Projet 6 : Catégorisez automatiquement des questions

[Attirez votre lecteur avec un résumé attrayant. Il s’agit généralement d’une brève synthèse du document. Lorsque vous êtes prêt à ajouter votre contenu, cliquez ici et commencez à taper.]

[Sous-titre du document]

Sommaire

Table des matières

[I. Contexte et Objectifs 2](#_Toc50934356)

[II. Sélection des données 2](#_Toc50934357)

[III. Nettoyage & pré traitement des données 4](#_Toc50934358)

[IV. Bag of Words 5](#_Toc50934359)

[1. Approche « Termes les plus fréquents » 5](#_Toc50934360)

[2. Approche TF-IDF (term frequency-inverse document frequency) 6](#_Toc50934361)

[V. Sujets des Post 6](#_Toc50934362)

[1. Approche supervisée 7](#_Toc50934363)

[2. Approche semi supervisée avec modélisation de sujets/topics (LDA) 7](#_Toc50934364)

[VI. Génération du modèle de prédiction des tags 9](#_Toc50934365)

[VII. Résultats 9](#_Toc50934366)

Table d’illustration

[Figure 1 : Modèle de la base de données de Stack OverFlow 3](#_Toc50934312)

[Figure 2 : Exemple extraction 4](#_Toc50934313)

[Figure 3 : Réprésentation TSNE & KMean après Pre processing 5](file:///C:\Users\naru_\Documents\Projet6_CategorizationAutomatiqueQuestions\SourceGitV3\TextCategorization\Projet%206%20Data%20Scientist%20_Catégorisation%20des%20posts%20automatiquement.docx#_Toc50934314)

[Figure 4 : Repésentation TSNE & KMean Avant Pre processing 5](file:///C:\Users\naru_\Documents\Projet6_CategorizationAutomatiqueQuestions\SourceGitV3\TextCategorization\Projet%206%20Data%20Scientist%20_Catégorisation%20des%20posts%20automatiquement.docx#_Toc50934315)

[Figure 5 6](file:///C:\Users\naru_\Documents\Projet6_CategorizationAutomatiqueQuestions\SourceGitV3\TextCategorization\Projet%206%20Data%20Scientist%20_Catégorisation%20des%20posts%20automatiquement.docx#_Toc50934316)

[Figure 6 7](#_Toc50934317)

[Figure 7 8](#_Toc50934318)

[Figure 8 8](#_Toc50934319)

[Figure 9 9](#_Toc50934320)

# Contexte et Objectifs

Le Site Internet Stack Overflow est un site internet d’entraide autour de l’IT.

Pour pouvoir solliciter l’aide des autres Internautes, il faut associer des tags aux sujets et questions posés.

Etant un utilisateur occasionnel de ce site, je sais qu’il est parfois difficile de trouver le bon tag pour le post que l’on vient de rédiger.

C’est pourquoi, je voudrais proposer un système pour aider les internautes en leurs proposant une liste de tags qu’ils peuvent associer au post qu’ils viennent de rédiger.

Dans ce rapport, je vais vous présenter mes démarches pour mettre en place deux modèles de prédiction de tag à partir des données de Stack Overflow

Dans un premier temps, je présente l’origine des données utilisées pour la génération du model

Ensuite, j’explique comment, j’ai nettoyé et préparé ces données

Ensuite, je présente deux méthodes de génération de « Bag of Words », nécessaire à l’entrainement du modèle de prédiction (« mot les plus fréquents » et TF-IDF)

Ensuite, je présente deux approches utilisées pour associés soit les tags aux posts grâce aux Bag of Word (méthode supervisé) et soit des sujets issus des bag of Word (méthode semi supervisée (LDA))

Et pour finir, je comparerais les modèles de prédictions entrainées avec via les deux approches précédentes.

# Sélection des données

Stack Overflow propose un site internet pour extraire les données des posts présents sur le site internet.

Il s’agit d’une page web qui nous permet d’écrire des requêtes SQL et de les exécutées sur la base de données du site. Toutefois, cette interface présentes des contraintes comme le temps d’exécution de la requête ou un nombre maximum de lignes de résultats (50 000 lignes maximum).

