Clustering automatique de texte

MAI 10

Titouan Galdemard,
Jean Rolland,
Kayané Robach

Description des outils théoriques utilisés

Lemmatisation

On lemmatise notre corpus pour le nettoyer : enlever les majuscules, éviter les mots fonctionnels comme les stop words (ensembles de mots très fréquents tels que les déterminants ou les prépositions) et pour empêcher que les mots puissent apparaître sous des formes différentes (pluriel de noms, verbes conjugués, accord des genres…). On traite donc l'ensemble du vocabulaire pour ne garder que les racines ou lemmes des mots : c'est-à-dire la forme qui correspond à leur entrée dans le dictionnaire.

TF IDF

« Term Frequency – Inverse Document Frequency » est une méthode utilisée dans le traitement d'informations sur un corpus de textes. On désignera par $\mathcal{R}$ le *« Répertoire »* de tous les mots qui composent le corpus (noté $\mathcal{C}$). Cette méthode associe à chaque réponse (noté r_j) un vecteur, de dimension celle de $\mathcal{R}$, dont les coordonnées correspondent aux scores TF-IDF de chaque mot de $\mathcal{R}$.
Ce score est construit de la manière suivante. La première partie *« Term Frequency »* d'un mot i fait référence à sa fréquence brute (noté $f_{r_j,i}$) dans la réponse r_j où il est présent. On obtient donc l'importance de ce mot dans cette réponse. La deuxième partie *« Inverse Document Frequency »* d'un mot, nous informe sur l'importance de ce mot dans le corpus. Elle est calculée par $log\left(\frac{Card(\mathcal{C})}{Card(r_j\,;\, i \in r_j)}\right)$. Donc, plus un mot sera rare dans le corpus, plus son IDF sera grand et inversement. Ainsi, le score final retenu est le produit de ces deux derniers, il évalue l'importance du mot sur le corpus.
Cependant, ce score présente des limites. Si un mot est présent dans une unique réponse, son score IDF associé sera très élevé mais le mot ne peut apporter aucune information pertinente aux réponses. De même si la fréquence d'un mot est très forte. Nous prendrons en compte ces deux remarques dans la suite du projet.

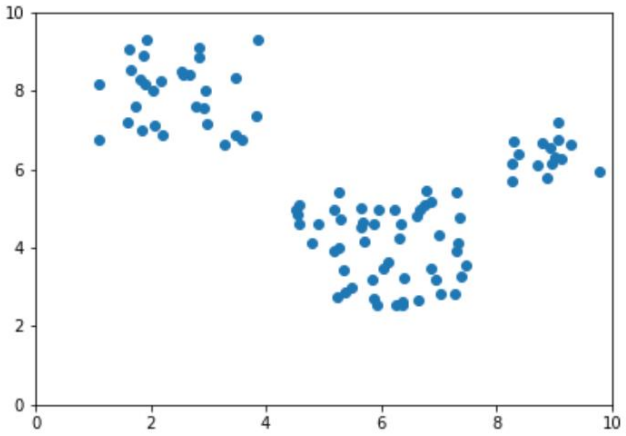
Distance Cosinus

La distance cosinus (ou similarité cosinus) permet d'évaluer la similarité entre des vecteurs de même dimension en mesurant l'angle entre ces vecteurs. Avec la méthode TF IDF on obtient pour chaque réponse un vecteur des scores TF-IDF, qui représente l'importance relative de tous les mots du répertoire $\mathcal{R}$ dans la réponse en question. La fonction de distance cosinus calcule le cosinus entre deux vecteurs réponses. La valeur -1 indique des vecteurs opposés, 0 des vecteurs indépendants et 1 des vecteurs similaires. Deux vecteurs dont la distance cosinus est proche de 1 représentent donc 2 réponses semblables, qui emploient le même vocabulaire et à la même fréquence.

