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](#_Toc480377030)

[Contexte 1](#_Toc480377031)

[L’existant 1](#_Toc480377032)

[Définition du projet 2](#_Toc480377033)

[L’architecture du projet 2](#_Toc480377034)

[Les besoins de l’entreprise 2](#_Toc480377035)

[Le travail des données 3](#_Toc480377036)

[Les analyses 4](#_Toc480377037)

[Les données 4](#_Toc480377038)

[L’analyse fréquentielle 4](#_Toc480377039)

# Définition du sujet

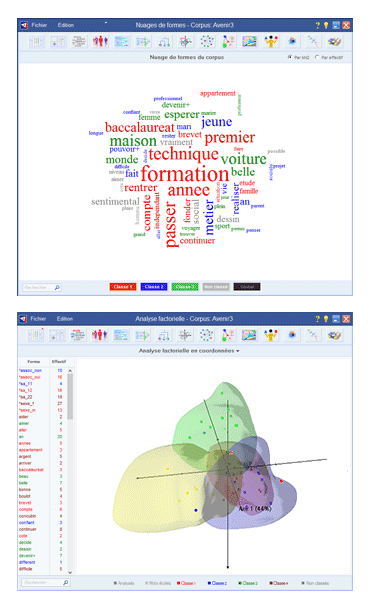
## Contexte

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

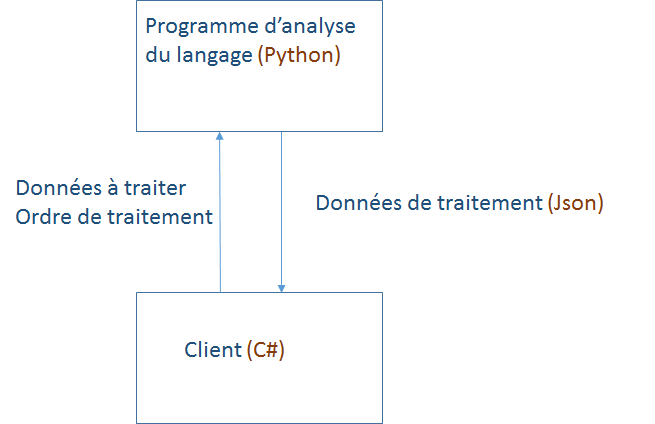
Vu les travaux qui ont été menés jusqu’ici par les développeurs et les chercheurs, vu les besoins de l’entreprise Quorum, il est apparu que mon travail s’axerait sur une analyse statistique des corpus fourni. Cependant, l’analyse statistique est limitée quant à son utilisation. Il est donc intéressant 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. 

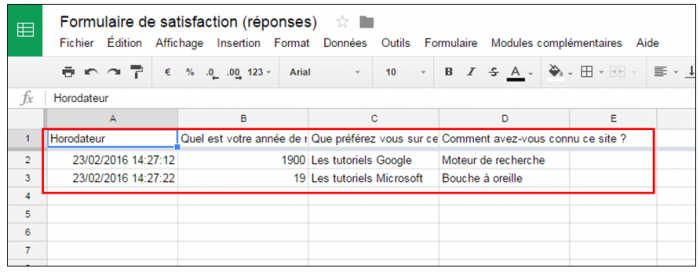
Le travail que je fournis doit pouvoir s’insérer dans la plateforme de Quorum. Il ne leur est donc pas nécessaire d’avoir une interface graphique. J’ai donc choisi de réaliser une interface en C#. Cette interface reçoit les commandes de l’utilisateur (notamment le recueil du corpus) envoie ces données 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 une représentation graphique des mots clefs et des thèmes abordées. Si le temps le permet une représentation en nuage de mots sera également mise en place.