C’est pourquoi, il nous faut faire une première étape pour affiner la sélection des données qui nous permettront par la suite d’entrainer nos modèles de prédiction de tags.

Une première étude de la base de données nous permet de voir qu’un grand nombre d’informations du site ne seront pas utiles à la résolution de notre problématique.

Une image contenant moniteur, écran, télévision, capture d’écran

Description générée avec un niveau de confiance très élevé

Figure : Modèle de la base de données de Stack OverFlow

Ainsi, nous récupérerons uniquement les champs body, title et tags de la table post pour les posts qui ne sont pas clos qui ont eu des réponses acceptées.

Comme nous voulons proposer des tags pour les futures posts de type question, nous ferons donc aussi le lien avec la table PostType afin de n’extraire que les posts de type question.

Le site propose environ 60 000 tags différents. Ceci est beaucoup trop pour pouvoir générer et entrainer un modèle efficace. Il convient donc de limiter ce nombre en identifiant les tags les plus intéressant : les tags qui ont le plus de posts associés. C’est dans ce but que j’ai limité ma requête uniquement aux tags qui ont eu plus de 10 000 posts associés (soit environs 700 tags).

Par ailleurs, il existe plus de 50 000 000 posts sur le site internet, il convient donc aussi de réduire ce nombre.

C’est pour cela que j’ai limité ma requête uniquement aux posts:

* dont une réponse a été acceptée,
* qui ont des tags parmi les plus utilisés,
* qui ne sont pas supprimés,
* qui ont une note (supérieure à 0)

J’ai limité ma requête aux 50 000 premiers retours en triant mes résultats par notation des posts et par date de création en privilégiant les posts mieux notés puis les plus récents.

Une image contenant capture d’écran

Description générée avec un niveau de confiance très élevé

Figure : Exemple extraction

Cette requête me permet de générer un fichier csv qui me servira de données d’entrée pour mes modèles de prédictions.

# Nettoyage & pré traitement des données

Afin de permettre au traitement du modèle de prédiction d’être le plus efficace possible, il convient de préparer les données afin d’en faire ressortir les données intéressantes.

Un rapide parcours du fichier csv, extrait via le site de Stack Overflow, nous suffit à comprendre qu’il ne sera pas directement utilisable pour entrainer un modèle de prédiction. En effet, le contenu des post (colonne body) contient des balises HTML, de la ponctuation et des caractères dans plusieurs casses. C’est pourquoi, il nous faut préparer les données en supprimant les balises html et les signes de ponctuations et en uniformisant la casse des textes.

Par ailleurs, il existe de nombreux mots qui sont présents dans la colonne body et qui n’apporte pas de sens au contenu. Il s’agit de ce que nous appelons les « Stop Words » : des mots du langages courant mais qui n’apporte pas de sens au contenu. Cependant, les conserver pour entrainer notre modèle peut pénaliser les performances et les résultats de ce dernier.

Ainsi, comme le contenu des messages sont en anglais, j’ai supprimé les « stop words » anglais de mes données d’entrainements.