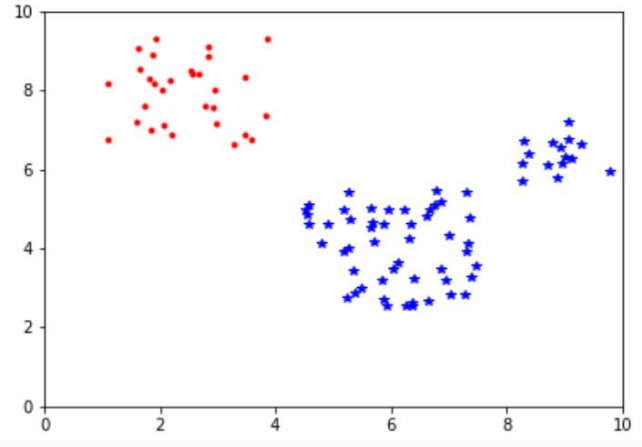
K Means

Le but de créer une fonction K Means à terme est de pouvoir identifier et regrouper différents types de réponses pour un corpus donné. Afin de vérifier que nous sommes en mesure de créer cette fonction correctement il nous est demandé de l'appliquer à l'ensemble des réponses de nos données (donc aux réponses des quatre dossiers du grand débat) et de vérifier que la fonction est bel et bien capable de reformer les quatre groupes initiaux auxquels appartenaient les réponses.
D'abord appliquée à un groupe de points aléatoires dans un espace en deux dimensions nous généralisons ensuite la fonction dans un espace à n dimensions (n étant la taille du vocabulaire utilisé). Cela nous permet aussi de vérifier l'efficacité de notre fonction qui mesure la distance cosinus.

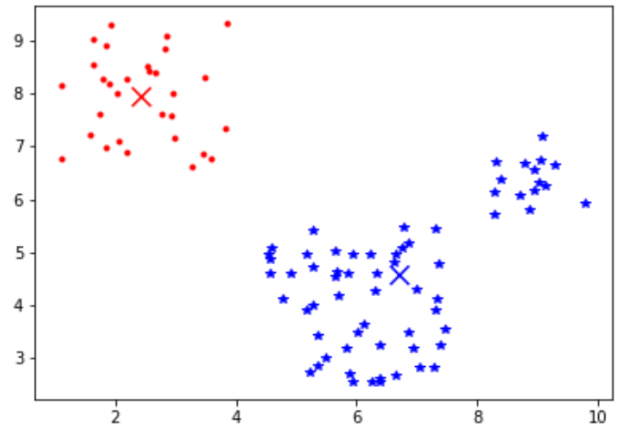
Voici le fonctionnement de notre fonction : Dans un espace en deux dimensions nous utilisons le K Means sur des groupes de points générés aléatoirement. Nous générons des points au hasard : les centres initiaux. Le but de la fonction est de faire évoluer les centres jusqu'à ce qu'il n'y ait plus d'amélioration à faire pour qu'ils représentent réellement les barycentres de chacun des groupes de points. On décide qu'il n'y a plus d'amélioration à faire quand la distance cosinus entre le précédent barycentre et le nouveau est inférieure à une certaine déviation entrée en paramètre (que nous posons par défaut à hauteur de 0.01)



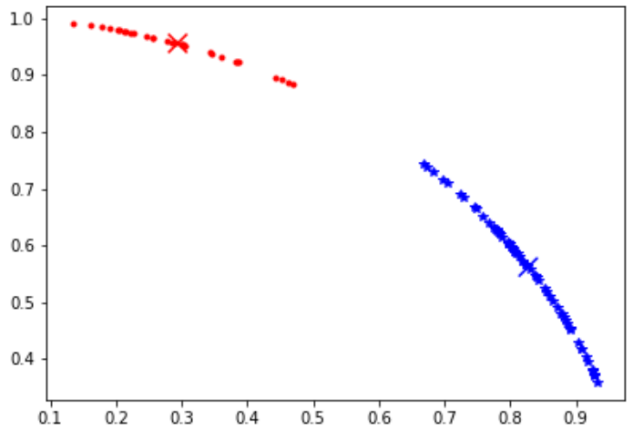
1. Génération aléatoire de 3 groupes de points



2. En termes de distance cosinus, ils forment 2 groupes



3. Barycentres finaux trouvés par la fonction K Means qui fonctionne avec la distance cosinus



4. Vérification de la fonction de distance cosinus sur les 2 groupes de points

Afin d'obtenir un résultat concluant avec notre ensemble de réponses nous réduisons le vocabulaire. Les mots dont le score IDF est grand ne sont pas utiles pour atteindre notre objectif, ce sont des mots trop rares ; ils ne nous permettront pas de différencier les dossiers parents. De la même manière les mots dont l'IDF est petit sont trop fréquents et ne permettent pas l'identification de différents groupes. Après différents tests sur les contraintes de score IDF pour aboutir à une différentiation correcte nous choisissons de sélectionner les mots dont l'IDF est inférieur à 7,5 et supérieur à 4. Cette contrainte réduit le vocabulaire à 1314 mots. On applique alors le K Means dans un espace de dimension n.

