Rapport écrit du Projet de fin de semestre
Amina AHAMADA (n°21101279) et Jules COËNON (n°21103772)

Table des matières

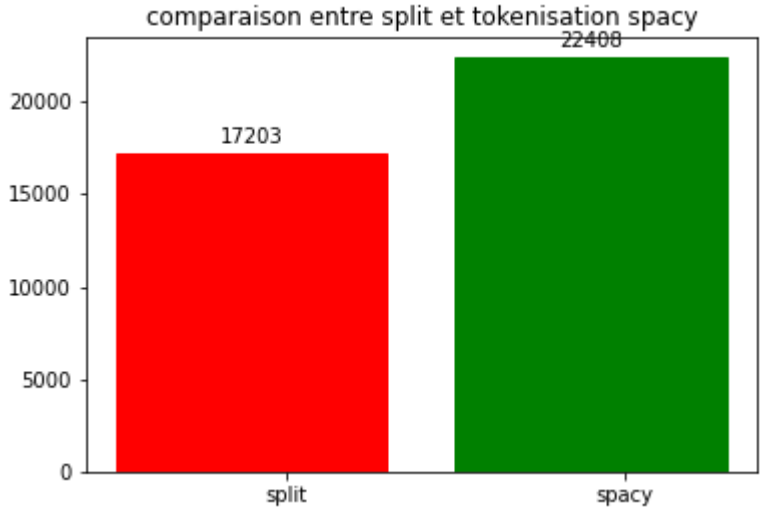
I. Présentation du projet

Ce projet a pour but de créer une représentation du vocabulaire de trois langues à partir d'un corpus multilingue, nous avons choisi le corpus français, anglais et espagnol car ces langues utilisent l'alphabet latin ce qui nous permettra d'avoir des résultats cohérents et de faciliter l'analyse des résultats. Pour réaliser cet objectif nous avons utilisé différents outils déjà disponibles dans python et la librairie spacy et nltk. Nous avons procédé à la réalisation de vocabulaire de deux manières. Premièrement nous avons tokenizer les corpus avec la fonction split et le tokenizer disponible dans la librairie spaCy, cela nous a permis de comparer le fonctionnement de ces deux outils. Puis nous avons lemmatiser et détecter les entités nommées à l'aide des outils de la librairie spacy. Après avoir suivi ces étapes, nous avons déterminé que générer le vocabulaire en se basant sur l'étiquetage grammatical rendrait le processus plus efficace.Nous avons également testé la réalisation du vocabulaire avec nltk pour pouvoir comparer ces deux processus.Finalement, nous avons réalisé le clustering en n-grammes à partir des vocabulaires créés par le biais de l'étiquetage grammatical.

