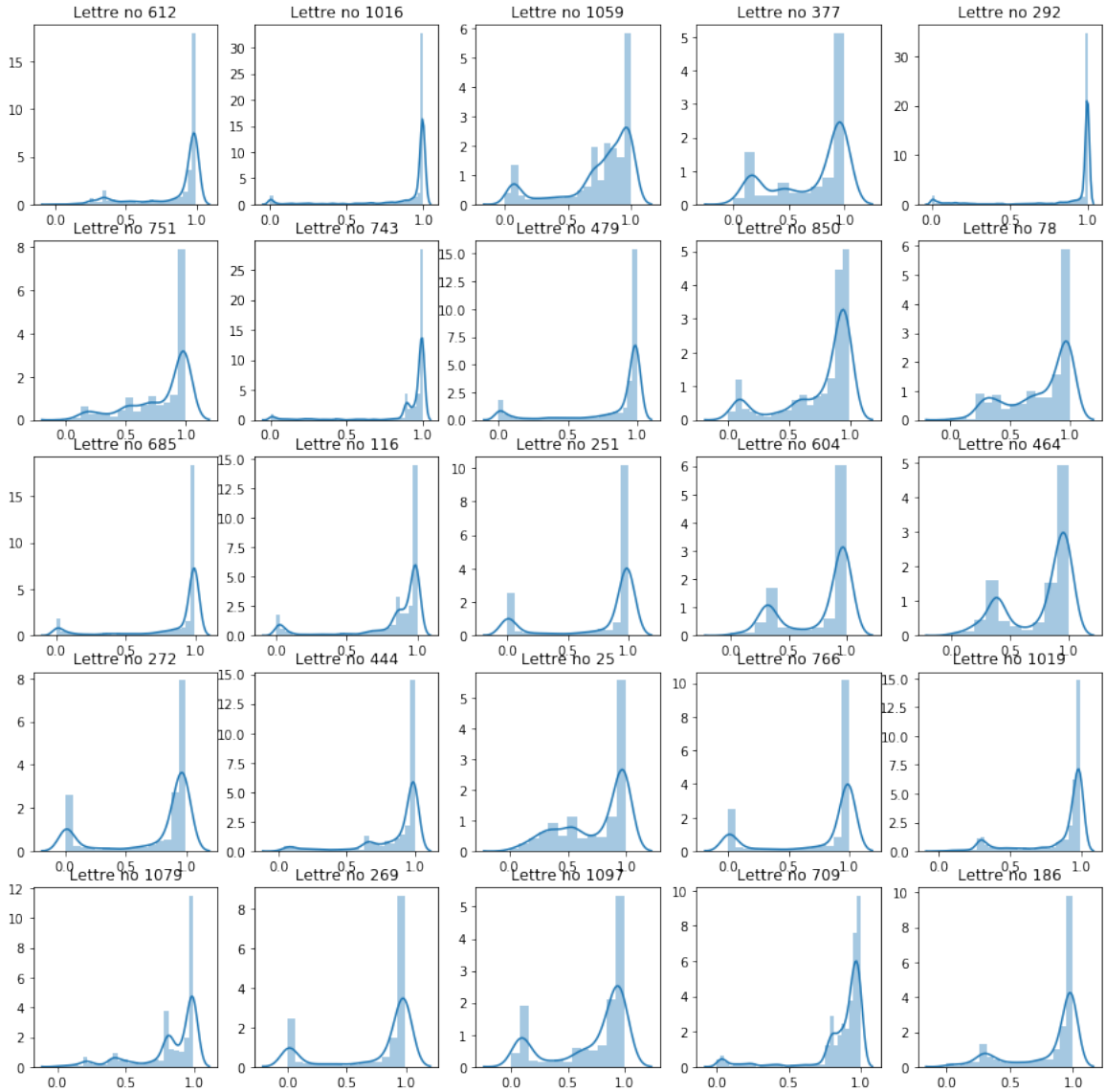


Figure 14: Répartitions des distances avec les autres lettres pour quelques lettres en représentation LDA avec la distance cosinus. On peut souvent y remarquer les deux pics au début et à la fin.

