

**RAPPORT
PROJET D'APPLICATION INGE2**

MAJEURE BIG DATA

**APPLICATION WEB POUR L'ANALYSE ET LA
CLASSIFICATION AUTOMATIQUE DES COMMENTAIRES
CLIENTS**

Projet réalisé par
Camille BOUBERKA
Salomé LAMARTINIE
Naomi MARRACHE

Projet encadré par
Salma REBAI-JRIBI

REMERCIEMENTS

Nous souhaitons remercier dans un premier temps toute l'équipe pédagogique de l'ESME SUDRIA, qui, à travers ce projet stimulant, nous a fait vivre une expérience enrichissante et a su nous fournir les bagages nécessaires à la réalisation de cette application très complète, nous poussant à solliciter nombre de nos acquis du semestre.

Nous souhaitons également remercier tout particulièrement Mme Salma Rebai-Jribi, à l'initiative de la majeure Big-Data, qui par son enseignement, a su nous épanouir en tant qu'élèves, et nous a naturellement conduites vers le parcours Big-Data dans lequel nous ne cessons d'évoluer. Nous souhaitons lui témoigner toute notre reconnaissance pour l'aide qu'elle nous a apporté lors de notre projet, pour sa disponibilité et son suivi en tant qu'encadrante, mais aussi en tant que professeur et que personne. Mme Rebai-Jribi est une personne humaine et entière, de bonne humeur communicative, sympathique et à l'écoute. Son œil expert dans le domaine du Big-Data nous a aidé à aller jusqu'au bout de notre projet, et nous a été de bons conseils dans des moments de doute.

SOMMAIRE

I.	<i>INTRODUCTION GENERALE</i>	6
II.	<i>PROBLEMATIQUE ET SOLUTION PROPOSEE</i>	6
1.	Contexte et Motivation	6
2.	Objectif du projet	7
3.	Analyse de la problématique et méthodologie adoptée	7
4.	Choix techniques	8
III.	<i>Conception et implémentation de la solution</i>	9
1.	Architecture générale de l'application de classification de sentiments	9
2.	Collecte de données	10
a)	Choisir du site et de la catégorie de restaurant	10
b)	Méthodologie utilisée	12
c)	Techniques utilisées	12
d)	Langages/ Librairies utilisées.....	12
e)	Etape de réalisation/algorithme	13
f)	L'output/résultat de l'algorithme :	17
3.	Nettoyage et exploration	18
a)	Le nettoyage.....	18
b)	Exploration et visualisation des données	19
c)	Analyse de mots ou groupes de mots	24
4.	Préparation des données	29
a)	Pré traitement des données.....	29
b)	Stemming & lemmatization.....	30
5.	Entraînement du modèle	31
a)	La vectorisation	31
b)	Modèle de Machine Learning.....	33
c)	Les hyperparamètres.....	36
d)	Pipeline.....	39
e)	Modèle final sélectionné	40
6.	Exposition du modèle	41
a)	Le Front-end grâce à Flask.....	41
b)	Convention pour l'arborescence de Flask	42
c)	Pickling	42
d)	Détection de la langue	42
e)	Design.....	43
IV.	<i>Conclusion et Perspectives</i>	45

Table des figures

Figure 1 : Étape d'un processus de Machine Learning	8
Figure 2: Librairies utilisées	9
Figure 3: Architecture globale de l'application.....	9
Figure 4: Ordre de création de nos fichiers csv et python	13
Figure 5: Organigramme du fichier scraping_URL.py.....	14
Figure 6: Organigramme de notre fichier de scraping_reviews.py.....	15
Figure 7: Organigramme de notre fonction de scraping	16
Figure 8: Information sur les datasets	19
Figure 9: Net promoteur score	20
Figure 10 : Graphique et échelle initiale.....	20
Figure 11: Graphique de répartition du dataset avec labélisation finale.....	21
Figure 12: Graphique de répartition de la note en fonction de l'année	21
Figure 13: Distribution du nombre d'avis en fonction du mois.....	22
Figure 14: Diagramme en boite de la longueur du commentaire selon le label	22
Figure 15 : Distribution de la longueur des commentaires positifs et négatifs.....	23
Figure 16: Matrice de corrélation entre la longueur et le label.....	24
Figure 17:Graphique des mots unitaires les plus fréquents dans le dataset positif.....	25
Figure 18: Graphique des mots groupés les plus fréquents dans le dataset positif.....	25
Figure 19: Graphique des deux mots groupés les plus fréquents dans le dataset négatif	26
Figure 20: Groupes de mots les plus fréquents dans le dataset négatif	26
Figure 21: Étude des points d'exclamation pour les datasets négatifs et positifs	27
Figure 22: Wordcloud des datasets négatifs et positif avant nettoyage	28
Figure 23: Wordcloud des datasets négatifs et positifs après nettoyage.....	28
Figure 24: Exemple appliqué de stommatisation	30
Figure 25: Exemple appliqué de lematisation.....	30
Figure 26: Exemple de stommatisation et lematisation d'un de nos commentaires.....	31
Figure 27: Exemple appliqué d'un TF-IDF	32
Figure 28: Exemple appliqué d'un CountVectoriser	33
Figure 29: Schéma explicatif du Random Forest.....	34
Figure 30: Schéma explicatif d'une régression linéaire.....	35
Figure 31: Schéma du séparateur à vaste marge (SVM).....	35
Figure 32 : Comparaison du GridSearch et du RandomSearch	37
Figure 33: Accuracy vs Prédiction.....	38
Figure 34: Principe de la matrice de confusion	38
Figure 35: Recherche des meilleurs paramètres à travers d'un pipeline	40
Figure 36: Tableau des résultats finaux obtenus avec les différents modèles	41
Figure 37: Arborescence de Flask.....	42
Figure 38: Schéma de l'utilisation de l'application web.....	44

I. INTRODUCTION GENERALE

Dans le cadre de notre quatrième année du cycle ingénieur, et ayant comme majeure Big Data au sein de l'ESME Sudria, il nous est proposé un projet de 3 mois. Ce projet nous donnant l'opportunité de mettre en pratique nos connaissances et nos compétences, nous a été présenté au travers d'un cahier des charges dont la finalité revient à la conception et au développement d'une interface web permettant l'analyse et la classification automatique d'avis (commentaires) saisis par des utilisateurs. Cette classification est basée sur un modèle d'analyse de sentiment entraîné sur des avis antérieurs. Cette thématique est liée au domaine d'apprentissage automatique (Machine Learning) et de Traitement de Langage Naturel (NLP). Le NLP est une technique d'intelligence artificielle qui vise à permettre aux ordinateurs de comprendre le langage humain, à travers l'analyse de traitement de texte. Grâce aux algorithmes de Machine Learning, on entraîne des données, dans le but d'analyser le langage humain et de prédire une réponse. De nos jours, ces domaines sont en plein essor et permettent la résolution d'une multitude de problèmes dans des secteurs très variés comme pour la traduction automatique, la vérification orthographique, les chatbots, ou encore les assistants personnels comme Siri et Alexa, par exemple.

C'est dans ce cadre que s'inscrit notre projet d'analyse sentimentale des commentaires écrits par des utilisateurs. Pour répondre au besoin exprimé, à savoir la prédiction des sentiments, nous nous sommes appuyées sur les méthodes d'apprentissage supervisé que nous avons vu en cours de Machine Learning. Ce projet fut pour nous enrichissant puisqu'il nous a permis de mobiliser l'ensemble de nos connaissances acquises en cours que nous avons changé en compétences, et nous a permis d'aller plus loin avec un projet de traitement de données textuelles, allant de la collecte des données à la création d'une API REST, permettant à des utilisateurs d'analyser un commentaire.

Le présent rapport détaille les différentes étapes par lesquelles nous sommes passées pour réaliser notre projet. Dans un premier temps, nous présenterons la problématique, les objectifs de notre projet ainsi que la solution proposée au problème à travers la présentation de son architecture globale. Nous aborderons dans un second temps la conception et l'implémentation des différentes étapes nécessaires au processus de résolution du problème. Enfin nous conclurons sur le résultat obtenu et les possibles perspectives d'amélioration de notre solution.

II. PROBLEMATIQUE ET SOLUTION PROPOSEE

1. Contexte et Motivation

Le suivi de la perception d'une marque et l'évolution de sa réputation auprès de ses clients constitue un enjeu majeur pour les entreprises. Celles-ci cherchent à comprendre au mieux le consommateur, afin de remanier leurs différentes stratégies marketing et d'améliorer, par exemple, leur expérience client. En effet, la majeure partie des consommateurs consulte des

avis clients en ligne avant de passer à l'acte d'achat, plaçant leur décision entre les mains des commentaires (négatifs ou positifs) partagés par d'autres consommateurs après avoir pesé le pour et le contre. Cela est d'autant plus valable pour les entreprises proposant des services comme la restauration. Ainsi l'étude des avis clients est incontournable pour fidéliser, gagner et ne pas perdre de clients potentiels.

C'est dans ce contexte, et afin d'améliorer leur réputation et le retour des clients sur le web, que les entreprises doivent analyser avec efficacité les avis et commentaires partagés sur leur site. Il est important de détecter immédiatement et automatiquement les avis négatifs au sujet d'un service donné afin d'identifier le problème rencontré et de trouver une solution à ce dernier. Cela permettra d'améliorer un produit/service initialement défectueux, d'éviter d'autres plaintes du même type et d'assurer une bonne qualité de services à l'entreprise et ainsi une répercussion directe sur la satisfaction client. D'autre part, une réponse rapide à un commentaire négatif permet d'améliorer la relation client et de temporiser le sentiment du client mécontent à travers une bonne réactivité aux messages, des propositions d'alternatives et de solutions à son problème. Elle permet également de mieux comprendre les attentes de celui-ci et d'améliorer l'expérience du client.

2. Objectif du projet

L'objectif de ce projet est de développer un application web permettant de détecter si l'avis saisi par un utilisateur correspond à une opinion « positive » ou « négative ». Pour ce faire, nous allons nous appuyer sur les techniques de traitement de données textuelles ainsi que d'apprentissage automatique, afin de proposer un modèle de prédiction pertinent. Ce modèle de prédiction de sentiment, sera entraîné sur des avis récupérées antérieurement.

Sans perte de généralité, nous avons choisi de réaliser cette application en se basant sur des commentaires liés au domaine de la restauration, dans la mesure où les jeux de données accessibles sur Internet étaient beaucoup plus fournis et pertinents que ceux d'autres domaines.

3. Analyse de la problématique et méthodologie adoptée

Nous avons affaire à un problème de classification binaire classique dans lequel nous allons extraire de la connaissance à l'issue d'un apprentissage supervisé sur des données antérieures labellisées.

Pour traiter notre problématique de prédiction, nous allons suivre les différentes étapes d'un processus typique de Machine Learning. On distingue les sept étapes clés suivantes, illustrées par la figure 1 :

- **Collecte des données** : Rassemblement des données nécessaires à l'entraînement de Machine Learning.
- **Préparation des données** :

- Nettoyage des données : identification et correction des données altérées/inexactes/ non pertinentes.
- Visualisation des données/ Recherche de corrélation.
- Prétraitement des données : nettoyage du texte, suppression des emojis, suppression des stop-words, regroupement des mots, vectorisation.
- **Choix du modèle de Machine Learning** : Sélection d'un sous-ensemble de modèles de Machine Learning appropriés au NLP.
- **Entraînement du modèle** : Implémentation des modèles de ML d'apprentissage supervisé et entraînement des données sur le trainSet.
- **Test et évaluation du modèle** : Évaluation sur les données de test.
- **Ajustement des paramètres** : Réglage des hyper-paramètres pour éviter des problèmes d'overfitting.
- **Déploiement du modèle** : Création d'une interface web graphique permettant à un utilisateur d'évaluer un commentaire.

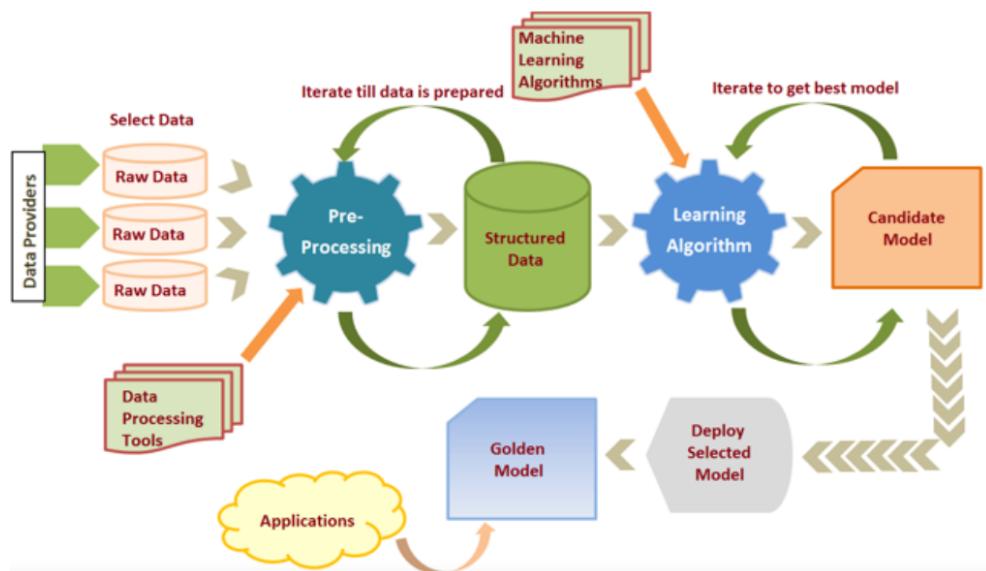


Figure 1 : Étape d'un processus de Machine Learning

4. Choix techniques

Tout au long du projet, nous avons utilisé le langage python, que ce soit pour la collecte des données, l'implémentation de l'interface graphique ou pour l'analyse des données et l'implémentation des algorithmes du Machine Learning. Il nous a fallu maîtriser également les langages HTML, CSS et Javascript pour la partie collecte de données. Par ailleurs, nous avons utilisé plusieurs bibliothèques OpenSource pour la réalisation des différents composants de notre application, entre autres les librairies Sélénum, BeautifulSoup, Scikit-learn, NLTK, Textblob, langig, Spacy, Matplotlib, Flask. La figure 2 donne un aperçu sur les différentes bibliothèques utilisées, catégorisées selon leurs fonctionnalités.

