2019/2020

# BUREAU D'ETUDE

## CLUSTERING AUTOMATIQUE DE TEXTE

Magistère 1ère année
Informatique S6

Amani Alexandra
Djétodjidé Golbé Succès
Lebrun Claire
She Huixin

Tuteur : Muller Philippe

Le grand débat de 2019 est un débat français mis en place par le gouvernement dans le but d'engager le dialogue et de s'informer sur les attentes des citoyens. Afin que l'Etat Français puisse envisager des solutions, il est intéressant de relever les avis et sujets qui ressortent le plus des réponses aux problématiques majeures.

Le but de cette étude est d'analyser automatiquement les contributions libres du grand débat politique de 2019, afin de construire des catégories de contributions. Pour répondre à ce sujet, nous avons utilisé plusieurs outils de *Text Mining* à l'aide de Python. Nous avons à notre disposition un ensemble de réponses libres divisée en quatre grands thèmes : la fiscalité et les dépenses publiques, l'organisation de l'Etat des services publics, la transition écologique, la démocratie et la citoyenneté.

# I/ Description des outils théoriques utilisés

Pour analyser les réponses, nous allons procéder en plusieurs étapes durant lesquelles nous allons utiliser différents outils théoriques.

Notation : Chaque thème correspond à un corpus. Un corpus est composé de plusieurs réponses que l'on appellera texte.

## 1. Tokenisation

La tokenisation d'un texte permet de décomposer le texte en une liste de mots. Dans notre cas, c'est la première étape à faire avant d'analyser le vocabulaire. La tokenisation est utile pour analyser les mots d'un texte. Cet outil ne permet cependant pas à supprimer les éléments de ponctuation mais permet de les repérer plus facilement.

## 2. Lemmatisation

Tout d'abord nous avons besoin d'un texte prétraité afin d'effectuer des analyses sur les mots qui ont le plus de sens. Pour cela nous allons appliquer un outil de lemmatisation à chaque corpus. Cela consiste à associer à chaque mot sa forme normalisée. La forme normalisée d'un mot (ou forme canonique) correspond à sa forme de base par exemple pour un verbe son infinitif et pour un nom son masculin singulier. L'idée étant de ne conserver que le sens des mots utilisés dans le corpus.

Pour cette étape, nous utilisons le package « SpaCy », librairie de traitement automatique de langage.

Une fois que nous avons lemmatisé les corpus, nous allons pouvoir sélectionner les mots qui n'ont pas de sens, on les appellera les mots vides. Cela correspond aux déterminants, pronoms, auxiliaires ou encore mots de liaisons qui n'apportent pas de valeur informative et que l'on peut maintenant repérer grâce à la lemmatisation. Ceux-ci sont très fréquents et ralentissent le travail, nous allons alors les supprimer.

## 3. TF-IDF (Term-Frequency - Inverse Document Frequency)

Pour répondre à notre problème, nous avons besoin de repérer des mots pertinents de chaque corpus. Pour cela il existe plusieurs méthodes. Parmi elles, le TF IDF est une méthode de pondération qui ne considère pas le poids d'un mot dans un corpus comme sa fréquence d'apparition uniquement, mais pondère cette fréquence par un indicateur qui indique si ce mot est commun ou rare dans tous les textes.

Cet outil consiste à calculer deux indices pour chaque mot :

Le **TF** *(term frequency)* : indique la fréquence d'apparition du mot dans un texte.

$$TF_{mot,texte} = \frac{nombre\ d'apparitions\ du\ mot\ dans\ le\ texte}{nombre\ de\ mots\ dans\ le\ texte}$$

L'**IDF** (*Inverse Document Frequency) :* mesure l'importance d'un mot dans le corpus, calculé comme ceci :

$$IDF_{mot} = \log\left(\frac{nombre\ total\ de\ textes}{nombre\ de\ texte\ qui\ comportent\ le\ mot}\right)$$

On obtient un indice

$$TF\ IDF_{mot,texte} = TF \times IDF$$

Plus le TF IDF d'un mot est grand, plus ce mot est pertinent dans le corpus. Grâce à cet outil, nous avons pu sélectionner la liste des mots pertinents dans chaque corpus.

## 4. Vectorisation et similarité cosinus.

La vectorisation d'un texte consiste à créer une matrice qui, pour chaque texte, associe à chaque mot pertinent son TF IDF. Si le mot n'apparait pas dans un texte, on entre alors la valeur zéro. On peut créer un vecteur de fréquences ou de TF IDF. Dans ce projet, comme nous avons sélectionné comme indice de pondération le TF IDF, mais il est aussi possible de le faire avec la fréquence d'apparition des mots

On obtient une matrice de la forme (par exemple) :

*Tableau 1: exemple de matrice de vectorization d'un texte*

| Texte/Mot | Impôt | Taxe | Revenu | Payer | Public | Entreprise | Aide | Santé |
|---|---|---|---|---|---|---|---|---|
| **1** | 0 | 2,3 | 0 | 7,5 | 2,1 | 0 | 0 | 0,2 |
| **2** | 1,1 | 0,7 | 0 | 0 | 0 | 1,7 | 0 | 0 |
| **3** | 1 | 4,6 | 9,1 | 8,2 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

A l'aide de cette matrice, nous pouvons alors calculer la similarité cosinus. Ce calcul sert à analyser les ressemblances entre les textes. Pour calculer la similarité du cosinus on doit calculer le produit scalaire des deux vecteurs (calculés ci-dessus) et le diviser par le produit des normes des deux vecteurs comme suit :

$$\cos\ (doc1, doc2) = \frac{\langle doc1, doc2 \rangle}{\|doc1\|.\|doc2\|}$$

On peut alors déduire si deux textes sont semblables ou non : une valeur proche de 0 indique une indépendance des deux textes tandis qu'une valeur proche de 1 indique des textes plus similaires. Les valeurs intermédiaires permettent d'évaluer le degré de similarité entre les deux textes.

La distance cosinus qui indique le degré de différence entre deux textes se déduit de ce calcul :

$$distance\cos(doc1, doc2) = 1 - similarité\cos((doc1, doc2))$$

Dans ce projet, nous allons utiliser la similarité cosinus pour analyser la ressemblance entre deux textes.