Vecteur	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
ACP	maladie, santé, entreprise, pouvoir, charge, mme, prise, public, demande, ministre	maladie, santé, lyme, prise, patient, charge, cancer, malade, atteindre, fibromyalgie	entreprise, cancer, salarié, pourcent, économie, pme, euro, finance, commerce, compte	cancer, enfant, pédiatrique, recherche, école, dépistage, public, financement, décès, euro	école, compte, administration, million, euro, impayer, enseigner, denier, crédibilité, déficit
LSA	maladie, entreprise, santé, pouvoir, charge, mme, public, demande, prise, français	maladie, santé, lyme, prise, patient, charge, cancer, malade, atteindre, fibromyalgie	carte, gris, ant, usager, préfecture, service, titre, site, délai, sécuriser	cancer, enfant, pédiatrique, recherche, école, dépistage, public, financement, décès, euro	école, compte, administration, million, euro, impayer, enseigner, denier, crédibilité, déficit
LDA	tabac, cigarette, commune, logiciel, buraliste, tiers, prix, paquet, retrait, public	boulangerie, journée, administratif, paneterie, frontalier, tabac, effet, dispositif, gérant, hausse	art, métier, loi, roumanie, exercice, lme, fromager, oniam, examiner, lyme	commerce, jour, autisme, entreprise, plan, mélanome, 2015, maladie, 2017, demande	gris, carte, service, retraité, délai, ligne, plateforme, sécuriser, demande, dématérialisation
NMF	000, pajemploi, pajot, pakistan, palais, palette, palier, palliatif, pallier, palmarès	charge, prise, patient, santé, atteindre, maladie, traitement, solidarité, autorité, soin	carte, gris, véhicule, problème, intérieur, demande, service, particulier, mois, agence	cancer, pédiatrique, recherche, enfant, financement, 500, adolescent, allouer, leucémie, maladie	école, administration, compte, million, euro, impayer, enseigner, denier, crédibilité, gage

Figure 15: 10 mots les plus importants des 5 premiers topics pour chaque représentation vectorielle

7.2 Thèmes identifiés par les représentations vectorielles

7.3 Graphiques des clustering

Nous avons appliqués les algorithmes de clustering présentés sur chacune des représentations vectorielles, toujours sur le corpus réduit. Les 6 graphiques des pages suivantes représentent :

Critères de qualité de clustering pour un k-means pour différents k	Umap des clusters trouvés avec K-means
Distances de linkage du clustering hiérarchique	Umap des clusters trouvés avec X-means
Dendrogramme du clustering hiérarchique	Umap des clusters trouvés avec le clustering hiérarchique

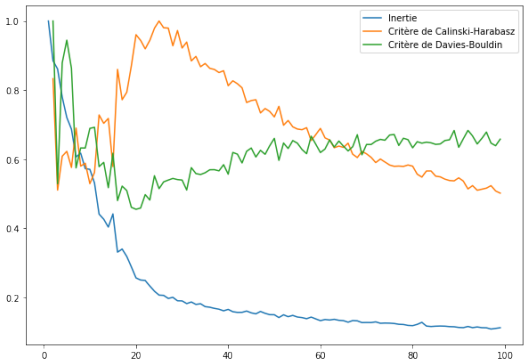
Les arguments entre parenthèse pour les algorithmes signifient :

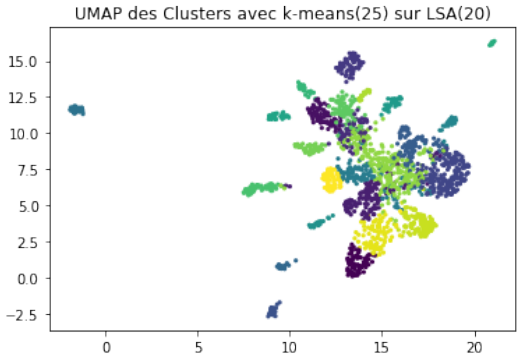
- k-means(k) : K-means avec k clusters
- (k)x-means($min : max$) : X-means entre min et max ayant obtenu k clusters
- Agg(t) : Agglomerative clustering avec le threshold (seuil) t.

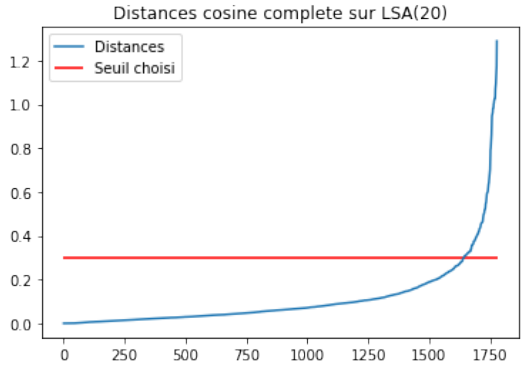
Et pour les représentations vectorielles, les nombres entre parenthèse indiquent le nombre de dimensions.