Collecte des données	Apprentissage automatique	Traitement automatique des langues	Visualisation	Interface graphique web
<ul style="list-style-type: none"> - BeautifulSoup • - Sélénum 	<ul style="list-style-type: none"> • - Scikit-learn 	<ul style="list-style-type: none"> • NLTK • Textblob • Langid • Spacy 	<ul style="list-style-type: none"> • Matplotlib 	<ul style="list-style-type: none"> • Flask

Figure 2: Librairies utilisées

III. Conception et implémentation de la solution

1. Architecture générale de l'application de classification de sentiments

La figure 3 illustre l'architecture globale de notre application web de prédiction. Dans les sections suivantes, nous détaillons la conception et les détails d'implémentation de chacune de ses composants :

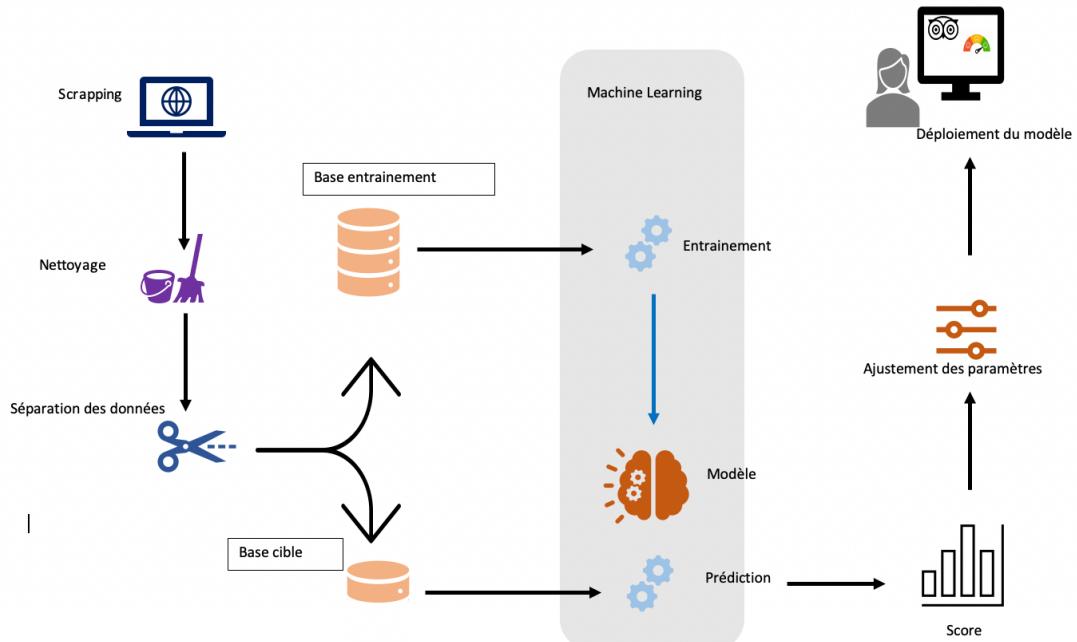


Figure 3: Architecture globale de l'application

2. Collecte de données

a) Choisir du site et de la catégorie de restaurant

Initialement, et dans le but de construire le Dataset d'entraînement, nous nous sommes demandé depuis quel site, nous pouvons collecter des données pertinentes et assez structurées. Nos réflexions nous ont rapidement dirigé vers le site TripAdvisor. Par la suite la question était sur quel type de restaurants nous souhaitions travailler. Il nous fallait une catégorie de restaurant regroupant un nombre d'avis assez important avant d'avoir un dataset équilibré et donc performant lors de son entraînement.

Au départ, nous souhaitions travailler sur un projet qui nous semblait réellement utile et qui marquait notre quotidien. C'est ainsi que notre premier choix s'est porté vers la restauration de la Réunion, ayant un membre du groupe résidant là-bas. Malheureusement, nous nous sommes vite rendu compte que la quantité de données fournies avec ces filtres n'était pas suffisante aux vues du nombre de restaurants et de commentaires comparé à un projet de taille en Big Data.

Nous avions donc voulu analyser les avis sur tous les restaurants de Paris. Mais nous nous sommes rendu compte que, pour chaque type de cuisine, il y avait des mots clefs récurrents, et un champ lexical propre à chacun. Par exemple, les avis des restaurants italiens et asiatiques ont un champ lexical totalement différent dû à leurs plats différents. Pour faciliter l'analyse, il nous a fallu donc travailler sur un seul type de cuisine.

Finalement, notre choix s'est porté vers la construction d'un dataset de commentaires sur **les restaurants à Paris de cuisine française**.

Pour cela, nous devions récupérer l'url de TripAdvisor qui nous donnait accès à la liste des restaurants prédéfinis au-dessus. Dans la catégorie « restaurant » du site, nous avons appliqué le filtre « cuisine française ».

Nous avons rencontré à ce moment un autre problème. Lorsque l'on cochait sur le site « cuisine française », cela revenait à appliquer un filtre sur tous les restaurants de Tripadvisor et nous avions donc le résultat voulu.

Cuisine
<input type="checkbox"/> Française
<input type="checkbox"/> Européenne
<input type="checkbox"/> Asiatique
<input type="checkbox"/> Italienne
Afficher plus ▾

Cependant, ce filtre n'était pas pris en compte dans notre URL et si nous revenions le lendemain sur la même URL le filtre « cuisine française » n'était plus là. Nous tombions donc sur la page suivante.

URL1 : https://www.tripadvisor.fr/Restaurants-g187147-Paris_Ile_de_France.html

The screenshot shows the Tripadvisor search interface for Paris restaurants. The search bar at the top contains the query "Meilleurs Restaurants à Paris, Île-de-France". Below the search bar, there are filters for date (22/09/2020), time (20:00), and number of people (2 personnes). A button to "Rechercher un restaurant" is present, along with a "Voir les restaurants avec chèques-cadeaux" link. On the left, a sidebar includes a map with a "Voir la carte" button, a COVID-19 section, and filters for restaurant type (checked for "Restaurants"), delivery service, and takeout. The main content area displays two rows of restaurant cards. The first row is titled "Livraison possible" and includes four cards: "Le Cherine" (Lebanese, 305 avis, €€-€€€), "Joyti Restaurant" (Indian, 392 avis, €€-€€€), "New Jawad Longcha..." (Indian, 2230 avis, €€-€€€), and "Baoli Bao" (Asian, Fusion, 394 avis, €€-€€€). Each card has a "Commander en ligne" button. The second row is titled "Terrasse extérieure disponible" and includes four cards: "Le Cirque" (French, 294 avis, €€-€€€), "Il Etait Un Square" (French, 3438 avis, €€-€€€), "Petit Boutary" (French, 391 avis, €€-€€€), and "Pizzeria Arrivederci" (Italian, P..., 871 avis, €€-€€€). Each card also has a "Commander en ligne" button. A "Tout afficher" link is located at the bottom right of each row.

Afin de contrer ce problème, nous nous sommes rendu compte que lorsqu'on effectue une recherche « cuisine française » dans la barre de recherche présente sur le site Tripadvisor cela avait une influence sur l'URL et nous permettait donc de retrouver la liste des restaurants avec le filtre voulu.

URL2 : https://www.tripadvisor.fr/Restaurants-g187147-c20-Paris_Ile_de_France.html

The screenshot shows the Tripadvisor search interface for French restaurants in Paris. The search bar at the top contains the query "Meilleurs restaurants français à Paris". Below the search bar, there are filters for date (21/09/2020), time (20:00), and number of people (2 personnes). A button to "Rechercher un restaurant" is present, along with a "Voir les restaurants avec chèques-cadeaux" link. On the left, a sidebar includes a map with a "Voir la carte" button, a COVID-19 section, and filters for restaurant type (unchecked for "Restaurants"), delivery service, and takeout. The main content area displays a list of restaurants. At the top, it says "7248 résultats correspondent à vos filtres Supprimer tous les filtres" and "Trier par : Pertinence". A banner at the top right says "Économisez jusqu'à 50 % dans des restaurants de Paris en réservant sur Tripadvisor" and "Voir toutes les offres". The list includes two cards: "Le Boui Boui" (French, 149 avis, €€-€€€) and "I. Il Etait Un Square" (French, Steakhouse, 3438 avis, €€-€€€). Each card has a "Réserver" button.

On peut constater que dans la dernière photo le filtre « Français » a été enregistré. Pour résoudre ce problème, une deuxième méthode s'offrait à nous, si nous n'avions pas trouvé de résolution grâce à l'URL. Nous aurions par exemple pu faire appel à Sélénum afin de cocher la case cuisine française à chaque fois que le lien était ouvert.

b) Méthodologie utilisée

Une fois notre type de restaurant défini, et le filtrage appliqué au site, nous avons dû procéder à la récupération des avis de chaque restaurant. Nous avons donc dû nous demander quelles étaient les informations à récupérer sur chaque avis. Voici le détail de ce que peut contenir un avis Tripadvisor : une note de 0 à 5, un titre, un commentaire, des éventuelles photos, la date de publication, la date de visite, le nombre de réactions des utilisateurs sur l'avis en question ainsi que le mode de publication (depuis un mobile ou non).

Nous avons retenu parmi ces informations : le titre, la note, le commentaire et la date de visite. Ici la note est intéressante car elle nous permet de labelliser le commentaire. Cela sera expliqué en détail dans la partie « exploration de données ».

c) Techniques utilisées

Étant donné que nos données se trouvent en quantité importante sur le Web, il est d'usage d'utiliser des techniques existantes pour « gratter » nos données. C'est dans ce contexte que nous avons utilisé deux méthodes : le scraping et le crawling.

Le scraping est une technique permettant d'extraire, de « gratter » des informations d'un ou de plusieurs sites web de manière totalement automatique. Il s'agit de scripts, chargés d'extraire ces informations. Le scraping a ses limites, et surtout dans notre projet. Il sert uniquement à récupérer des données statiques. Si la page utilise des données dynamiques, le scraping n'est pas suffisant. C'est là qu'intervient le crawling. Le crawling permet de collecter, de « ramper » sur le contenu d'une page Web. Cela peut être représenté par un robot qui navigue sur des pages web : On appelle robot de crawl, ou spider, le logiciel dont la mission est d'explorer le web et de capter les ressources d'une page web en suivant des liens ou à partir d'une liste. Dans le cadre du projet, nous avons utilisé le crawling à travers Sélénum.

d) Langages/ Librairies utilisées

Pour la collecte de données, nous avons utilisé BeautifulSoup et Sélénum, qui sont deux librairies python.

Beautifulsoup est utile pour le scraping, qui analyse la syntaxe d'un document HTML. L'utilisation de cette librairie nous a permis de récupérer les informations qui nous intéressaient, contenues dans les balises HTML que nous avions sélectionnées (titre, note, commentaire, date de visite).

Comme dit auparavant, BeautifulSoup se limite aux données statiques, c'est pourquoi nous avons eu besoin de Sélénum.

Sélénum imite le comportement d'un navigateur et a été utilisé pour le crawling afin d'accéder à certains boutons comme « afficher plus » si les commentaires étaient longs, ou encore pour cliquer sur les pages suivantes.

Afin de stocker les données dans un tableau, nous avons utilisé la librairie Pandas de python. Pandas, permet de manipuler des données à analyser. Elle permet de stocker les données dans des tableaux, des Dataframes avec des variables dans des colonnes et des individus dans les lignes.

Nous avons donc utilisé principalement des librairies de python. Malgré cela, il était nécessaire d'avoir une compréhension des langages HTML, CSS (qui nous permettaient de cibler les balises) et JavaScript. En effet, la compréhension des langages HTML et CSS était un prérequis afin de pouvoir analyser les pages web et savoir où prélever les informations qui nous intéressaient.

e) Etape de réalisation/algorithme

Nous avons choisi de récupérer les URL de chaque restaurant. Au départ, nous voulions scraper commentaire par commentaire les commentaires d'un restaurant, puis revenir à un autre restaurant pour faire de même, mais le crawling prenait beaucoup trop de temps, et il était moins laborieux pour nous de récupérer au préalable les URL de chaque restaurant.