## 5. Le clustering et l'algorithme Kmeans.

Pour trouver les catégories de contributions, nous allons appliquer l'algorithme Kmeans à nos textes. Cette méthode de clustering consiste à laisser l'algorithme classer nos textes selon leur ressemblance. Un cluster correspond à un groupe de textes qui se ressemblent, il possède un centroïde.

Le fonctionnement de clustering avec K Means est itératif en quatre étapes. Imaginons que l'on souhaite regrouper nos données en k clusters :

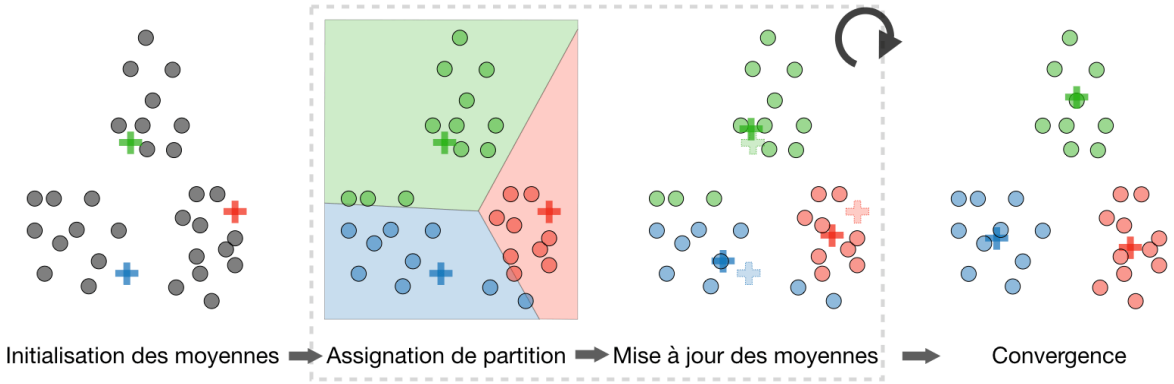



*Figure 1: illustration de la méthode K means*

**1ère étape** : On définit aléatoirement k vecteurs qui seront les centroïdes de nos clusters.

**2ème étape** : Dans cette étape, on associe chaque texte à son centroïde le plus proche. Pour cela on calcule la distance cosinus entre le texte et chaque centroïde et on affecte le texte à celui qui est le plus proche.

**3ème étape** : Une fois chaque texte associé à un centroïde, on recalcule la position du centroïde. On déplace le centroïde au milieu du cluster en calculant la moyenne des points.

**4ème étape** : On recommence les étapes 2 à 3 avec les nouveaux centroïdes. On s'arrête lorsque les centroïdes convergent vers une position d'équilibre.

Pour trouver le bon nombre de clusters, on utilise « Méthode de coude » qui consiste à tracer l'évolution de la somme des distances des textes au centroïde le plus proche en fonction du nombre de clusters. Dans ce graphique, on peut trouver un coude qui est le nombre de clusters optimal.

# II/Choix de conception et description des algorithmes

## Partie 1 : Traitement des fichiers

Dans cette partie, nous avons utilisé deux principales fonctions dans le but de nettoyer le texte.

**Lemmatisation**

On utilise la fonction “lemma” du package SpaCy qui traite le texte de la manière suivante : pour chaque mot dans le corpus, cette fonction lui associe son type (déterminant, verbe, adjectif etc.) et sa forme normalisée. La fonction renvoie pour chaque texte une liste des mots associés à leur type et à leur forme normalisée.

**Suppression des mots**

Cette fonction permet de supprimer des corpus les mots qui n’apportent pas de sens (aussi appelés les “stopwords”). Pour cela, il a fallu décider quels mots devaient être supprimés. Nous avons remarqué mots sans sens sont par exemple les déterminants, les adverbes, les pronoms, les chiffres ou encore les auxiliaires. En fait, ce sont des “mots vides”, un mot vide est un mot communément utilisé dans un langage. Ceux-ci apparaissent très souvent dans un texte, ils ne permettent pas de discriminer les documents

Nous avons utilisé deux outils :

- Premièrement, la fonction de lemmatisation précédente pour sélectionner les mots qui faisaient partis de types énoncés ci-dessus et nous les avons retirés du corpus
- Ensuite, une liste de “mots vides”. Dans chaque texte, pour chaque mot, si celui-ci est dans la liste des mots vides, on le supprime.

Cette fonction renvoie une liste de listes (liste de textes) qui ne continent que les mots qui apportent du sens dans le corpus.

## Partie 2 : Analyse de texte

Dans cette partie, nous avons construit un ensemble de fonctions dans le but d’analyser les textes des corpus avec le TF-IDF

**Fréquence d’apparition d’un mot**

Cet algorithme prend en entrée un mot et un texte. Il renvoie en sortie la fréquence d’apparition de ce mot dans le texte. Nous utilisons l’attribut « .count » dans la liste des mots du texte pour compter le nombre de fois que le mot apparait

**Stockage des fréquences : TF**

Dans cette fonction, on calcule et on stocke pour chaque texte les mots et leur fréquence (qui correspond au TF).

Pour cela, on utilise la fonction décrite ci-dessus. Pour chaque texte, on calcule la fréquence des mots qui le constituent et on les stocke dans un dictionnaire. Cette fonction prend donc un corpus en entrée et renvoie une liste de dictionnaire, chaque dictionnaire correspond à un texte.

**Calcul de l'IDF**

Cette fonction sert à calculer l'IDF de tous les mots d'un corpus mis en paramètre et renvoie un dictionnaire qui a pour clé le mot et pour valeur son IDF.

Tout d'abord, on crée une liste contenant tous les mots du corpus en évitant les doublons. Ensuite pour chaque mot dans cette liste de mots on compte le nombre de texte dans le corpus qui le contient. Après, pour chaque mot on applique la formule de l'IDF décrite plus haut :

$$IDF_{mot} = \log\left(\frac{nombre\ total\ de\ textes}{nombre\ de\ texte\ qui\ comportent\ le\ mot}\right)$$

Ensuite nous donnons des scores à chaque mot pour pouvoir favoriser les mots qui reviennent le plus souvent. Pour chaque mot d'un texte on multiple son IDF par le nombre de fois qu'il revient dans chaque texte du corpus.