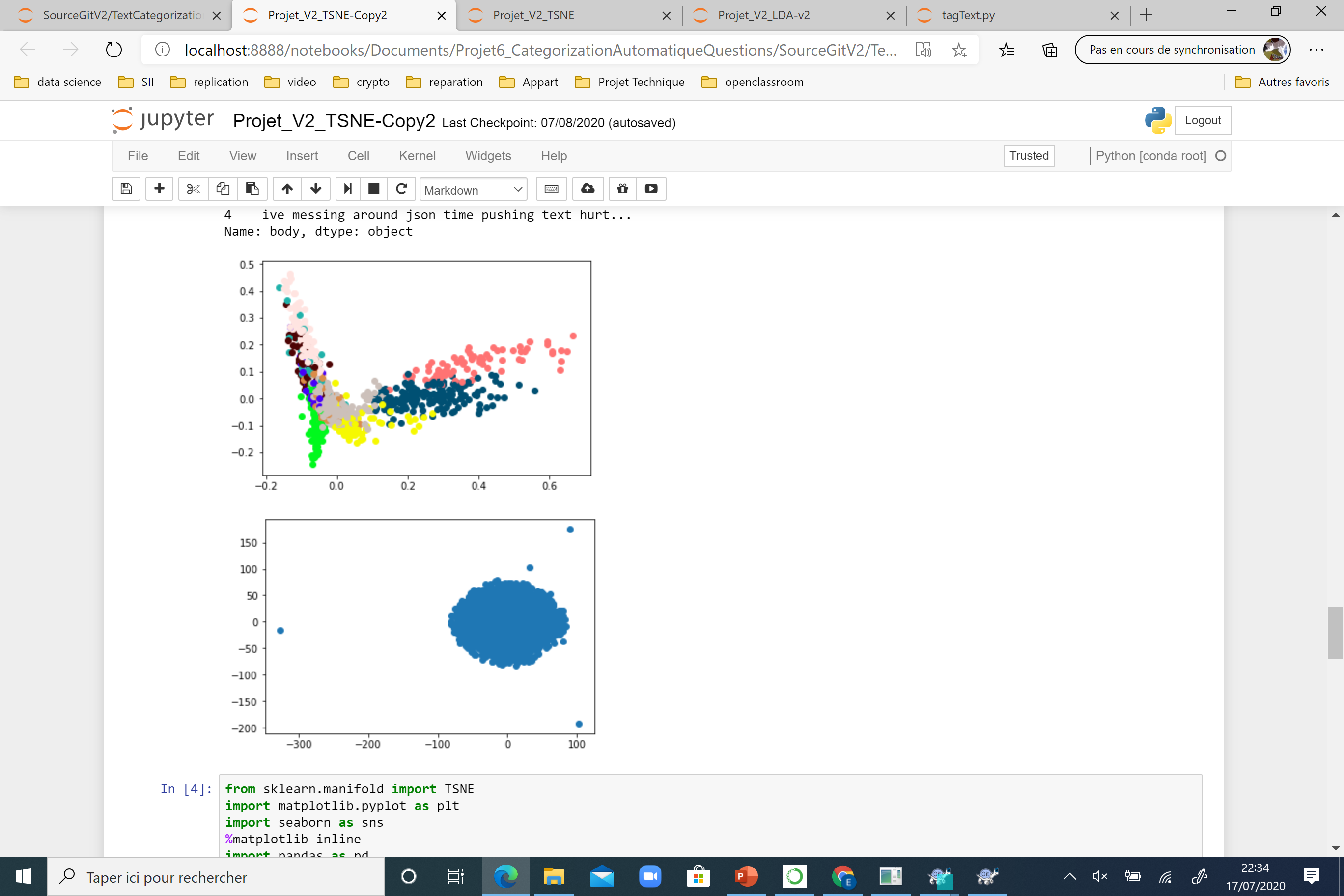
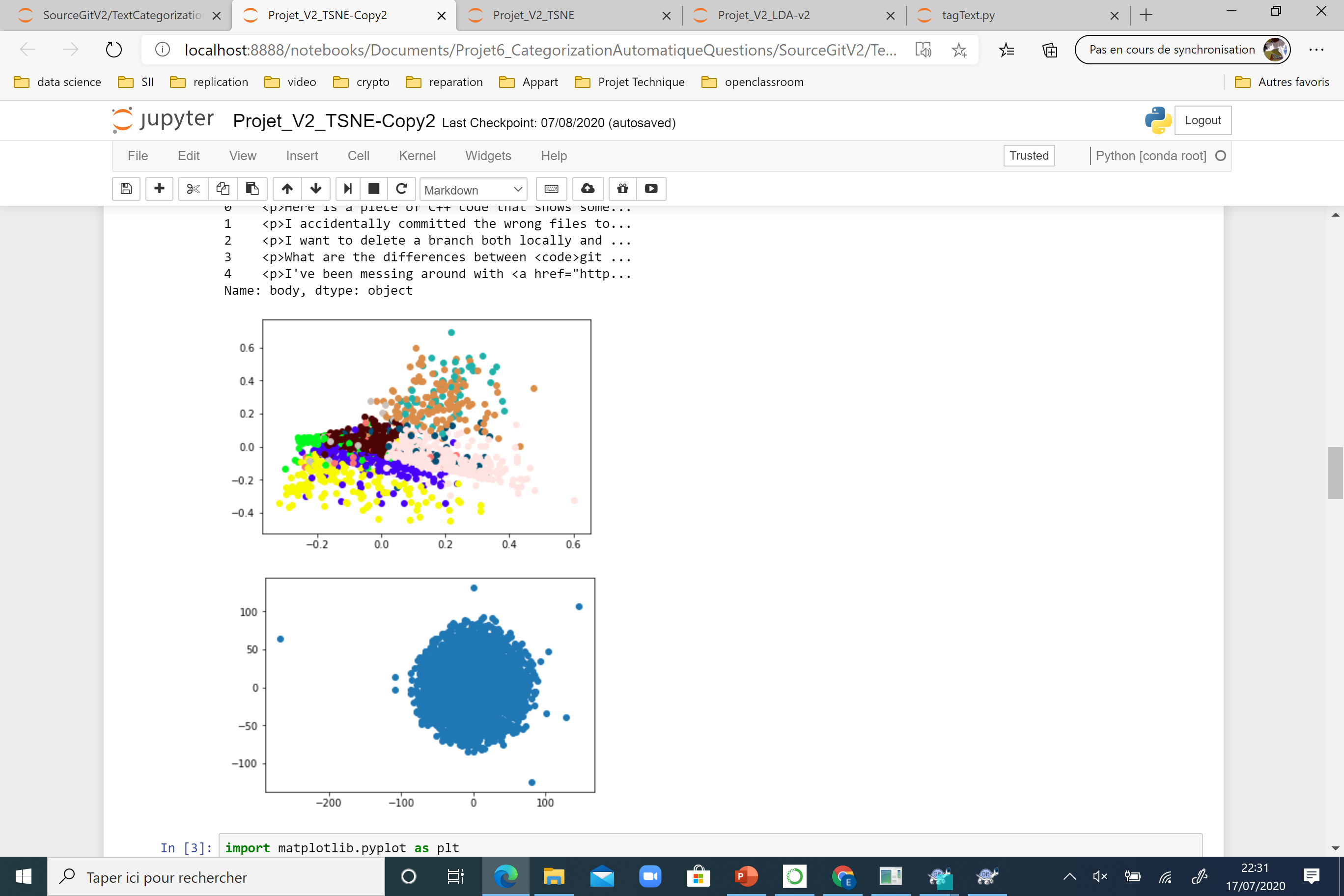
Les deux figures précédentes montrent la répartition des mots de l’ensemble de mes données d’entrée avant et après nettoyage et pré traitement selon des regroupement TSNE ou KMean.