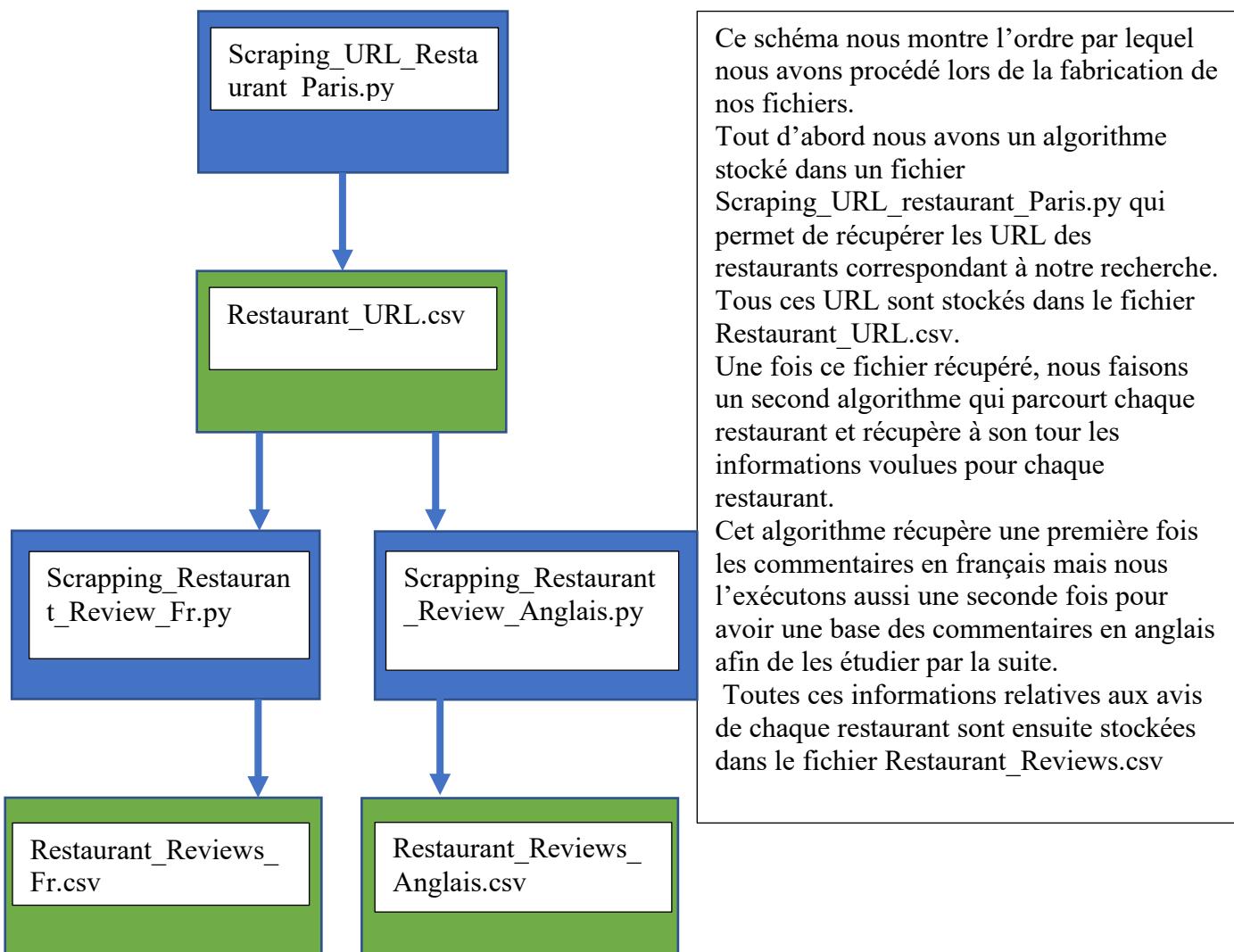


Figure 4: Ordre de création de nos fichiers csv et python

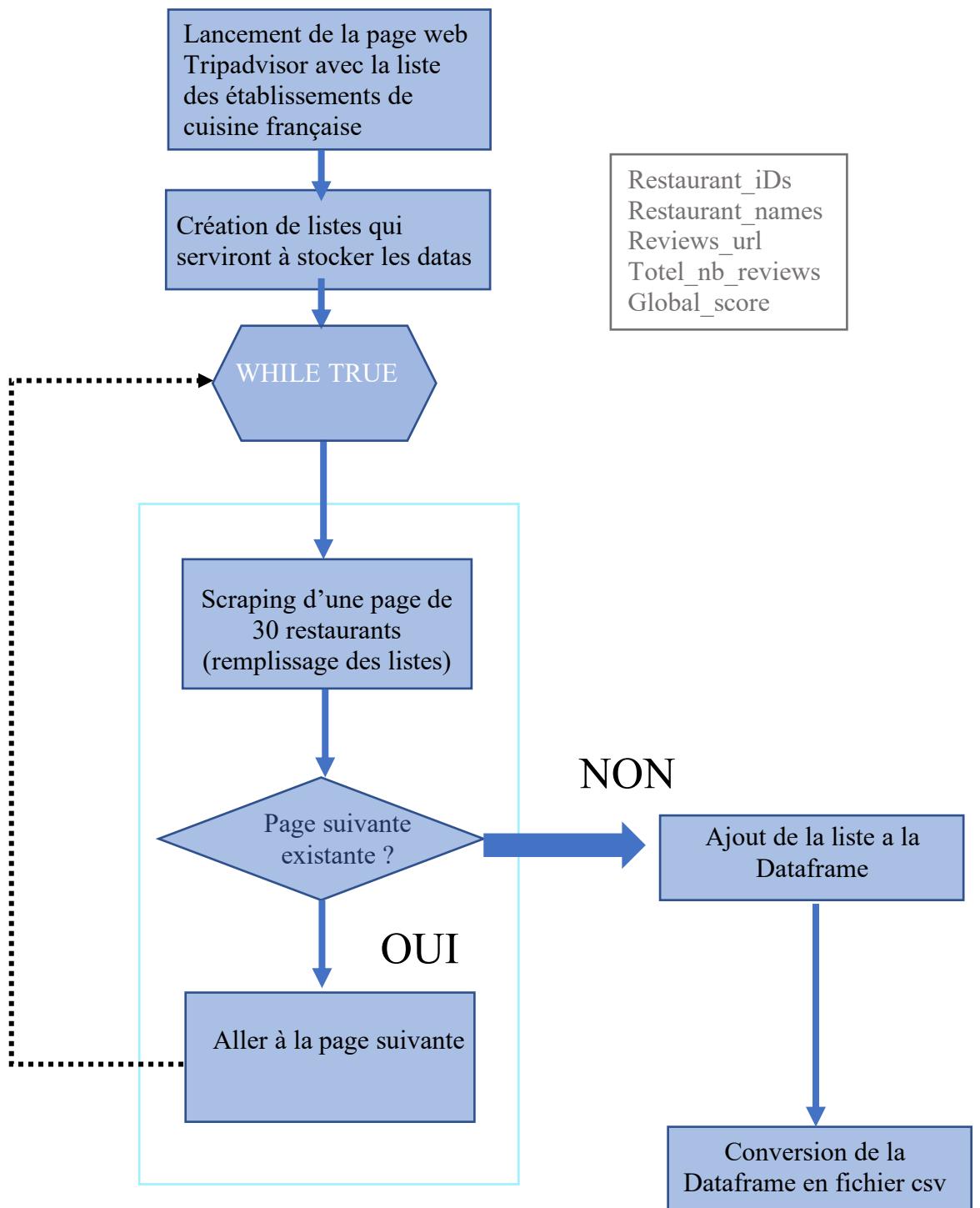


Figure 5: Organigramme du fichier scraping_URL.py

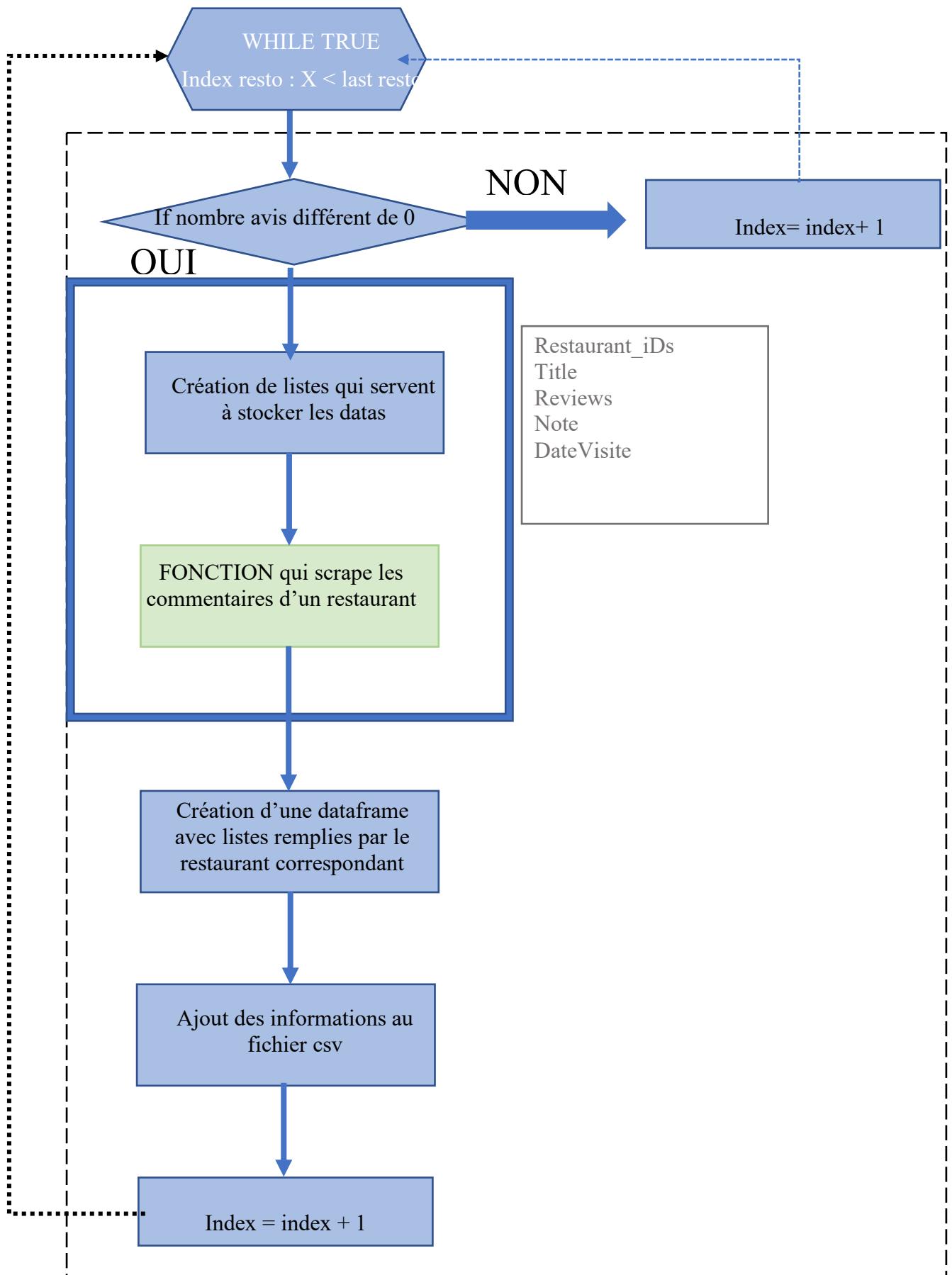


Figure 6: Organigramme de notre fichier de scraping_reviews.py

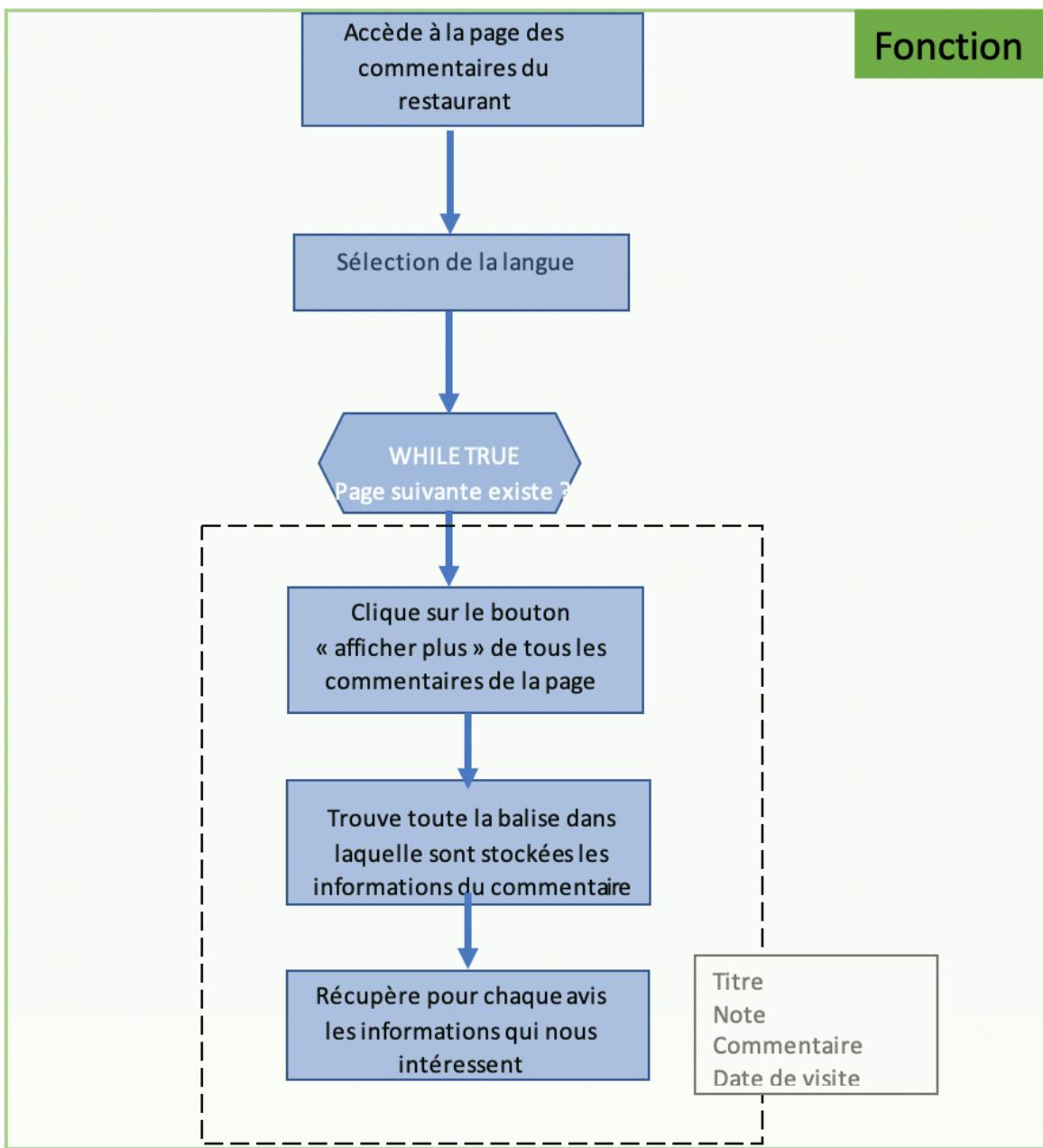
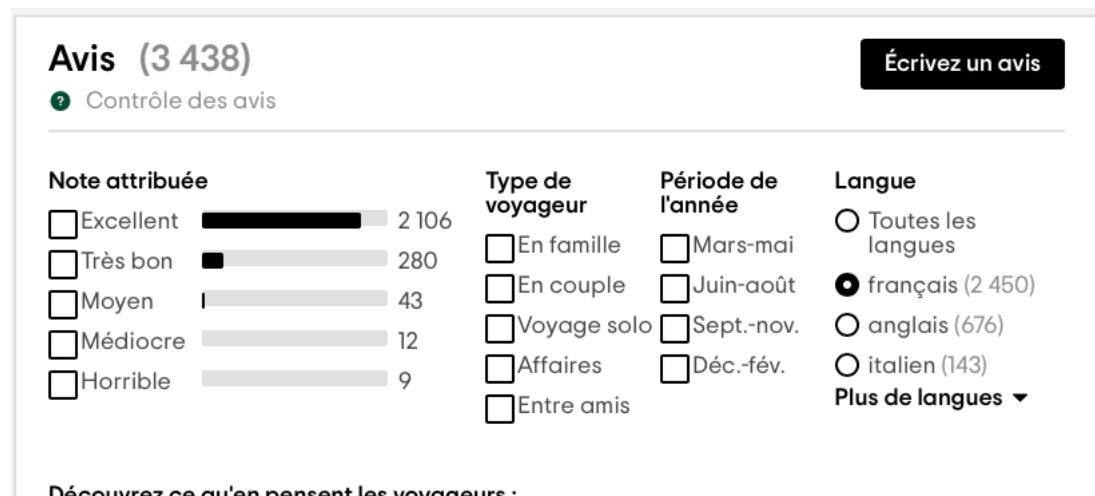


Figure 7: Organigramme de notre fonction de scraping

A travers ces trois schémas, on retrouve l'organisation complète de notre scraping. Nous avions un premier algorithme qui récupérait tous les URL de chaque restaurant et les stockait dans un fichier csv. Le second schéma représente notre boucle qui permettait de passer à travers chaque page et d'enregistrait les données. Le dernier schéma explique la fonction qui intervient dans le second schéma. Celui-ci a pour but d'expliquer comment nous avons procédé afin de récupérer sur chaque page de commentaire l'ensemble des commentaires du

restaurant. Il faut savoir qu'il y avait environ une dizaine de commentaires sur chaque page. Un restaurant avait donc plusieurs pages de commentaires. Nous avons effectué le même travail sur des commentaires en anglais.

Le scraping des commentaires en anglais est possible grâce à l'ajout d'une ligne dans le code qui nous permet de sélectionner la langue (voir photo). La sélection de la langue se fait sur la page de chaque restaurant comme on le voit ci-dessous pour un de nos établissement.



f) L'output/résultat de l'algorithme :

Après avoir exécuté nos deux algorithmes, on obtient donc deux fichier csv comme expliqué sur le schéma 1.

Le premier fichier csv « Restaurant_URL.csv » contient la liste des url des restaurants de cuisine française, soit un fichier de 7 224 url. Si l'on regarde la page Tripadvisor, on remarque lors de notre recherche, qu'il y a exactement 7 247 restaurants différents. Cette différence s'explique par le fait que nous n'avons pas récupéré les restaurants qui n'avaient aucun avis, puisqu'il n'aurait rien apporté à notre dataset final, dont le but était de regrouper des avis. Le fichier « Restaurant_URL.csv » contient donc 7 224 url de restaurants avec pour chacun un numéro d'identifiant du restaurant, le nom, l'url, le nombre d'avis et le score général.

Le second fichier csv « restaurant_Reviews.csv » contient pour chaque restaurant du fichier « Restaurant_URL.csv » l'identifiant du restaurant (qui est le même que dans le fichier où est stocké l'url), le titre du commentaire, le commentaire, la note, la date de visite et le nom du restaurant. Nous avons donc deux fichiers finaux « all_restao_reviews_fr.csv » avec 219 425 commentaires de restaurants ainsi que le même fichier en anglais contenant 190 298 commentaires.

3. Nettoyage et exploration

a) Le nettoyage

Le nettoyage des données d'un Dataset est nécessaire afin d'améliorer son utilisation future. Il permet de réduire les incohérences afin de prendre des décisions plus précises. Le nettoyage consiste à identifier et corriger les données altérées, inexactes ou non pertinentes. Ce processus permet donc de résoudre le problème des valeurs manquantes ou doubles, aberrantes ou encore nulles. Avec un Dataset propre, nous pourrons avoir une meilleure analyse de nos données et une meilleure efficacité.

Globalement, le nettoyage des données s'est fait au moment même du scraping, dans la récupération des données.

Avant de récupérer un commentaire et un titre, nous avons procédé à la suppression des tabulations et retours à la ligne inutiles qui pouvaient être gênants lors de la visualisation de notre fichier csv.

Il nous a fallu récupérer le nombre d'avis pour chaque restaurant afin de pouvoir ignorer ceux qui n'ont aucun commentaire dès la phase de scraping de celui-ci.

Nous nous sommes rendu compte que, parfois, des avis ne présentaient pas de date de visite, c'est ainsi que, plutôt que d'enregistrer l'avis dans la Dataset sans date, et de faire un nettoyage à posteriori, nous avons jugé plus pertinent de prendre directement la date de publication du commentaire qui était toujours présente.