**Calcul du TF-IDF**

A l'aide des deux dernières fonctions, on calcule le TF-IDF de chaque mot et pour chaque texte. Pour cela on crée une liste de dictionnaire. Chaque dictionnaire correspond à un texte. Les clés des dictionnaires sont les mots et les valeurs correspondent au produit de leur TF et de leur IDF

## Partie 3 : Nuage de mots

Pour cette partie, on utilise le package « *word cloud* ». Celui- ci crée un nuage de mot permettant de visualiser les mots importants d'un ensemble de textes. Nous appliquons la fonction *word cloud* à notre corpus nettoyé. Pour cela, nous avons créé un document comprenant l'ensemble de nos textes nettoyés. Il est nécessaire de définir plusieurs paramètres de notre nuage de mots comme le nombre de mots que l'on veut afficher dans notre nuage de mot, la police maximale et la police minimale, la couleur de fond, la couleur des mots ou encore appliquer un masque. Les masques permettent que les mots prennent la forme d'une image choisie. Par exemple, nous pouvons appliquer la forme de la France (exemple avec le corpus sur le thème de la fiscalité) :


*Figure 2 : exemple de nuage de mots avec le corpus fiscalité*

## Partie 4 : Distance entre texte

**Motspluspertinents**

Nous cherchons les n mots plus pertinents en nous basant sur leur IDF. Plus l'IDF du mot est grand plus il est pertinent. Le dictionnaire comportant les mots d'un corpus en clés et leur IDF comme valeurs sont les entrées. On range en fonction des IDF les éléments du dictionnaire en ne gardant que les n premiers. On conserve dans une liste les n mots (clés).

Sortie : une liste de n mots ayants les plus grands IDF du corpus.

**Vectorizer TF IDF**

L'algorithme vectorizer TFIDF transforme un texte en un vecteur de mots plus pertinents. Après avoir calculé les IDF, TF, TFIDF du texte, on effectue une boucle de la longueur du dictionnaire de TF IDF dans laquelle pour chaque index i :

Pour chaque mot pertinent :

- si le mot est la clé d'index i du dictionnaire, on conserve son TFIDF dans une liste, sinon on lui attribue 0 comme TFIDF. On conserve cette liste dans un dictionnaire que si elle est différente du vecteur nul (pour éviter les textes n'ayant aucun mot pertinent).

Sortie : une transformation en tableau du vecteur comportant les listes des mots pertinents (en colonne) et leur TFIDF dans chaque texte du corpus (en ligne).

**Similarité_cosinus**

Cet algorithme nous permet de calculer la similarité cosinus de deux textes. Nous avons en entrées les vecteurs fréquences des deux textes. Nous calculons la similarité cosinus qui est le produit scalaire des deux vecteurs et le diviser par le produit des normes des deux vecteurs. Nous ne pouvons calculer cette distance cosinus que si les vecteurs sont de mêmes longueurs.

**Maxsimilaire**

Cet algorithme nous permet d'attribuer à un texte un autre texte du corpus avec lequel il a la plus grande similarité cosinus. Nous commençons d'abord par calculer la similarité cosinus entre ce texte et tous les autres textes du corpus, ensuite nous rangeons par ordre décroissant les différents textes et leurs similarité cosinus avec le texte mis en entrée, et en fin renvoyons en sortie la plus grande similarité cosinus et l'indice du texte associé à celle-ci.

## Partie 5 : Classification non supervisée

**Classification avec l'algorithme du k-mean**

Comme le résultat final change en fonction des centroïdes de départ, nous allons faire plusieurs essais (nombre de fois d'essai est de « N ») sur les différents centroïdes pour trouver la meilleure classification parmi les essais. Pour répéter N fois, nous faisons une boucle de range(N) :

- Étant donné que nous devons classer les textes dans K clusters, nous définissons K centroïdes aléatoirement avec les coordonnées comprises dans la plage des limites des coordonnées des vecteurs de texte et avec les mêmes dimensions des vecteurs de texte. Nous conservons les centroïdes dans un dictionnaire indexé par le numéro de cluster.
- Nous faisons une boucle de « while » **:**

- On calcule la similarité cosinus pour chaque point de texte et chaque centroïde
- Nous affectons les textes au cluster dans lequel le centroïde qui lui est le plus proche, c'est-à-dire avec une similarité cosinus la plus grande et on les conserve dans un dictionnaire indexé par le cluster qu'ils appartiennent à.
- Pour chacun des clusters que nous venons de former, nous déplaçons leur centroïde au milieu du cluster en calculant la moyenne des points
- Nous recommençons jusqu'à ce que les centroïdes ne changent pas après une reformation des clusters

- Nous calculons la somme des similarités cosinus des textes au centroïde du cluster où ils appartiennent et nous la conservons dans un dictionnaire comme la clé avec sa valeur qui est une liste de 2 dictionnaires : l'un qui conserve les points de texte pour chaque cluster et l'autre qui conserve les centroïdes.

Après la boucle de range(N), on cherche la plus grande valeur dans les clés du dictionnaire (la plus grande somme des similarités cosinus) et la sortie est la liste qui correspond à cette plus grande valeur.

(On remarque que dans le cas des vecteurs TF-IDF, les similarités cosinus sont comprise entre zéro et un et la similarité cosinus plus proche de 1 indique que les deux textes sont plus similaires.)

**L'algorithme pour trouver le meilleur nombre de cluster**

Nous définissons d'abord un nombre de cluster « N » afin de trouver le meilleur nombre de cluster dans la limite de « N ».

Nous faisons une boucle de « range(N) » :

- Nous faisons d'abord la fonction k-mean que nous avons défini dans l'étape précédente afin de savoir auquel cluster les textes appartiennent
- Nous calculons la somme des similarités des textes au centroïde du cluster où ils appartiennent à et la conservons dans une liste

Nous avons maintenant une liste des sommes des similarités en fonction du nombre de cluster et nous pouvons la visualise dans un graphique.

# III/Quelques statistiques

Dans cette partie, nous allons présenter quelques statistiques sur les textes et les corpus.