# Les analyses

## 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.

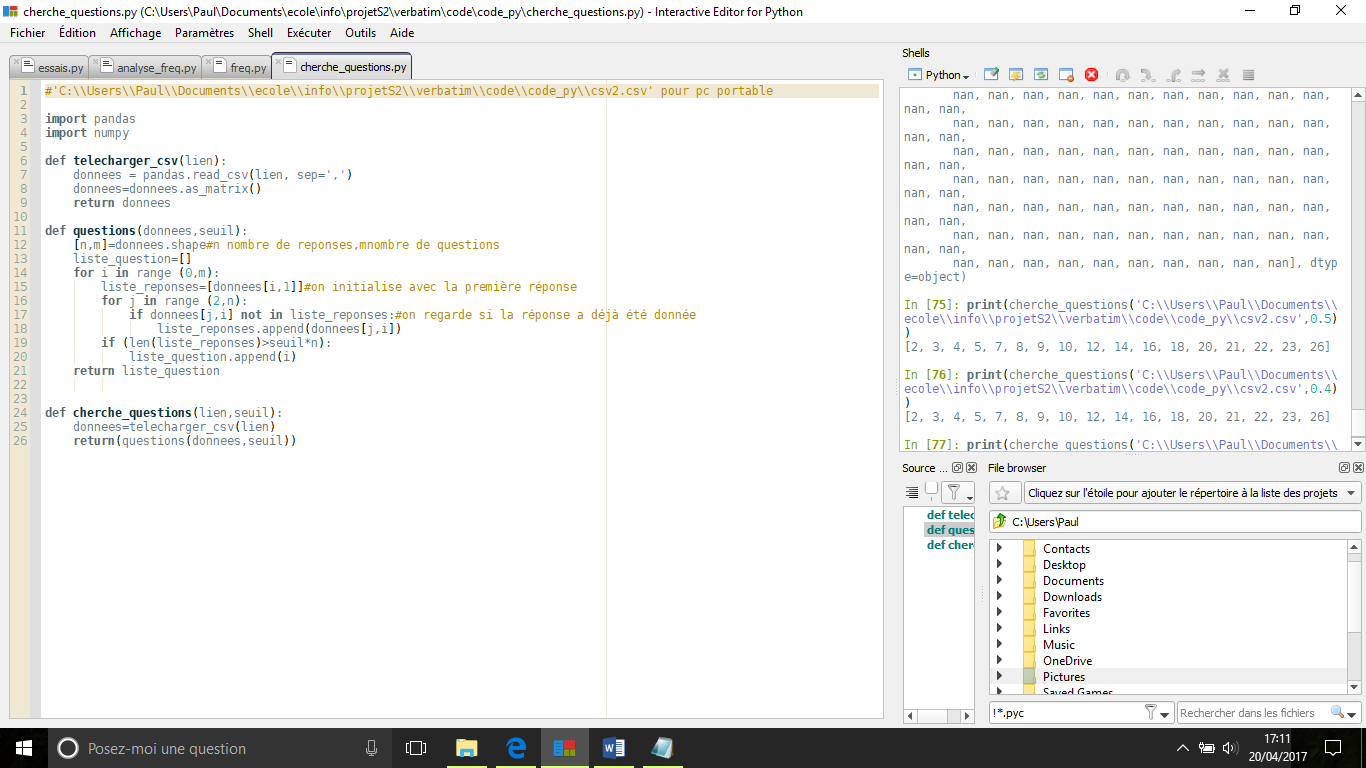
Enfin la conversion de l’élément NaN en chaine de caractère n’est pas possible. Il existe un moyen de supprimer ces éléments mais qui nécessite un traitement plus lourd et l’utilisation du module nltk. Ceci est utilisé dans la suite du projet.

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).



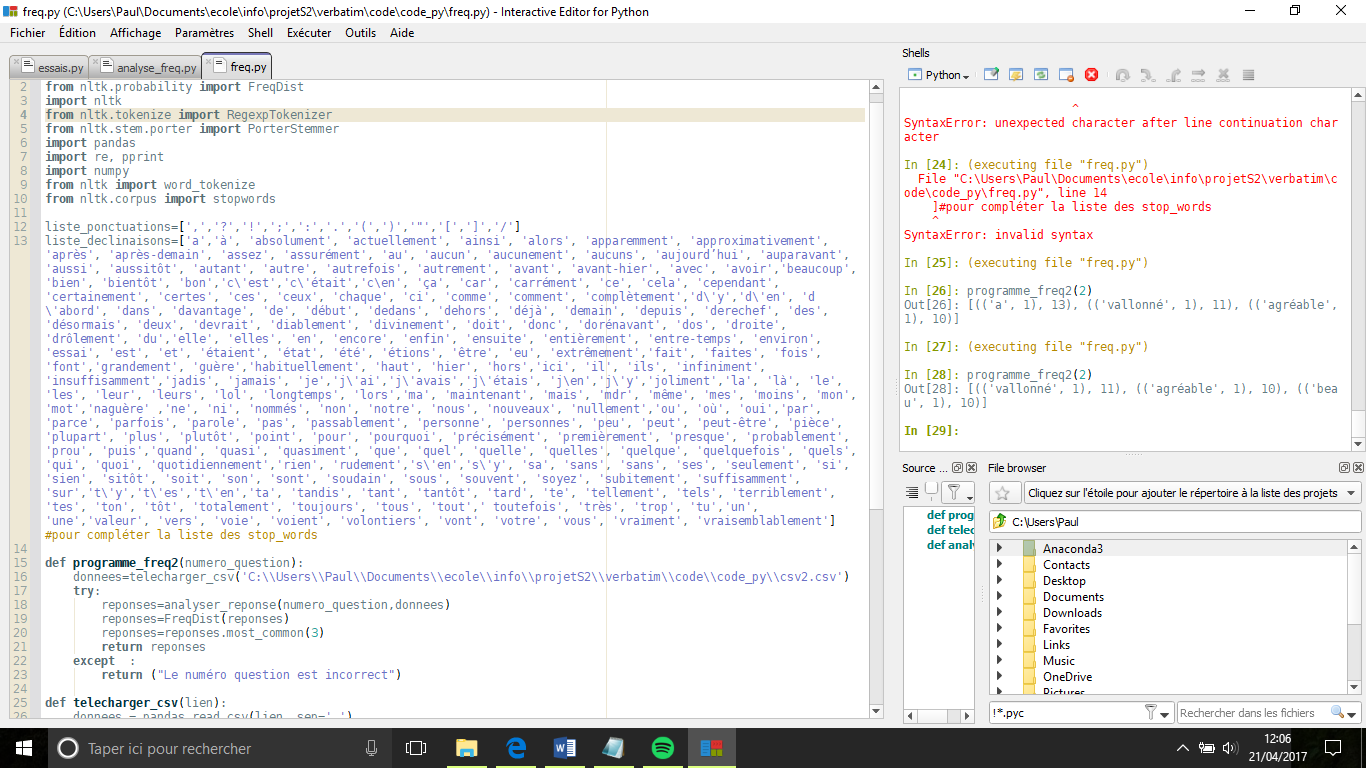
### L’analyse fréquentielle

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.