Lors de l'utilisation de notre Dataset, nous avons eu quelques erreurs sur notre code qui nous ont permis de nous rendre compte que certaines valeurs récupérées étaient vides, par exemple, certains commentaires n'avaient pas de titre et cela nous a posé problème : nous obtenions des tailles différentes au sein du dataset en fonction de la variable. Nous avons aussi pu constater que certains des commentaires étaient en chinois et donc que l'algorithme ne les comprenait pas. Il a donc fallu effectuer un nettoyage du dataset malgré nos conditions lors de la récupération des données.

Afin de repérer les langues utilisées dans notre dataset nous avons utilisé la librairie « langid » qui nous a permis de supprimer les commentaires qui n'étaient pas en français dans les datasets français et anglais.

```

fr = pd.read_csv('all_reviews_Fr_clean.csv', sep=';')

fr.shape
(218360, 6)

fr.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 218360 entries, 0 to 218359
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   restau_ID   218360 non-null  int64  
 1   Titre        218360 non-null  object  
 2   Note         218360 non-null  float64 
 3   Commentaire  218360 non-null  object  
 4   DateVisite   218360 non-null  object  
 5   restaurant   218360 non-null  object  
dtypes: float64(1), int64(1), object(4)
memory usage: 10.0+ MB

en= pd.read_csv('all_reviews_En_clean.csv', sep=';')

en.shape
(115331, 6)

en.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 115331 entries, 0 to 115330
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   restau_ID   115331 non-null  int64  
 1   Titre        115331 non-null  object  
 2   Note         115331 non-null  float64 
 3   Commentaire  115331 non-null  object  
 4   DateVisite   115330 non-null  object  
 5   restaurant   115330 non-null  object  
dtypes: float64(1), int64(1), object(4)
memory usage: 5.3+ MB

```

Figure 8: Information sur les datasets

b) Exploration et visualisation des données

i. *Un dataset équilibré*

Après avoir scrapé un certain nombre d'établissements, qui nous ont valu un dataset assez gros, nous avons voulu labéliser nos données. Deux problèmes se sont alors posés à nous. Après quelques recherches, nous avons établi une échelle de séparation basée sur le « net promoteur score » (Figure 7 : Net promoter Score), qui explique que sur une échelle de 0 à 10 les promoteurs sont uniquement les personnes qui mettent une note de 9 ou 10 sur 10, et les personnes insatisfaites mettront elles une note pouvant aller jusqu'à 6/10. Nous avons donc établi une échelle qui notait le commentaire négatif pour une note allant jusqu'à 3/5, positif au-dessus de 4 (voir figure 8) et neutre sinon.

Grâce à cette nouvelle classification, nous obtenions un graphique plus juste, et équilibré (Figure 9).

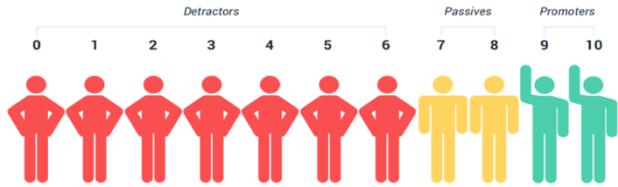


Figure 9: Net promoteur score

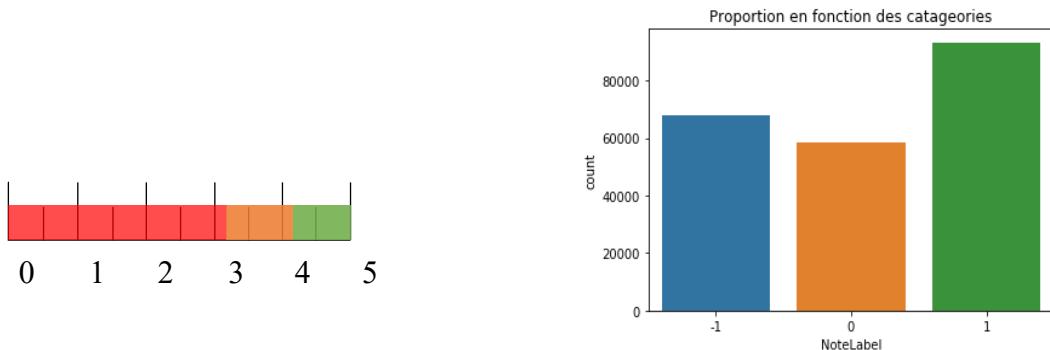


Figure 10 : Graphique et échelle initiale

Nous avons cependant décidé de partir sur une classification binaire des commentaires, soit positif soit négatif puisque nous trouvions au final assez difficile de déterminer si un commentaire était réellement neutre. Il y avait souvent une tendance plutôt positive ou négative dans la finalité. Partir sur une labélisation binaire nous a aussi permis d'être plus sûr au niveau de la prédiction de notre algorithme de machine Learning puisqu'il était difficile de déterminer la neutralité d'un mot. Les mots bons ou mauvais sont facilement associables à un sentiment positif ou négatif. Mais peu de mot son vraiment neutre. Un commentaire pouvait être neutre dans le cas où il y avait autant de négatif que de positif dedans mais cela aurait été plus difficile à mesurer.

Nous obtenons donc le graphique suivant (figure 10) avec un dataset comprenant 70% de commentaires positifs et 30% de commentaires négatifs. Ce qui est assez équilibré pour avoir de bonne prédiction grâce à nos modèles de Machine Learning utilisés par la suite. Par ailleurs, nous utiliserons différentes techniques pour pallier les problèmes qui peuvent être engendré par ce petit déséquilibre de la distribution de nos classes prédites, entre autres les techniques d'échantillonnage et de « Cross-validation ». Nous reviendrons sur ces aspects dans la section entraînement du modèle.

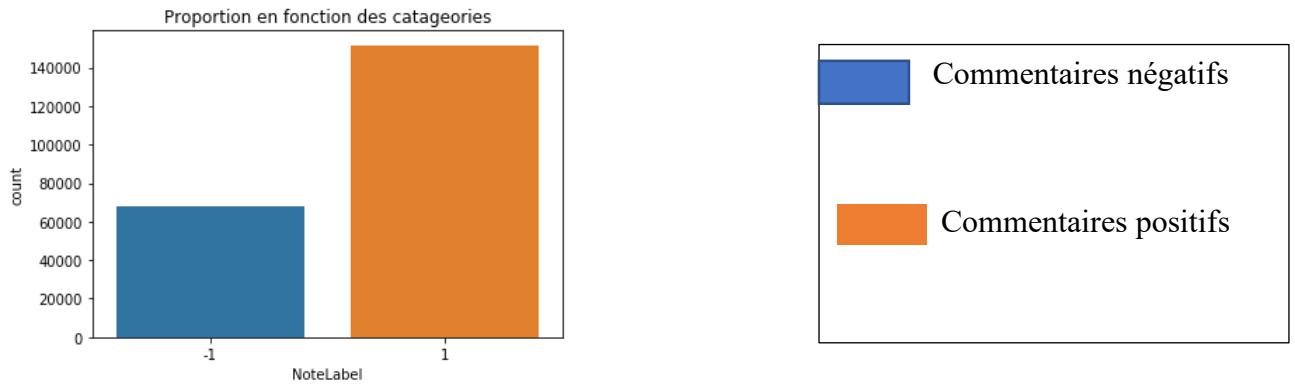


Figure 11: Graphique de répartition du dataset avec labélisation finale

ii. Note en fonction du temps (années+ mois)

Nous avons étudié la distribution temporelle des avis en fonction de leur label entre 2007 et 2020 afin de voir s'il y avait une corrélation possible. Nous avons obtenu les deux graphiques suivants :

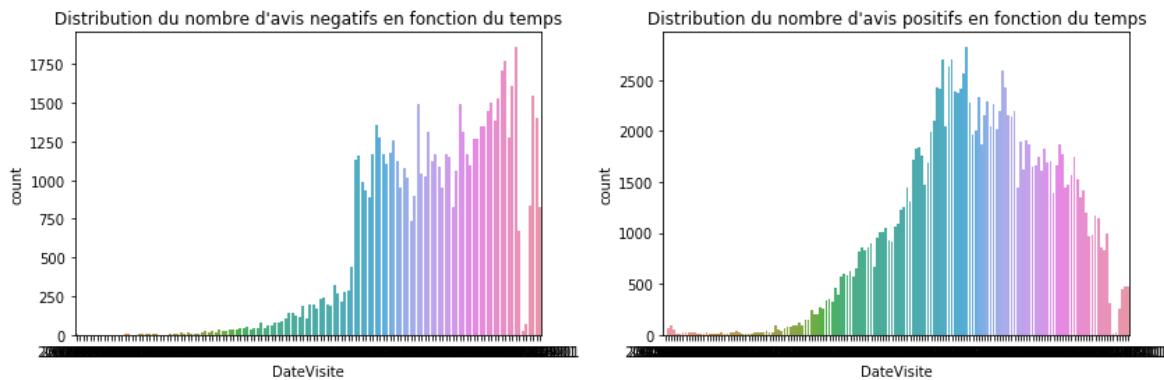


Figure 12: Graphique de répartition de la note en fonction de l'année

On ne remarque pas de relation précise entre la note et le temps puisque les deux graphiques sont très poches. Cependant, on peut voir qu'il y a une réelle croissance des commentaires sur le site Tripadvisor aux alentours de 2010 dû à l'essor d'internet.

Nous nous sommes tout de même interrogées sur l'existence d'un lien plus subtile en rapport avec le temps, soit un lien entre les mois de l'année et les notes. Pour cela nous avons groupé les commentaires par mois.

Nous observerons ci-dessous les graphiques résultants de ce regroupement.

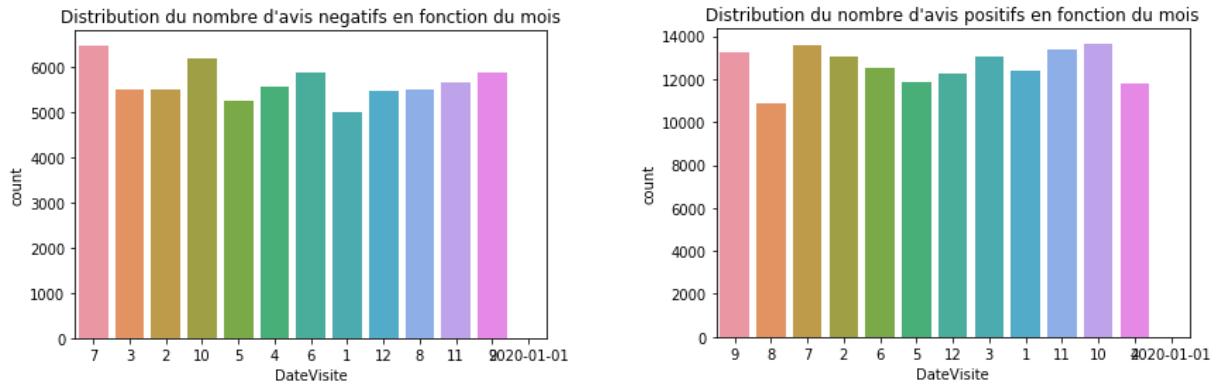


Figure 13: Distribution du nombre d'avis en fonction du mois

On remarque à nouveau que les tendances sont similaires, ainsi la piste temporelle n'est pas pertinente.

Nous nous sommes alors dirigées vers une autre piste : existe-t-il une relation entre la longueur du commentaire et la note attribuée par le client ?

iii. Taille du commentaire en fonction de la note

Pour cela nous avons affiché le nombre de mots pour les commentaires neutres et positifs.

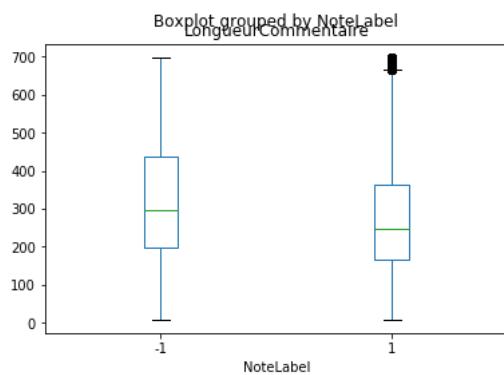


Figure 14: Diagramme en boite de la longueur du commentaire selon le label

En observant le premier graphique, nous pouvons constater que, malgré la présence de certaines valeurs aberrantes qui ne sont pas à prendre en compte, la longueur des commentaires ne semble globalement pas corrélée à la catégorie du commentaire. C'est la raison pour laquelle nous avons affiché la médiane, qui reste proche pour chaque label. Nous observons une légère tendance à écrire plus pour des commentaires négatifs.

Cependant, en tenant compte de la distribution et de la moyenne de chacun des graphiques il semble compliqué d'y voir une corrélation entre la taille du commentaire et le label de celui-ci. Nous avons donc mis cette piste de côté.

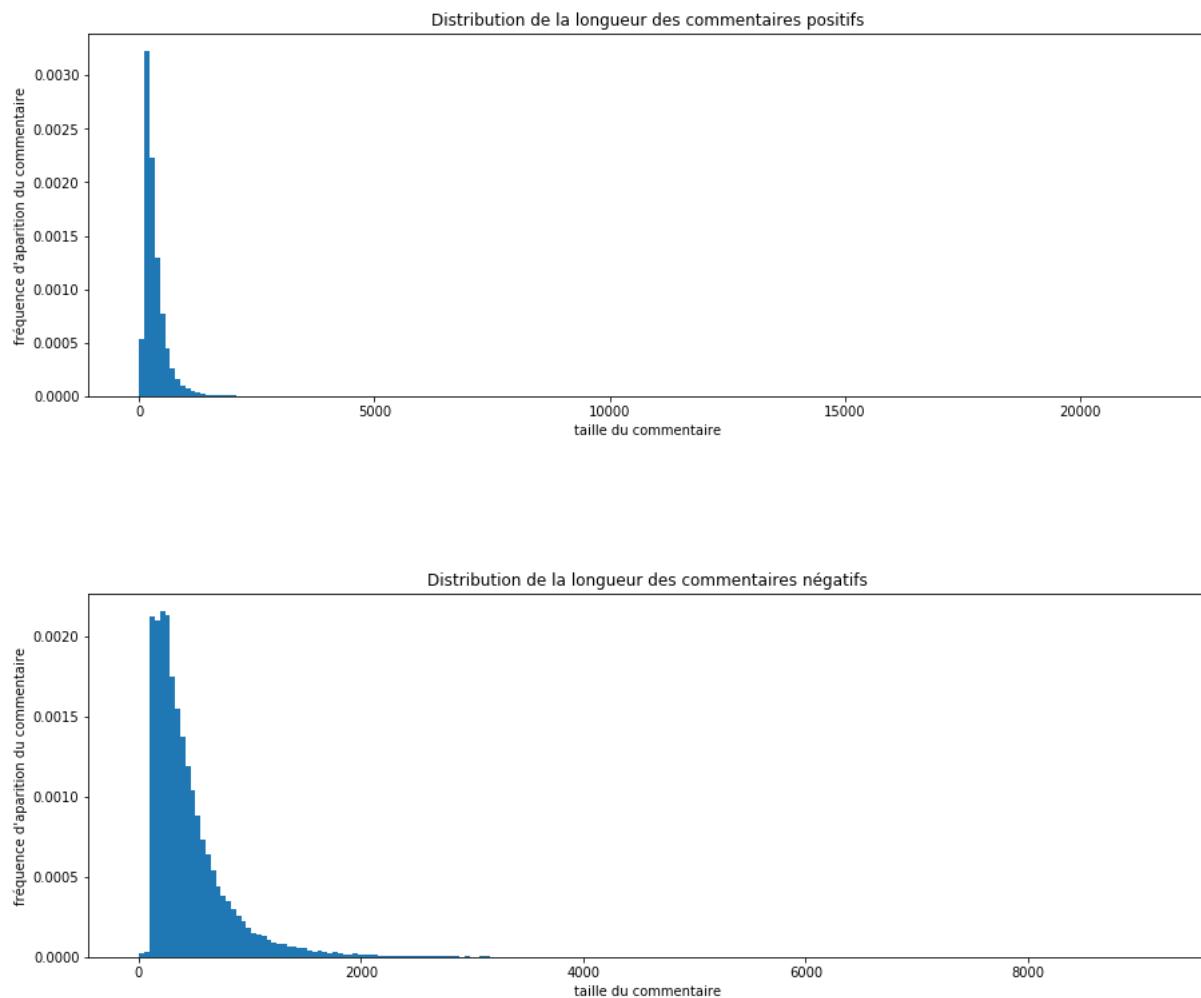


Figure 15 : Distribution de la longueur des commentaires positifs et négatifs

iv. *Matrice de corrélation*

Nous avons effectué une matrice de corrélation pour avoir la corrélation exacte entre les deux variables suivantes : la longueur du commentaire ainsi que le label attribué. Il y a une corrélation négative de 0,18 entre la longueur du commentaire et le label, ce qui nous conforte dans notre prédition précédente, à savoir qu'un commentaire long à tendance à correspondre à un avis négatif.