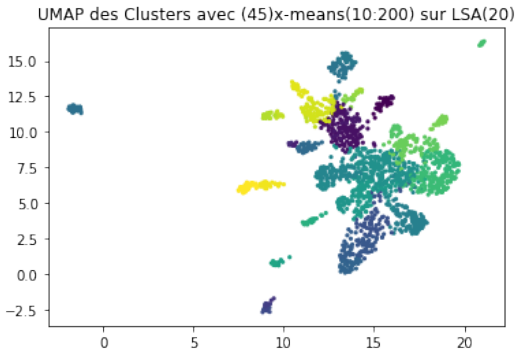
7.3.1 LSA

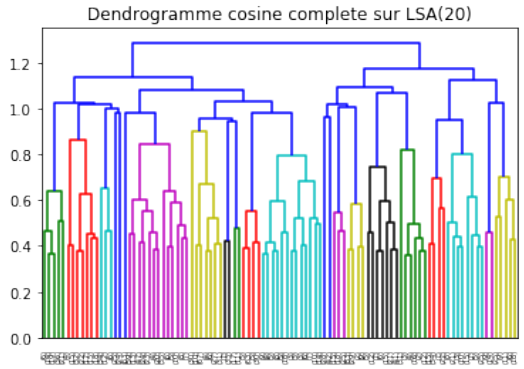
	K-MEANS	X-MEANS	HAC
Nombre de clusters	25	45	83
Inertie	47.15	35.97	
Critère de Calisnki-Harabasz	283.32	211.29	
Critère de Davies-Bouldin	1.14	1.13	
Seuil			0.3