Choix de conception et description des algorithmes

Nous utilisons la librairie sklearn et notamment le transformateur TfidfVectorizer pour calculer les scores IDF et TF-IDF quand cela est nécessaire. Ce transformateur est très pratique car il nous permet de récupérer simplement ces scores. Au lieu de stocker pour chaque mot leur fréquence dans un corpus nous nous contentons du score IDF qui capture cette information. De manière analogue nous remplaçons le stockage des fréquences de chaque mot par texte par le calcul du score TF-IDF qui capture plus d'informations et qui nous est plus utile.

Nous avons créé trois gros algorithmes.

Le premier sert à retourner un fichier contenant tous les fichiers du grand débat traités (voir code *Partie 1 Traitement des fichiers*). Nous utilisons pandas pour récupérer les données des fichiers du grand débat. Nous gardons toutes les réponses libres à toutes les questions des quatre fichiers, tout en mémorisant pour chaque réponse le dossier parent duquel elle provient ainsi que la question à laquelle elle est adressée et le numéro d'identification de cette question. S'ajoute à cela le numéro d'identification de l'auteur de la question (que nous n'utiliserons pas). Le document obtenu est ensuite traité avec spacy afin de le « nettoyer », c'est à dire le lemmatiser, enlever les petits mots inutiles, les majuscules, la ponctuation…
Afin de traiter nos documents de manière efficace nous supprimons du *pipeline* (liste des processus de nettoyage réalisés par spacy) de spacy les processus *ner* (*name entity recognition*) et *parser* dont nous n'avons pas besoin pour nettoyer nos textes. Au lieu de faire une simple boucle for dans laquelle nous réaliserions le traitement de tous les textes uns à uns, nous avons choisi d'ajouter un processus au *pipeline* afin de nettoyer tous les textes, processus qui s'adapte mieux au fonctionnement de spacy.
On mémorise donc dans le fichier final la réponse telle quelle ainsi que sa forme nettoyée sous forme de liste de mots. Ce processus de nettoyage dure environ deux heures.

Le second algorithme créé un nuage de mots (voir code *Partie 2 & 3 Word_Cloud_generation*), notamment un nuage des mots les plus importants qui apparaissent dans l'ensemble des documents réponses à une question donnée. Nous récupérons les données du fichier précédant grâce à la librairie pandas ; nous choisissons une question à l'aide de son numéro d'identification en entrant un numéro de la liste des questions dans « qID_selected ». Grâce aux différentes librairies de Python matplotlib, wordcloud, sklearn et numpy nous créons une fonction « genWordCloud » qui génère un nuage de mots (30 par défaut) à partir d'une liste de textes qui correspond aux réponses choisies mises sous forme de listes. Pour cela nous utilisons le score IDF des mots afin de faire apparaitre leur poids dans le corpus. Les mots importants ont un score IDF faible, ce sont ceux que l'on retrouve le plus souvent dans le corpus. Par la suite, on inverse ces scores pour que plus celui d'un mot soit petit, plus le mot apparaisse gros sur l'image. Ainsi nous pouvons facilement visualiser les mots importants de l'ensemble des réponses à une question.

Enfin, le dernier algorithme utilise la distance cosinus comme mesure de la similarité entre deux documents (donc deux réponses) afin d'appliquer la méthode des K Means qui permet de regrouper ensemble des documents « homogènes » (voir code *Partie 4 & 5 Distance Cos & K Means*). Pandas nous permet de récupérer les données textuelles, ensemble des documents que nous utilisons.
Pour le fonctionnement du K Means, dans le but d'alléger la quantité de mémoire utilisée nous utilisons les sparse matrix de scipy. Ces matrices fonctionnent comme les matrices de numpy mais ont la particularité de ne pas utiliser de mémoire pour les cases dont la valeur est 0. C'est très utile pour réduire la taille de nos données puisque quand on procède au calcul des scores TF IDF d'une réponse il y a beaucoup de cases dont la valeur vaut 0 puisqu'aucune des réponses ne comprend tous les mots du répertoire. Grâce à cela nous réduisons considérablement la quantité de RAM utilisée.
Nous sélectionnons du vocabulaire dans le répertoire $\mathcal{R}$ des mots de l'ensemble des documents selon des critères autour du score IDF des mots. Pour ce faire nous choisissons un intervalle de valeurs des IDF. Trente-deux mots du vocabulaire ont un IDF compris entre 0 et 4, parmi ces mots on trouve interdiction, transport, installation, saisir, public, économique … Ce sont des mots très fréquents dans le corpus, on peut penser qu'ils ne seront pas très pertinents pour différencier les quatre groupes initiaux. Parmi