Comme nous l'avons vu précédemment, nous ne prendrons pas en compte cette corrélation puisque la moyenne des commentaires en fonction de label est relativement proche, et donc ne nous aiderait pas à améliorer notre modèle.

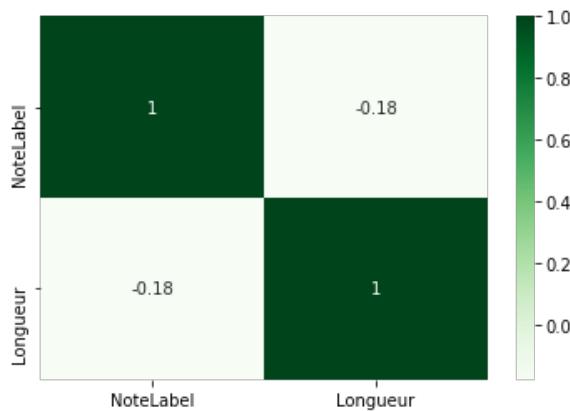


Figure 16: Matrice de corrélation entre la longueur et le label

c) Analyse de mots ou groupes de mots

Dans le cadre de notre analyse de sentiments par les consommateurs et plus précisément l'analyse de leurs commentaires, nous pouvons parler de « Text processing » : il s'agit d'un traitement de texte électronique. Nous avons cherché les mots qui revenaient le plus souvent en fonction du label attribué au commentaire et, pour permettre cette analyse, nous avons utilisé le NLP (Natural Language Processing), qui vise à traiter automatiquement le langage pour en extraire une signification. Dans un premier temps, nous avons séparé notre dataset en deux, afin d'avoir un premier dataset comprenant uniquement les commentaires positifs, et un second comprenant les commentaires négatifs.

Afin d'extraire les mots les plus utilisés, nous avons eu besoin de « tokeniser » nos commentaires, c'est-à-dire que nous avons transformé les différents commentaires en une suite de tokens individuels. On peut le comprendre de la façon suivante : un token représente un mot, ou un groupe de mots. Ainsi le commentaire « J'ai bien mangé » sera décomposé en quatre tokens : « j' », « ai », « bien », « mangé ». Parmi les mots employés par les clients, on retrouve des mots utilisés très fréquemment dans notre langage, qui compteraient comme les mots les plus fréquemment employés. Cependant, ces mots ne sont pas intéressants pour nous puisqu'ils n'apportent aucune information sur le sentiment du consommateur, comme par exemple « l' », « du », « plus » etc. Ces mots sont appelés « stop-words », et nous avons pris soin de les retirer de nos datasets. De même, nous avons retiré les mots « cuisine », « plat », « restaurant », « Paris », « table », « endroit » et « brasserie », puisqu'ils apparaissent dans beaucoup de commentaires, que ce soit positifs ou négatifs et n'ont donc aucune signification selon les labels.

Par la suite, nous avons cherché à afficher les mots et les groupes de mots les plus importants, en choisissant des tokens simples, en 2-uplets ou 3-uplets.

Nous avons tout d'abord commencé notre étude par les commentaires positifs.

Nous avons regardé quels étaient les 30 mots seuls qui apparaissaient le plus. Voici le graphique que nous obtenons (figure 16).

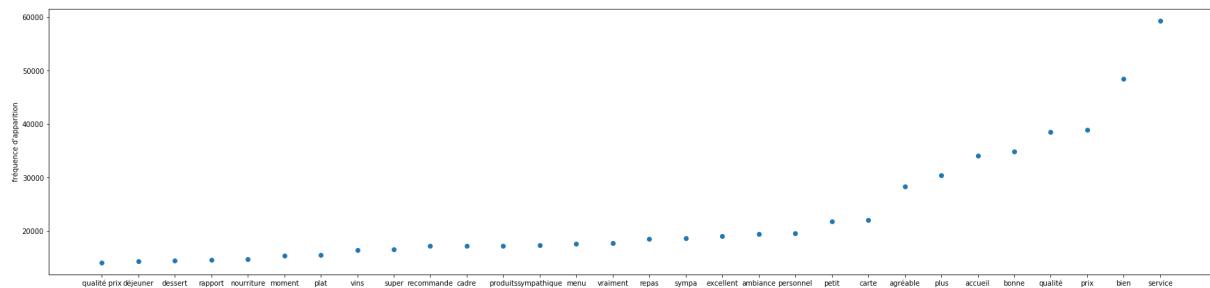


Figure 17: Graphique des mots unitaires les plus fréquents dans le dataset positif

Nous pouvons constater que le mot « service » est celui qui revient le plus souvent avec presque 60 000 apparitions, suivi du mot « bien » avec environ 50 000 apparitions. Apparaissent ensuite les mots « prix », « qualité », « bonne », « accueil », « plus », « agréable », « carte », « excellent », « recommande » et encore bien d'autres.

On remarque que ces mots ont souvent une connotation positive, mais n'oublions pas que ces mots sont tirés de phrases et celles-ci peuvent donner un tout autre sens au mot employé. Prenons par exemple le mots prix afin d'illustrer cette idée. « Prix » peut en réalité signifier « bon rapport qualité prix » ou encore « hors de prix ». Nous avons donc décidé d'étudier les mots positifs en les regroupant par trois ou quatre, et en affichant les 6 groupes de mots qui avaient le plus d'apparition (Figure 18).

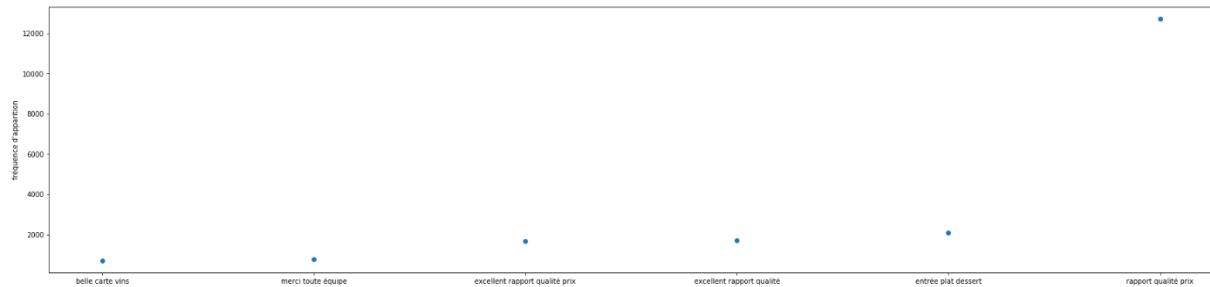


Figure 18: Graphique des mots groupés les plus fréquents dans le dataset positif

Ce graphique nous montre que les 6 groupes de mots qui reviennent le plus souvent dans notre dataset sont « rapport qualité prix » avec environ 12 500 apparitions, « rapport qualité prix », « entrée plat dessert », « excellent rapport qualité », « excellent rapport qualité prix », « merci toute l'équipe », « belle carte vins ». En analysant ces groupes de mots, on voit que les mots « rapport », « qualité » et « prix » reviennent souvent ensemble. Cependant s'ils sont également souvent présents dans les commentaires négatifs, cela ne nous apporterait pas grand-chose, il faudrait alors étudier uniquement le groupe « excellent rapport qualité prix » qui a uniquement aux alentours des 2 100 apparitions.

Nous avons donc poursuivi notre étude par l'affichage de graphiques similaires, mais avec le dataset négatif. Nous avons cependant fait le choix d'afficher des groupes de mots de deux, puisque les mots réguliers comme « prix », « qualité » et « service » revenaient trop souvent,

comme dans le dataset positif. Observons donc le résultat sur la figure ci-dessous (Figure 19).

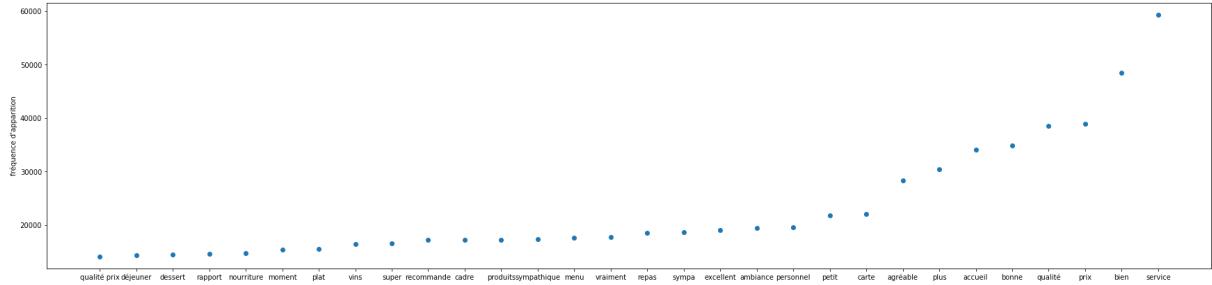


Figure 19: Graphique des deux mots groupés les plus fréquents dans le dataset négatif

On comprend que les deux groupes de mots qui ont eu le plus de résultats sont « rapport qualité » et « qualité prix » qui sont assez similaire à l'étude du dataset positif. Cependant, les quelques groupes de mots suivants sont surprenants, on retrouve : « passez chemin », « boire verre », « service rapide », « non plus », « entrée plat » et d'autres. Dans « passez chemin », on retrouve bien le côté négatif du dataset mais dans les autres, l'avis négatif est un peu plus difficile à déceler.

Nous avons donc, comme fait précédemment, poursuivi l'étude avec des groupes de trois ou quatre mots, comme dans la figure 20 ci-dessous.

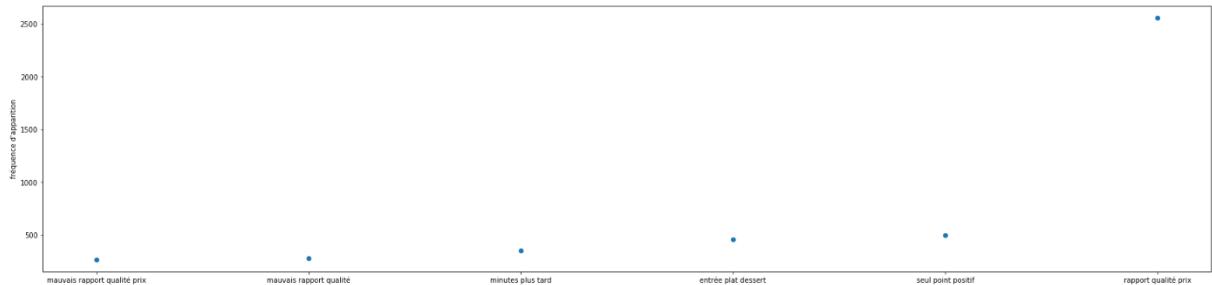


Figure 20: Groupes de mots les plus fréquents dans le dataset négatif

Ainsi, le groupe de mots qui revient le plus est une fois de plus « rapport qualité prix », suivi de « seul point positif », « entrée plat dessert », « minute plus tard », « mauvais rapport qualité ». On observe donc qu'il y a des groupes de mots qui reviennent, peu importe la nature du commentaire (positif ou négatif).

On peut donc en conclure qu'il y a bien des mots qui reviennent plus souvent en fonction du label.

On peut également se rendre compte que, parfois, les 3-uplets et les 2-uplets peuvent être plus révélateur du label. Cependant la négation n'est pas toujours prise en compte et peut

complètement renverser le sens de l'opinion laissé dans le commentaire vis-à-vis du label. Nous ne pouvons toutefois pas conclure sur le nombre optimal de n-uplets à choisir, et cela fera l'objet de recherches afin de voir quel nombre nous mènera au meilleur accuracy score.

Il y a également une piste que nous avions en tête et voulions explorer : y-a-t-il une relation entre le nombre de points d'exclamation et le label ?

count	67903.000000	count	93241.000000
mean	1.220830	mean	1.186377
std	2.656377	std	2.009744
min	0.000000	min	0.000000
25%	0.000000	25%	0.000000
50%	0.000000	50%	0.000000
75%	1.000000	75%	2.000000
max	69.000000	max	137.000000

Figure 21: Étude des points d'exclamation pour les datasets négatifs et positifs

D'après les résultats obtenus en figure 20, on peut remarquer que les moyennes des datasets positifs et négatifs sont assez proches. On remarque également que, ne regardant les écarts-type (std) que pour le dataset négatif, on est à 2,65 et pour le positif à 2,00. Une fois de plus, ces valeurs sont assez proches pour que l'on puisse les exploiter lors de notre prédiction finale. On constate cependant, en regardant la valeur à 75% des commentaires positifs, qu'ils ont 2 points d'exclamation « !! » contre un pour le neutre. Les valeurs étant trop proches, la piste de la ponctuation par les « ! » ne nous aidera pas lors de notre prédiction de label.

d) WordCloud

Afin d'étudier les mots les plus fréquemment utilisés en fonction du label, nous avons cherché dans un premier temps à afficher un WordCloud, qui permet de les mettre en évidence. Cet affichage aide à la visualisation des mots les plus importants, et pour y parvenir nous avons dû importer WordCloud de la librairie WordCloud. Voici les nuages de mots issus de nos commentaires négatifs et positifs.



Figure 22: Wordcloud des datasets négatifs et positif avant nettoyage

Beaucoup de mots présents dans ces deux WordCloud sont communs aux sentiments positifs et négatifs. Nous avons donc décidé d'enlever une certaine liste de mot afin que l'on puisse mieux apercevoir le sentiment et les mots importants, donnant du sens à notre analyse. Voici la liste des mots que nous avons enlevé :

```
[ 'serveur', 'plat', 'restaurant', 'cuisine', 'table', 'client', 'service', 'serveurs', 'serveur', "c'est", 'verre', 'serveuse', 'service', 'commande', 'tables', 'bar', 'bars', 'plats', 'verre', 'vin', 'clients', 'personnel', 'eur o', 'salade', 'entrée', 'dessert', 'repas', 'terrasse', "j'ai", 'manger', 'endroit', 'quartier', 'carte', 'café', 'gout', 'viande', 'pain', 'prendre', 'lieu', 'boisson', 'brasserie', 'commandé', 'patron', 'côté', 'demande ', 'Paris', "c'était", 'menu', 'servi', 'salle', 'menu', 'jour', 'assiette', 'dit', 'toilettes', 'toilette', 'bref', 'vraiment', 'boire', 'ami', 'amis', 'cuisine', 'paris', 'nourriture', 'après', "qu'il", 'surtout', 'dire', 'produit', 'c hoix', 'établissement', 'moment', 'payer', 'part', 'cadre', 'beaucoup', "qu'on", "n'est", 'plu', 'euros', 'pri', 'demandé', 'cocktail', 'frite', 'plutôt', 'sauce', 'maison', 'plus', "n'était", 'frites', 'place', 'n'a" ]
```



Figure 23: Wordcloud des datasets négatifs et positifs après nettoyage

4. Préparation des données

a) Pré traitement des données

Après avoir nettoyé et exploré nos données, il nous a fallu les préparer avant d'implémenter nos modèles.

