“

Groupe 5 : Filtrage

BENOIS Elise, BEZY Cédric, BRANDAO Clément,
BRIENS Maxime, CAMINADE Adrien, COGNIN Tom,
LAFAURIE Paul, PINSON Aurélien

GROUPE n°5
Projet inter-promotion
Année universitaire : 2017/2018

S O M M A I R E

1. Objectif du groupe

1.1. Intervention au sein du projet

Ce projet a pour but d'aboutir à un site web qui permettra de récupérer des articles de divers journaux. Ces derniers serviront à connaître les opinions positives ou négatives sur un sujet donné et de pouvoir avoir une analyse spécifique à la recherche de l'utilisateur. Afin de parvenir à ce résultat, nous avons été réparti en différents groupes, disposant chacun d'un objectif défini et précis. Au total, dix groupes ont été constitués, dépendant les uns des autres afin de parvenir à mettre en place une base de données et un site optimisé.

La mission de notre groupe consistait à réaliser le filtrage du texte. Ce traitement se compose d'un nettoyage du texte (suppression des stop-words, des signes de ponctuation) avec une lemmatisation des mots afin de les regrouper en fonction de leur racine. De plus, nous effectuons par la suite un "POS-tagging" dans le but de connaître la nature du mot et une recherche d'entité nommé.

Ce travail a été utilisé par deux groupes afin de réaliser une analyse sémantique du texte mais aussi à effectuer de la prédiction. Le travail de POS-tagging et de lemmatisation de notre groupe renvoie des informations utilisées afin de rendre des analyses sur la positivité d'un article ou encore permettre la prédiction statistique d'événement. Auparavant, nous avons reçu les articles de journaux par le serveur. Ces articles étaient au format "*json*". Ces derniers ont dû être renvoyé au serveur et à la base de données, après leur passage dans notre système de filtrage pour leur donner un format adéquat afin de faciliter leur utilisation par les autres groupes.

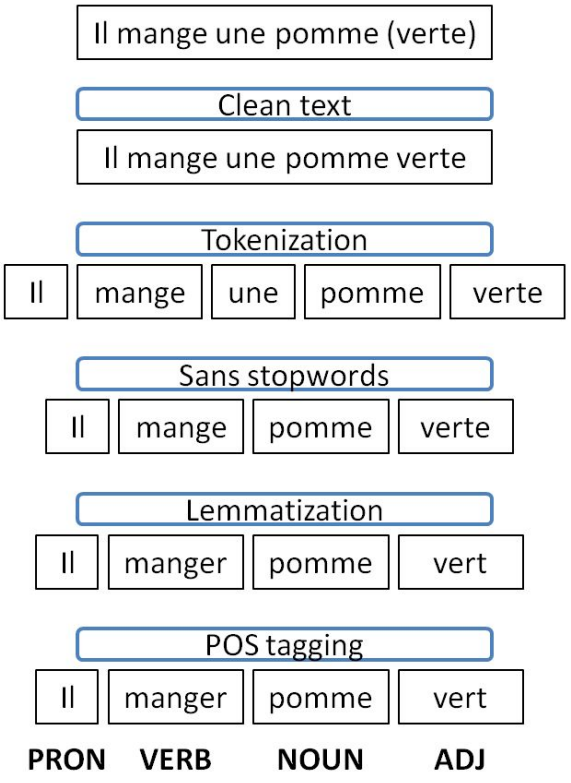
1.2. Explication globale de la mission principale

Après avoir réussi à importer les articles au format json de la base, notre mission consiste à effectuer toutes les étapes du *Natural Language Processing* (NLP), afin de fournir aux groupes d'analyse un texte correctement filtré. Une première version brute a été fournie rapidement, puisque l'avancement de certains autres groupes dépend de notre filtrage.

