Rapport de projet

*Analyse automatique de verbatim*

Ce document présente mon travail sur un projet informatique en deuxième année à l’ENSC

Paul LAURENT

28/04/2017

Table des matières

[Définition du sujet 1](#_Toc481067845)

[Contexte 1](#_Toc481067846)

[L’existant 2](#_Toc481067847)

[Définition du projet 2](#_Toc481067848)

[L’architecture du projet 3](#_Toc481067849)

[Les besoins de l’entreprise 3](#_Toc481067850)

[Le travail des données 3](#_Toc481067851)

[Les analyses 4](#_Toc481067852)

[Les données 4](#_Toc481067853)

[L’analyse fréquentielle 5](#_Toc481067854)

[Le programme 5](#_Toc481067855)

[Les tests 7](#_Toc481067856)

[L’analyse ontologique 8](#_Toc481067857)

[Le programme 8](#_Toc481067858)

[Les tests 9](#_Toc481067859)

[L’analyse par allocation de Dirichlet latente 11](#_Toc481067860)

[Le programme 11](#_Toc481067861)

[Les tests 12](#_Toc481067862)

[Le clustering hiérarchique agglomératif 13](#_Toc481067863)

[Le programme 13](#_Toc481067864)

[Tests 14](#_Toc481067865)

[L’interface 17](#_Toc481067866)

# Définition du sujet

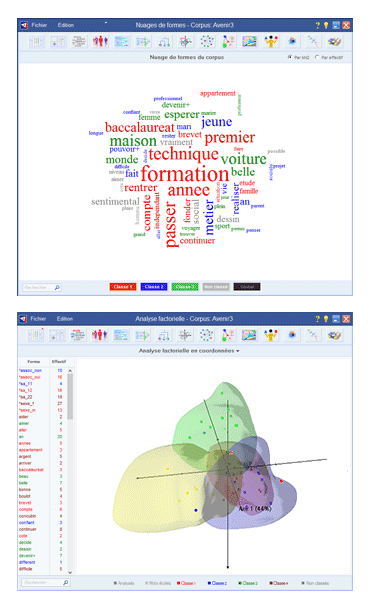
## Contexte

A l’approche des campagnes électorales de 2017, tout un travail de terrain est mené par des bénévoles de partis ou des instituts de sondage afin de comprendre les tendances, et de prévoir quelles idées sont plébiscitées par la population. Ce travail est plus largement utilisé lors de consultations citoyennes, qui permettent aux élus de connaître l’opinion, les attentes et les besoins de leurs administrés.

Lors de ces consultations, des personnels viennent questionner directement des citoyens. Les questions revêtent deux formes, les questions ouvertes et les questions fermées. Les questions fermées attendent comme réponse des choix prédéfinis (comme un QCM). Elles peuvent être analysées automatiquement simplement. Les questions ouvertes quant à elle laissent la personne questionnée développer ses idées et ne possède donc pas de réponse prédéfinie.

C’est dans ce contexte que l’entreprise Quorum cherche à développer un outil permettant de fournir à ses client une analyse automatique des réponses aux questions ouvertes.

## L’existant

Le traitement automatique du langage est apparu avec la capacité de calcul des ordinateurs. Les chercheurs voulaient pouvoir automatiser certaines taches comme la traduction. Reposant sur des méthodes statistiques il a été approfondi dans certains domaines comme la correction automatique de textes.

Aujourd’hui certains logiciels fournissent une analyse syntaxique de corpus et offrent une robustesse importante.

C’est le cas de Alceste

Ce logiciel issu d’un travail entre le CNRS et la société Image fourni une étude statistique de corpus Ce logiciel est programmé en R et python (pour respectivement les parties statistique et traitement).

L’entreprise Proxem est leader dans le traitement de données textuelles. Elle propose le traitement de données pour une meilleure satisfaction client, l’analyse du climat social et des capacités des employés, la cartographie du marché d’une entreprise et de l’analyse de contenus.

Le logiciel Statistica permet à son utilisateur de réaliser des études statistiques de ses données grâce à des fonctionnalités codées en VB.

De nombreuses études ont été développées ou sont en cours sur le traitement du langage sans que celles- ci n’ait était implémentées. Ces travaux concernent aussi bien des collocations (associations habituelles de mots), que la formation automatique de dictionnaires techniques.

## Définition du projet

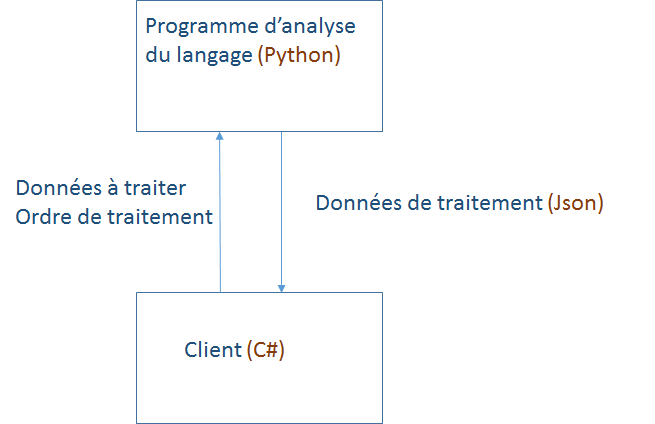
Vu les travaux qui ont été menés jusqu’ici par les développeurs et les chercheurs, vu les besoins de l’entreprise Quorum, il est apparu que mon travail s’axerait sur une analyse statistique des corpus fourni. Cependant, l’analyse statistique est limitée quant à son utilisation. Il est donc intéressant de travailler notamment sur de l’analyse sémantique des corpus afin de pouvoir extraire des thèmes qui auront été développés par la population questionnée.

Par conséquent, il a été décidé que mon travail serait axé sur :

* Une analyse statistique robuste (utilisable directement par l’entreprise).
* Une interface graphique qui me permette d’améliorer mes compétences.
* Une analyse sémantique aussi robuste que possible, afin que l’entreprise puisse finalement l‘utiliser.

# L’architecture du projet

## Les besoins de l’entreprise

Le projet s’inscrit dans une démarche de construction mutuelle. Ainsi je fourni à l’entreprise Quorum une recherche sur un sujet qu’il n’ont pas le temps de développer, tandis qu’ils m’apportent un cadre professionnel, et des critiques expertes sur mon travail. 

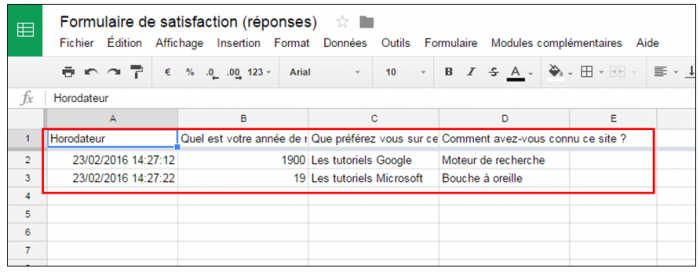
Le travail que je fournis doit pouvoir s’insérer dans la plateforme de Quorum. Il ne leur est donc pas nécessaire d’avoir une interface graphique. J’ai donc choisi de réaliser une interface en C#. Cette interface reçoit les commandes de l’utilisateur (notamment le recueil du corpus) envoie ces données au programme d’analyse du langage qui fournit à nouveau au client les résultats de l’analyse. Ces données seront fournies au format Json pour favoriser l’intégration au sein du système Qurorum.

Le choix du langage python pour l’analyse des données s’est fait sur la base de plusieurs critères. C’est un langage que je maitrise et dont plusieurs programmes d’analyse du langage naturel font l’utilisation comme Alceste. Il possède donc de nombreuses bibliothèques fournissant des outils robustes pour l‘analyse du langage. Par ailleurs, il s’agit du seul langage fournissant ces bibliothèques de traitements.

## Le travail des données

Le projet s’inscrit donc sur plusieurs niveaux. Le premier niveau concerne l’analyse du langage naturel.

Le programme analyse le corpus et ressort pour une même question les mots utiles qui ressortent le plus. Il renvoie au client une information de fréquence de mots. Il renvoie également les thèmes abordés dans la question pour les personnes interrogées.

Le second niveau concerne l’interface client. Cette interface doit permettre le téléchargement de données issues d’un Google form. Ces données sont représentées sous forme de tableau CSV. Le tableau de données téléchargé, l’utilisateur peut choisir les questions qu’il souhaite analyser. Le retour se fait par l’affichage des thèmes abordées.

# Les analyses

## Existantes

* LSA

    Anlayse sémantique latente: 62% de pertinence.

Cette analyse permet de comparer des corpus de texte pour connaître leur proximité sémantique. Chaque corpus est représenté par un vecteur V dont la dimension est le dictionnaire du corpus. Chaque composante du vecteur a une norme égale à l’importance du terme dans le corpus. Pour quantifier l’importance du terme dans le corpus on peut utiliser le TF-IDF. Cette matrice est simplifiée pour s’adapter aux ressources de l’odinateur. Il s’agit d’une méthode de pondération basée sur la fréquence d’apparition du mot dans le corpus. On se situe donc dans un espace vectoriel composé des différents corpus. On calcule la similarité cosinus (qui est une mesure de l'écartement dans l’espace vectoriel des corpus). Si deux corpus ne possèdent rien en commun, ils seront orthogonaux dans l’espace considéré donc la similarité sera nulle. Limitations : non traitement de la polysémie, on utilise le modèle du sac de mot (pas d’analyse grammaticale), et on se base sur une distribution gaussienne contrairement à la distribution de poisson qui a été observée.