Cette étape de prétraitement des données est primordiale pour une bonne analyse sentimentale car il est important d'avoir des commentaires prêts à l'emploi et dont on peut en extraire des informations, c'est-à-dire un texte clair sans ponctuation, sans smileys, le tout au même format.

Dans ce processus de prétraitement des données, nous avons été amenées à supprimer les colonnes qui ne nous étaient pas utiles dans l'analyse de sentiment, et plus précisément les colonnes nous renseignant sur le nom du restaurant, son id, la note (puisque celle-ci a été labellisée) et la date de visite. Après avoir concaténé le titre du commentaire et le commentaire lui-même dans la colonne 'Commentaires', nous avons pu également supprimer la colonne 'Titre'. Faire ceci était très intéressant car les titres des commentaires sont souvent révélateurs du sentiment, puisqu'ils résument généralement l'expérience qu'a eu le client. Nous nous retrouvons alors avec un dataset plus petit, comprenant une colonne commentaire (titre + commentaire) et le sentiment associé au commentaire (1 ou -1 en fonction de positif ou négatif).

Nous avons ensuite procédé à un second « nettoyage du texte » à travers la suppression des digits, des ponctuations (qui ont été remplacées par un espace), des emojis et enfin des suites d'espaces que nous avons remplacé par un unique espace.

Nous avons également converti tout le texte en minuscule, afin que le même mot écrit de façons différentes soit perçu comme un même mot et non pas deux mots différents (par exemple : « Mange », « mange » et « MANGE », qui auraient été perçus comme trois tokens différents et deviennent alors un unique token « mange ». Sans cette étape, notre vectorisation aurait été biaisée puisque le poids des mots n'aurait pas été le même et nous aurions donc eu une analyse approximative.

Comme nous l'avons vu précédemment dans la partie Nettoyage, nous avons également procédé à la suppression des stop-words ainsi qu'à la suppression des caractères uniques afin d'optimiser nos résultats (les caractères uniques ne forment pas de mots et sont donc inexploitables car ne veulent rien dire). Cependant, dans la partie concernant les commentaires en anglais, nous avons pris soin de définir une liste de mots à conserver, à savoir ["n't", "not", "no"], qui nous permet d'indiquer la négation de certains termes et ainsi en conserver le sens. Nous n'avons pas eu à le préciser pour le Dataset en français puisque ces mots n'étaient pas pris en compte par la fonction appelée.

Enfin, nous avons procédé au Pos-Tagging, qui signifie « part of speech tagging ». Cela revient à identifier le type des mots dans un texte (nom, verbe, adjectif et adverbe), et est utile pour améliorer la performance de la lemmatisation, que nous aborderons dans la partie suivante.

Pour parvenir à faire cela, nous avons utilisé les librairies nltk, spacy, et re (regular expression).

b) Stemming & lemmatization

Après avoir tokenisé les mots et supprimé les stop-words de nos commentaires, il était important de regrouper les mots dont la racine est la même, et qui auraient donc la même signification. C'est à travers le processus de stemming, qui signifie « racine », que nous avons pu regrouper ces mots dont la racine est identique et dont le sens profond est également le même. Le stemming permet donc de transformer par exemple des adjectifs en mots, ou de regrouper des verbes conjugués en tronquant leurs dérivations et leurs accords.

Prenons l'exemple de « accueil », on pourrait retrouver dans les différents commentaires les mots « accueillants », « accueillis », « accueillante », « accueillirent » etc.

La stemmatisation va permettre de transformer tous ces mots en « accueil ».

Ceci permettrait à notre modèle de ne pas avoir à analyser trop de mots différents mais qui signifient la même chose, et donc de faciliter la prédiction du commentaire, en réduisant le temps de prédiction.

Nous avons également effectué sur nos commentaires un processus de lemmatisation. La lemmatisation, en opposition à la stemmatisation (troncature à la racine), va permettre de ramener un terme à sa forme la plus simple (forme masculin singulier, par exemple pour la langue française.), dans le même but final que la stemmatisation.

Voici deux tableaux permettant d'illustrer ces propos, que nous avons trouvé sur « blog.bitext.com ».

Form	Suffix	Stem
studies	-es	studi
studying	-ing	study
niñas	-as	niñ
niñez	-ez	niñ

Principe de la stemmatisation

Principe de la Stemmatisation :
 -Suppression des suffixes
 -Troncature à la base de la racine du mot
Avantage :
 Rapidité
Inconvénient :
 Précision

Figure 24: Exemple appliqué de stemmatisation

Form	Morphological information	Lemma
studies	Third person, singular number, present tense of the verb study	study
studying	Gerund of the verb study	study
niñas	Feminine gender, plural number of the noun niño	niño
niñez	Singular number of the noun niñez	niñez

Principe de la lemmatisation

Principe de la lemmatisation :
 Mettre le mot sous sa forme la plus simple (suppression du pluriel, un seul genre).
Avantage :
 Précision et compréhension
Inconvénient :
 Temps de prédiction plus long

Figure 25: Exemple appliqué de lemmatisation

```

df_test.Commentaire[2]
'Agréable moment passé dans ce restaurant. Très bel accueil. Le burger était très bon! Je recommande vivement.'

df_test.StemmedComments[2]
'agréabl moment pass dan ce restaur . tres bel accueil . le burg était tres bon ! je recommand viv .'

df_test.LemmatizedComments[2]
' agréable moment passer dans ce restaurant . très bel accueil . le burger être très bon ! je recommande vivement .'

```

Figure 26: Exemple de stommatisation et lemmatisation d'un de nos commentaires

5. Entraînement du modèle

a) La vectorisation

Après avoir regroupé sous forme de fichiers csv nos commentaires en anglais et en français (all_review_FR.csv et all_reviews_EN.csv) qui ont été au préalable nettoyés, labélisés, stommés et lemmatisés, nous avons dû procéder à une étape de vectorisation avant d'entraîner notre modèle. La vectorisation permet de compter la fréquence d'un mot dans un texte et extraire des corrélations entre les termes. Son but est de transformer les tokens en nombres afin qu'ils soient exploitables dans nos modèles de Machine Learning.

Nous avons testé deux types de vectorisation :

Le TF-IDF (TermFrequency - InverseDocumentFrequency)

Le TF-IDF est une statistique numérique destinée à refléter l'importance des mots dans un document, où TF représente la fréquence du mot dans le document et IDF permet de réduire l'influence des mots les plus fréquents.

Cela revient à mesurer l'originalité d'un mot en comparant le nombre de fois où il apparaît dans un commentaire, avec le nombre fois où il apparaît dans notre dataset entier.

Ainsi, avec ce vecteur, les mots qui apparaîtront dans la plupart des commentaires seront perçus comme non pertinents, et les mots rares auront un plus gros poids.

Comme on le voit dans le schéma ci-dessous, pour chaque phrase on prend la fréquence du mot puis on lui applique une fonction log par rapport au nombre de fois où elle apparaît dans tout le document ce qui nous donne au final une importance à chaque mot. Les mots « this » et « is » ont un poids nul puisqu'ils apparaissent dans toutes les phrases du dataset anglais, contrairement au mot « good » et « bad ».

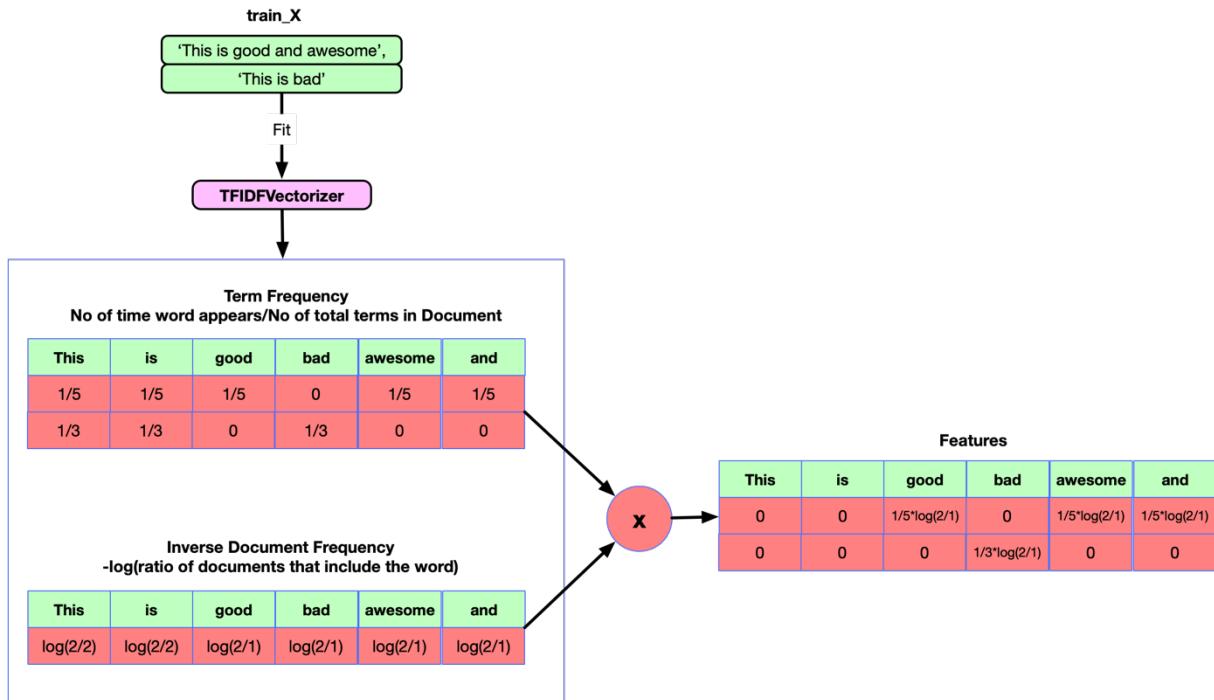


Figure 27: Exemple appliqué d'un TF-IDF

Le Count Vectorizer :

Le deuxième type de vectorisation que nous avons testé est le Count Vectorizer. Il permet de convertir notre collecte de commentaires en un vecteur de nombres, tout comme le TF-IDF. Il s'appuie lui, non pas sur la fréquence inverse du mot, mais sur la fréquence du mot.

Le poids d'un mot augmente proportionnellement au nombre d'occurrences de celui-ci dans le corpus.

Ainsi, un mot fréquemment employé par les clients aura un poids très important, contrairement à un mot utilisé 5 fois par exemple. C'est également la raison pour laquelle il a été important de s'affranchir des stop-words, dont le poids aurait été le plus important. Comme expliqué dans le schéma suivant, on voit que « this » apparaît souvent alors que ce n'est pas un mot clés, d'où la suppression de certains mots réalisée en amont. Les mots comme « bad », « good » ou encore « awesome » seront retenus comme mot clés.

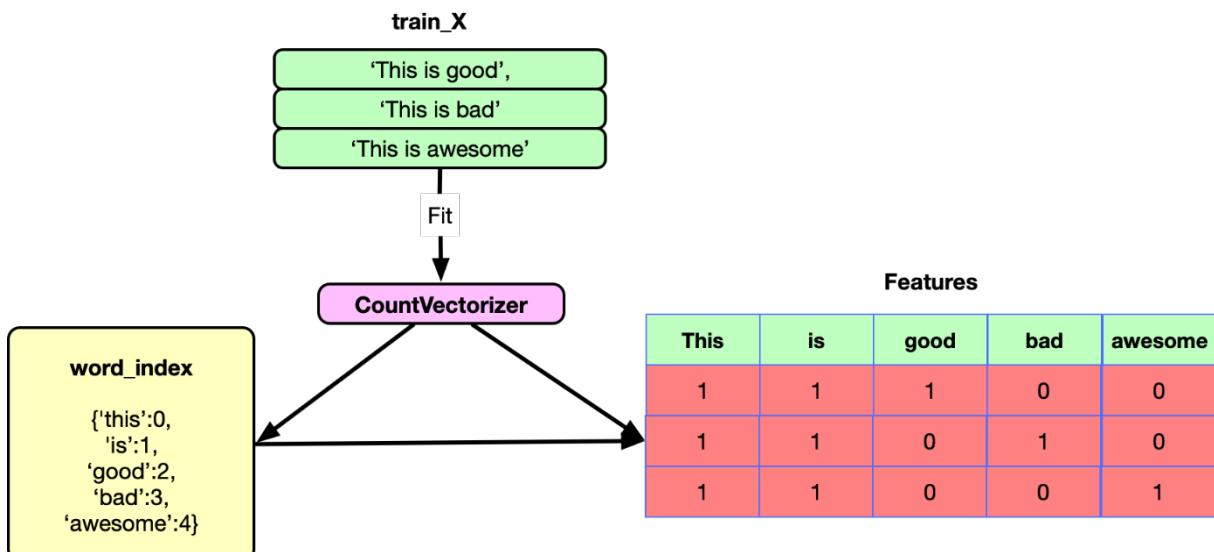


Figure 28: Exemple appliqu   d'un CountVectoriser

Cette vectorisation facilite l'apprentissage de notre mod  le de Machine Learning afin qu'il puisse mieux diff  rencier l'importance de chaque mot.

b) Mod  le de Machine Learning

Pour parvenir    notre analyse sentimentale des commentaires, nous avons utilis   des techniques de NLP (programmation neurolinguistique) afin d'en extraire les « features » (mots) les plus pertinents et informatifs, comme vu dans les parties pr  c  dentes. Nous cherchons    pr  dire au mieux un sentiment, en fonction du commentaire.

Trois mod  les de Machine Learning ont   t     tudi   :

Dans le but d'identifier le meilleur classificateur des commentaires, nous avons impl  ment   et entra  in   trois mod  les d'algorithme d'apprentissage supervis   que nous avons compar  . Les mod  les de Machine Learning que nous avons test  s sont le Logistic Regression (mod  le de r  gression) et le Random Forest Classifier (mod  le de classification) et le SVM, car ce sont les algorithmes les plus souvent utilis  s pour les analyses de textes et de sentiments.

Random Forest Classifier

Le Random Forest Classifier est une m  thode de bagging (entra  nement d'un seul algorithme de Machine Learning sur diff  rents sous-ensembles), c'est un algorithme efficace et qui donne g  n  ralement de bons r  sultats m  me pour des dataset d閞閏imb  l  s (m  me avec du 80%-20%).

L'arbre de d  cision est l'unit   de base du Random Forest. Il s'agit d'un mod  le simple qui tend    r  soudre un probl  me de Machine Learning en le mod  lisant comme une suite de d  cisions en fonction des d  cisions qui ont   t   pr  ises ult  rieurement. Cependant, avec cette m  thode

nous ne savons pas si l'ordre des décisions a rendu l'arbre optimal. Donc en modifiant l'ordre de ces décisions nous aurions peut-être eu un bien meilleur arbre de décision.

C'est pour cela que le Random Forest intervient en introduisant en plus la randomisation. Il permet de construire plusieurs arbres de décision de 2 à l'infini (en fonction du paramètre exigé dans la fonction), le modèle sélectionne ensuite toutes les prédictions de chaque arbre décisionnel, et la prédiction finale est issue d'un vote des prédictions de chaque arbre.

On fait donc la moyenne des prévisions de plusieurs modèles indépendants pour réduire la variance et donc l'erreur de prévision.