Figure : Réprésentation TSNE & KMean après Pre processing

Figure : Repésentation TSNE & KMean Avant Pre processing

Nous voyons que les données semblent plus faciles à regrouper et donc à utiliser après le nettoyage et le pré traitement.

# Bag of Words

Après avoir préparé le texte à analyser, il faut identifier les termes les plus importants pour catégoriser les posts initiaux.

Une méthode pour proposer un modèle de prédiction de texte revient à représenter les textes par des « Bags of Words » présent au sein de notre corpus et à les associer à des tags. Ensuite, il faut transmettre le tout à notre modèle d’entrainement afin qu’il définisse une fonction qui permette d’obtenir les tags en fonction de ces « sacs ».

Ces « Bags of Words » peuvent être générés de nombreuses manières mais dans le cas de ce projet, j’ai choisi deux approches : Soit identifier les termes les plus fréquents soit utiliser l’algorithme TF-IDF.

### Approche « Termes les plus fréquents »

Une première approche consiste à identifier les mots de l’ensemble des posts et à les ordonner par nombre d’apparition décroissant. Ainsi les mots les plus présents dans l’ensemble des posts apparaitront en premier.

Ensuite, la représentation d’un texte se fait en vérifiant la présence de chacun des mots de la liste.

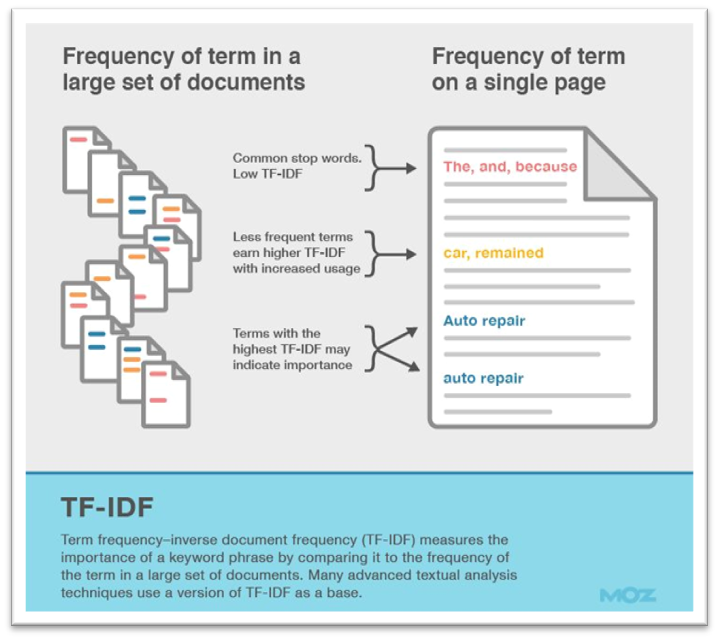
Cette approche permet de choisir le nombre de mots que nous acceptons dans la liste et ainsi de limiter la taille des « Bags of Words ».

Cependant, nous nous rendons vite compte que bien que les data soient pré traitées, il existe encore des mots très présents dans notre corpus mais qui n’apportent pas de sens au contenu. Ainsi, représenter les posts par les mots les plus fréquents ne semble pas un choix pertinent.

### Approche TF-IDF (term frequency-inverse document frequency)

Puisque la seule fréquence d’un mot n’est pas suffisante pour identifier les mots importants de notre corpus, il convient d’envisager une autre approche.

TF-IDF pondère les mots d’un corpus selon deux métrics :

* TF (Term-Frequency) qui majore l’importance des mots en fonction de leur fréquence d’apparition dans un texte du corpus,
* IDF (Inverse Document Frequency) qui minore l’importance des mots en fonction de leur fréquence d’apparition dans le corpus car il s’agit de mots « commun » et donc vraisemblablement moins importants pour la précision du contenu du texte.

Figure

Cette méthode, nous permet de générer les « Bag of Words » des documents du corpus

# Sujets des Post

Après avoir représenté les textes du corpus sous forme de « Bag of Words », il faut identifier le sujet de chaque texte.

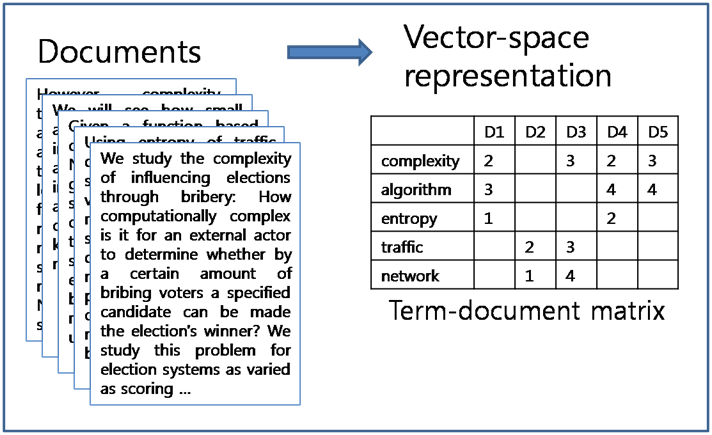
Pour ce faire, j’ai envisagé 2 approches :

* Une approche entièrement supervisée où les Bag of Words représentent les sujets,
* Une approche partiellement supervisée dans laquelle j’ai appliqué l’allocation de Dirichlet latente (LDA ou Latent Dirichlet Allocation) afin d’identifier automatiquement des topics / sujets.

### Approche supervisée

Grace aux étapes précédentes, j’ai un « Bag of Words » issue de l’approche TF-IDF pour chaque post initial.