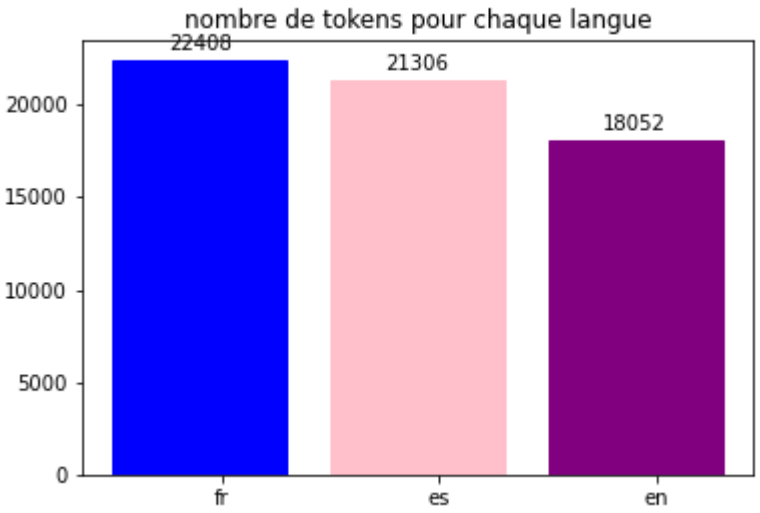
II. Développement

1. Tokenizer

Lors de la tokenization nous avons utilisé la fonction split et les différents tokenizer de spaCy, ainsi nous avons pu constater le fonctionnement des ces deux outils. La fonction split découpe le corpus à partir des espaces tandis que la librairie spaCy a des tokenizers adapter pour chaque langue. Les tokenizers de spacy prennent les ponctuations et les sauts à la ligne comme des tokens. Ainsi d'après cette observation nous avons déduit de que la liste créer avec la tokenization de spacy devait être plus longue que la liste créer avec la fonction split, puisque split n'ajoute pas les saut de lignes ni les signes de ponctuation quand ils sont collés au mot. Ci-dessous nous avons représenté la taille des listes de token avec split et spaCy pour le corpus français.



Nous pouvons constater que split possède moins d'élément que la liste créer avec la tokenization de spaCy, de ce fait les éléments de la liste généré avec la fonction split devraient majoritairement être des mots. Mais la la fonction split n'est pas un tokenizer puisqu'il se contente de séparer le texte en fonction des espaces entre les mots, donc il ne s'adapte pas au texte comme le fait spaCy. C'est pour cela que nous avons fait notre représentation graphique à partir de la tokenisation réalisé avec SpaCy



Nous pouvons constater que le français à un nombre de tokens plus élevé que les deux autres langues.

2. Lemmatisation et détection d'entités nommées.

Nous avons réalisé la lemmatisation et la détection d'entités nommées en créant les fonctions *lemmatisation et entite_nommee .*Après avoir observé les résultats nous avons constaté que l'outil de détection d'entité nommée de spaCy semble défaillant, en effet plusieurs éléments ont été détectés comme étant des entités nommées bien que ce ne soit pas le cas. Dans le corpus de l'anglais "approximately 10 billion" est considéré comme étant un nom propre, dans le corpus espagnol "Controles de la pesca más rigurosos" et dans le corpus français la lettre "l" ont été identifiés comme étant des entités nommées. Ainsi le résultat de la détection des entités n'est pas fiable. De plus la liste des lemmes de chaques langues contient des ponctuations, ce qui nous semble inutiles dans la cas du processus de lemmatisation.

3. Réalisation du vocabulaire

Pour la réalisation du vocabulaire, nous avons décidé de retirer les noms propres, stop words, les signes de ponctuation et les espaces. Pour cette étape, nous avons mis en pratique deux méthodes: une qui a nécessité la création d'une fonction de notre part et une autre qui utilise la librairie *NLTK (Natural Language Toolkit),* que l'on qualifie de méthode automatique.
Dans les deux cas, une liste du vocabulaire sera renvoyée et enregistrée au format JSON.

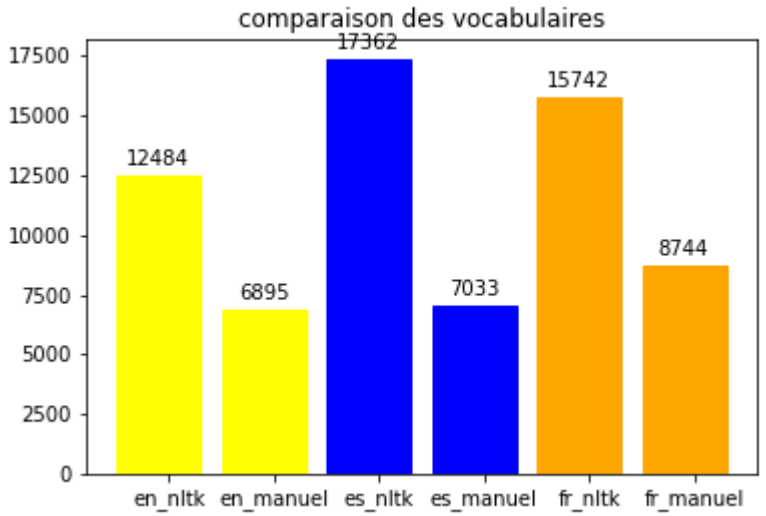
a. Manuellement

Nous avons utilisé le pos-tagging, cela permet de sélectionner les éléments que nous souhaitons retirer pour créer notre vocabulaire. En effet, en assignant à chaque