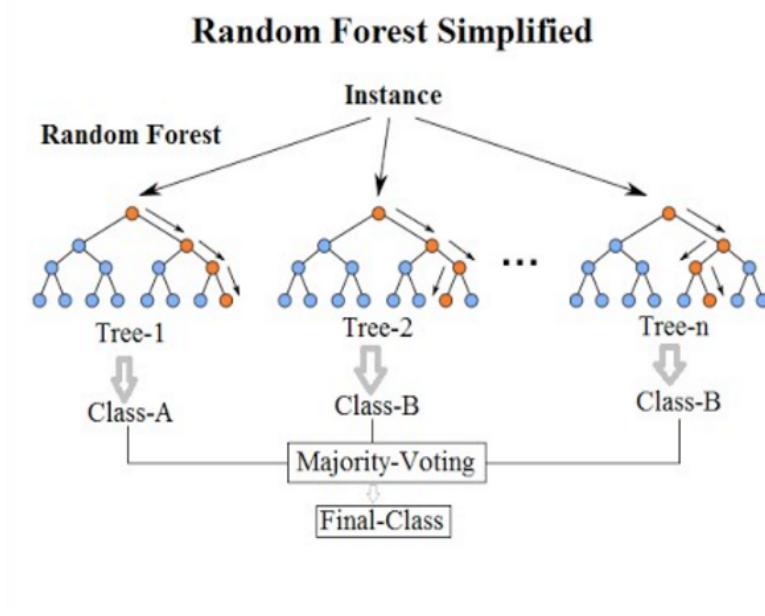


Figure 29: Schéma explicatif du Random Forest

Régression logistique :

La méthode de régression logistique peut être utilisée sur des variables catégorielles (comme pour notre projet, sentiment positif ou négatif) ou sur des variables continues (l'âge ou la taille). Elle étudie la probabilité qu'un évènement se réalise ou non, en associant à un vecteur de variable aléatoire ($x_1, x_2, x_3\dots$) une variable binomiale « y » ce qui nous donne une courbe bornée entre 0 et 1. Les résultats s'analysent de la façon suivante :

- I. Si la probabilité est supérieure ou égale à 0,5 alors la prédiction correspond aux données étiquetées à 1 (sentiment positif).
- II. Si la probabilité est inférieure à 0,5, alors la prédiction correspond aux données étiquetées à 0 (sentiment négatif).

On obtient donc au final deux résultats possibles : soit le commentaire est considéré comme négatif soit comme positif.

En suivant le schéma suivant, prenons par exemple que chacun de ces points est un commentaire qui a donc été classé suivant la courbe. Si le point sur le graphique est en dessous de 0,5 (axe rouge) il est considéré comme négatif et si il est au-dessus de l'axe, il est positif.

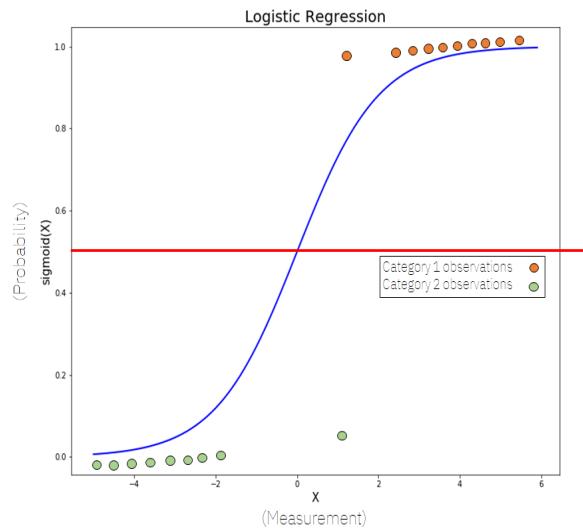


Figure 30: Schéma explicatif d'une régression linéaire

SVM :

Le troisième modèle de Machine Learning que nous avons utilisé est le SVM (Support Vector Machine).

Il s'agit d'un classificateur de régression linéaire qui convient à notre cas puisqu'il s'agit de construire une fonction permettant la prédiction d'un ensemble dans un problème à deux classes. On se pose la question suivante : l'ensemble appartient-il à la classe -1 ou +1 ?

Il s'agit plus particulièrement de rechercher une surface de séparation entre les deux classes. On va séparer les deux classes par un hyperplan, et on définit une marge comme la distance entre le plus proche exemple d'apprentissage et l'hyperplan de séparation.

Le SVM va chercher le séparateur qui maximise la marge, que l'on appelle « séparateur à vaste marge » et définit deux vecteurs de support, se trouvant à égale distance de l'hyperplan de séparation.

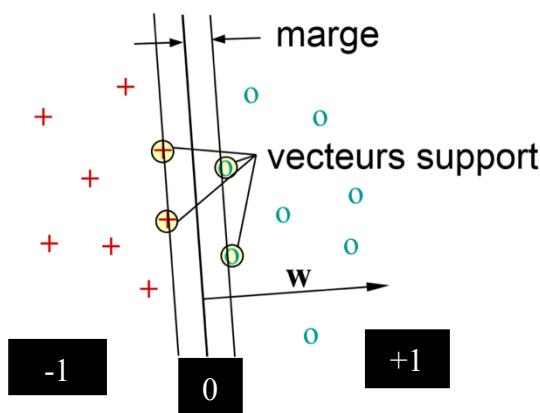


Figure 31: Schéma du séparateur à vaste marge (SVM)

Une fois que le séparateur est trouvé, la classification d'un nouvel exemple se fait par une simple décision :

La sortie de cette fonction est interprétable de la façon suivante :

- III. $f(x)=0$: l'élément se trouve sur la frontière de séparation, il n'y a pas de décision
- IV. $f(x) > 0$: l'élément est de classe +1
- V. $f(x) < 0$: l'élément est de classe -1

Utiliser cette méthode est pratique car elle permet d'éviter les erreurs de discrimination non linéaire et permet une bonne prédiction part sa fonction qui vise à séparer deux classes avec le moins d'erreurs possibles.

c) Les hyperparamètres

avons dû faire une recherche des meilleurs paramètres à utiliser pour chaque modèle : c'est à nous de définir ces hyper-paramètres avant l'entraînement du modèle afin qu'il soit le plus efficient possible.

Pour choisir les meilleurs hyper-paramètres, nous avons fait appel à deux méthodes : le **GridSearchCV** et le **RandomizedSearchCV**. Leur but commun est de nous retourner la meilleure combinaison d'hyper-paramètres à utiliser selon notre modèle. Le mode de fonctionnement de ces deux méthodes sera détaillé ci-après.

Le **GridSearchCV** est une méthode qui va prendre en compte une « grille » de paramètres de différentes valeurs et va entraîner séparément les différentes combinaisons possibles. Il va ensuite analyser les performances de chaque hyper-paramètre à travers un score et permettre un choix de la combinaison d'hyper-paramètres la plus performante. Comme chaque combinaison d'hyper-paramètres est essayée, cela peut être fastidieux et prendre beaucoup de temps. Le GridSearchCV est basé sur la validation croisée, qui permet de donner des évaluations plus précises en faisant la moyenne de plusieurs itérations avec des données au hasard. Cela permet également d'éviter les problèmes d'Overfitting (sur-apprentissage), où le modèle a du mal à généraliser.

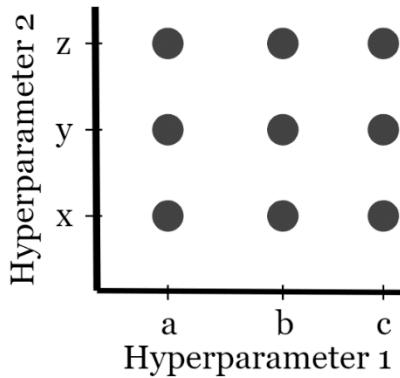
Le **RandomizedSearchCV**, lui, implémente une recherche aléatoire sur les paramètres, où chaque paramètre est échantillonné à partir d'une distribution sur les valeurs de paramètres possibles. Dans cette méthode, on peut contrôler explicitement le nombre de combinaisons de paramètres tentés et le nombre d'itérations. En bref, le RandomizedSearchCV échantillonne un nombre de candidats avec une liste d'échantillons spécifiés.

Après avoir étudié ces deux méthodes de paramétrisation, nous nous sommes finalement tournées vers le GreadSearchCV. Nous avons favorisé des paramètres optimaux plutôt qu'une exécution plus rapide mais au détriment de la précision de nos hyper-paramètres si nous utilisions le RandomizedSearchCV.

Grid Search

Pseudocode

```
Hyperparameter_One = [a, b, c]
Hyperparameter_Two = [x, y, z]
```



Random Search

Pseudocode

```
Hyperparameter_One = random.num(range)
Hyperparameter_Two = random.num(range)
```

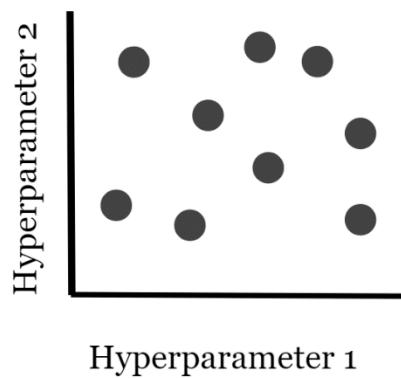


Figure 32 : Comparaison du GridSearch et du RandomSearch

Ces deux méthodes nous ont donc permis de tester différentes combinaisons d'hyperparamètres. Afin de savoir quel était la meilleure combinaison, nous avons travaillé sur des métriques d'évaluation comme l'accuracy et la matrice de confusion pour notre cas.

Nous avons principalement utilisé un GridSearchCV puisqu'il nous permettait de tester des possibilités plus précises. Nous avons choisi pour le Count Vectoriser et le TF-IDF les hyperparamètres suivants à tester :

-vectorizer_max_df : [0.5, 0.75, 1.0], il permet d'ignorer les termes qui ont une fréquence supérieure à la valeur entrée.

-vectorizer_ngram_range : [(1, 1), (1, 2), (1,3)], qui est représenté sous la forme (a,b). Avec a représentant la limite supérieure et b la limite inférieure de la plage de valeur pour différents n-grammes à extraire. Les n-grammes représentent les groupes de mot possibles. Cela signifie ici que l'on étudie la possibilité de prendre dans une phrase un mot seul, deux mots à côté, ou trois.

-vectorizer_max_features : [2000, 4000], il permet de créer un vocabulaire qui prend en compte uniquement les max_feautures de fréquences supérieures aux valeurs entrées.

Pour les classifiers, chaque modèle a ses propres hyper-paramètres. Nous avons choisi pour le Logistic Regression de tester le classifier_C avec les valeurs [0.01, 0.1, 1, 10, 100] qui représente l'inverse de la force de régularisation. Plus la valeur est petite, plus la régularisation

est forte. Nous avons aussi testé deux paramètres pour le penalty « l1 » et « l2 ». Ces paramètres sont utilisés pour spécifier la norme de pénalisation.

Pour le RandomForest, nous avons décidé de tester trois hyper-paramètres. Le n_estimators avec les valeurs [10, 50, 100, 200, 500], permet de choisir le nombre d'arbre dans notre forêt. Le paramètre max_features qui prend comme valeurs ['auto', 'log2', 'sqrt'], permet de tester le nombre de fonctionnalités à prendre en compte lors de la recherche de la meilleure répartition.

La valeur « auto » et « sqrt » signifient que max_features = $\text{sqrt}(n_features)$ et la valeur « log2 » signifie que max_features = $n_features$. Le classifier_criterion prend les valeurs ['gini', 'entropy'] et permet de mesurer la qualité d'une scission.

La valeur 'gini', elle, mesure l'impureté de gini et « entropie » pour le gain d'information.

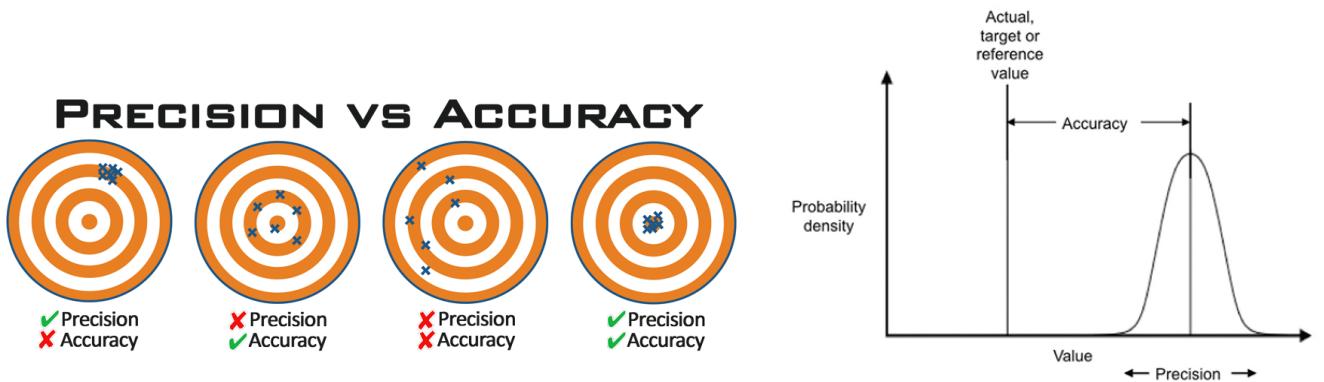


Figure 33: Accuracy vs Prédiction

L'**accuracy** représente la précision d'un algorithme de classification d'apprentissage. C'est un moyen de mesurer la fréquence à laquelle l'algorithme classe correctement un point de données. La précision est le nombre de points de données correctement prédites parmi tous les points de données.

La **matrice de confusion** va elle traduire de la performance du modèle à prédire, en nous informant sur le nombre de prédictions correctes pour chaque classe et le nombre de prédictions incorrectes pour chaque classe, en fonction de la classe prédite. Chaque ligne du tableau correspond à une classe prédite et chaque colonne correspond à une classe réelle.

On peut l'illustrer de la façon suivante :

		Classe réelle	
		-	+
Classe prédite	-	True Negatives (vrais négatifs)	False Negatives (faux négatifs)
	+	False Positives (faux positifs)	True Positives (vrais positifs)

Figure 34: Principe de la matrice de confusion

En se basant sur la matrice de confusion, on peut utiliser plusieurs métriques pertinentes pour l'évaluation des algorithmes entraînés. Nous allons prendre un code pour les formules des métriques suivantes :

Vrais Positifs = TP ; Vrais négatifs = TN ; Faux positifs = FP ; Faux négatif = FN ; P= Actual Positive

I. Accuracy

L'accuracy est le rapport des échantillons. Il correspond à la proportion de classes correctement prédictes. On la calcule de la façon suivante :

$$\text{Accuracy} = \frac{TP+TN}{\sum \text{Observations}} \text{ Avec } \sum \text{Observations} = TP+TN+FP+FN$$

Le calcul de l'Accuracy est précis car prend en compte les positifs et négatifs. Mais ne représente cependant pas les données quand celles-ci sont mal équilibrées.

II. Précision

$$\text{Précision} = \frac{TP}{TP + FP}$$

Une augmentation de la précision indique qu'il y a peu de faux positifs, et donc que le modèle est pertinent.

III. Recall

$$\text{Recall} = \frac{TP}{P} \text{ Avec } P = TP+FP$$

Le Recall ou rappel, peut également être appelé sensibilité. Il nous informe le taux de vrais positifs, c'est-à-dire la proportion de positifs que l'on a correctement identifié.

d) Pipeline

Après avoir prédéfini une structure des différentes étapes à réaliser pour mettre en œuvre une solution, nous avons fait le choix d'utiliser un outil presque indispensable dans l'univers de l'apprentissage automatique : Le pipeline. On parle plus grossièrement d'une collection d'étapes qui peuvent être exécutées en tant que flux de travail réutilisable.

Le pipeline de scikit-learn nous a permis de construire, itérer et produire rapidement notre modèle. Dans notre cas, le pipeline est constitué de deux grandes étapes : Un vectorizer (TF-IDF ou CountVectorizer), et un classifier (algorithme de Machine Learning de sklearn).