A partir des « Bag of Words », je génère un vecteur qui compte, pour chaque post, le nombre d’occurrences des chacun des mots presents dans les Bag of Words. Ainsi pour chaque post, j’obtiens un vecteur qui le représente.



Figure

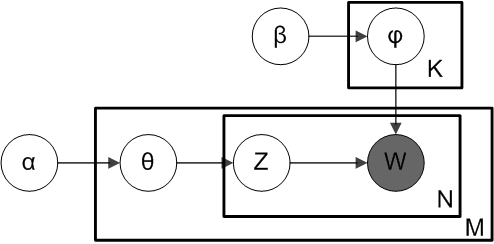
### Approche semi supervisée avec modélisation de sujets/topics (LDA)

Il existe des algorithmes qui permettent de faire ressortir des sujets automatiquement

La loi de distribution Latent Dirichlet Allocation est un de ces algorithmes.

Il s’agit d’une méthode non-supervisée générative basée sur les hypothèses suivantes :

* Chaque document du corpus de texte est un ensemble de mots sans ordre,
* Chaque document est composé de plusieurs thèmes dans différentes proportions qui lui sont propres,
* Chaque mot possède une distribution associée à chaque thème,
* zn représente le thème du mot wn



Figure

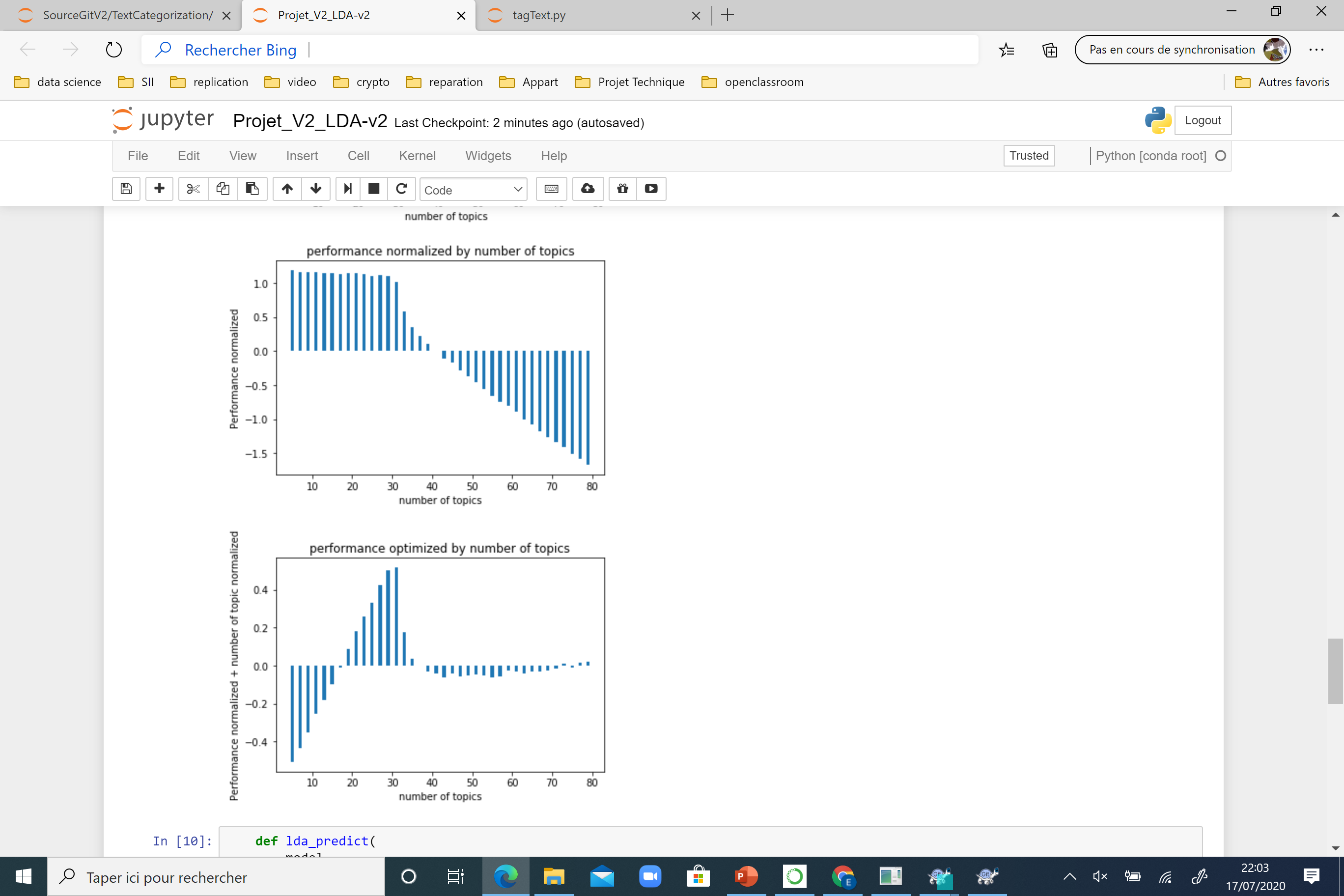
Un modèle génératif définit une probabilité de distribution jointe sur les différentes variables identifiées, à la fois observées et latentes