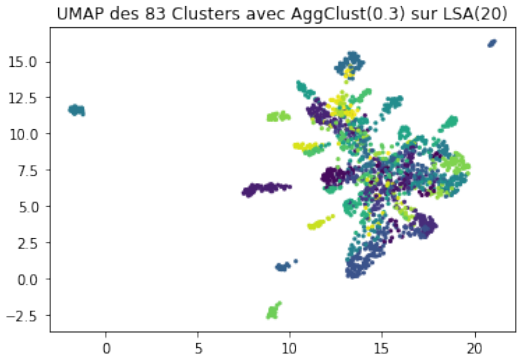






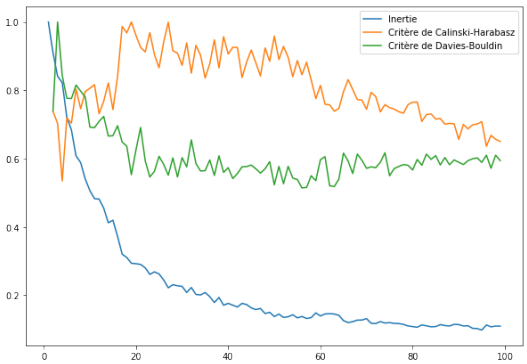


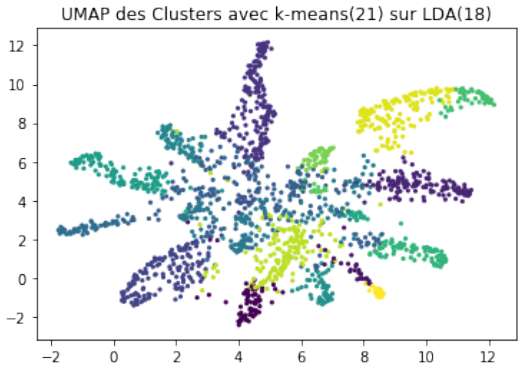


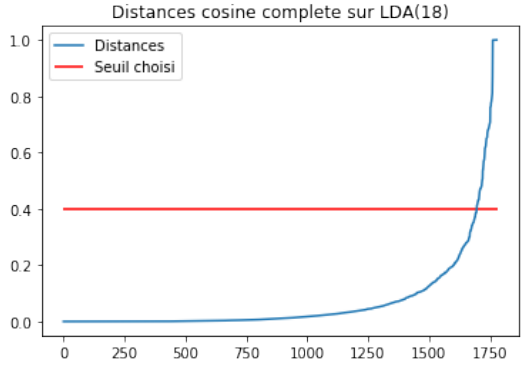


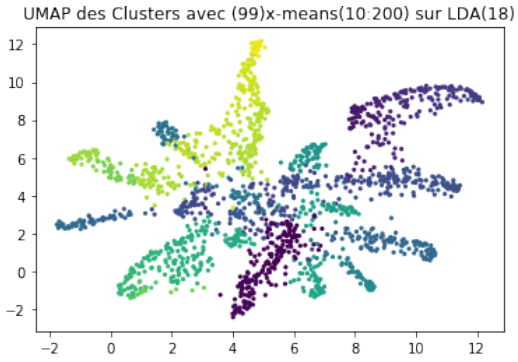
7.3.2 LDA

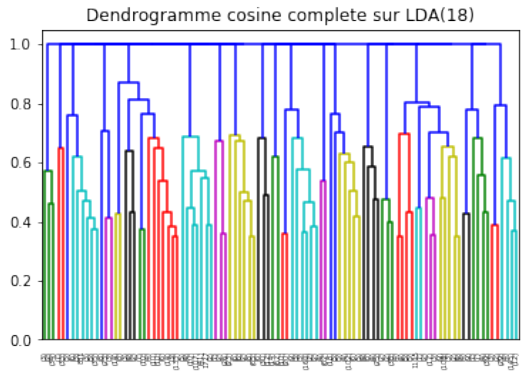
	K-MEANS	X-MEANS	HAC
Nombre de clusters	21	99	84
Inertie	5363935.39	1985900.63	
Critère de Calisnki-Harabasz	221.96	142.50	
Critère de Davies-Bouldin	1.29	0.78	
Seuil			0.4