## Nombre de mots par corpus.

*Tableau 2 Quelques statistiques sur les textes :*

| Corpus | Fiscalité | Démocratie | Ecologie | Services |
|---|---|---|---|---|
| **Nombre de mots** | 20 193 935 | 25 521 123 | 16 503 290 | 11 701 021 |
| **Nombre mots moyen par texte** | 37 | 25 | 33 | 33 |
| **Nombre mots moyen par texte après nettoyage des mots vides.** | 16 | 13 | 11 | 14 |

### Temps de chargement des fonction principales

Dans cette partie, nous présentons le temps de chargement des principales fonctions de notre projet. Les temps sont calculés pour le corpus concernant la fiscalité

| Fonction | Temps de chargement |
|---|---|
| **Lemmatisation** | 3 sec |
| **Suppression des mots vides** | 1.5 sec |
| **Calcul des TF-IDF (Sur 1000 textes)** | 1 sec |
| **Trouver les mots pertinents (Sur 1000 textes)** | < 1 sec |
| **Vectorisation (Sur 100 mots pertinents)** | 2.34 sec |
| **Calcul de la similarité cosinus (Sur 100 mots pertinents)** | < 1 sec |
| **Calcul des clusters** | 255.3 sec |

## IV/ Exemples d'utilisation de TF-IDF

Nous avons utilisé l'outil TF-IDF permettant le repérage de mots pertinents présenté dans la première partie de ce rapport. Dans cette partie, des exemples d'utilisation du TF IDF sont présentés.

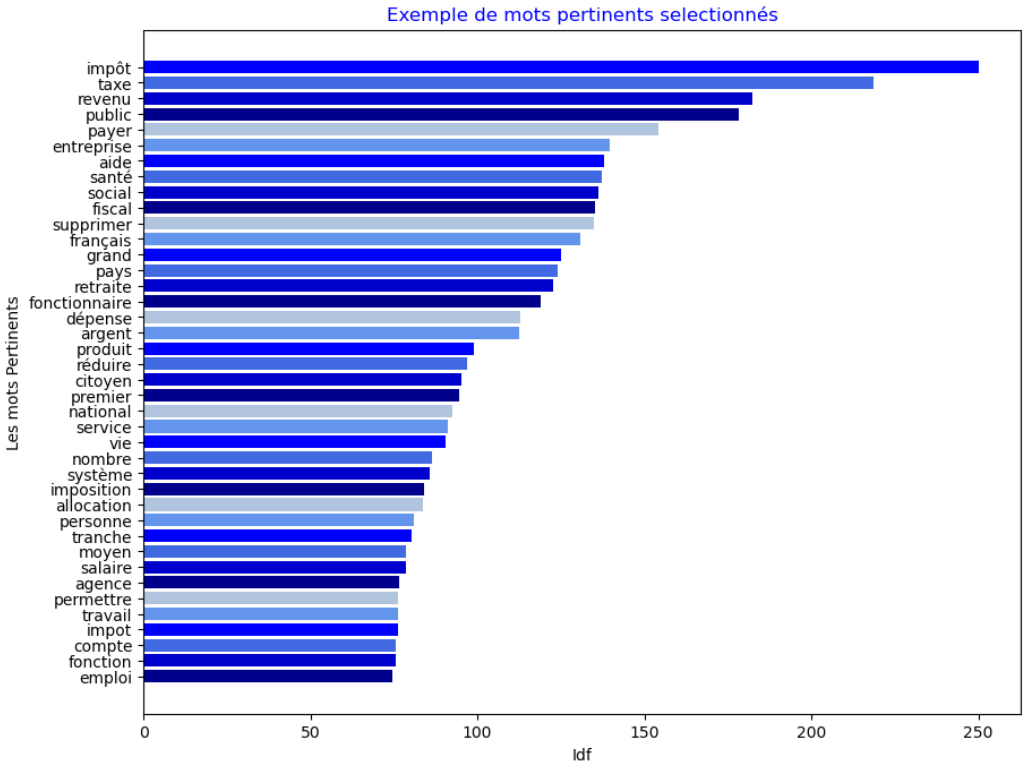
Nous allons utiliser un échantillon de 10 textes des réponses libres concernant la fiscalité. Après avoir nettoyé le texte en enlevant les mots qui n'apportent pas de sens et la ponctuation, nous obtenons une liste des mots utiles par textes, voici un extrait :

'objectif', 'réduire', 'fracture', 'social', 'diminuer', 'impôt', 'taxe', 'produit', 'premier', 'nécessité', 'augmenter', 'smic', 'prime', 'activité', 'moyen', 'réduire', 'fraude', 'fiscal', 'imposer', 'grand', 'groupe', 'gafa', 'renforcer', 'taxe', 'transaction', 'financier', 'bulletin', 'info', 'diminuer', 'taux', 'retraité', 'percevoir', 'minute', 'impôt', 'salaire', 'retraite', 'compris', 'csg', 'retraité'…

On calcule ensuite le TF IDF de chaque mot et textes. Si on regarde les mots les plus pertinents selon le TF IDF, c'est-à-dire ceux qui ont le plus grand TF IDF, on obtient comme mots les plus pertinents :

| **Mot** | **Revenu** | **Impôt** | **Emploi** | **Retraite** | **Taxe** | **Retraité** | **Salaire** | **Taux** | **Diminuer** |
|---|---|---|---|---|---|---|---|---|---|
| **TF IDF** | 182.25 | 61.86 | 24.86 | 20.43 | 20.35 | 18.35 | 15.62 | 13.32 | 10.17 |

On peut aussi sélectionner les mots pertinents d'un texte grâce à leur IDF uniquement. Le graphique suivant montre un exemple de cette technique pour le corpus fiscalité avec les 40 mots les plus pertinents.




## V/ Etude de la pertinence de TF-IDF pour la détection des mots courants de la langue

Dans cette partie nous étudierons la pertinence de la méthode TF-IDF pour reconnaitre les mots courants. Comme indiqué dans la partie I.3 plus le TF est grand plus un mot est présent dans un texte et plus son IDF est grand, dans moins de document il se situe textes du corpus. L'indice TF-IDF parait donc une parfaite combinaison pour évaluer l'importance d'un mot pour un texte dans un corpus. En effet, l'importance augmente proportionnellement au nombre de fois qu'un mot apparaît dans le document mais est compensée par la fréquence du mot dans le corpus. Dans notre cas étant donné que les texte d'un corpus ont tous un même thème, les mots ayant les plus grand TF-IDF sont des mots importants en lien donc avec ledit thème et aussi les mots courants de la langue.