Par la suite, nous avons enrichi la version initiale du filtrage pour obtenir une version plus propre et au bon format. Nous avons donc amélioré au mieux le système de nettoyage des textes à l'aide de la suppression des signes de ponctuation et d'expressions régulières. Nous avons également perfectionné la tokenization en utilisant une librairie plus efficace avec la langue française (librairie spacy). Après coup, nous avons composé une liste de stop-words plus étoffée et nous avons amélioré le processus de lemmatisation et de POS-tagging afin de permettre des analyses plus probantes par les autres groupes. D'autre part, nous avons géré les problèmes d'importation et d'exportation des données dans la base.

Le Natural Language Processing (NLP) se compose de plusieurs étapes. Dans un premier temps, le texte reçu subit un nettoyage afin qu'il puisse perdre ses éléments de ponctuation. Puis, dans un second temps, nous segmentons le texte en une liste composée de chaque mot de l'article. Les mots inutiles sont expulsés de la liste, puis à l'aide de la lemmatisation, une autre liste ressort avec la racine de chaque mot utilisé dans le texte. Le POS-tagging permet de savoir la nature du mot.

NB: Nous conservons les points dans les textes et transformons tout les symboles de fin de phrase en point, afin de conserver le découpage original du texte, nécessaire à l'analyse sémantique du groupe 6.



2. Organisation

2.1. Répartition du travail (humain)

2.1.1. Constitution du groupe et répartition des tâches

Le groupe est constitué de

- Elise Benois : stop-words, tf, rapport ;
- Cédric Bezy : communication, import export json, filtrage, rapport ;
- Clément Brandao : SpaCy, intégration, tf-idf, qualité, gestion des formats de sortie ;
- Maxime Briens : gestion, communication, intégration, serveur ;
- Adrien Caminade : nltk, expression régulière, rapport ;
- Tom Cognin : SpaCy, tf-idf, qualité, stop-words, gestion des formats de sortie ;
- Paul Lafaurie : filtrage, optimisation, intégration, tf-idf ;
- Aurélien Pinson : SpaCy, expression régulière, rapport.

2.1.2. Planning Prévisionnel

	Lundi	Mardi	Mercredi	Jeudi	Vendredi		Lundi	Mardi	Mercredi	Jeudi	Vendredi
lemmatisation & tokenisation (NLTK)	X	X									
Amélioration de la lemmatisation & tokenisation	X	X	X								
POS-tagging & entités nommées		X	X	X	X						
Filtres stopwords & mots rares		X	X	X	X						
Mesures de qualité du filtrage			X	X			X	X	X	X	
Integration & qualité				X	X		X	X	X	X	
Préparation des POST				X	X		X	X	X	X	X
Automatisation des processus (CRON) - Publication sur serveur				X	X			X	X	X	X
Expressions régulières pour améliorer le filtrage				X	X		X	X	X		
Rédaction du rapport							X	X	X	X	X
Préparation du point											X

2.1.3. Travail Préparatoire

Le travail préparatoire a servi à définir les différents axes de travail lors du projet inter-promo de 2018. Cela consiste à détailler toutes les techniques et outils sur lesquelles s'appuyer pour réaliser les tâches relatives au filtrage de texte qui seront effectuées durant le projet. Nous avons convenu l'installation de plusieurs logiciels tels que Anaconda ou GitKraken mais aussi des librairies qui nous ont été utiles par la suite. Anaconda est un logiciel zźproposant un environnement Python. GitKraken, quant à lui, nous a permis de partager le travail effectué par l'ensemble du groupe. De plus, ce logiciel permet d'enregistrer toutes les versions réalisées. Ensuite, nous avons commencé à nous renseigner sur la librairie NLTK mais également sur ses alternatives. Ce travail préparatoire a permis d'avoir une visualisation plus précise du travail à faire ainsi que d'obtenir des exemples de NLP.

