Projet - Text Mining

Nicolas Moine

Rémi Delage

Thomas Schuler

[I - Contexte](#_roo57fz638mc)

[II - Le scrapper](#_ynlivfum8b7b)

[III- Présentation de la base de donnée finale](#_4jiee6fgspya)

[IV - Présentation du modèle SVC](#_hoqs6fyfdkaj)

[V - Résultat](#_o9yt9ewoqpkr)

# I - Contexte

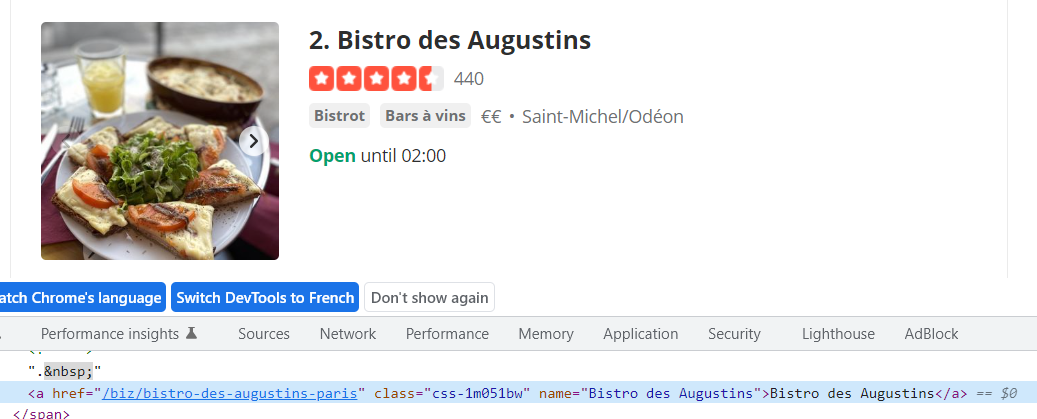
Dans le cadre de notre projet en text mining, nous avons décidé de traiter des commentaires sur les restaurants. En effet, nous souhaitons savoir s' il est possible de proposer une note aux utilisateurs en fonction de leur commentaire, en accord avec leur souhait ou leur retour. Cela pourrait permettre de normaliser un peu les notes attribuées aux restaurants, en suggérant une note à un client postant un commentaire en fonction des notes attribuées avec un commentaire similaire.

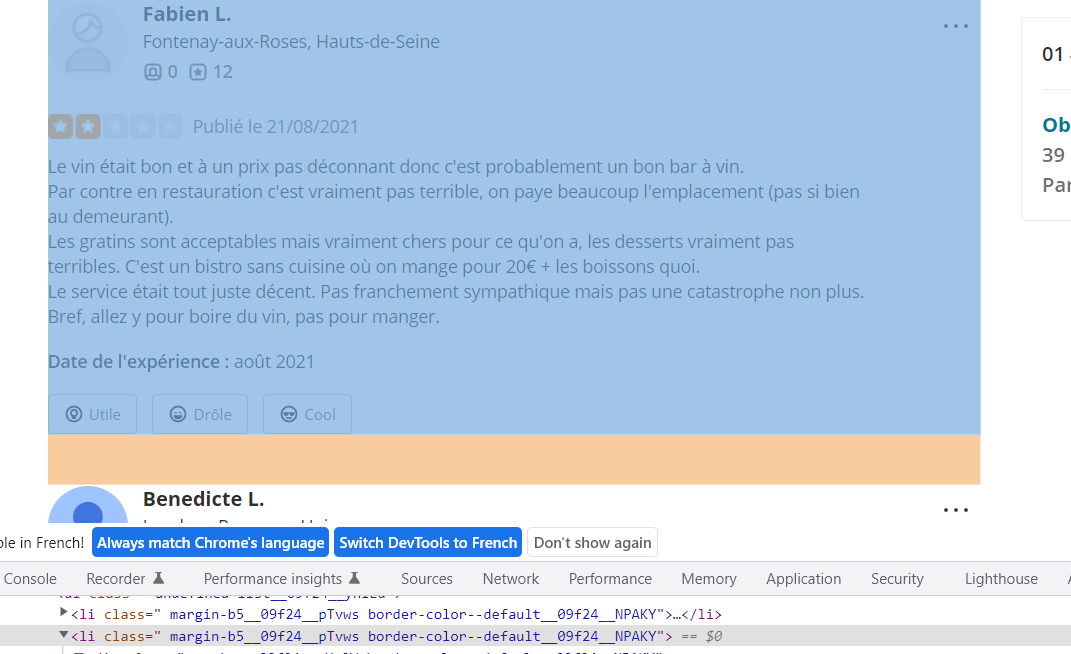
Pour ce faire, nous sommes allés récupérer de l’information directement en ligne via une plateforme. Nous vous expliquerons dans un premier temps comment est conçu notre scrapper et comment la base de données est construite. Puis ensuite les défis rencontrés et les méthodes pour pouvoir éviter d’être bloqué par les sites.