Nous calculons donc les TF-IDF des textes du corpus Fiscalité et relevons les 200 mots ayant les plus grands TF-IDF grâce à la fonction *motspluspertinen*ts :

['impôt', 'taxe', 'revenu', 'retraite', 'aide', 'fiscal', 'public', 'payer', 'français', 'social', 'entreprise', 'allocation', 'o', 'tranche', 're venir', 'santé', 'retraité', 'droit', 'pays', 'supprimer', 'personne', 'citoyen', 'système', 'produit', 'grand', 'dépense', 'service', 'salai re', 'suppression', 'vie', 'année', 'travail', 'mois', 'compte', 'national', 'fonctionnaire', 'argent', 'nombre', 'fonction', 'société', 'p ermettre', 'réduire', 'mettre', 'impot', 'moyen', 'emploi', 'familial', 'charge', 'csg', 'premier', 'habitation', 'annuel', 'politique', 'i mposition', 'taux', 'minimum', 'enfant', 'niveau', 'commune', 'formation', 'monde', 'donner', 'augmentation', 'chômage', 'avan tage', 'économique', 'augmenter', 'élu', 'ressource', 'cotisation', 'activité', 'etat', 'économie', 'contribuable', 'baisser', 'riche', 'é ducation', 'prendre', 'place', 'haut', 'foyer', 'complémentaire', 'nouveau', 'pension', 'priver', 'salarié', 'employeur', 'valeur', 'ass urance', 'niche', 'financier', 'question', 'dividende', 'créer', 'temps', 'tva', 'sécurité', 'taxer', 'action', 'verser ', 'baisse', 'étranger', 'cas', 'ensemble', 'classe', 'collectivité', 'point', 'fraude', 'investissement', 'besoin', 'achat', 'corporation', 'local', 'transport', 's eul', 'européen', 'part', 'association', 'imposer', 'rapport', 'montant', 'dire', 'pense', 'revoir', 'fin', 'ancien', 'gestion', 'travailler', 'jour', 'servir', 'mesure', 'compris', 'coût', 'famille', 'habitant', 'population', 'foncier', 'milliard', 'nombreux', 'maladie', 'agence ', 'peuple', 'utilisation', 'remplacer', 'contribution', 'effort', 'contribuer', 'rétablir', 'situation', 'professionnel', 'total', 'ministre', 'rémunération', 'possible', 'seuil', 'bénéfice', 'vivre', 'gouvernement', 'salarier', 'cher', 'capital', 'heure', 'petit', 'financement', 't

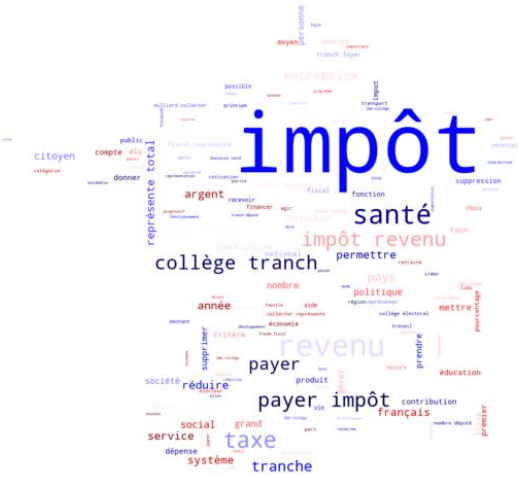
ype', 'soin', 'consommation', 'paie', 'budget', 'président', 'évasion', 'financer', 'actif', 'général', 'réduction', 'région', 'dette', 'condition', 'gens', 'comprendre', 'avenir', 'proposition', 'finance', 'territoire', 'indemnité', 'personnel', 'base', 'nature', 'utiliser', 'actuel', 'diminuer', 'forme', 'investir', 'écologique', 'nécessité', 'département', 'débat', 'répartition', 'attribution', 'administration']

Les mots surlignés (bleu : nom commun, vert : verbe) font partie des mots les plus courant de la langue française selon *encyclopedie-incomplete.* On remarque que la fonction *motspluspertinents* basée sur la méthode TF-IDF relève non seulement les noms communs en lien avec le thème du corpus, et les verbes en lien avec le corpus les plus courants. En utilisant un corpus de texte ayant des thèmes différents les uns des autres pourrait mettre plus en évidence cette propriété, puisque les mots qui reviendront les plus souvent donc qui auront les plus grand TF-IDF dans chaque texte seront des mots courants.

## VII/ Nos résultats

Dans cette partie, nous fixons d'abord le nombre de répétition dans la fonction kmeans qui est de 50.

- Tester avec 4 clusters et vérifier si on peut retrouver les grands thèmes de départ.



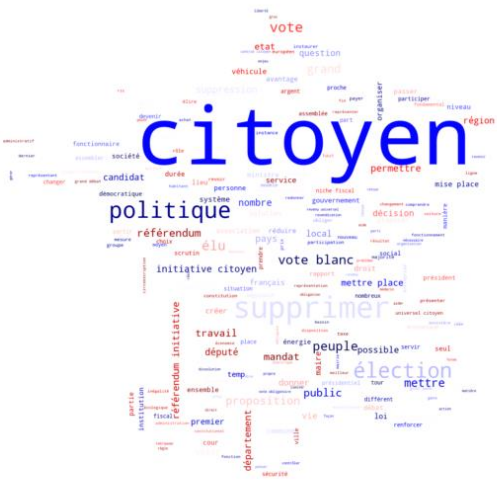





*Figure 2 : Les nuages de mots de l'ensemble de textes*

Nous identifions facilement que le premier cluster correspond bien au thème de fiscalité, où nous trouvons les mots : impôt, revenu, taxe, santé, réduire etc. Le deuxième cluster correspond au thème démocratie, où nous trouvons les mots : citoyen, politique, élection, vote blanc, élu, député, mandat etc. Le troisième cluster correspond plus au thème d'écologie, où nous trouvons les mots : local, transport, commun, réduire, taxe, gestion, baisse, consommation, etc. Le quatrième cluster correspond plus au thème de service, les mots qui sont en évidence dans ce cluster sont : aide, social, donner ; vie, travail, personne, population, etc.