2.1.4. Planning Effectif

	Lundi	Mardi	Mercredi	Jeudi	Vendredi		Lundi	Mardi	Mercredi	Jeudi	Vendredi
lemmatisation & tokenisation (NLTK)	X	X									
Amélioration de la lemmatisation & tokenisation	X	X	X								
POS-tagging & entités nommées		X	X	X	X						
Filtres stopwords & mots rares		X	X	X	X						
Mesures de qualité du filtrage			X	X			X	X	X	X	
Integration & qualité				X	X		X	X	X	X	
Préparation des POST				X	X		X	X	X	X	X
Automatisation des processus (CRON) - Publication sur serveur				X	X			X	X	X	X
Expressions régulières pour améliorer le filtrage				X	X		X	X	X		
Rédaction du rapport							X	X	X	X	X
Préparation du point											X

Comme nous pouvons le voir sur le planning effectif, nous avons plutôt pris de l'avance durant la première semaine. Nous avons profité de cette avance pour intégrer une version améliorée de filtrage dès la version 1, basée sur le package SpaCy.
En revanche, la seconde semaine ayant été ponctuée de problèmes de communication entre les différents groupes et d'intégration dans la base de données, nous n'avons pas pu passer autant de temps que prévu pour rechercher des améliorations possibles des processus de lemmatisation.

2.2. Outils utilisés

Durant ce projet, nous avons utilisé les outils suivants :

- **Python** utilisé pour la programmation. Nous avons utilisé les packages ci-dessous :
  - NLTK : traitement automatique des langues ;
  - spaCy : niveau avancé de traitement de langage ;
  - re : transformation des chaînes de caractères par expressions régulières ;
  - pickle : sauvegarde de données sous forme de fichiers ;
  - datetime : format de date (pour obtenir la date du jour);
  - numpy : manipulation de structures de données (liste, dictionaire) ;
  - json : création et lecture de fichiers au format json ;
  - os : gestion de dossiers et fichiers;
  - tqdm : barre de progression lors d'une exécution.
- **Github** : site web permettant de gérer les projets et de poster les fichiers.
- **Gitkraken** : interface graphique permettant d'utiliser Git et d'accéder au projet Github afin de partager les dernières modifications.
- **Putty / WinSCP** : application SSH utilisée pour accéder au serveur.

2.3. Agencement des fichiers

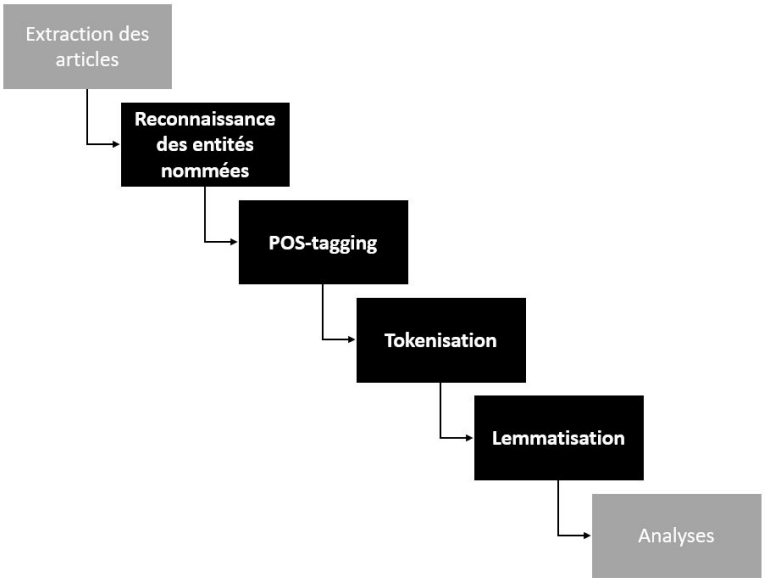
Afin de manipuler les fichiers à partir de Github identiquement pour chaque collaborateur, nous avons créé une arborescence similaire à celle du serveur. L'agencement de nos fichiers devait respecter le chemin **Data/source_press_article/[date_import]/[newspaper]/[article]** afin de tester le bon fonctionnement sur nos machines. Ce chemin est modifié par le chemin du serveur avant l'intégration sur ce même serveur.

Au même niveau que le dossier **Data**, nous avons donc placé la fonction **main**. Cette fonction a pour but de récupérer les fichiers .json dans la partie **Data**. Ensuite, le filtrage de texte est appliqué. Une fois que le filtrage de texte est réalisé, nous renvoyons les données dans le format adéquat que ce soit dans le serveur ou dans la base de données.