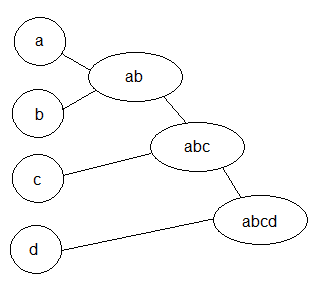
* PLSA:

Se base sur la LSA. Cependant, plutôt que de se baser sur une étude algébrique,

Il cherche à calculer la probabilité d'occurrence d’un mot dans un document, en fonction de la probabilité du sujet, de la probabilité du document sachant la probabilité du sujet et la probabilité du mot sachant le sujet.

* HAC:

clustering hierarchique : construction de groupes homogènes d’après un calcul de distance (ex distance euclidienne ou de malhanobis) et fusion de proche en proche des paquets crées



* Naive Bayes :

repose sur une catégorisation a priori de mots. On calcule à partir de ce corpus la probabilité d’un mot d’appartenir à une catégorie.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| doc/word | w1 | w2 | w3 | w4 | w5 |
| cat1 | 0.82 | 0.75 | 0.50 | 0.11 | 0.20 |
| cat2 | 0.18 | 0.25 | 0.50 | 0.89 | 0.80 |
| doc3 | 2 | 0 | 1 | 0 | 0 |
| doc4 | 0 | 0 | 1 | 4 | 1 |

On peut donc calculer la probabilité qu’un document soit dans une catégorie. ça se rapproche d’une approche ontologique.

* LDA

C’est un modèle génératif probabiliste dérivé du PLSA permettant de retrouver les thèmes abordés par un texte. L’algorithme suppose qu’un texte est un ensemble de sujet et que chacun des mots du texte est responsable de la création d’un des sujets.

L’algorithme prend tous les mots présents dans le corpus et l’assigne à l’un des K sujets cherchés et ce de manière aléatoire;

Puis on calcule la probabilité que le sujet k soit dans dans le document suivant le nombre de mots assignés au-dit sujet (proba (sujet| corpus)), ainsi que la probabilité que le mot soit celui considéré suivant le nombre de fois où il est assigné au sujet.

Puis on redonne au mot un nouveau sujet suivant la probabilité (proba (du sujet| corpus)\*proba(mot|sujet)

|  |  |  |
| --- | --- | --- |
| Algorithme | Avantages | Inconvénients |
| LSA | Algorithme d’algèbre linéaire, simple | Pertinence de (40-60%) |
| PLSA | version probabiliste de LSA | pertinence de 40-60% |
| HAC | pertinence de 84% | Pas d’implémentation en nltk |
| Naïve Bayes | Meilleure pertinence | Nécessite une intervention extérieure |
| LDA | en partie implémentée en python | Pertinence de 40% |

D’autres algorithmes existent mais se basent sur de l’intelligence artificielle.

Vu le volume de données qu’on possède, il est plus pertinent de passer par des algorithmes sans apprentissage, donc l’intelligence artificielle est exclue.

L’algorithme Naïve Bayes nécessite une intervention extérieure qui est également exclue.

Il est donc intéressant de développer les algorithmes LSA, PLSA, HAC et LDA. L’algorithme LDA étant une amélioration des deux autres, il n’est qu’utile de regarder les algorithmes LDA et HAC.

## Les données

Les entreprises partenaires de Quorum fournissent des données sous forme de fichier au format csv. Il s’agit de tableaux de données dont chaque colonne correspond à une colonne et chaque ligne à la réponse d’une personne.

Les questions sont donc soit des questions ouvertes, soit des questions fermées, soit des réponses à QCM.

Les questions fermées sont des questions dont les réponses sont « oui » ou « non », les QCM impliquent l’apparition de plusieurs réponses identiques,

Pour sélectionner les questions ouvertes, plusieurs options ont été envisagées : un travail sur les questions a été envisagé pour voir si certaines formes de questions pouvaient signifier qu’elles étaient des questions fermées. Mais les réponses aux QCM n’étaient pas supprimées puisqu’il s’agit de questions ouvertes pour lesquelles on propose des choix. C’est donc un travail sur les réponses qui a été entrepris.

Pour chaque question, on compare les réponses entre elles pour vérifier la présence de réponses identiques. Puisque l’on se trouve sur des échantillons suffisamment faibles, la probabilité de répondre de manière identique à une réponse est faible en cas de réponse ouverte.

Un critère discriminant est la présence de deux fois la même réponse pour une même question. Cependant, la présence d’éléments NaN dans le fichier compromet ce critère.

Plusieurs possibilités de sont possibles : soit supprimer ces éléments, soit assouplir le critère discriminant. La suppression des éléments NaN est possible grâce aux fonctionnalités du module pandas. La fonction dropna() supprime les lignes pour lesquelles un élément NaN est présent. Cependant ce n’est pas dans notre intérêt puisque l’on ne souhaite pas traiter les données de manière personnelle. Ainsi nos lignes n’ont pas d’intérêt en tant que tel. Ce qui importe ce sont les réponses aux questions et pas les corrélations entre les différentes colonnes : ie une réponse d’une personne affectée à une autre ne pose pas de problème. Ainsi les données n’étant pas forcément de bonne qualité, le test de cette fonctionnalité a montré la suppression de toutes les lignes du tableau de données. Une fonctionnalité du module numpy permet de supprimer un élément particulier. Cependant nous manipulons des listes de chaines de caractère, tandis qu’il considère NaN comme un entier. On lève donc une exception.

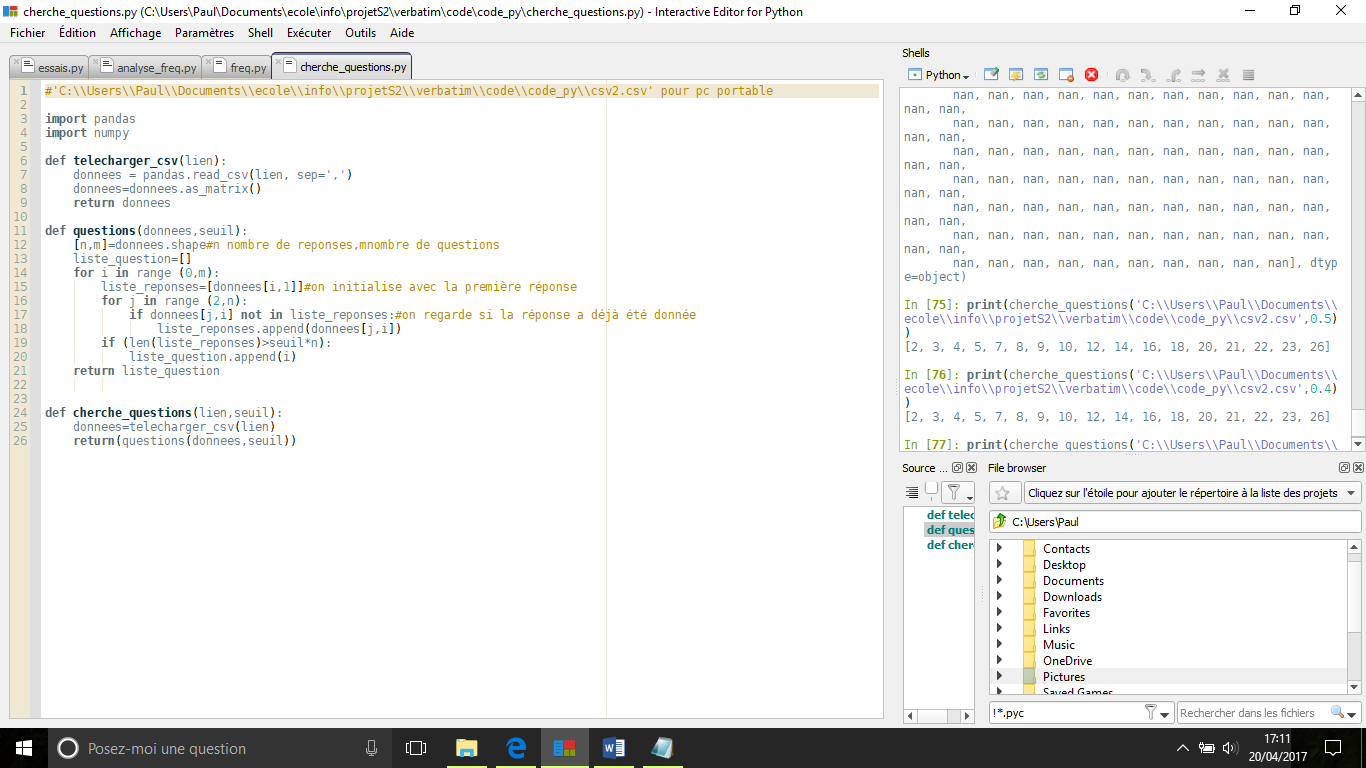
Enfin la conversion de l’élément NaN en chaine de caractère n’est pas possible. Il existe un moyen de supprimer ces éléments mais qui nécessite un traitement plus lourd et l’utilisation du module nltk. Ceci est utilisé dans la suite du projet.

L’utilisation d’un critère discriminant est plus flexible mais peut nécessiter son ajustement par un opérateur.

Dans mon cas, les questions 0,6,13,15,17,19,23 sont des questions fermées. De manière empirique, un seuil de 0,2 permet de discriminer les questions ouvertes à une exception prête : la colonne 23. Cette colonne ne contient en effet que des éléments NaN. Il s’agit probablement d’une erreur du fournisseur puisqu’il n’y a aucun intérêt à analyser une colonne vide. La suppression manuelle de cette colonne paraît être la solution la plus cohérente puisqu’elle est facilement identifiable.

Ainsi on vérifie que le nombre de réponses différentes est supérieur à 20% des personnes interrogées.

Toutes les réponses ouvertes ont été discriminées avec cette technique. Cependant elle implique une complexité calculatoire assez importante puisqu’il faut parcourir tout le tableau de données et réaliser des tests dessus. On se trouve face à une complexité calculatoire en o(n3).