Les mots les plus en évidence dans ces quatre nuages de mots sont impôt, citoyen, public et aide qui apparient dans les quatre thèmes. Nous remarquons que les quatre thèmes sont liés, par exemple on peut dire que « augmenter les taxes sur la consommation des énergies est bien pour l'écologie » et cette phrase est dans le thème d'écologie et aussi de fiscalité. C'est pour cette raison que les quatre clusters ne correspondent pas parfaitement aux quatre thèmes : fiscalité, démocratie, écologie et service. Par ailleurs, dans l'étape de trouver les mots pertinents, il est probable que les mots pertinents ne sont pas parfaitement répartis dans les quatre thèmes, par exemple, pour les 400 mots pertinents peut-être il y a 150 de mots dans le thème de fiscalité et 50 de mots dans le thème de démocratie car nous pouvons facilement faire un lien entre la fiscalité et les trois autres thèmes donc les mots de fiscalité peut-être plus mentionnés.

- Tester aussi la méthode à l'intérieur de chaque grand thème pour trouver k sous-thèmes qui vous paraissent faire du sens.

  - Fiscalité

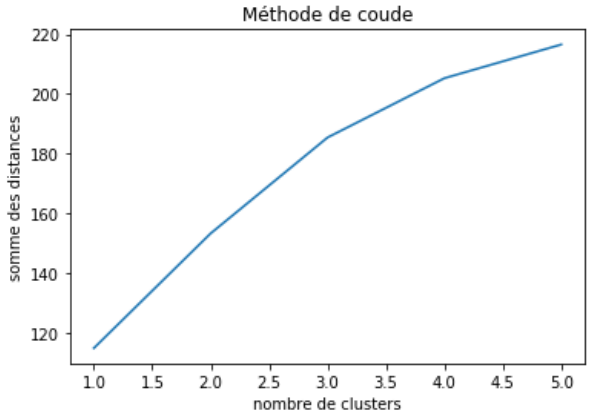



*Figure 3 : Graphique sur le Méthode de coude pour les textes dans le thème de fiscalité*

Nous remarquons que pour toutes nos méthodes de coude, l'éboulis est croissant du fait que nous avons utilisé la similarité cosinus et non la distance. En effet, une variance élevée signifie que les textes sont proches.

Nous voyons ici que le coude est dans le nombre 3 ou 4 et on prendre 4 clusters pour avoir plus de précisions.

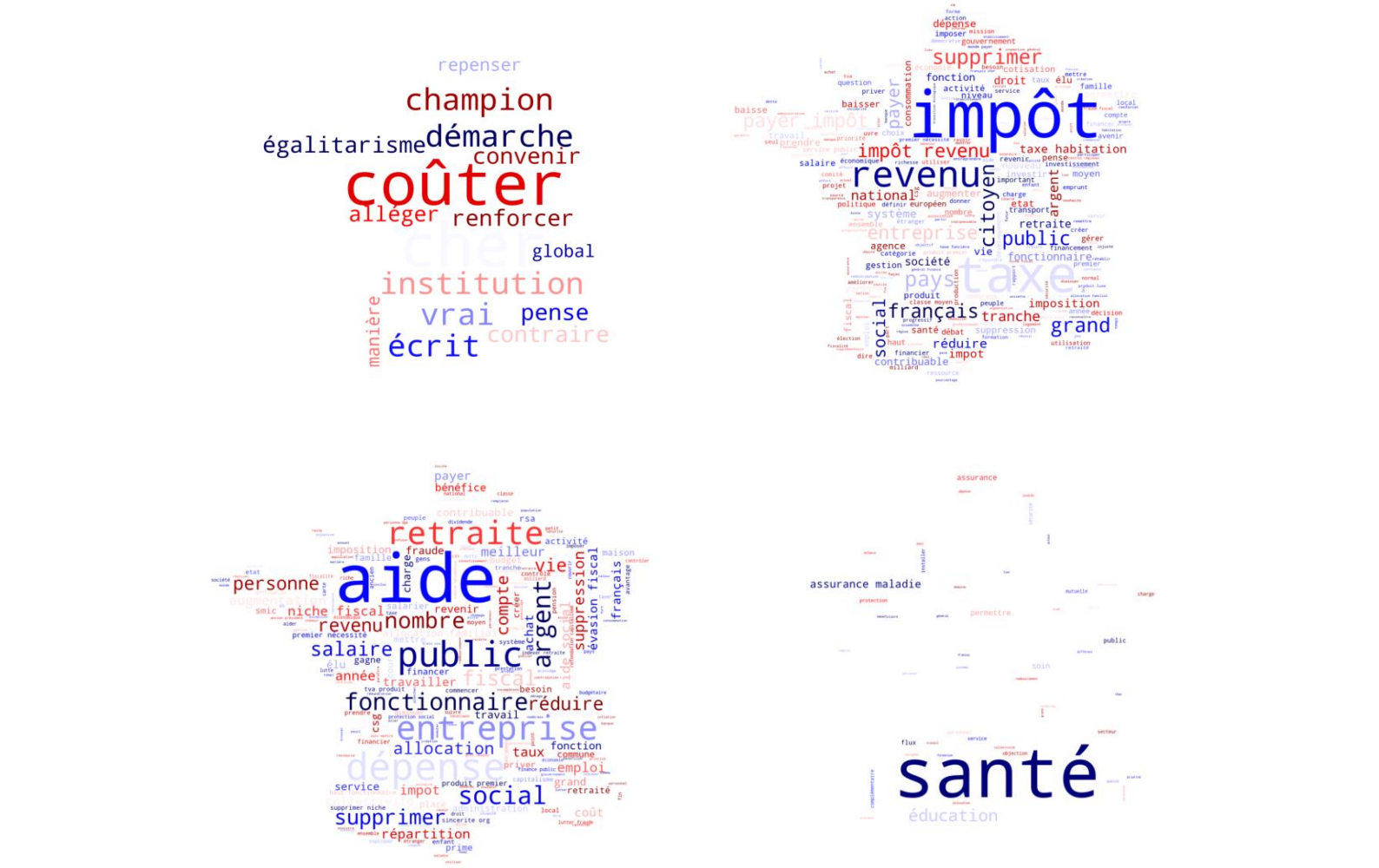



*Figure 4 : Les nuages de mots de Textes de fiscalité*

Nous relevons ici les sous thèmes des textes ayant un regard critique sur la fiscalité en France, le thème des impôts, ceux concernant les aides aux entreprises et aux personnes et les réponses concernant la santé. Nous remarquons qu'il y a deux sous thèmes principaux, pour lesquels beaucoup de réponses ont été rattachées : les impôts et les aides.