les mots dont l'IDF est compris entre 12 et 15 on trouve cheddar, frite, picard, apocalyptique, médiatrice, cacophonique ou encore fusible. On trouve aussi beaucoup de mots qui ont été mal tapés et dont la lemmatisation ne s'est pas faite correctement : credibl, demarr, epac, peubl, cytoyenn. Ces mots sont très rares dans le corpus ils ne sont donc pas utiles à la différentiation des groupes de textes. Plus de 4000 mots ont un IDF compris entre 8 et 10, alors que moins de 1900 ont un IDF inférieur à 8.
Après de nombreux tests quant à la sélection d'IDF à faire nous avons décidé de constituer le vocabulaire avec les mots dont l'IDF s'étale entre 4 et 7,5 car la différentiation des groupes se fait mieux. Cela réduit notre vocabulaire à 1314 mots ; élargir l'intervalle ne nous permet pas d'obtenir un meilleur résultat… Pour réaliser la fonction K Means nous nous aidons de sklearn, numpy et scipy ainsi que de la fonction de distance cosinus créé dans un même temps.
Ensuite afin de vérifier la qualité de notre fonction nous la testons et essayons de retrouver nos quatre groupes : services, écologie, fiscalité et démocratie à partir de notre ensemble de vocabulaire (1314 mots). Cependant un problème récurrent apparait : la fonction n'identifie pas le groupe du questionnaire sur les services, comme on peut le voir dans l'exemple ci-dessous. Les réponses provenant de ce questionnaire se trouvent reparties dans les autres groupes alors que chacun des autres thèmes domine au moins un groupe.

	démocratie	services	fiscalité	écologie
1er groupe	**270754**	169925	127585	56295
2ème groupe	**513720**	128428	271972	346028
3ème groupe	36276	489	**116793**	131
4ème groupe	41879	20277	4583	**83787**

Afin d'analyser les limites de notre fonction et de comprendre ses failles nous regardons le pourcentage de mots communs (parmi les mots du vocabulaire sélectionné) entre un des fichiers et les trois autres.

Ecologie et les autres	60%
Démocratie et les autres	66%
Fiscalité et les autres	69%
Services et les autres	70%

On remarque que services est le fichier qui a le plus de mots communs avec les autres fichiers tandis qu'écologie et démocratie qui sont les deux groupes qui semblent se former le mieux avec notre fonction K Means ont un plus petit pourcentage de mots communs avec les autres fichiers. Néanmoins, ces résultats ne nous permettent pas d'expliquer le problème puisque les différences de pourcentages sont minimes. Si on s'intéresse au pourcentage de mots les plus spécifiques d'un fichier qui appartiennent au vocabulaire réduit mais qui ne sont pas présents dans les autres fichiers, on se rend compte qu'il y a seulement 2,6% de mots spécifiques provenant du fichier services quand il y en a 3,8%, 5,3% et 7,5% provenant respectivement des fichiers fiscalité, démocratie et écologie. Cette différence dans le pourcentage de mots spécifiques au questionnaire des services vis-à-vis des autres questionnaires parmi les mots du vocabulaire réduit pourrait expliquer le disfonctionnement de notre fonction K Means.