Enfin, dans notre dossier **Functions**, nous retrouvons en tout sept fichiers servant pour le filtrage. Nous les détaillerons ultérieurement.

3. Déroulement de la mission principale

3.1. Développement brut et immédiat



Afin de ne pas bloquer d'autres groupes, nous avons dû développer un système de filtrage dès le début du projet. Cette première version n'était pas optimale et a été améliorée par la suite. Ce n'était qu'un filtrage brut du texte qui possédait plusieurs imperfections. Pour cette méthode, nous avons uniquement utilisé la librairie NLTK. Cette librairie est simple d'utilisation cependant elle est plus ou moins complète pour la langue française. Elle permettait d'effectuer toutes les tâches du filtrage, de la reconnaissance d'entités nommées à la lemmatisation (voir ci-dessus).

3.2. Elaboration d'une version plus aboutie

Pour la version plus élaborée, nous avons utilisé essentiellement la librairie SpaCy. Elle possède plusieurs modèles d'analyse de la langue française. De plus, sa reconnaissance d'entités nommées est plus aboutie car par exemple, elle ne prend pas en compte tous les mots avec une majuscule. En effet, le premier mot de chaque phrase n'est pas nécessairement reconnu contrairement à NLTK. En outre, Spacy possède une méthode de

traitement de texte plus orientée objet plus commode à manipuler que les arbres de NLTK. En ce qui concerne l'intégration, nous avons fait face à de nouvelles problématiques avec

l'implémentation sur le serveur avec de nouveaux codes et le temps d'exécution à limiter. Nous avons également effectué différents exports selon les groupes de réception avec la gestion des formats de sorties.

Désormais, nous allons vous présenter les différentes script que nous avons créé.

- Le script **g5_import_json** permet d'importer les articles de type "*json*" provenant du robot du groupe 4 ayant la structure suivante : [date / chemin-source / Journal / Texte de l'article]. Grâce à la librairie « datetime », nous pouvons obtenir la date du jour afin d'extraire, chaque jour, les nouveaux articles. Cette fonction a comme sortie un dictionnaire d'articles.

- Le script **g5_tokenize** prend en entrée le texte de l'article et le retourne en séparant les différents mots (tokenisation). En sortie, nous avons donc une liste de mots individuelles de type token sur lesquelles nous pouvons effectuer des analyses de POS-tag, lemmatisation, etc.

- Le script **g5_clean_text** est indispensable pour gérer les ambiguïtés qu'impliquent tous les symboles et la ponctuation d'un texte. Par exemple, la phrase suivante est composée d'apostrophes et d'un nombre séparé d'un espace : « Aujourd'hui, j'ai 200 500 euros. ». Si on utilisait une tokenisation basique, elle deviendrait : "Aujourd" / "hui" / "," / "j" / "ai" / "200" / "500" / "euros". La fonction supprime les symboles et les ponctuations qui ne sont pas voulues. Cependant, elle utilise les expressions régulières afin de transformer une majorité de mots ou nombres pouvant être mal filtrés. Après utilisation, notre exemple devient : "Aujourd_hui" / "j" / "ai" / "200_500" / "euros". En reprenant l'exemple ci-dessus, on constate que le mot «aujourd_hui » est considéré comme un seul token. De plus, nous remarquons que la virgule est supprimée, et enfin, le séparateur de milliers de "200_500" est remplacé par un underscore et le nombre est considéré comme une seule entité.

- Le script **g5_database_posts** prend en entrée le texte tokenisé et retourne deux listes. Une des listes comporte chaque mot tokenisé et la seconde comporte le mot tokenisé avec sa nature. Par exemple, [«chat», « manger »] devient [« chat », «manger »][« noun », « verb »].

- Le script **g5_handing_entity** prend en entrée le texte tokenisé et construit des liaisons entre les mots considérés comme entité. Par exemple, [«Steve » « Jobs »] devient [« Steve Jobs PERSON »]. Cette fonction renvoie en sortie une liste d'entité

nommée et une liste d'entité nommée avec un underscore à la place d'un espace pour que chaque entité nommée soit reconnue comme un et un seul token.