Ensuite nous avons réalisé un notebook pour travailler sur notre base et voir ce qui était réalisable pour analyser les commentaires et en tirer une note similaire à celle vraiment donnée par l’utilisateur. Nous vous expliquerons cela dans un deuxième temps.

# II - Le scrapper

Pour récupérer nos données, nous avons décidé de faire un scrapper en python. Le 1er choix à faire était de choisir le site sur lequel nous allions récupérer les commentaires. Nous comptions nous appuyez sur un site parmi les sites de références tel que tripadvisor ou thefork. Cependant ces sites mettent en place des api payants, donc bloquent toute tentative de connexion via un scrapper. Nous avons donc trouvé le site yelp, site moins connu mais recueillant quand même un bon nombre de commentaires. Le scrapper fonctionne donc ainsi :

* On choisit une localisation via le filtre de l’application nous avons choisi Paris et Lyon, deux grandes villes françaises avec une forte concentration de restaurants très différents. Le site propose différents types de classement, nous choisissons et récupérons les restaurants avec le plus de commentaires. Nous les récupérons grâce à la balise <a> qui contient les liens vers les restaurants. Nous récuperons le lien grâce à l’option href et le nom grâce à l’option name. 
* Ensuite grâce au lien des restaurants récupérés, nous allons sur la page du restaurant. Nous récupérons le prix du restaurant grâce à la balise "span", contenant la class="css-1ir4e44". Nous trouvons ensuite les blocs commentaires graçe à la balise li :



* Nous récupérons ensuite le nombre d'étoiles grâce à la div contenant la référence de l’image des étoiles, contenant l’aria-label = x étoiles.

# 

Nous récupérons ensuite le texte contenu dans le bloc text de la balise p.

Les pauses aléatoires faites dans le script sont dû à un algorithme empechant le minage automatique sur le site. L’algorithme repère les les hautes fréquences de navigation ainsi que les temps de navigation récurrents (10s par exemple).

# 

# II- Présentation de la base de donnée finale

Nous nous retrouvons avec une base de test de 3166 avis de 254 restaurants différents. Les étoiles allant de 1 à 5, on remarque une disparité entrele nombre d’étoiles faibles (1 et 2) par rapport aux étoiles hautes (4 et 5).

# IV - Présentation du modèle SVC

Le modèle de SVC, machine à vecteur de support, a vocation à classifier la donnée en différentes catégories. Ici pour rappel on a 5 catégories, on cherche donc à discriminer le vocabulaire entre les 5 différentes catégories.

L’idée de ce modèle est de maximiser la distance entre les échantillons via l’information et pour y arriver on positionne l’information sous forme de vecteur de support.

Ensuite on calcule la marge, qui n’est autre que la distance entre les vecteurs de support, et on essaye de la maximiser en réitérant autant que nécessaire.

Dans notre modèle, on a modifié les paramètres de telle sorte que les poids entre les différentes catégories soient pondérés. En effet, comme nous avons plus d’étoiles supérieures à 3 que des 1 ou 2 étoiles, il paraît important de balancer le poids pour équilibrer notre base d’apprentissage.

Afin d’obtenir un résultat de prédiction de notre modèle, nous allons d’abord séparer notre base de données en une partie d’apprentissage contenant 80% de nos observations et le reste servant à tester notre modèle.

Nous commençons par ajuster notre modèle sur notre base “train”.

Une fois le modèle ajusté, nous appliquons sur les observations de la base test que le modèle n’a jamais vu.

# V - Résultats

Avec les données que nous avons extraites du site cible, notre score oscille autour des 40% malgré des tentatives de fine tuning. Même si notre score paraît relativement bas, il est essentiel de prendre en compte deux éléments négligés ici :

* D’abord nous travaillons sur une segmentation en 5 éléments. Lorsqu’on vote pour la note d’un restaurant, il arrive très fréquemment que quelqu’un soit surpris que son voisin ne mette que 4 alors que son commentaire est dithyrambique. En ce sens si de façon aléatoire on a une chance sur 5 de trouver la bonne note, via notre modèle force est de noter que de façon significative, nous faisons mieux que l’aléa. En effet, un tirage aléatoire ne nous donnerait un score que de 20%.

Nous faisons donc ici 2 fois mieux qu’un tirage aléatoire.

* Dans un second temps, les données de texte lorsqu’on leur retire leur mot de liaison, perdent du sens. En effet, il ne faut pas négliger les figures de style qui sont valorisées à travers les connecteurs logiques qu’on retire ici.

Pour conclure, si notre modèle peut paraître faible, un modèle de scoring de la perte du modèle permettrait de mettre en valeur que sa capacité à distinguer un bon résultat d’un mauvais est plutôt forte.