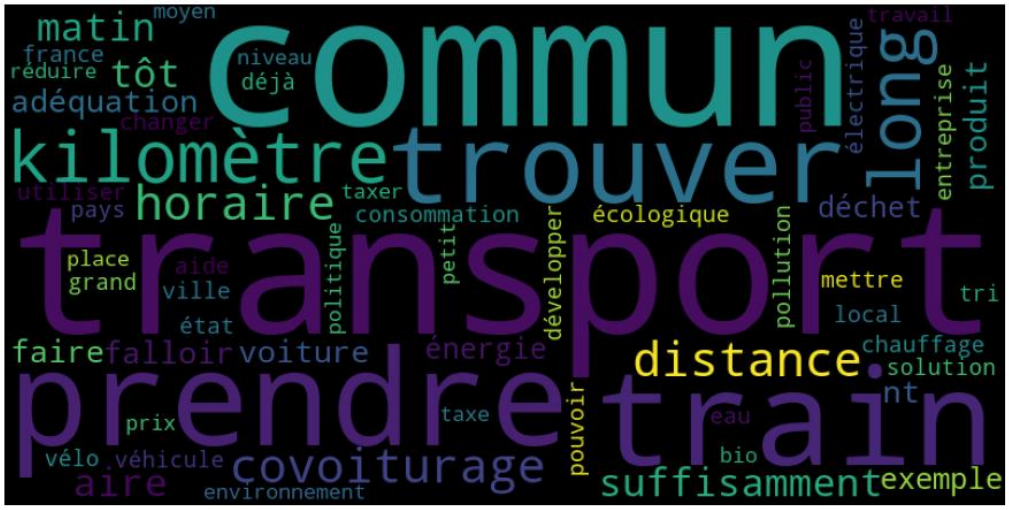
Mots spécifiques à Ecologie	7,5%
Mots spécifiques à Démocratie	5,3%
Mots spécifiques à Fiscalité	3,8%
Mots spécifiques à Services	2,6%

En conséquence du fait que nous ayons du mal à identifier les quatre groupes correspondant aux quatre dossiers, les nuages de mots définis par les groupes que nous pouvons former avec le K Means ne donnent que peu de sens. Il est alors plus intéressant de regarder les groupes que la fonction peut former au sein d'un même thème. Quand on regarde les nuages de mots qui se génèrent par ses groupes on remarque des thèmes qui font du sens.