- Le script **g5_integration** a pour but de créer un dictionnaire à l'aide du fichier des stop-words créé en parallèle. Ce module permet de prendre en compte soit l'intégralité des informations de l'article, soit uniquement le texte. Sa sortie dépend de ses paramètres d'entrée. Si on ne prend que le texte, la sortie est un dictionnaire dont chaque compartiment contient des informations pour chaque mot. Sinon, elle correspond à une liste des mots lemmatisés avec leur POS-tag correspondant. Ce résultat est obtenu en combinant l'utilisation des fonctions clean_text, pos, et named_entity.

- Le script **g5_tfidf** permet de calculer le TF-IDF, relatif à l'importance d'un mot dans un texte. Nous avons dans un premier temps construit la fonction TF (qui compte le nombre d'occurence d'un lemme dans un document) qui est utilisé dans deux situations. En effet nous avons utilisé la fonction TF pour les calculer sur un maximum d'article (environs 10 000) pour ensuite calculer les idf (log du nombre d'article contenant un mot). Cette liste d'IDF est stockée en format pickle.

  La deuxième utilisation de cette fonction, est celle qui se fera de manière continue lors de la mise en fonctionnement finale du site, se fait lors de l'appel de la fonction principale (main) lors du calcul du TF-IDF des mots de l'article que l'on parcours, en multipliant son tf (dans l'article parcouru) par l'idf correspondant dans le pickle.

- **g5_main** est le seul script présent en dehors du fichier « functions ». Cette fonction nécessite l'utilisation de g5_import_json, g5_tag-text et g5_tfidf. Elle récupère les données de type json dans le fichier Data et applique tous les traitements de filtrage. Nous y retrouvons le calcul des tf. Cette fonction renvoie les informations traitées selon le format demandé sur le serveur ou sur la base de données. Dans un premier temps, cette fonction permet d'importer un par un chaque fichier json et filtre le contents de l'article ainsi que son titre. Ensuite, elle calcule le tf associé aux mots du texte et finalement exporte les informations sous différents formats détaillés en annexe.

4. Missions complémentaires

4.1. Optimisation des fonctions

En outre l'application des méthodes de lemmatisation et de tokenisation, il est nécessaire d'appliquer d'autres filtres sur le texte. Nous avons constitué une liste de stop-words. Cette liste rassemble tous les mots qui sont très ou trop fréquents, et qui n'ajoutent pas de sens réel, et brouilleraient une analyse statistique. Par exemple "le", "la", "de", "du", "ce", etc.

Pour effectuer ce filtre, nous avons dans un premier temps utiliser le package NLTK. Cependant, certains mots tels que "les" n'apparaissaient pas dans les stop-words. Afin de compléter cette liste de manière efficace, nous avons choisi de travailler avec le package SpaCy. Alors, nous avons une liste de stop-words complète qui répond à nos attentes.

Afin de permettre la compréhension de notre code par nos partenaires de travail et d'améliorer le travail en équipe avec les autres groupes, nous avons commenté une grande partie de notre code et respecté la charte PEP 8 demandée par le groupe 10 (Qualité). Par exemple, nous avons placé le numéro de notre groupe devant chacune de nos fonction. De plus, nous avons évité la présence de chiffre dans nos variables mises en anglais. D'autre part, nous avons aéré notre code et apporté plusieurs modifications de lisibilité. Pour finir, nous avons optimisé le temps d'exécution du code. L'optimisation est surtout passée par l'utilisation de numpy (librairie codée en C) et de manière de coder inspirées du wiki python.

4.2. Gestion des demandes extérieurs

"Le **TF-IDF** (de l'anglais term frequency-inverse document frequency) est une méthode de pondération souvent utilisée en recherche d'information et en particulier dans la fouille de textes." *{Wikipédia}*. Les **TF** constituent le décompte d'un mot donné dans un article, et les **IDF** eux sont le résultat d'un log qui permettent de déterminer l'importance d'un mot: plus il est proche de 0, plus le mot est présent et donc importe que peu pour déterminer le sens de l'article, et plus il est élevé moins le mot apparaît et donc permet de préciser l'analyse sémantique du texte.