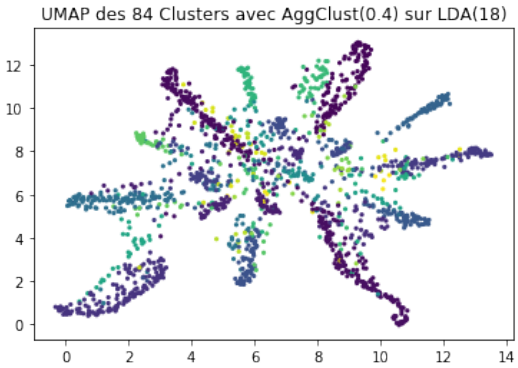






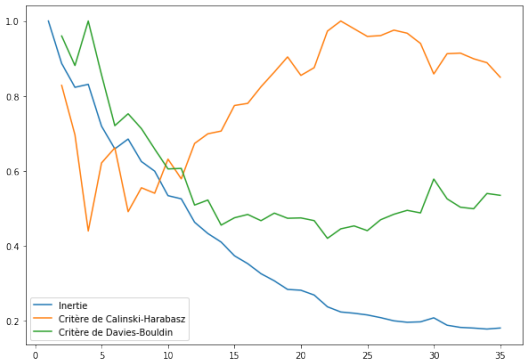


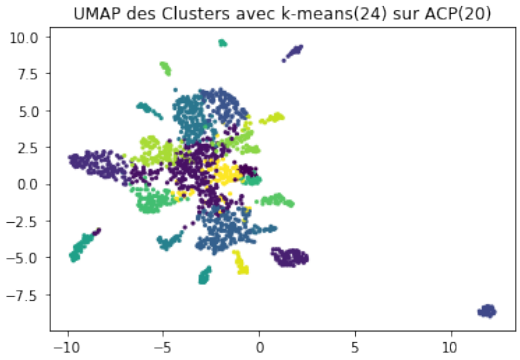


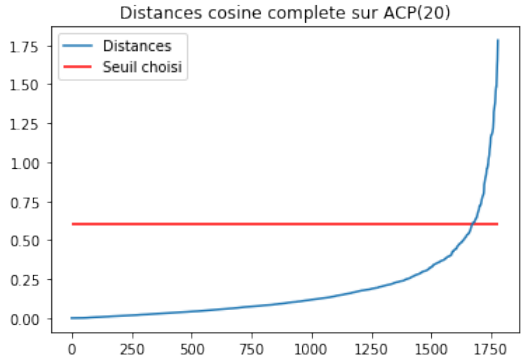


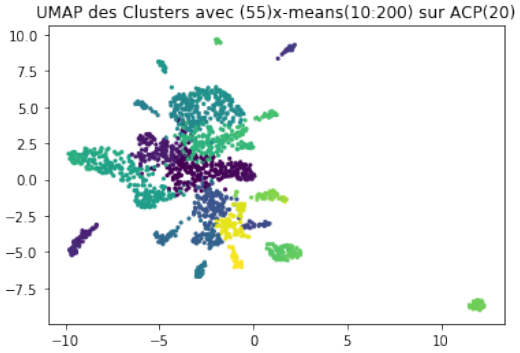
7.3.3 ACP

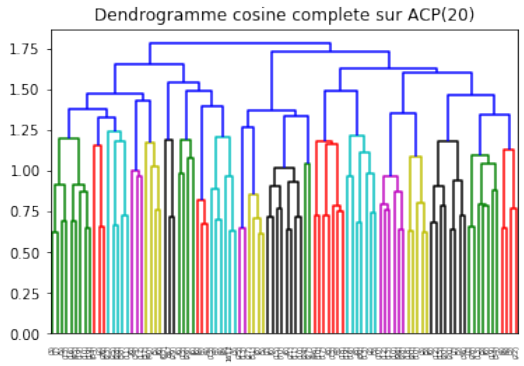
	K-MEANS	X-MEANS	HAC
Nombre de clusters	24	55	109
Inertie	151.05	33.51	
Critère de Calisnki-Harabasz	273.95	189.38	
Critère de Davies-Bouldin	1.07	1.16	
Seuil			0.6