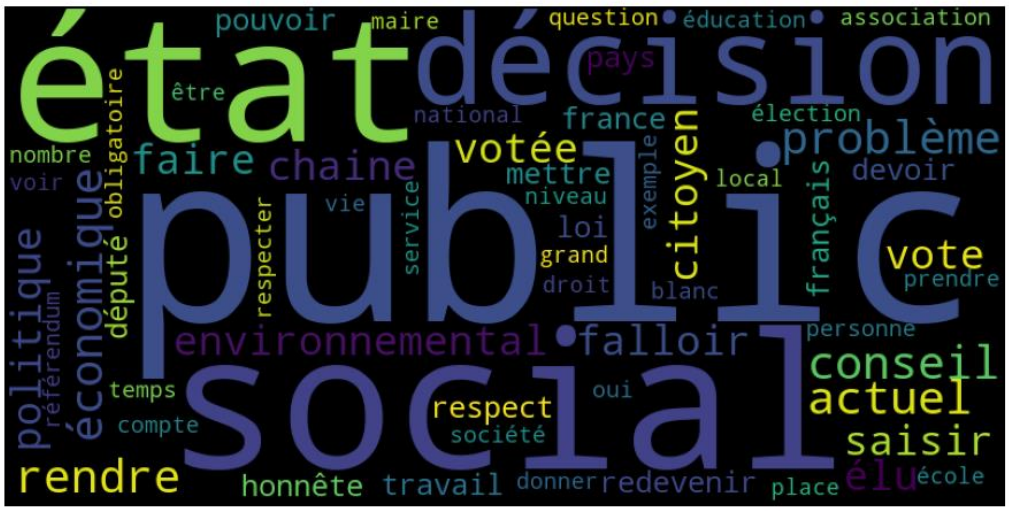
Présentation des résultats

Représentation en nuage de mots des 4 thèmes

ecologie

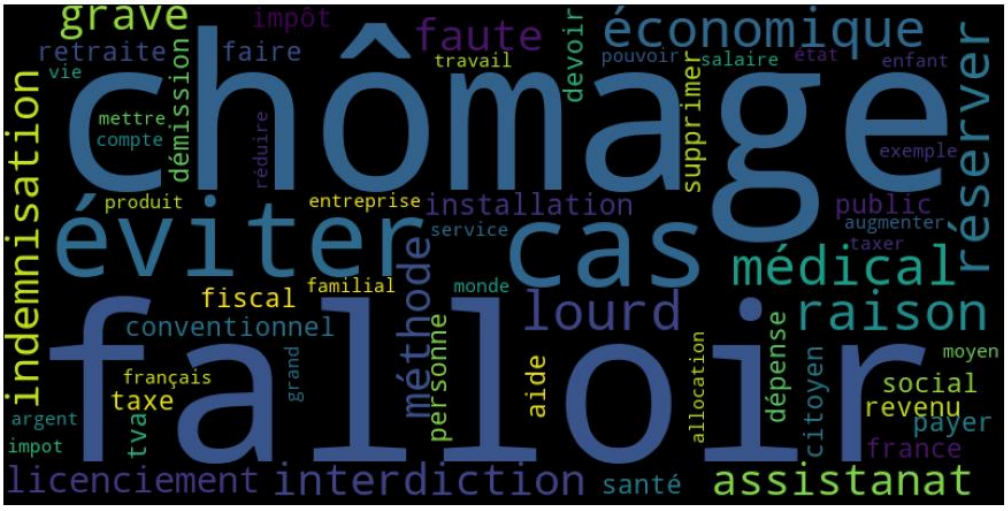


democratie



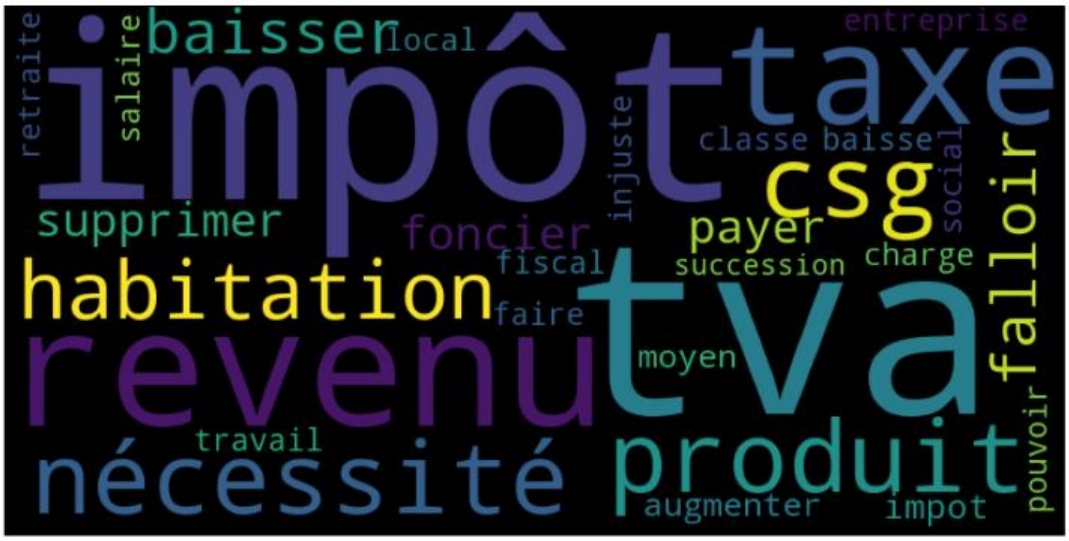
services

fiscalite



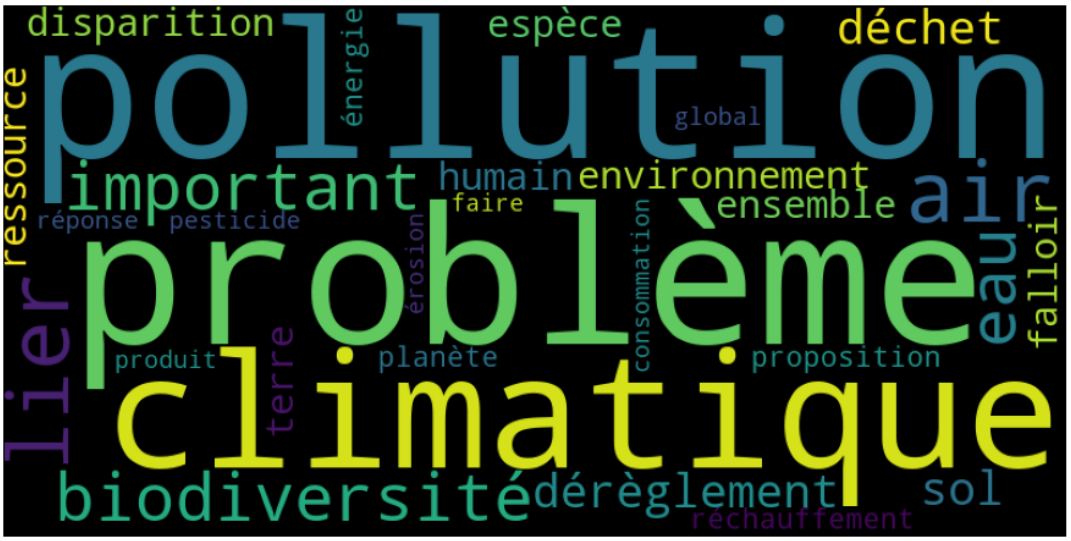
Nuage de mots pour des questions

Quels sont selon vous les impôts qu'il faut baisser en priorité ?