Tout de même, l'entraînement du pipeline selon les différents paramètres prend du temps, et pour cela, nous avons rajouté une étape de mise en cache des transformateurs du pipeline (CountVectorizer/TF-IDF) dans un dossier spécifié. Ainsi, le transformateur n'est pas calculé/entraîné à chaque fois si les paramètres et les données d'entrée sont identiques, et cela permet donc une optimisation du temps de calcul. Le GridSearchCV qui nous permettra d'obtenir la meilleure combinaison d'hyper-paramètre sera entraîné sur un pipeline.

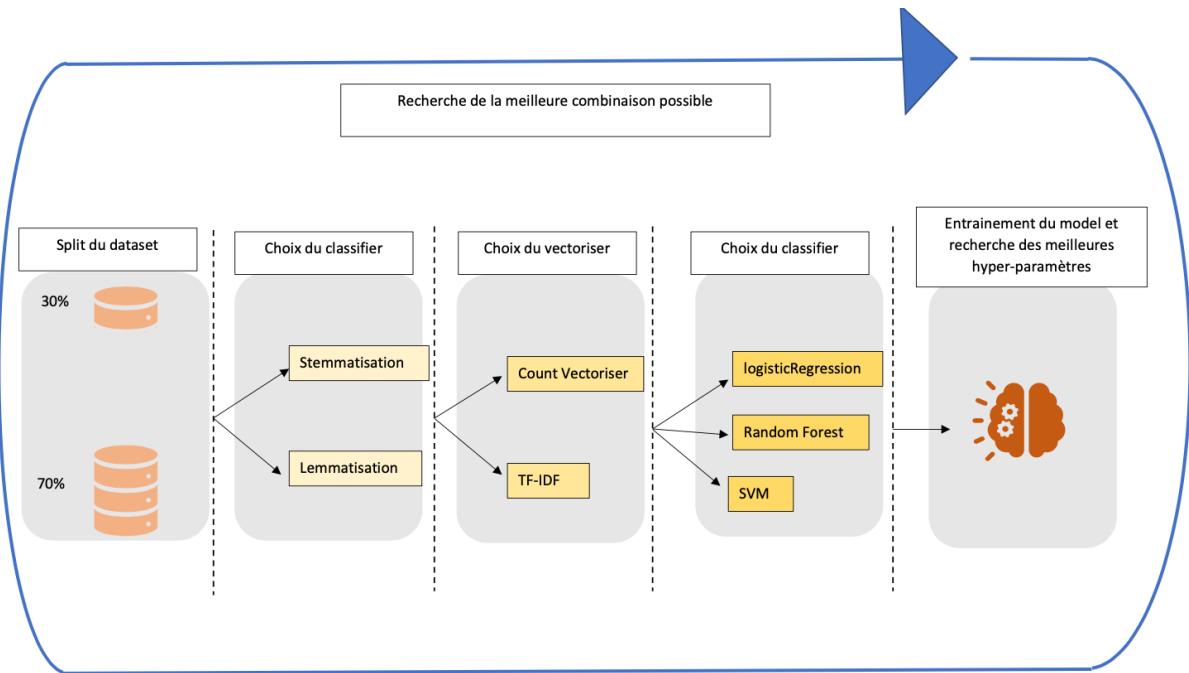


Figure 35: Recherche des meilleurs paramètres à travers d'un pipeline

e) Modèle final sélectionné

Afin de choisir quel modèle utiliser, nous avons dû effectuer plusieurs tests avant de les comparer. En effet, il ne s'agit pas seulement d'effectuer un choix entre l'utilisation du RandomForest, du LogisticRegression ou du SVM, mais de comparer plusieurs combinaisons possibles à savoir : quel choix de classificateur ? pour quelle méthode de traitement de texte ? pour quel type de vectorisation ? et pour quels hyper-paramètres ? Tant de questions dont le but final est d'obtenir la meilleure analyse de sentiments.

Nous allons tester ces différentes combinaisons à travers l'appel d'une fonction qui va prendre en paramètre la méthode (lemmatisation ou stemming), le type de vectorisation (TF-IDF ou CountVectorizer) et le classificateur (LogisticRegression, RandomForest), avec pour chacun les hyper-paramètres que nous avons prédéfinis en amont.

Pour cela, nous avons procédé à l'affichage des meilleures hyper-paramètres, du meilleur accuracy score et du meilleur score d'entraînement avec les meilleurs hyper-paramètres sectionnés ainsi qu'une matrice de confusion.

Type de word	Type de vectorisa	Type de classifier	Résultats paramètre	Matrice confusion	Accuracy Score
--------------	-------------------	--------------------	---------------------	-------------------	----------------

Stem	tfidf	Log	Classifier=1 Classifier__penalty :'l2'	[[14504 2456] [1535 36362]]	0.927247
Stem	cv	Log	Classifier=0.1 Classifier__penalty :'l2'	[[14355 2605] [1493 36404]]	0.9252966
...

Figure 36: Tableau des résultats finaux obtenus avec les différents modèles

Parmi toutes les combinaisons possibles, notre choix final se fera en fonction de la matrice de confusion et de l'accuracy score les plus performants.

6. Exposition du modèle

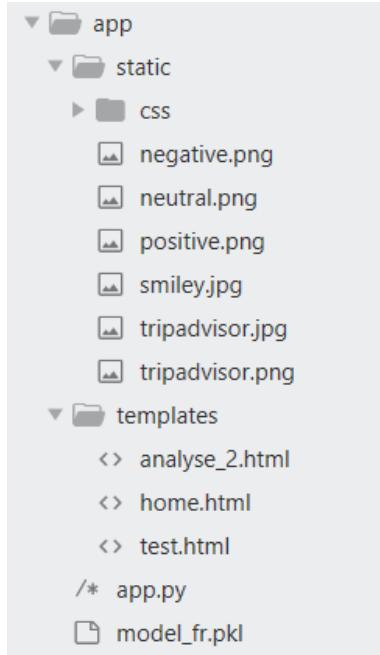
L'objectif de notre projet est finalement de pouvoir permettre de prédire si un commentaire d'un restaurant est positif ou négatif. Par ailleurs, on a pu voir jusque-là que l'on avait finalement trouvé un bon modèle de ML pour prédire. Dès lors, une simple méthode *predict* appliquée sur ce modèle suffit. Mais cela n'est pas très pratique, et le but est de déployer une solution accessible à chacun (codeur ou pas)

C'est donc dans cette idée que nous avons pensé au déploiement du modèle de Machine Learning choisi sur une interface web graphique, accessible sur le localhost avec le port 5000. Il s'agit d'entrer un commentaire dans une zone de saisie, de la valider puis d'avoir en retour la prédiction à savoir le sentiment correspondant.

a) Le Front-end grâce à Flask

Jusqu'à présent, nous codions en python pour travailler sur nos modèles de Machine Learning. Nous avons utilisé Flask qui est une librairie de python qui permet de faire simultanément du back-end et du front-End. Nous avons notre partie statique, front End en HTML, qui contient des variables python. Ces variables python seront utilisées et appelées dans notre back-end afin de pouvoir exécuter nos fonctions de prédiction.

b) Convention pour l'arborescence de Flask



Une appli en Flask respecte généralement une convention. Il s'agit de séparer le fichier principale (app) en deux fichiers : static et templates.

Par convention, les modèles (templates) et les fichiers statiques (static) sont stockés dans des sous-répertoires de l'arborescence des sources Python de l'application (voir figure à côté).

Figure 37: Arborescence de Flask

c) Pickling

Après avoir choisi notre modèle ainsi que les paramètres les plus optimaux, il faut entraîner ce modèle sur toute la base puis sauvegarder le modèle sous forme de fichier.pkl à l'aide du module pickle de Python. Pickle sert à sérialiser, soit convertir une hiérarchie d'objets Python en un flux d'octets. Un tel fichier peut donc passer par le "unpickling" pour pouvoir permettre de manipuler le modèle en python depuis un fichier de flux d'octets. Cette étape est indispensable dans la mesure où le but de ce projet et d'exposer un modèle ML. On aurait pu entraîner ce modèle dès que l'utilisateur rentrerait un commentaire à analyser, mais cela serait beaucoup trop long et inapproprié. Pour cela il est donc indispensable d'enregistrer le modèle afin de déployer facilement un modèle de machine Learning sur une interface (pour nous il s'agit d'un site web utilisant la librairie python FLASK).

d) Détection de la langue

Nous avons voulu rendre notre application utilisable avec des commentaires anglais et français (deux langues sur lesquelles nous avions entraîné notre modèle). Afin de reconnaître la langue du commentaire, nous avons utilisé au début la méthode « détecte » de la librairie

langdetect, mais nous avons changé pour Textblob qui a une meilleure performance de détection de la langue surtout pour les commentaires de petite taille.

e) Design

Afin de rendre agréable l'utilisation de notre application aux utilisateurs, le déploiement de notre module via interface web devait être un minimum esthétique. C'est ainsi que nous avons cherché à styliser nos pages. Pour ce faire, nous avons utilisé Bootstrap car c'est un framework open source de développement web orienté interface graphique. Nous avons donc appliqué à certaines balises un style existant, comme pour les boutons. Nous avons également utilisé du Css, pour rendre harmonieux notre partie front. La figure 38 montre un aperçu de notre application web.

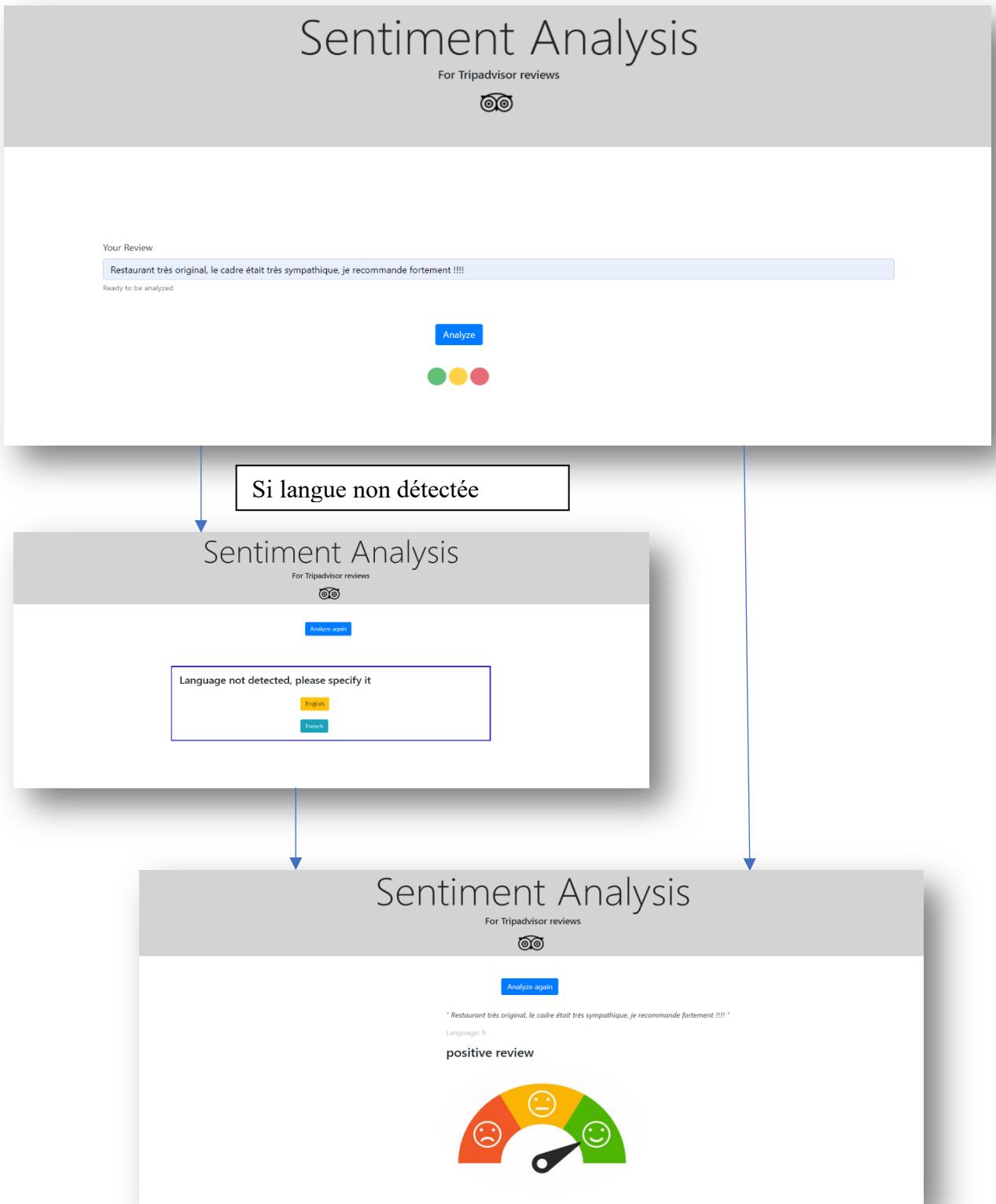


Figure 38: Schéma de l'utilisation de l'application web

IV. Conclusion et Perspectives

Pour rappel, l'objectif de notre projet était de prédire le sentiment d'un commentaire en le tapant directement sur une page web. Pour parvenir à faire cela, nous sommes passées par plusieurs étapes. Nous avons tout d'abord récupéré deux Dataframes de centaines de milliers de commentaires dont une en anglais et une en français. Nous sommes parvenues à faire cela grâce aux méthodes de scraping et de crawling. Nous avons ensuite dû nettoyer nos Dataframes. En effet, cette étape est primordiale puisqu'elle permet la bonne utilisation de la donnée offrant par la suite de meilleurs résultats. Nous nous sommes retrouvés dans le cas de figure suivant : nous avons retrouvé des commentaires de langues inexploitables telles que le russe ou encore le mandarin au sein même de dans notre dataset français. Par la suite, la phase d'exploration de données nous a permis de repérer les éléments clés nécessaires à l'entraînement du Machine Learning. Afin de choisir un modèle efficace de Machine Learning nous avons dû utiliser un pipeline pour structurer nos étapes d'entraînement. Pour trouver la meilleure combinaison (Classifier/ Vectorizer/ Hyper-paramètres) nous avons utilisé un GridSearch. Finalement, après avoir trouvé la meilleure combinaison, nous avons déployé ce modèle sur une interface web via la librairie Flask permettant un back-end avec python. Notre modèle est donc utilisable par l'utilisateur et la prédiction d'un commentaire est à portée de mains.

Malgré ce résultat satisfaisant, nous distinguons plusieurs axes de perspectives. Tout d'abord nous aimeraions utiliser de nouvelles approches de résolutions comme le Naïves bayes, les réseaux de neurones et le Deep Learning qui pourraient sûrement nous apporter des résultats plus précis. Nous pourrions également tester de nouvelles métriques comme la précision, le recall, que nous vous avons présenté mais encore le roc_accuracy, par exemple. Nous souhaiterions également élargir notre plage de valeurs pour les hyper-paramètres afin d'avoir un modèle le plus optimal possible. Nous voudrions aussi tester les méthodes de réduction de dimension afin de pouvoir réduire le temps de calcul de l'algorithme de Machine Learning.

L'utilisation de framework de Big-data permettant du calcul distribué et donc offrant la possibilité de tester des algorithmes plus complexes en moins de temps aurait été intéressante à étudier puisque nous avons eu à ce niveau des problèmes liés à la vitesse d'exécutions de nos algorithmes.

Ce projet a été pour nous une introduction dans le monde du Big-data, nous offrant à la fois une perspective concrète sur ces champs d'applications mais encore une ouverture vers de nouveaux horizons sur son utilisation.