token une partie du discours ou classe grammaticale (*Part Of Speech*) on peut exclure les noms propres, déterminants ou encore les auxiliaires par exemple. On a créé la fonction *recuperer_vocab_manu* avec laquelle nous avons procédé au pos-tagging de chaque corpus et qui par la suite parcourt la liste de pos-tagging et vérifie l'étiquette *Part Of Speech* de chaque mot: s'il s'agit d'une étiquette qui n'est pas PROPN (nom propre), SPACE (espace vide), PUNCT (ponctuation) et que le mot n'est pas un stop-word alors ce mot sera ajouté à notre liste de vocabulaire. Malgré un fonctionnement exact de cette fonction, on a décidé de trouver une autre méthode plus simple et rapide à mettre en place,et que l'on pourra, par la suite, comparer à la manière dite manuelle.

b. Automatiquement

Pour cette méthode dite automatique, il a fallu télécharger la librairie *NLTK* et en importer les *stopwords* pour les différentes langues nécessaires. Les *stopwords* de la librairie *NLTK* sont un set regroupant un ensemble de mots. Ainsi, contrairement à la méthode manuelle, au lieu d'inclure dans une liste les éléments voulus, on élimine ceux présents dans la liste des *stopwords* pour ne garder que le vocabulaire des textes. Cette étape se fait au sein de la fonction *recuperer_vocab_auto.*

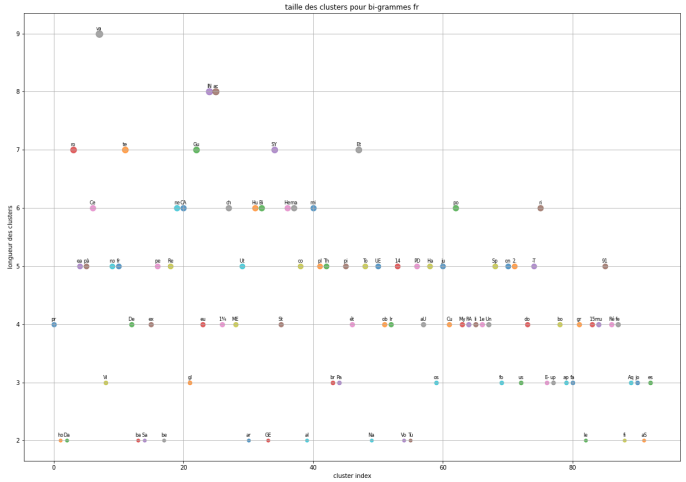


Nous pouvons observer que les vocabulaires réalisé avec la librairie *NLTK* sont plus longs que les vocabulaires générés manuellement. Après avoir effectuer une observation visuelle,cela s'explique par le fait que les vocabulaires automatiques contiennent des nom propres et des ponctuations contrairement aux vocabulaires manuels qui n'en possèdent pas. Bien que les vocabulaires générés automatiquement soient plus longs, ils sont plus rapides à être réalisés par la machine, ainsi pour l'anglais le temps d'exécution du vocabulaire automatique est de 0,087 seconde et 0,32 seconde pour le vocabulaire manuel.