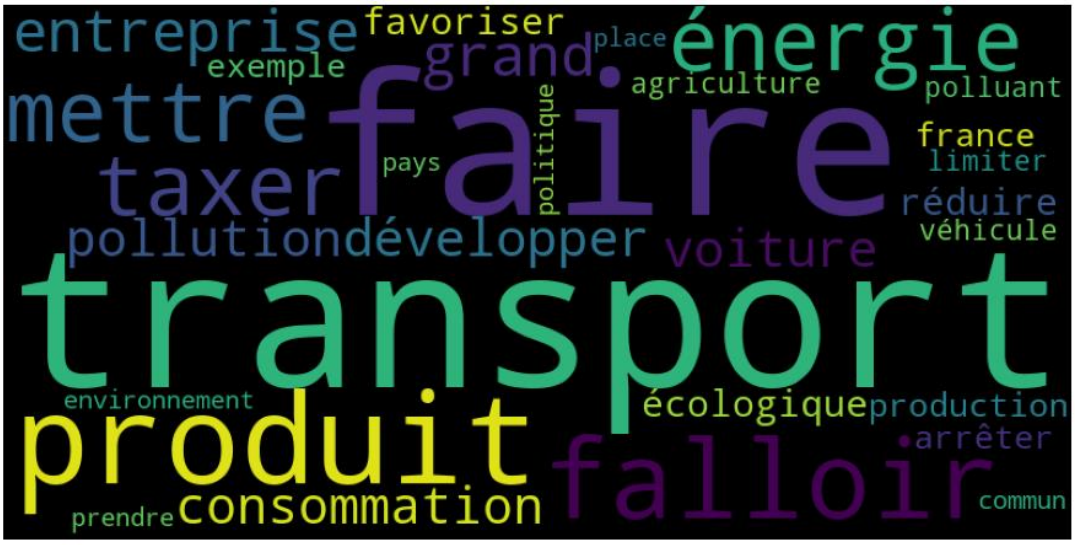


Quels sont les comportements civiques qu'il faut promouvoir dans notre vie quotidienne ou collective ?

Quel est aujourd'hui pour vous le problème concret le plus important dans le domaine de l'environnement ?