## L’analyse fréquentielle

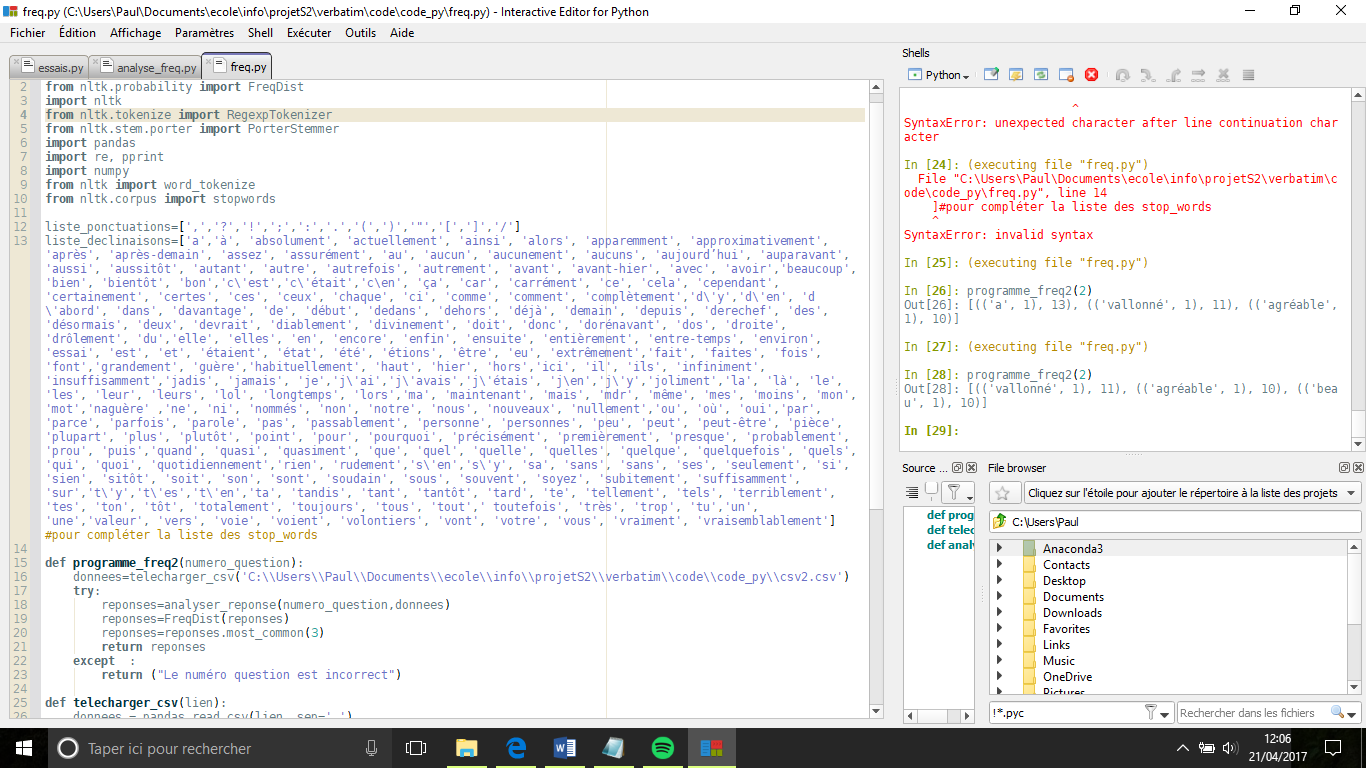
### Le programme

Une des pistes pour comprendre les points clefs des réponses des personnes est de faire de la sélection automatique de mots clefs. Cette sélection peut se faire par sélection fréquentielle. L’idée est de récupérer l’ensemble des mots répondus et de faire un tri fréquentiel.

Pour cela on utilise le module nltk qui permet de faire certaines opérations sur les chaines de caractères.

Après avoir télécharger le fichier csv, on ne s’occupe que de la colonne de la question considérée. On convertit chaque réponse en chaine de caractère pour éviter une exception dans l’utilisation du module. En effet, la présence d’éléments NaN va lever une exception. La chaîne de caractère est convertie en liste de chaine de caractère dont chaque chaine correspond à un mot. Ceci est possible grâce à la fonction word\_tokenize().

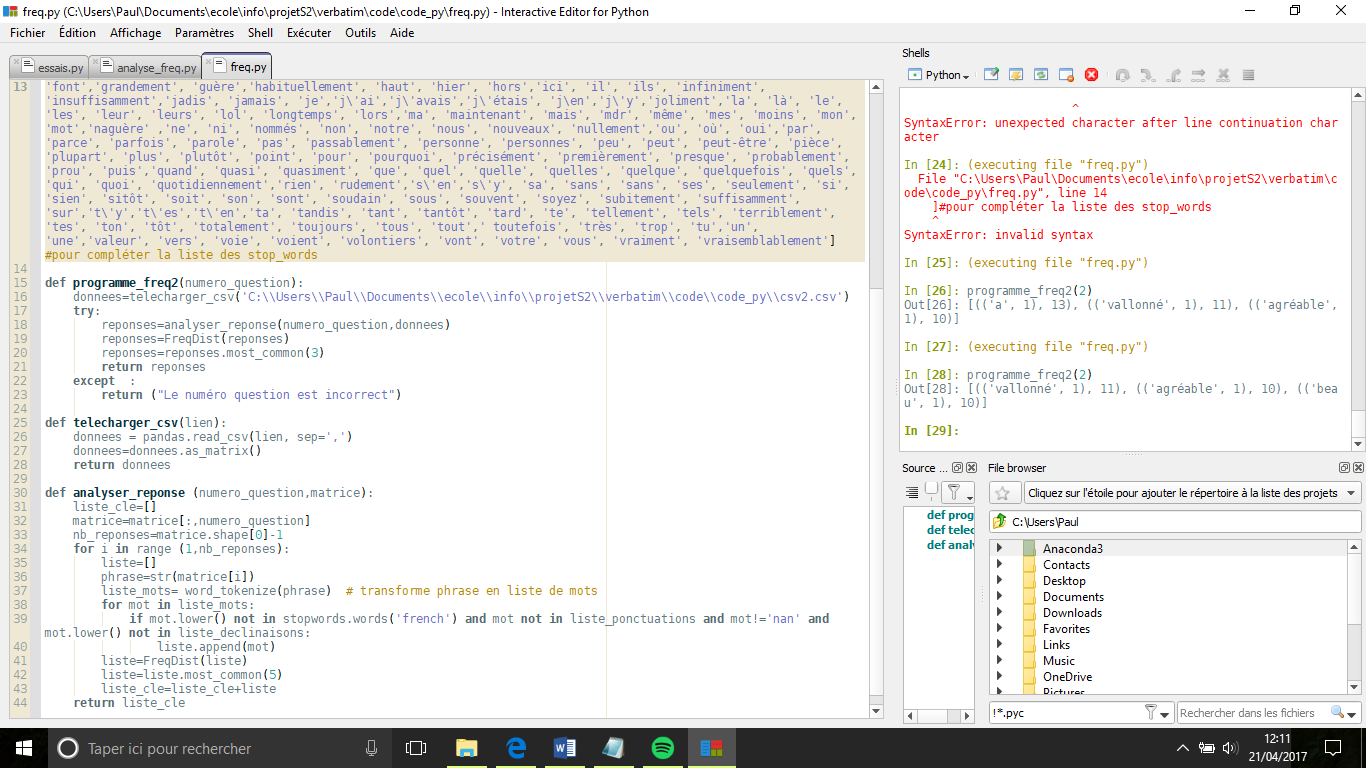
Tous les mots récupérés n’ont pas nécessairement d’importance dans la compréhension de l’opinion des personnes. En référencement cela s’appelle des stop\_words. La bibliothèque nltk propose des stop\_words pour la langue française. Cependant, après avoir trié les mots grâce aux stop\_words fournis, il est apparu qu’ils n’étaient pas suffisants. Premièrement la tokenisation isole la ponctuation, ensuite de nombreux mots ressortaient sans apporter de sens. La liste des stop\_words (qui contient principalement les conjugaisons des verbes « être » et « avoir ») a été complétée par une liste des ponctuations et une liste de stop\_words utilisés en référencement qui elle-même a été complétée pour s’adapter aux sorties que la tokenisation proposait. Ainsi certains mots qui pouvaient être porteurs de sens ont été enlevés de la liste, tandis que d’autres non spécifiés ont été ajoutés.



On stocke dans une liste, initialisée à chaque question les mots qui sont utiles à la compréhension et on réalise un tri fréquentiel grâce à la fonction freqDist(). On ne garde que les 5 premiers mots, puisqu’il s’avère que sélectionner plus de mots est inutile.

Les mots selectionnés sont ensuite stockés dans une liste qui contient les mots importants de chaque réponse. Cela permet d’éviter une pondération par la longueur de la réponse : une réponse plus longue, donc possédant un nombre plus important de mots n’impactera pas d’avantage la sortie du programme.

Cette dernière liste est enfin triée par la même fonction.



### Les tests

Pour évaluer la pertinence des résultats obtenus

|  |  |  |
| --- | --- | --- |
| Question | Analyse fréquentielle | Commentaire |
| Comment décrivez-vous ce territoire ? | Vallonnée, agréable, beau | Ok, analyse efficace |
| Qu’est-ce qui attire les nouveaux habitants | Campagne, calme, tranquillité | Ok, analyse efficace |
| Voiture, qu’en pensez-vous ? | Voiture, obligé, choix | Voiture n’apporte rien,  Obligé et choix peuvent faire référence à « je suis obligé je n’ai pas le choix » ou « je ne suis pas obligé, j’ai le choix » |
| Si vous aviez une baguette magique, que changeriez-vous ? | Jeunes, lumbres, territoire, enfants, commerces | On comprend que les jeunes doivent être pris en considération, une amélioration des commerces doit être envisagée. Cependant « Lumbres » et « territoire » ne dégagent une idée facilement identifiable |

On peut augmenter le nombre de mots pour éventuellement percevoir plus facilement le sens. Ce qui a été fait pour la dernière question.