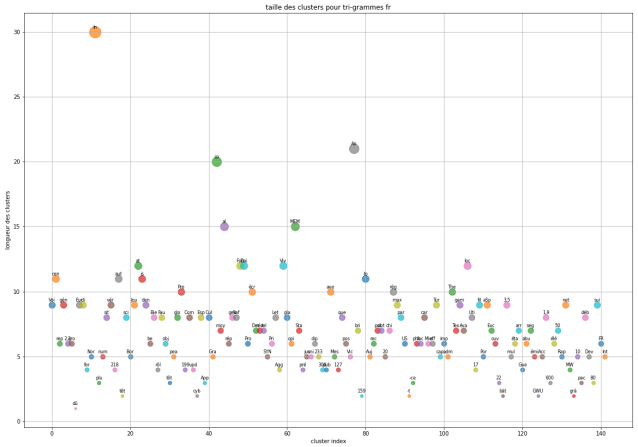
4. Partitionnement des données en n-grammes

Nous avons adapté le programme de clustering utilisé lors d'un TD précédent afin de permettre à l'utilisateur de spécifier le nombre de chiffres à partir desquels les mots seront segmentés.

a. Partitionnement en bigrammes et trigrammes
   Nous avons effectué le clustering des bigrammes et trigrammes à partir du vocabulaire français généré manuellement.

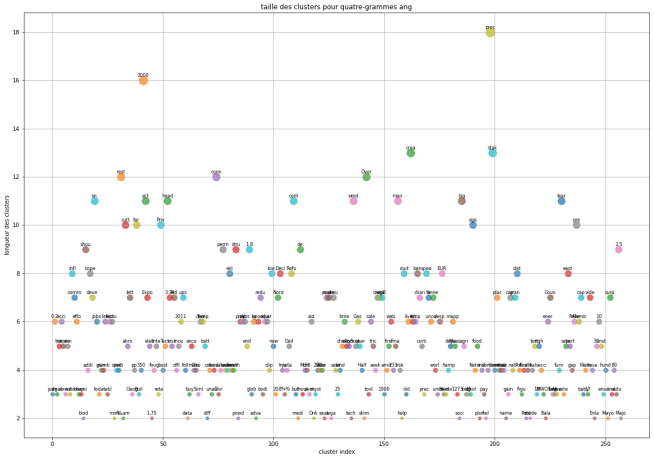


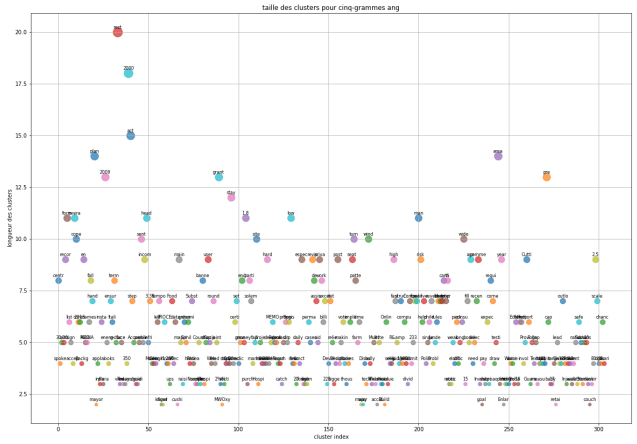
   Nous remarquons que la taille des clusters pour les bigrammes est majoritairement de 4 ou 5 termes.



   La taille des clusters pour les trigrammes est entre 5 et 10 termes.

b. Partitionnement en 4 grammes et 5 grammes
   Nous avons exécuté le clustering des 4 grammes et 5 grammes avec le vocabulaire anglais réalisé manuellement.





Nous pouvons observer que plus le chiffre à partir duquel les mots sont segmentés est grands plus il y a de clusters. On peut attribuer cela au fait que en moyenne la longueur des mots est d'environ 5,2 lettres en anglais et 5,4 lettres en français[1]. Ainsi plus le partitionnement se rapproche de 5 plus il y aura de mot proches, donc les clusters seront plus grands.

[1] https://www.inter-contact.de/en/blog/text-length-languages

III. Conclusion

Finalement, la réalisation de ce projet nous a permis de tirer plusieurs enseignements. L'utilisation d'outils et de traitement de texte en Python, nous a permis de mieux comprendre la richesse lexicale des langues choisies. De plus, nous avons pu identifier et approfondir nos recherches pour mieux comprendre la composition des mots en appliquant le clustering.