5. Difficultés rencontrées et solutions apportées

Durant ce projet, nous avons été confrontés à diverses difficultés sur les plans humains et techniques.

5.1. Problèmes humains

Parmi les quelques problèmes humains rencontrés, nous avons remarqué, au début, que nous ne pouvions pas travailler en même temps sur toute les parties du filtrage. En effet, des parties du traitement sont dépendantes les unes des autres ce qui a freiné leur développement.

De plus, nous avons rencontré des problèmes de communication avec les différents groupes en ce qui concerne le format des fichiers résultats JSON : en effet, entre les besoins de sorties de filtrage diffèrent parmi les groupes 2, 6 et 7, il a fallu trouver des solutions. Lors de la première semaine les formats exigés ont changés 2 à 3 fois par groupe, ce qui a ralenti notre livraison de la V1. De plus, nous avons été confrontés à la question : “quel groupe effectue le calcul des TF-IDF ?”. Question qui a peiné à trouver une solution, avant que la tâche nous soit attribué tardivement.

5.2. Problèmes techniques

Nous avons également rencontré diverses difficultés techniques, notamment l’installation du modèle pour la langue française de SpaCy sous Windows. En effet, il n’y a eu aucun problème ni sur MacOs ni sur Linux. Cependant, les collaborateurs utilisant Windows (la moitié du groupe) n’ont pas pu installer la version française, et par conséquent, n’ont pas pu travailler sur les fonctions nécessitant SpaCy, ni exécuter l’intégralité du code. Aussi, les tâches liées à l’utilisation de SpaCy ont été attribué aux collaborateurs disposant de la version française de SpaCy.

Nous avons également eu des difficultés avec le réseau Internet qui n’est pas très performant, ni régulier, à l’étage du projet, en plus d’être souvent saturé particulièrement le matin. Ce qui a été un problème pour les Push/ Pull sur Github et pour la recherche de solutions efficaces sur Internet. Nous avons donc utilisé nos téléphones afin de faire des points d’accès et être plus efficaces.

Enfin, nous avons eu des problèmes d'utilisation de Gitkraken, notamment au début du projet où nous ne savions pas nous en servir efficacement et avons généré de nombreux conflits par de mauvaises manipulations.