Les tests montrent un que l’analyse fréquentielle peut être suffisante pour dégager des thématiques de certaines questions, mais certainement pas pour toutes. Certains mots peuvent ressortir sans dégager de sens particulier et d’autres soulèvent des ambiguïtés.

## L’analyse ontologique

### Le programme

L’analyse ontologique cherche non plus à détecter seulement les mots clefs mais également à trouver leurs champs ontologiques pour détecter les thématiques liées au réponses. Pour cela on se base sur l’analyse fréquentielle qu’on a réalisée auparavant, sur laquelle on réalise une comparaison avec une ontologie.

Plusieurs choix se sont posés : soit développer une ontologie particulière, soit en utiliser une existante. Il existe de nombreuses ontologies mais la plupart sont en anglais. Il est possible de trouver les principales (en libre accès) sur le site <http://labs.mondeca.com/>.

Cependant, les ontologies disponibles ne couvrent qu’un seul champ et aucune couvrant les thématiques rencontrées lors des consultations citoyennes n’est disponible librement en français.

Les thématiques ayant été dégagés (sécurité, fiscalité, éducation, alimentation, santé, environnement, transports, loisirs, sport…) il aurait été possible de réaliser des ontologies, mais il aurait été question d’un projet à part entière.

Il existe cependant une ontologie qui cherche à être la plus complète possible : celle développée l’université par de Princeton appelée Wordnet. Cette ontologie est en anglais et un équivalent français existe appelé Wolf.

Le terme de base du wolf est le synset qui est un groupe de mots interchangeables dénotant un sens.

Les synsets sont reliés entre eux par des relations de l'ontologie

A ce titre, chaque synset se compose :

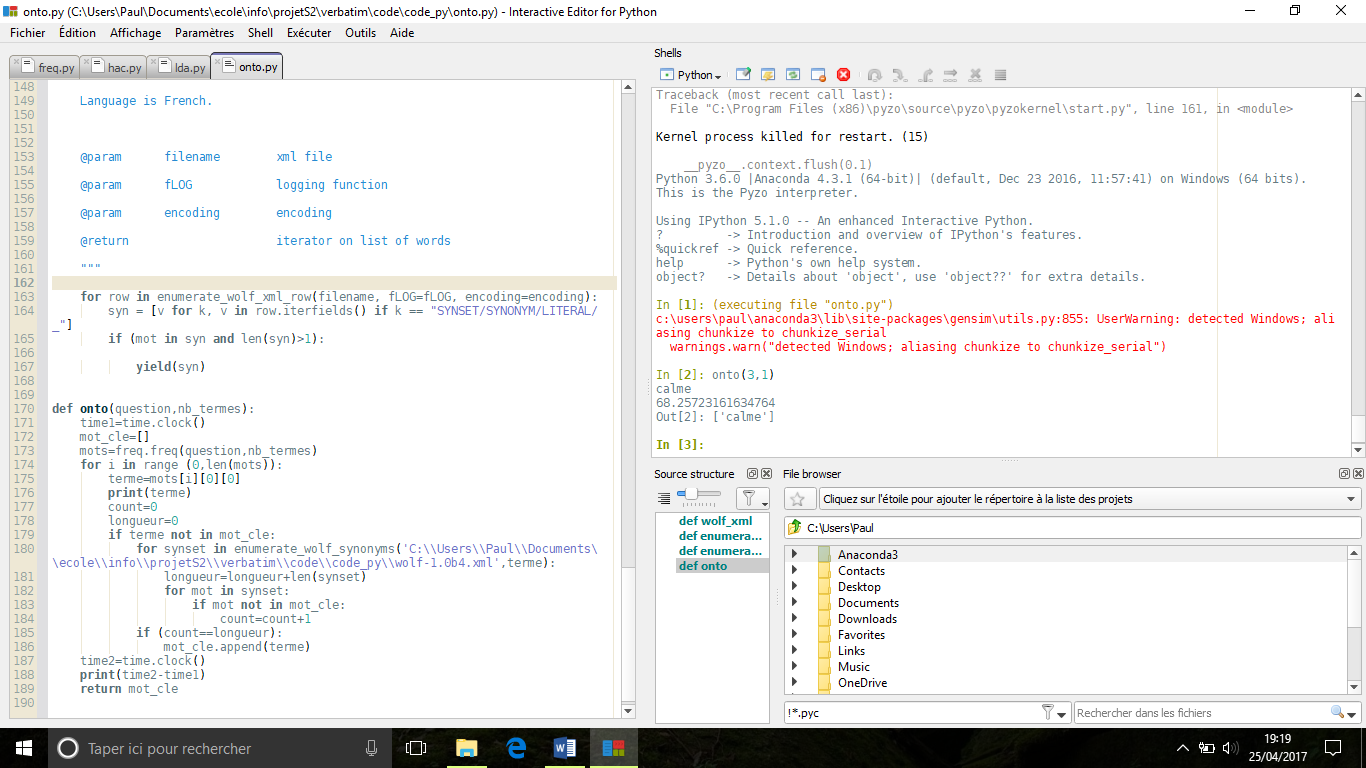
* D’un identifiant unique, présent dans sa sous balise ID. Cet identifiant est en fait le même que celui du synset original dans le Wordnet de Princeton.
* Une fonction grammaticale, présente dans la balise POS. Tous les mots composant un synset partagent la même fonction grammaticale.
* Une liste de mots (ou encore lexèmes), présente dans la balise Synonym. Elle représente la liste des mots composants le synset.
* Une liste de liens avec les autres synsets, présente dans la balise ILR. Les identifiants des synsets sont utilisés pour décrire les liens.
* Une note BCS, indiquant l’importance du synset. 1 : très important, 2 : important, 3 : relativement important, vide : peu important.
* Les autres balises indiquent, optionnellement, une définition, des exemples ou encore des renseignements par rapport à la traduction du synset depuis le Wordnet de Princeton.

Les relations existantes sont respectivement pour les noms et verbes les suivantes :

* Noms
  + Hyperonymie: Y est une hyperonymie de X si tous les X sont des types de Y (canin est un hyperonyme de chien)
  + Hyponymie: Y est une hyponymie de X si tous les Y sont des sortes de X (chien est un hyponyme de canin)
  + Termes coordonnés :Y est un terme coordonné de X si X et Y partagent une même hyperonymie (loup est un terme coordonné de chien)
  + méronymie : Y est un méronyme de X si Y est une partie de X (fenêtre est un méronyme de bâtiment)
  + Holonomie : Y est un holonome de X si X est une partie de Y (un bâtiment est un holonome de fenêtre)
* Verbes
  + Hyperonymie le verbe Y est une hyperonymie de X si l’activité X est un type d’activité Y
  + Toponymie: le verbe Y est une toponymie de X si l’activité Y fait l’activité X dans un certain sens
  + Implication: le verbe Y est impliqué par le verbe X si pour réaliser la tâche Y on doit faire la tâche X
  + Termes coordonnés: verbes partageant une même hyperonymie.

Le principe est donc de parcourir l’ontologie comme un fichier XML et de parcourir pour chaque terme le synset dont il fait partie. Ce synset comporte a priori les termes qui portent un sens commun à celui du mot. On compare donc les termes aux termes déja stockés pour vérifier qu’on n’ajoute pas un terme qui posséderait le même sens qu’un des mots déjà sélectionnés.

On parcourt donc les lignes du fichier XML, et on récupère tous les éléments présents dans les balises <SYNONYM>. Ces synonymes sont censés représenter des termes partageant un même sens.