Nous voyons ici dans le deuxième cluster,

- Démocratie

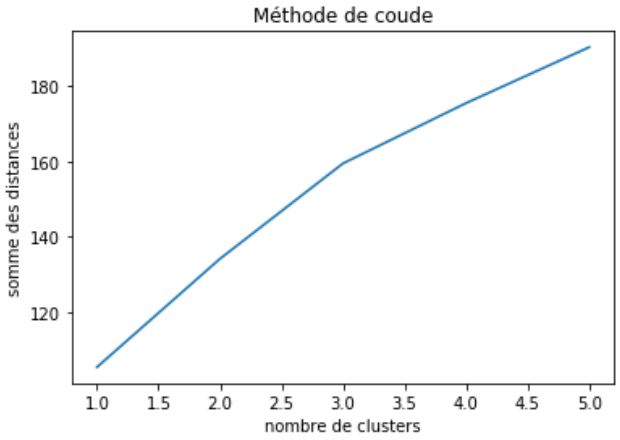



*Figure 5 : Graphique sur le Méthode de coude pour les textes dans le thème de démocratie*

Nous voyons ici que sur le graphique, le coude est dans le nombre 3 donc nous sélectionnons un nombre de cluster égal à 3.

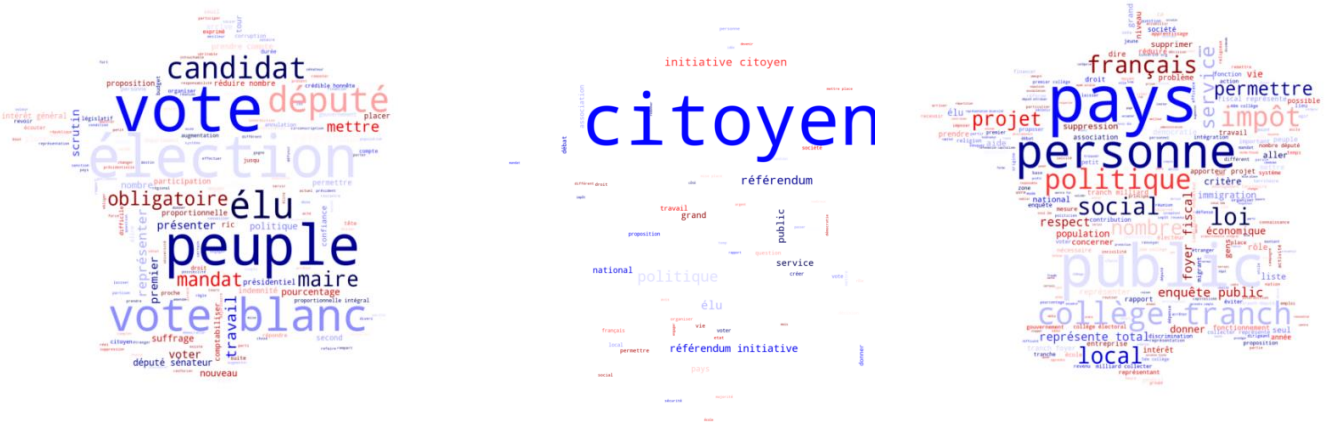



*Figure 6 : Les nuages de mots de textes de démocratie*

Nous relevons ici les sous thèmes de l'élection et vote, du pays et du citoyen. Nous remarquons que les deux sous thèmes qui ressortent le plus des réponses sont le sous thème de l'élection et vote et celui du pays. Le sous thème de l'élection est lié avec le candidat, le député, le vote blanc, le peuple et le sous thème de l'élection est lié avec l'impôt, la politique, la loi etc. ce sont des facteurs faisant fonctionner un pays.

- Écologie

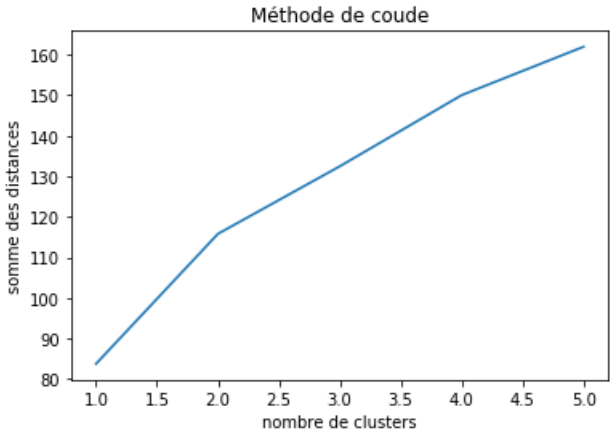



*Figure 7 : Graphique sur le Méthode de coude pour les textes dans le thème d'écologie*

Nous voyons ici que le coude est dans le nombre 2 ou 4 et nous prenons 4 clusters pour avoir plus de précisions.