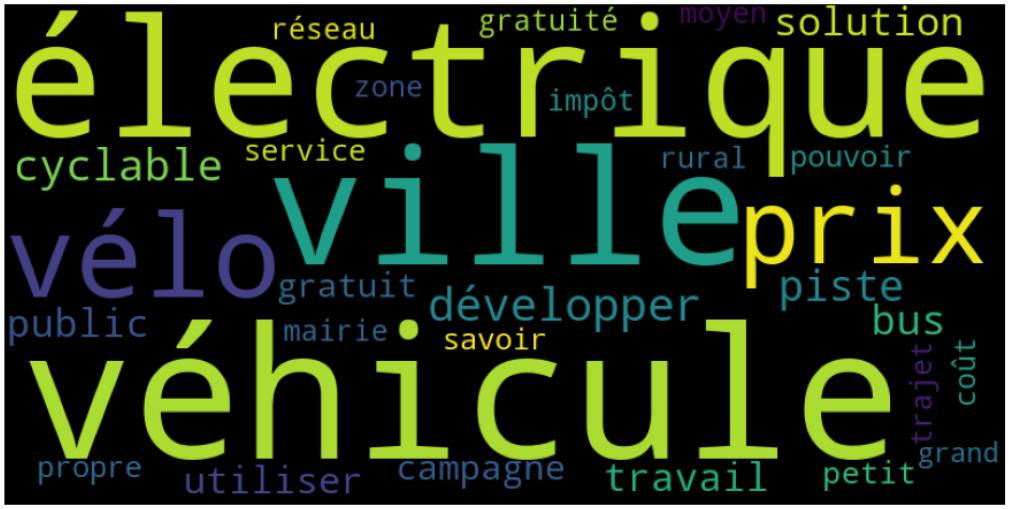


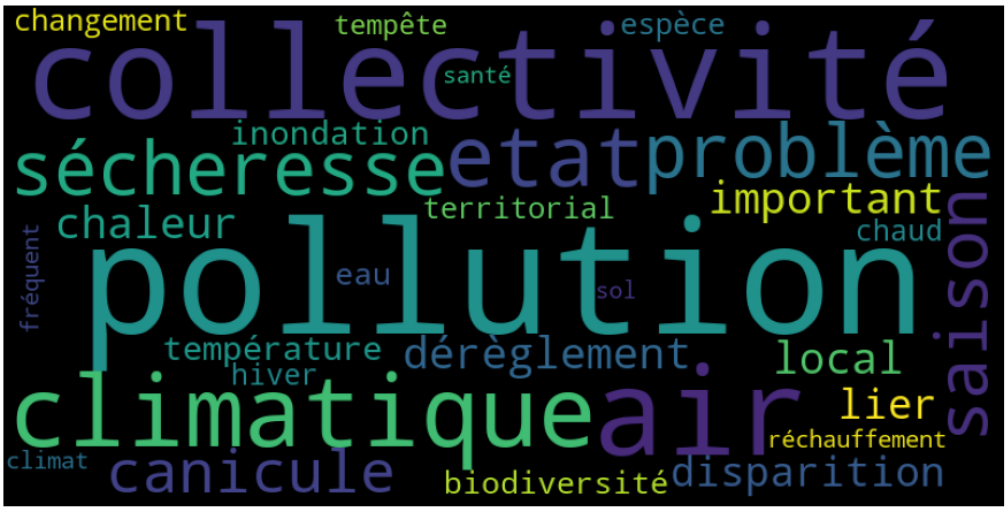
Que faudrait-il faire selon vous pour apporter des réponses à ce problème ?

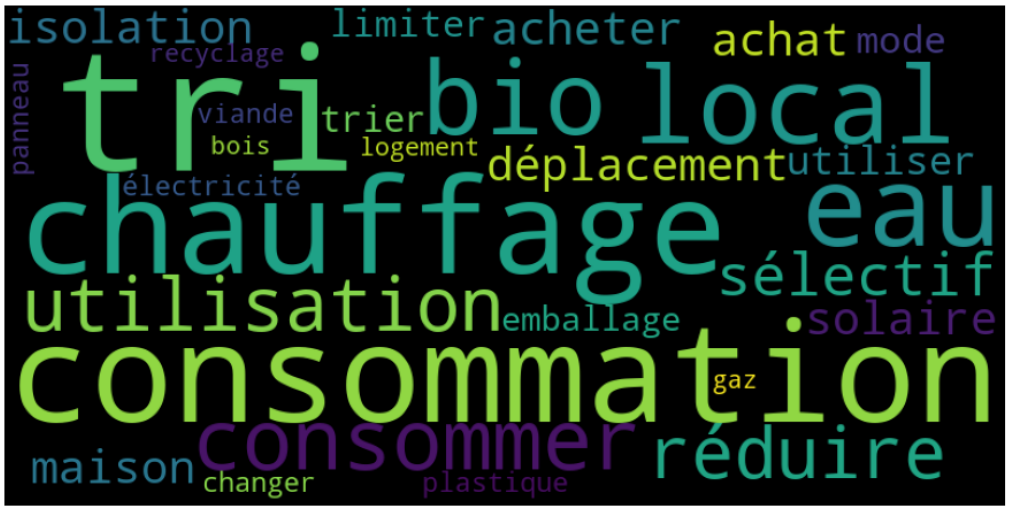


Nuage de mots des sous thèmes formés dans le dossier écologie

Quand on demande à la fonction K Means de produire six groupes dans le dossier écologie nous obtenons des résultats pertinents. On obtient par exemple : un groupe sur les transports, un autre sur les problèmes climatiques, et un sur la consommation.






isolation
limiter
acheter
achat
mode
recyclage
panneau
tri
bio
local
viande
trier
bois
logement
déplacement
utiliser
électricité
chauffage
eau
sélectif
utilisation
emballage
solaire
consommation
gaz
consommer
réduire
maison
changer
plastique