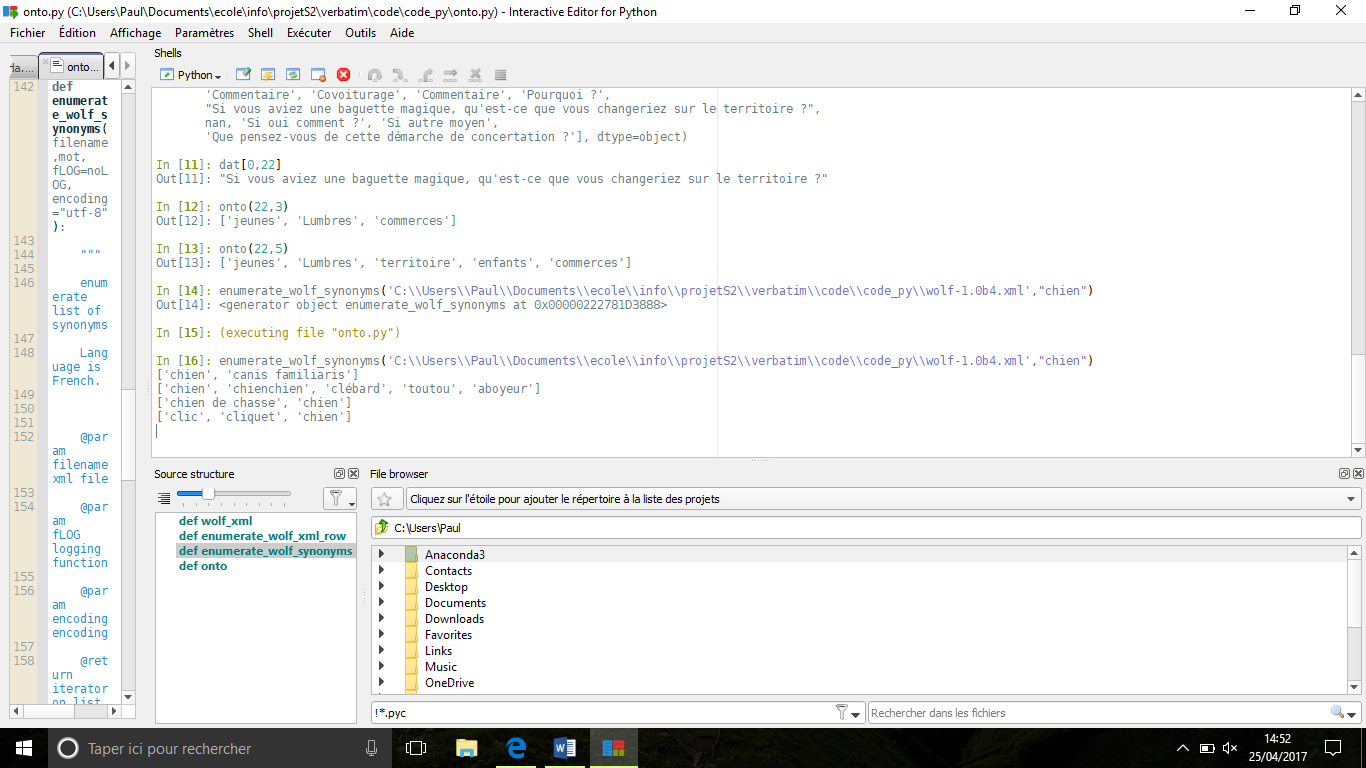
### Les tests

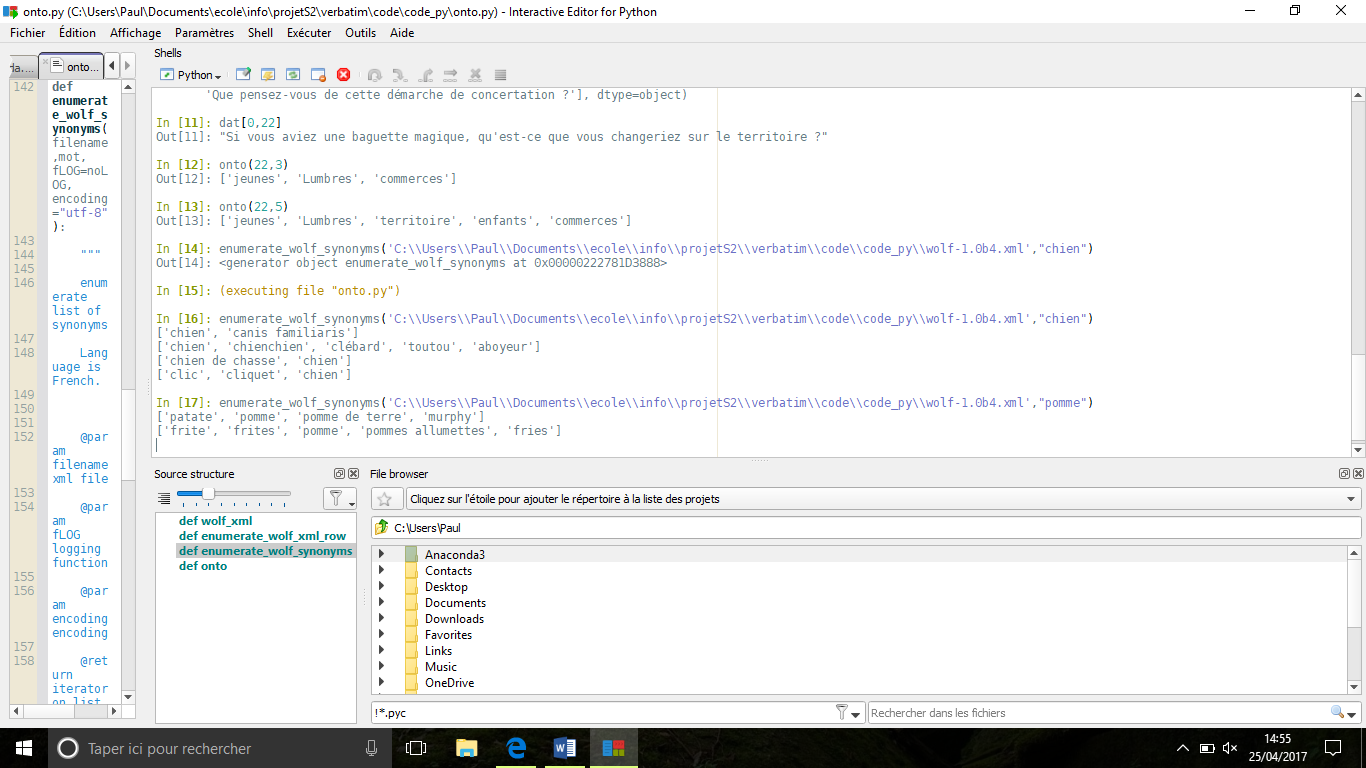
Pour évaluer la pertinence des résultats obtenus

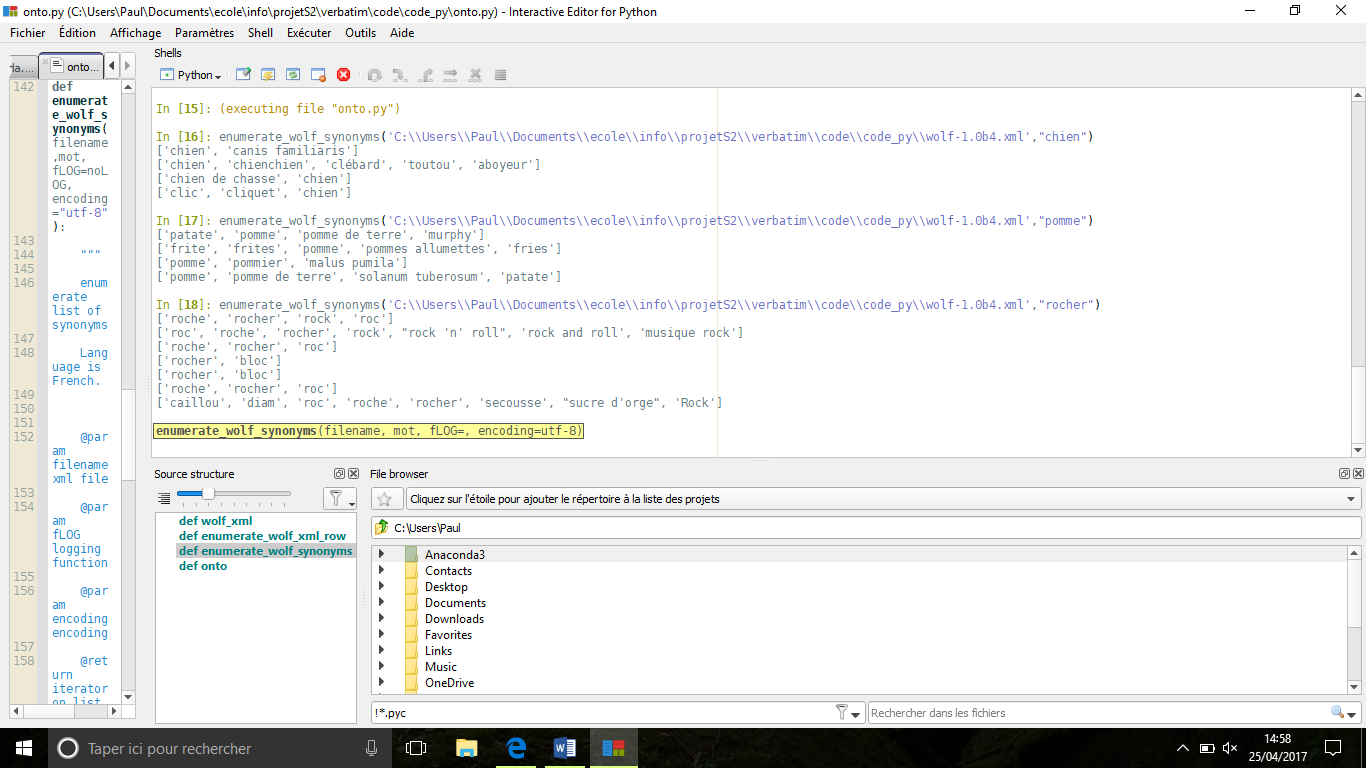
|  |  |  |
| --- | --- | --- |
| Question | Analyse fréquentielle | Commentaire |
| Comment décrivez-vous ce territoire ? | Vallonnée, agréable, beau | Ok, mais même résultat que l’analyse fréquentielle |
| Qu’est-ce qui attire les nouveaux habitants | Campagne, calme, tranquillité | identique |
| Voiture, qu’en pensez-vous ? | Voiture, obligé, choix | identique |
| Si vous aviez une baguette magique, que changeriez-vous ? | Jeunes, lumbres, territoire, enfants, commerces | identique |

Ce que l’on observe, ce sont les mêmes résultats que pour l’analyse fréquentielle seule.

Plusieurs explications permettent de comprendre : premièrement le WOLF est une ontologie très vaste mais peu pertinente : voici ce que nous propose l’ontologie pour les mots : «chien », « pomme »,  « rocher »







L’ontologie ne parvient pas à balayer tout le spectre des idées liées aux termes.

Par ailleurs, l’analyse fréquentielle réduit fortement le nombre de termes. Deux possibilités sont alors envisageables : soit réaliser un tri fréquentiel sans sélection, mais le temps de calcul devient alors très important sans réelle amélioration au vu de la pertinence de l’ontologie, soit créer une ontologie complète, possédant les thèmes qu’on souhaite pouvoir détecter.

Cette deuxième option paraît être la plus pertinente et permettrait de ne plus détecter des mots mais de les rapporter directement à leur sujet de base.