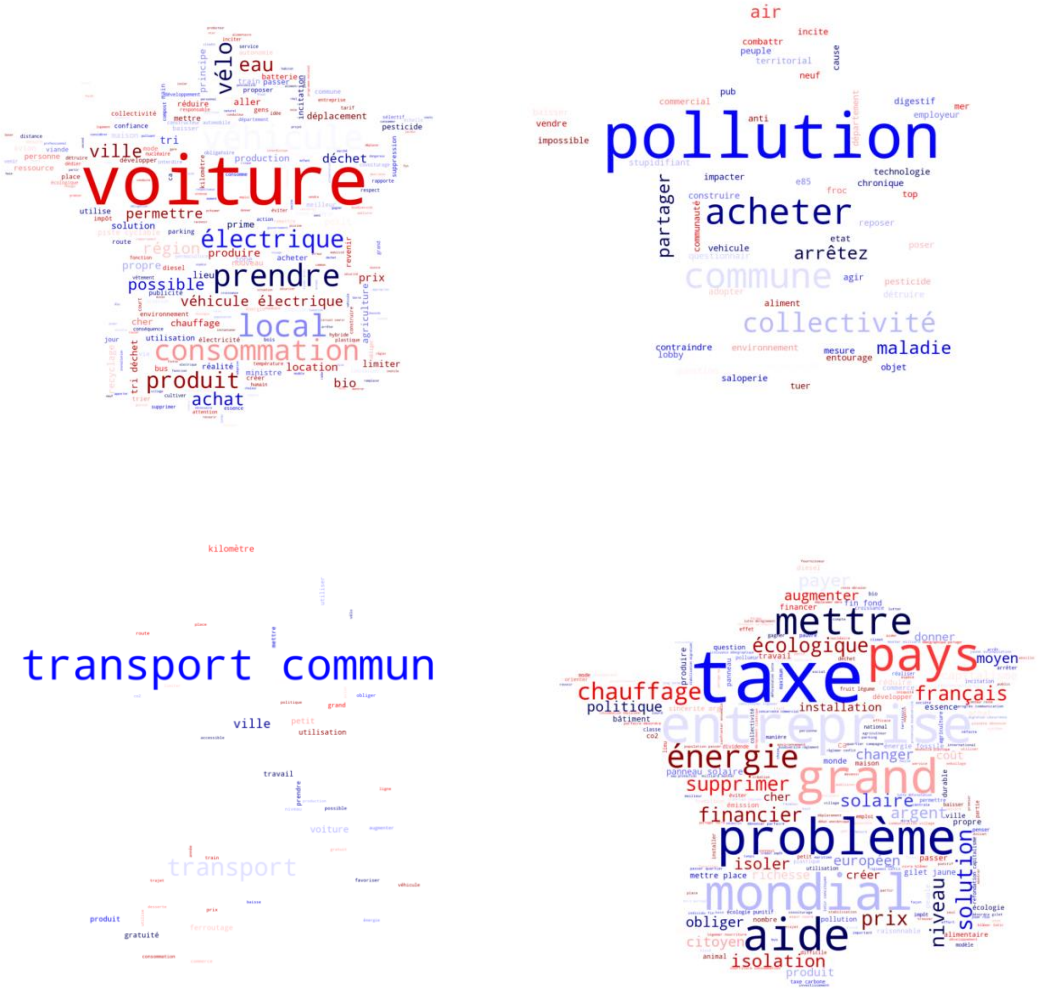



*Figure 8 : Les nuages de mots de textes d'écologie*

Nous relevons ici les sous thèmes de l'automobile, la pollution, le transport en commun et la taxation sur l'usage des énergies. On remarque que les deux sous thèmes qui ressortent le plus des réponses sont le problème des taxes sur l'énergie et sur l'utilisation de la voiture

- Services

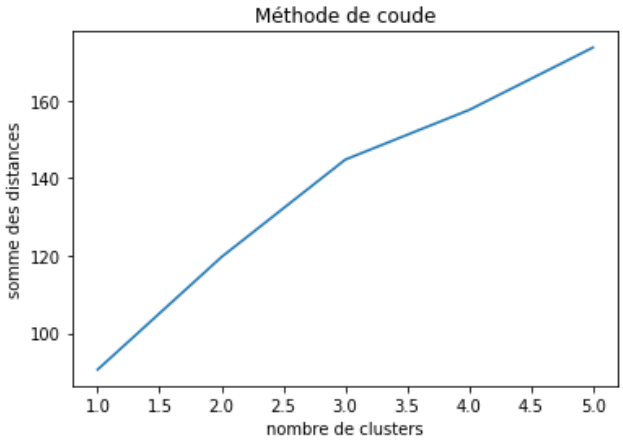



*Figure 9 : Graphique sur le Méthode de coude pour les textes dans le thème de services*

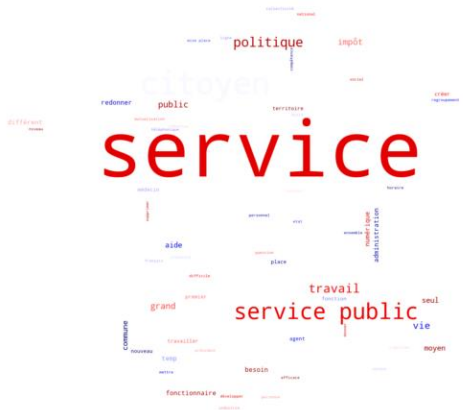




*Figure 9 : Les nuages de mots de textes de services*

Nous relevons ici les sous thèmes de la formation, de l'administration et du service public et l'impôt. Le troisième cluster rassemble des textes qui n'avaient pas beaucoup de ressemblance ou des réponses dans lesquels les participants n'avaient rien à signaler. Nous remarquons que le sous thème qui ressort le plus des réponses est la formation.

# VIII / Conclusion

Les différents outils de *text mining* nous ont permis d'analyser les réponses au grand débat politique de 2019. Nous avons pu déterminer, pour chaque sujet, plusieurs sous thèmes auxquels les français ont fait référence. Concernant la fiscalité, les sujets qui ressortent des réponses sont la diminution des impôts et des taxes, l'augmentation des aides, les entreprises ou encore la santé. Pour la démocratie, beaucoup de réponses concernaient le système des élections (vote, référendum…), les lois et la politique. Au sujet de l'écologie, les français ont soulevé le problème de la pollution des transports comme la voiture, des énergies et les solutions comme les transports en commun. Enfin, pour le sujet des services, on trouve comme sous thème l'administratif, les formations et les services publics.

Les outils utilisés comme le clustering ou encore la distance cosinus sont très répandus dans le traitement de données chez les entreprises par exemple. Ils sont notamment utilisés pour l'analyse des avis clients ou dans le domaine universitaire pour la détection du plagiat. Cependant, il existe beaucoup d'autres outils pour analyser un texte. Par exemple, d'autres mesures de similarités autres que la distance cosinus : la distance de Jacquard, la distance euclidienne etc. Il pourrait être intéressant d'analyser notre texte avec plus d'outils afin d'avoir des résultats plus précis. De plus, un des problèmes rencontrés lors de l'analyse concernait la taille des corpus. Avec les outils que nous avions à disposition, il n'était pas possible d'analyser l'entièreté des textes, nous avons donc concentré notre analyse sur un échantillon de textes.