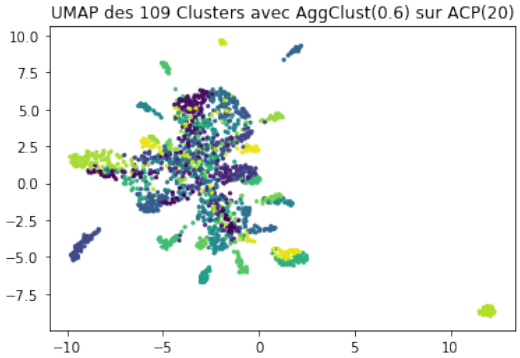






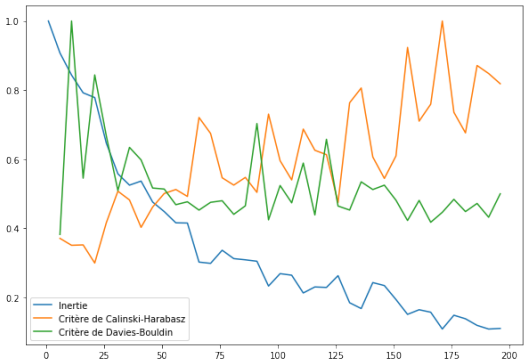


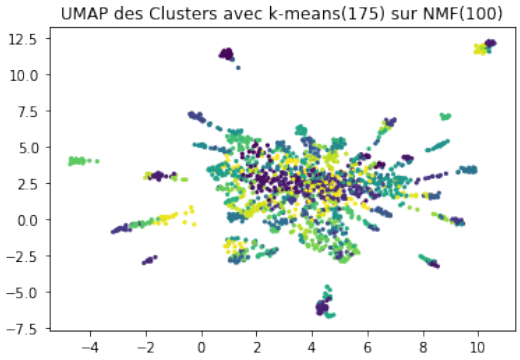


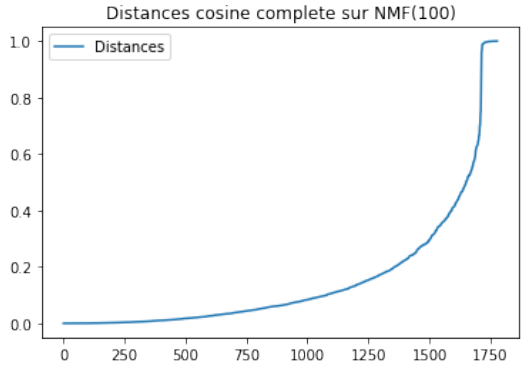


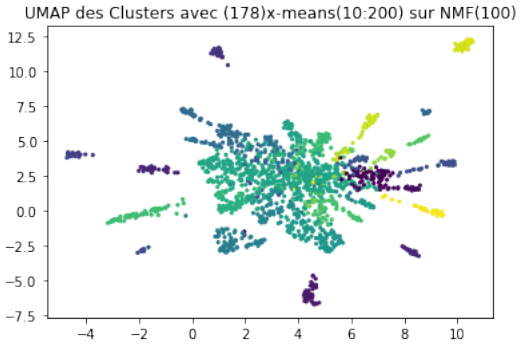
7.3.4 NMF

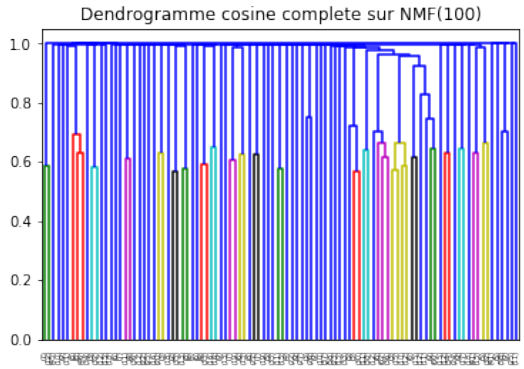
	K-MEANS	X-MEANS	HAC
Nombre de clusters	175	178	100
Inertie	16.39	10.62	
Critère de Calisnki-Harabasz	81.95	101.76	
Critère de Davies-Bouldin	1.18	0.61	
Seuil			Fixer le nombre de clusters à 100

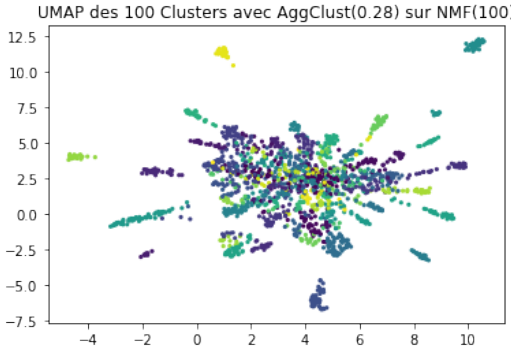












Pour le HAC, nous avons ici préféré fixer le nombre de clusters à 100, plutôt que d'utiliser la méthode du seuil.

References

[ABD+20] Samy Asma, Achille Baucher, Homer Durand, Thomas Genin, and Ibtissam Lachhab. Étude sur le renouveau politique en france. Rapport de Projet Industriel MAIN4 Polytech Sorbonne, 2020.

[BNJ01] David Blei, Andrew Ng, and Michael Jordan. In *Latent Dirichlet Allocation*, volume 3, pages 601–608, 01 2001.

[CH74] T. Caliński and J Harabasz. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

[DB79] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[DCLT19] J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[Des17] Bernard Desgraupes. Clustering indices. `https://cran.r-project.org/web/packages/clusterCrit/vignettes/clusterCrit.pdf`, Novembre 2017.

[DGK04] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means, spectral clustering and normalized cuts. *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 551-556, 07 2004.

[Fal19] Joris Falip. *Structuration de données multidimensionnelles : une approche basée instance pour l'exploration de données médicales.* PhD thesis, Université de Reims Champagne-Ardenne, 2019. Thèse de doctorat dirigée par Herbin, Michel et Blanchard, Frédéric.

[GCLL10] Maha Ghribi, Pascal Cuxac, Jean-Charles Lamirel, and Alain Lelu. Mesures de qualité de clustering de documents : Prise en compte de la distribution des mots clés. In Nicolas Béchet, editor, *10ième Conférence Internationale Francophone sur l'Extraction et la Gestion des Connaissances - EGC 2010*, Hammamet, Tunisia, January 2010.

[HSDD19] Amir Hadifar, Lucas Sterckx, Thomas Demeester, and Chris Develder. A self-training approach for short text clustering. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 194–199, Florence, Italy, August 2019. Association for Computational Linguistics.

[MH08] L. V. D. Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[MHSG18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3:861, 09 2018.

[PM02] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. *Machine Learning*, January 2002.

[SG03] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3(null):583–617, March 2003.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, L. Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

[WW07] Silke Wagner and Dorothea Wagner. Comparing clusterings - an overview. *Technical Report 2006-04*, 01 2007.

[XLG03] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, pages 267–273, January 2003.

[YZO10] Zhirong Yang, Zhanxing Zhu, and Erkki Oja. Automatic rank determination in projective nonnegative matrix factorization. In Vincent Vigneron, Vicente Zarzoso, Eric Moreau, Rémi Gribonval, and Emmanuel Vincent, editors, *Latent Variable Analysis and Signal Separation*, pages 514–521, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.