Enfin, la complexité de calcul est en o(n4), ce qui, au vu de l’importance des données présentes dans l’ontologie, présente un problème de temps de calcul. Cependant, la pertinence de l’utilisation d’une ontologie personnalisée (avec les sujets souhaités) pourrait facilement contrebalancer cet obstacle de temps de calcul.

Pour un test de temps, le programme a été lancé pour trouver un terme : 68 secondes ont été nécessaires.

## L’analyse par allocation de Dirichlet latente

### Le programme

L’allocation de Dirichlet Latente est un modèle génératif probabiliste dérivé du PLSA permettant de retrouver les thèmes abordés par un texte. L’algorithme suppose qu’un texte est un ensemble de sujets et que chacun des mots du texte est responsable de la création d’un des sujets.

Les réponses sont ici représentées par des vecteurs donc chaque élément est un mot du Dictionnaire.

Ici se trouve le principe de l’algorithme :

Pour toutes les réponses :

Pour tous les mots :

Assigner à mots un sujet aléatoire

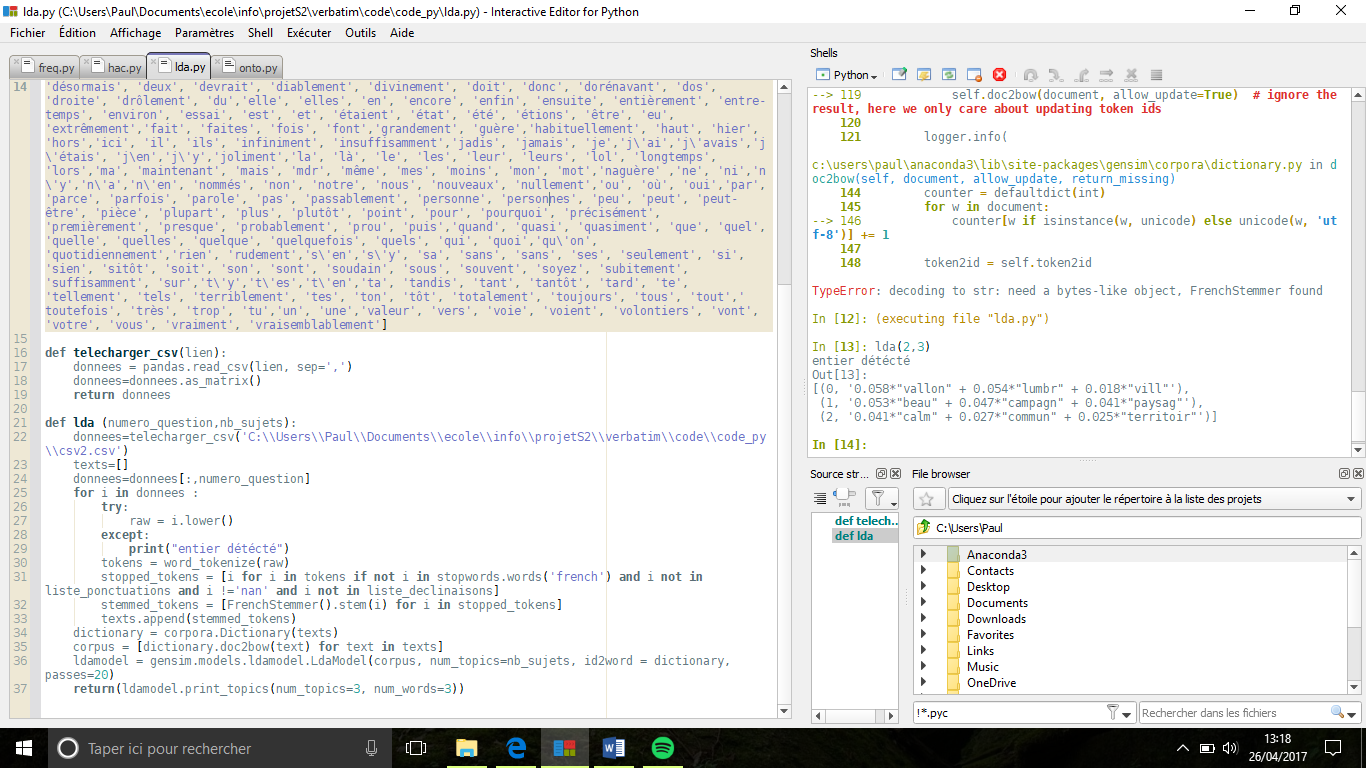
Pour toutes les réponses :

Pour tous les mots :

Pour tous les sujets :

\*

Mots sujet(max(\*)



La fonction FrenchStemmer permet de travailler sans extensions de mots pour ne pas ajouter des mots équivalents (par exemple « manger » et « mange »). On créer le dictionnaire de mots, composé de tous les mots.

Enfin le modèle LDA est appliqué, suivant un nombre de sujets recherchés, et le nombre d’itérations souhaitées (en effet, l’algorithme ne converge pas forcément, il est donc nécessaire de lui spécifier un nombre de passes à réaliser.

### Les tests

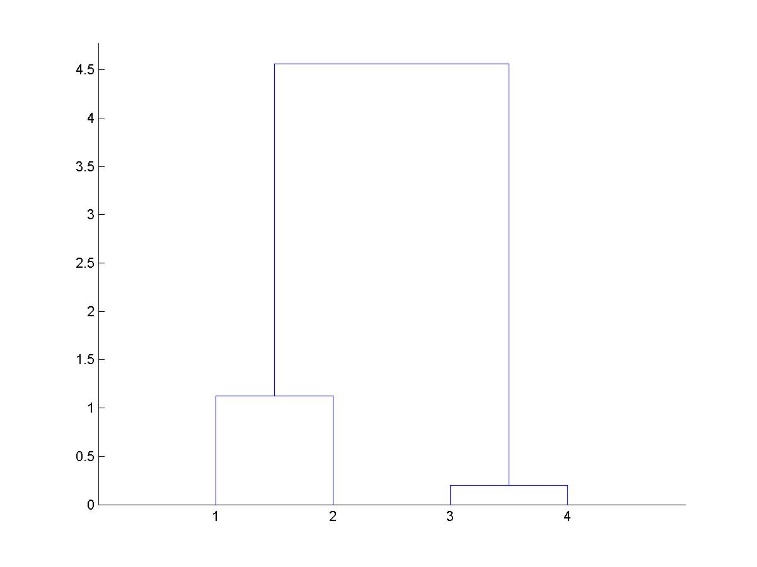
|  |  |  |
| --- | --- | --- |
| Question | Analyse fréquentielle | Commentaire |
| Comment décrivez-vous ce territoire ? | Vallon + Lumbr+ vill  Beau+campagn+paysag  Calm+commun+paysage | Le résultat est un peu plus précis que l’analyse fréquentielle |
| Qu’est-ce qui attire les nouveaux habitants | Tranquill+vill+terrain  Travail+st+Omer  Campagn+calm+ville | On peut travailler à saint Omer. |
| Voiture, qu’en pensez-vous ? | Choix+bus+voiture  Voiture+travail+faut  Transport+normal+commun | On comprend finalement que la voiture est nécessaire pour travailler  Les autres n’apportent pas d’information supplémentaire |
| Si vous aviez une baguette magique, que changeriez-vous ? | Commun+faut+transport  Petit+chos+amenagement  Jeun+lumbr+commerc | Il faut des transports en commun  Des petits réaménagements  La jeunesse de la ville |

L’analyse grâce au modèle LDA est beaucoup plus efficace que l’analyse fréquentielle. On a un gain de 100% de la compréhension des attentes de la population.

## Le clustering hiérarchique agglomératif

### Le programme

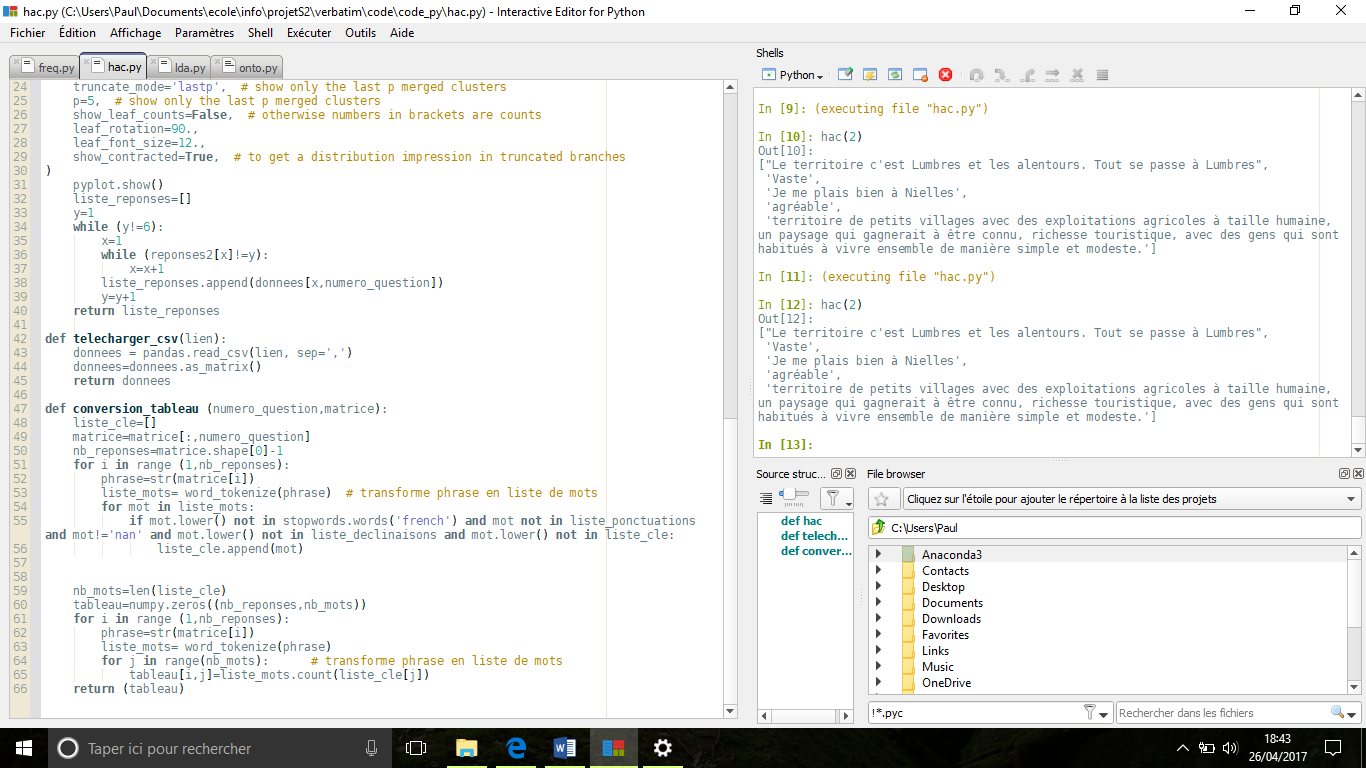
Le HAC (hierarchical agglomerative clustering) est une technique de classement de données. Le principe est simple : il s’agit de calculer des poids moyens pour chaque document et de regrouper les documents ayant les poids les plus proches.