6. Conclusion

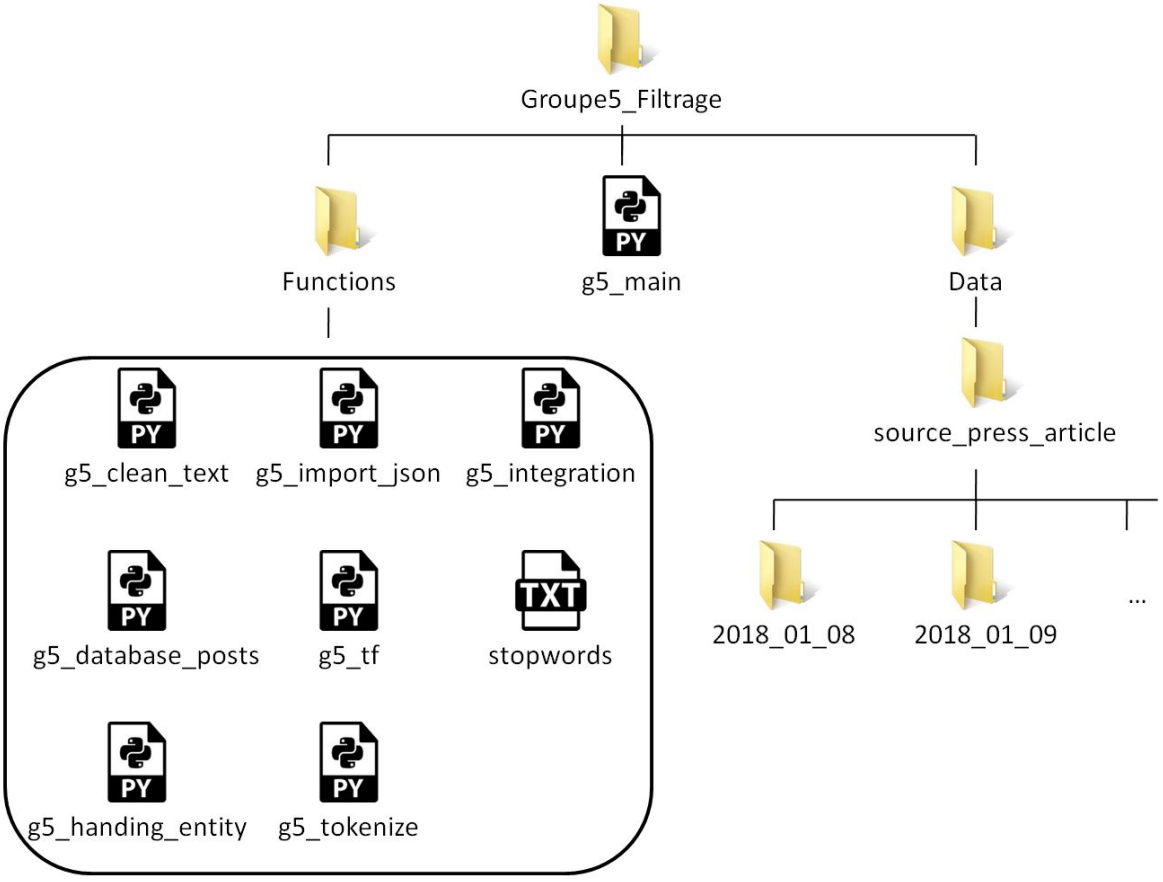
Pour conclure sur ces deux semaines de projet, nous avons eu un départ un peu tumultueux lorsqu'il s'agit d'échanger avec les groupes d'avant et d'après nous dans la chaine du projet, mais grâce aux interventions de Maxime avec les autres chefs de projet, notre environnement de travail a pu rester cordial, confortable et nous avons pu avancer sereinement vers les objectifs que nous nous étions fixés.

En ce qui concerne le travail effectué, nous avons respecté les tâches prévues lors du travail préparatoire. Cependant, la deuxième semaine a été plus compliquée dans la réalisation des tâches restantes. En effet, nous avons rencontrés certains problèmes avec l'intégration des données. Cela nous a empêché de nous concentrer sur l'amélioration de processus de lemmatisation.

Dans l'ensemble, le projet s'est déroulé correctement grâce à une bonne ambiance ainsi qu'une bonne implication du groupe.

7. Annexes

Architecture des fichiers



Format pour la base de données

```
[
  {
    "article":
        { "date_publication": "2017-05-24", "name_newspaper":"FIGARO",
        "surname_author": ["Patrick","George"]},
    "position_word":
    [
        {"POS_tag": "test1","type_entity": "PERSONNE","word": "bonjour1",
       "lemma" : "test1", "position":1, "title": true },
       { "POS_tag": "test2", "type_entity": "PERSONNE","word": "test4",
        "lemma" : "test2", "position":2, "title": true }
    ]
  }
]
```


Format pour le groupe sémantique


```
{
"id_art": "1951626bf77e415ea102ad04f1f9901f",
"title": "Emmanuel Macron veut <<un plan penitentiaire global>> ",
"newspaper": "Liberation",
"author": [" LIBERATION  avec AFP"],
"date_publi": "2018-01-15",
"content": {"1": {"word": “Prison", "lemma": "Prison", "POS_tag": "NOUN",
                    "type_entity": "MISC", "position": 1, "title": false},
            "2": {"word": "ou", "lemma": "ou", "POS_tag": "STOPWORD",
                    "type_entity": "", "position": 2, "title": false},
                    "words": ["Prison", "ou",...]},
"theme": " "
}
```