Puisque l'on a accès uniquement aux documents, on doit déterminer quels sont les thèmes, les distributions de chaque mot sur les thèmes, la fréquence d’apparition de chaque thème sur le corpus.

Une image contenant texte, carte

Description générée avec un niveau de confiance très élevé

Figure



Figure

# Génération du modèle de prédiction des tags

Maintenant que j’ai identifié les thèmes et la répartition des post dessus, je peux entrainer un modèle de prédiction.

Comme j’ai plusieurs tags associés à chaque post, j’utilise l’algorithme « Random Forest Classifier ».

La représentation de chaque post sous formes de vecteur d’appartenance aux thèmes/topics seront fournis comme données d’entrée.

Dans la requête initiale, j’ai, pour chaque post, 1 ou plusieurs tags. Je peux donc générer un vecteur de présence de tag que je vais ensuite fournir à mon modèle de prédiction comme données cible.

Ensuite pour connaitre les tags proposés pour un nouveau post, il faudra procéder de manière suivante :

* Nettoyer le contenu du post en utilisant le même algorithme que dans mon entrainement,
* Pré-processer le contenu du post en utilisant le même algorithme pour l’entrainement de mon modèle,
* Classer le contenu afin d’identifier le taux d’appartenance à chacun des thèmes/topics,
* Transmettre les résultats des 3 précédentes étapes au modèle de prédiction,
* Demander au modèle de prédiction la liste de probabilité de présence de chacun des tags ordonnés par taux décroissant.
* Choisir dans les X premiers tags et les proposé (X étant un nombre entier et supérieur à 0.

# Résultats

Maintenant que nous avons deux méthodes pour prédire les tags à proposer à l’internaute qui vient de saisir un nouveau post, il reste à voir lequel est le plus intéressant.

Comme le modèle doit proposer plusieurs tags, nous ne pouvons pas utiliser le score de l’algorithme RandomFOrestCLassifier. En effet, le score permet de donner le nombre de résultats juste pour la prédiction d’un tag pour un post. Hors, comme les posts d’entrainement ont plusieurs tag, le résultats de la prédiction d’un tag unique aura obligatoirement un mauvais score.

J’ai donc programmé une fonction qui permet de compter, sur un jeu de données de validation, le nombre de bon tag proposé les X premier prédit pour le post (ordonnés par probabilité décroissante)

Voici les résultats :

|  |  |  |  |
| --- | --- | --- | --- |
| Méthode | | Méthode supervisée | Méthode Partiellement supervisée |
| Bag Of Word | | Tf-idf | Tf-idf |
| Thèmes / Topics | | Chaque mot du Tf-idf correspond à un thème | LDA : topics généré automatiquement |
| Résultats : nombre de tags juste parmi les 20 premiers prédits | 1 Tag | 11,24% | 12,48% |
| Tous les tag | 1,36% | 1,52% |

La méthode Partiellement supervisée (avec la génération automatique de topic grâce à LDA) donne des résultats plus probants que l’autre méthode

API

Ouverture :

* tester les tags
* regrouper les tags