Dans le dendrogramme ci-contre, on voit que les éléments 1 et 2 ont été assemblés dans un même groupe (appelé cluster). Leur distance étant de 1. Les éléments 3 et 4 ont été assemblés dans un autre cluster, leur distance étant de 0,25. Ces deux clusters ont par la suite été regroupé dans un troisième. Leur distance respective étant de 4,5.

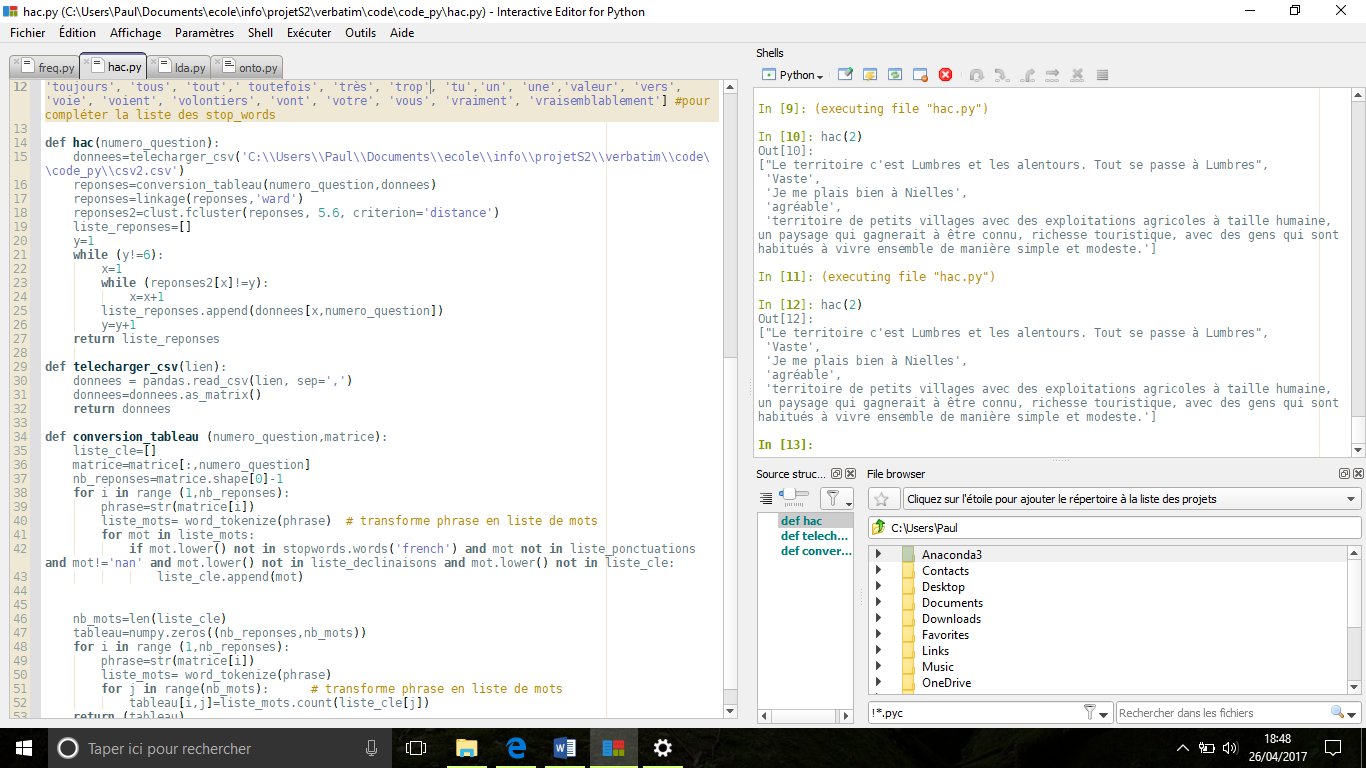
Ce que l’on appelle distance est une mesure de dissimilarité, c’est-à-dire de différence entre les groupes. Il peut s’agir en géométrie d’une distance mais il peut également s’agir d’autres mesures.

L’idée dans le projet de cette technique de réaliser le classement des réponses. Si on considère qu’une réponse est constituée de « x » sujets, eux-mêmes définis par les termes utilisés. On peut calculer une dissimilarité entre des questions, ces dissimilarités pouvant représenter une différence de sujets abordés. En choisissant le nombre de groupes que l’on souhaite former, on peut définir des groupes de réponses qui ont chacun des thématiques proches.

Ainsi, le programme ne va pas détecter des sujets, mais seulement des groupes de réponses semblables.



Pour implémenter cet algorithme de classification, il est nécessaire de travailler avec des vecteurs. Ces vecteurs sont de la taille du dictionnaire des réponses. Chaque coordonnée est le nombre de représentations du mot du dictionnaire dans la réponse. Ainsi on obtient un vecteur dont les coordonnées dépendent des mots qui le composent.



Après avoir obtenu un tableau de vecteurs, la fonction linkage procède au regroupement des clusters (donc des vecteurs) suivant leur poids. La méthode ward calcule la distance euclidienne entre les vecteurs et regroupe deux à deux les clusters ayant la distance la plus courte.

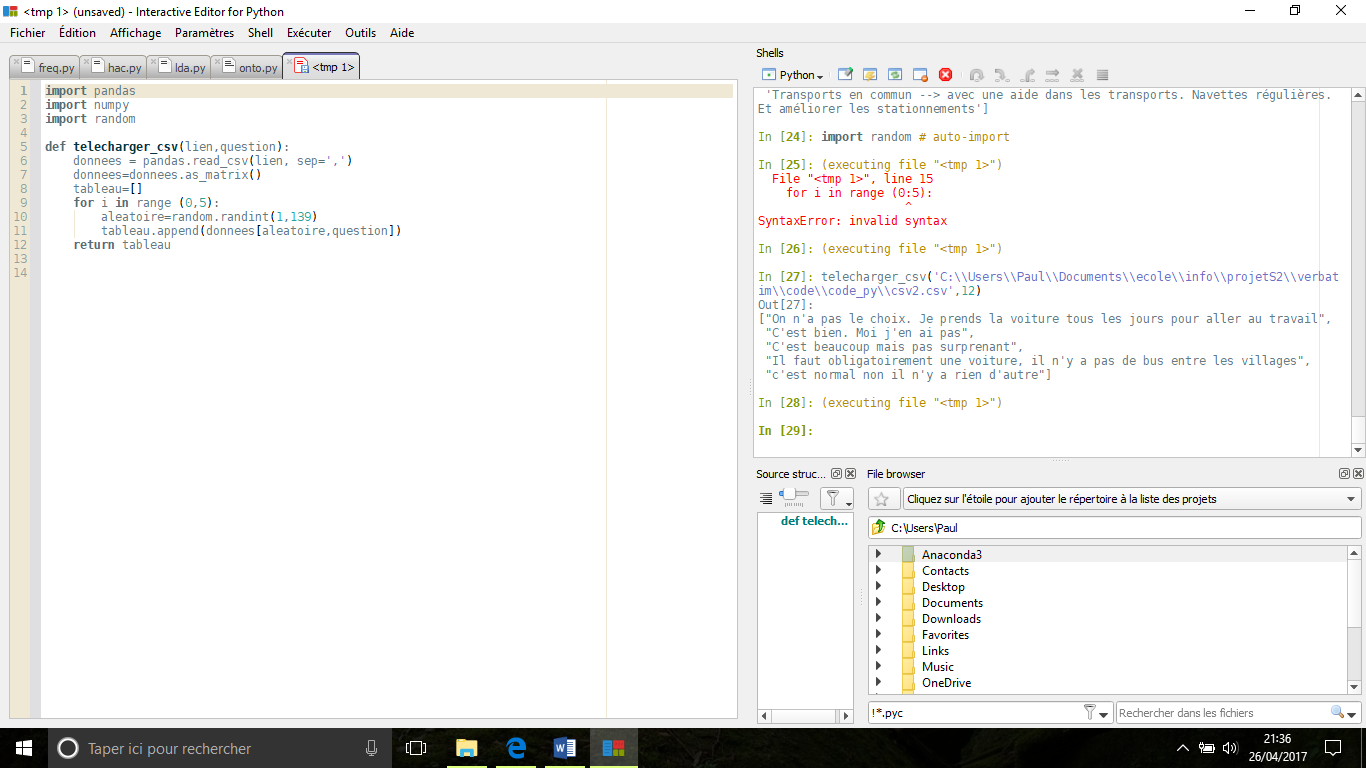
On affiche finalement une réponse de chaque cluster pour voir les thématiques abordées.

### Tests

|  |  |  |
| --- | --- | --- |
| Question | Analyse fréquentielle | Commentaire |
| Comment décrivez-vous ce territoire ? | "Le territoire c'est Lumbres et les alentours. Tout se passe à Lumbres",  'Vaste',  'Je me plais bien à Nielles',  'agréable',  'territoire de petits villages avec des exploitations agricoles à taille humaine, un paysage qui gagnerait à être connu, richesse touristique, avec des gens qui sont habitués à vivre ensemble de manière simple et modeste. | Information inutile « je me plais bien à nielles » et redondances. Non pertinent vis-à-vis des autres analyses |
| Qu’est-ce qui attire les nouveaux habitants | "On est à la campagne, au milieu de l'axe Boulogne - St Omer - Calais avec l'accès à l'autoroute et la pointe du littoral qui permet ça (la partie covoiturage)",  'Lumbres est une ville accueillante avec tout à proximité. ',  'les terrains pas chers',  "Le côté rural, les espaces, la tranquilité, le côté convivial avec les voisins qui n'existe pas forcément en ville",  "Les magasins, les médecins, l'envie de revenir à la campagne" | Précisions et meilleure compréhension des thèmes sortis précédemment |
| Voiture, qu’en pensez-vous ? | 'Il faut être véhiculé pour trouver du travail',  "C'est rare une voiture. Les familles en on 2 ou 3. On a besoin d'une voiture pour travailler ",  "on est tellement habitué à prendre la voiture que c'est normal. Et il n'y a pas de transports en commun",  'Pas vraiment le choix, il ne peut pas u avoir de bus partout',  'La voiture est une nécessité sur le territoire' | Permet de décider le sens de certains thèmes évoqués |
| Si vous aviez une baguette magique, que changeriez-vous ? | 'La piscine de Lumbres, un centre aquatique comme à St Omer',  "Il faudrait montrer que Lumbres est une ville à la campagne / avoir une identité écologique, patrimoine naturel et sportif en extérieur. Il faut des maisons médicalisées dans les villages, avec des permanences. Il faut des lieux pour se retrouver (artisans, locaux, marchands). Un lieu intergénérationnel où des personnes agées s'occuperaient des petits, des devoirs. Il manque l'hébergement sur le territoire, à part le golf qui n'apporte rien ",  "FAire quelque chose pour les jeunes, type MJC avec des encadrants. Améliorer les calendriers d'activités festives. Parfois 3 activités le même week end et puis rien le week end suivant",  "On aurait pu regrouper plusieurs écoles des 3 communes : bonninques (école vieille en plus), Audrehem et Clerques. Car ici à Audrehem c'est 1 RPI",  'Transports en commun --> avec une aide dans les transports. Navettes régulières. Et améliorer les stationnements' | Les thèmes qui étaient très approximatifs peuvent êtres précisés. |

Les résultats de cette analyse sont mitigés. Elle ne semble pas du tout utile pour les questions pour lesquelles les sujets avaient bien été définis auparavant. Cependant, là où l’analyse fréquentielle et par le modèle LDA n’avait pas permis de clairement identifier les sujets et où l’on pouvait rester indécis quant au sens à leur attribuer, cette analyse a permis de trancher sur le sens.

Cependant on peut se demander si la sélection au hasard de réponses n’aurait pas permis le même résultat. Le test suivant concerne les sujets qui ont été améliorés, les réponses sont choisies aléatoirement.



|  |  |  |
| --- | --- | --- |
| Voiture, qu’en pensez-vous ? | "On n'a pas le choix. Je prends la voiture tous les jours pour aller au travail",  "C'est bien. Moi j'en ai pas",  "C'est beaucoup mais pas surprenant",  "Il faut obligatoirement une voiture, il n'y a pas de bus entre les villages",  "c'est normal non il n'y a rien d'autre" | Des informations importantes ressortent. Pas nécessairement aussi pertinentes |
| Si vous aviez une baguette magique, que changeriez-vous ? | 'Du soleil, des commerces de proximité',  "plus de festivités sur Lumbres (Comme la fête médiévale qu'il y avait mais qui n'a jamais été reconduite)",  'nsp ',  'Parc pour se promener. Redynamiser Lumbres. Activités sur la place. Ils ont tout déporter ça fait plus vie de centre ville',  'Les taxes foncières sont trop élevées' | Les informations étaient plus complètes dans l’analyse hac mais la réponse était plus longue, exhaustive. |

Le bilan de la clusterisation est donc mitigé. Sur le principe elle est pertinente, mais la sélection aléatoire de données semble donner des résultats proches de ceux sélectionnés par clusterisation. Une meilleure définition des vecteurs pourrait éventuellement améliorer sa pertinence.

## L’interface

Concernant l’affichage des thèmes abordés, deux options ont été abordées. Premièrement, il a été tenté de déplacer le code développé sur un environnement IronPython. Cet environnement a été développé par Microsoft Visual Studio afin de de pouvoir développer des interfaces python grâce à l’environnement visual studio connu. Cependant, cet environnement est encore à l’état de développement. Ainsi la création des forms n’est pas simplifiée (il est nécessaire de créer tous les éléments un à un) et la version de python n’est pas toujours disponible. Il est donc difficile d’utiliser des librairies qui doivent être installées et dont les versions changent.

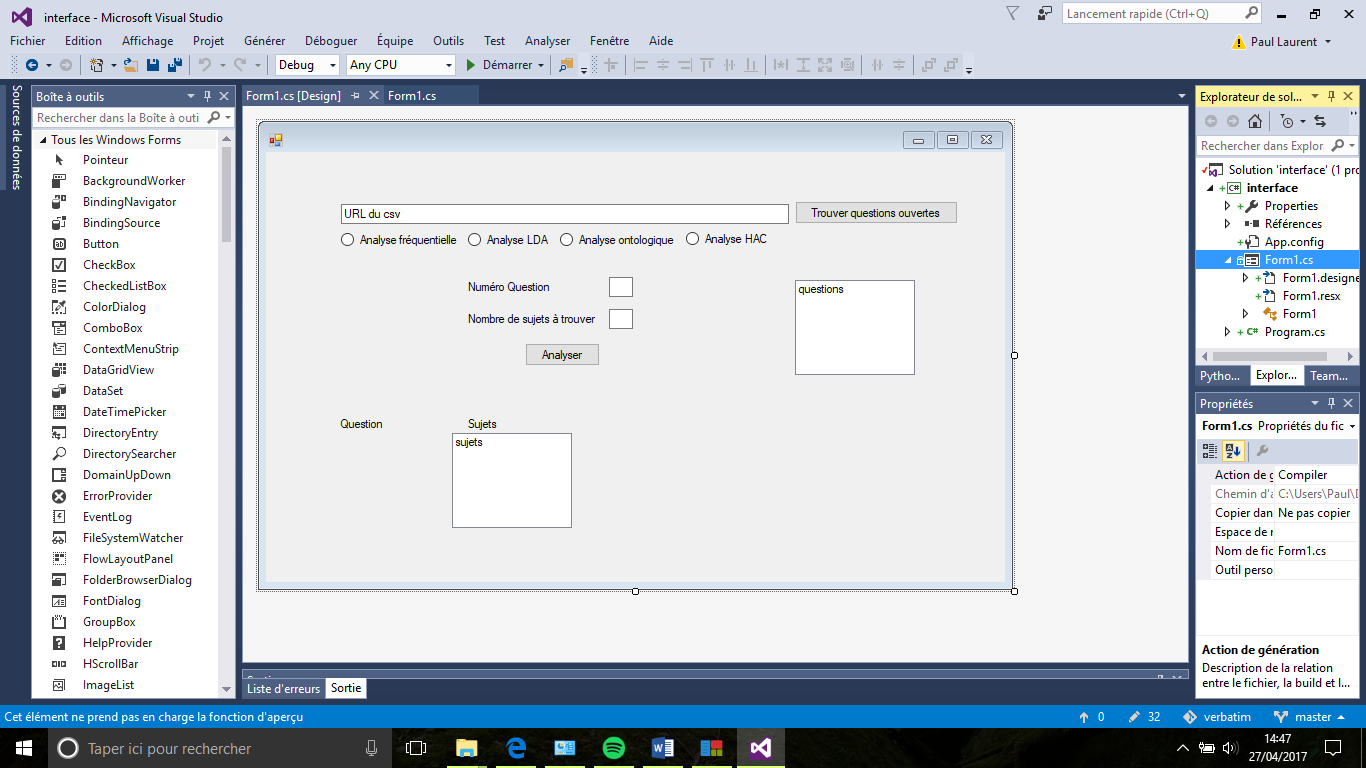
Cette option a donc été écartée et un format d’API a été choisi ; ce qui s’adapte parfaitement à la méthode de Quorum. L’idée est donc de coder en python et d’utiliser ce code via un server.

Un serveur Flask a été mis en place pour cela puisque flask permet de développer des applications python.

L’interface codée en c# réalise des requêtes http sur le serveur local qui retourne les données à l’interface. Les données sont envoyées et reçues au format Json. Ce format a l’avantage d’être compris par tous les langages, et d’être léger et efficace.

Le formatage des informations au format Json est dans notre cas simple puisqu’il ne s’agit que de chaines de caractères. Il n’y a pas d’arborescence complexe. L’utilisation de la fonction json.dumps() ne retournant pas nécessairement une donnée bien structurée, il est plus avantageux de construire le json, ce qui dans notre cas est simple.

La connexion au serveur n’ayant pas été réussie, l’interface ne peut être utilisée et les données n’ont pas été converties au format Json pour faciliter leur réutilisation au sein des fonctions python.



# La gestion de projet

Planning :

