

# Rapport de projet de synthèse

## SMART RESTAURANT RECOMMENDATION

Rédigé par l'équipe



ASHRAF Ali  
BAYO Mohamed II  
FAGOT Benoît  
SALEEM-AHAMED Mina  
VU Hong Phuc

*11 juin 2020*

Rapporteur :	KOTZINOS Dimitrios
Tutrice technique :	TZOMPANAKI Aikaterini
Encadrant de gestion de projet :	LIU Tianxiao
Formation :	Master 2 Informatique et Ingénierie des Systèmes Complexes Spécialité Systèmes Intelligents Distribués

## Remerciements

Nous souhaitons débuter ce rapport par les remerciements que nous devons à Madame Tzompanaki Aikaterini, co-responsable Master IISC de CY Cergy-Paris Université, d'une part, pour avoir accepté de tutorer techniquement notre projet après nous avoir proposé le sujet de celui-ci. D'autre part, nous la remercions pour sa disponibilité pour les rendez-vous ou bien par e-mail, pour sa présence lors de la pré-soutenance technique, pour ses précieux conseils sur le projet ainsi que pour ses remarques sur ce rapport. Et aussi les cours magistraux et séances de travaux pratiques qu'elle nous a dispensé pour le module de « Machine Learning » cette année.

Ensuite, il nous paraît opportun d'adresser nos remerciements à notre encadrant de gestion de projet, Monsieur LIU Tianxiao, responsable du thème "Gestion de projet" au sein de l'université, qui a validé notre sujet. Dans un premier temps, nous le remercions pour ses cours de qualité sur la gestion de projet disponibles sur sa page web, mais également pour tous les cours de gestion de projet qu'il nous a dispensé durant la licence ainsi que pendant nos années de master. Dans un second temps, nous tenons à le remercier pour sa présence, les différents points et rendez-vous qu'il a effectué avec l'équipe à propos de la gestion de projet, du projet en lui-même mais aussi de ce rapport.

Par la suite, nous remercions Monsieur LORANDEL Jordane, co-responsable Master IISC et co-responsable parcours IE du master IISC, pour sa présence lors de notre soutenance technique et qui nous a permis de revenir sur un point important du projet.

Puis nos remerciements vont aussi à Monsieur JEN Tao-Yuan, co-responsable du parcours SID (Master IISC), ainsi qu'à Madame MARINICA Claudia, maître de conférences, pour leurs cours magistraux et travaux pratiques dans le module de « Technique de datamining » de cette année. Nous remercions, particulièrement, Monsieur JEN pour sa disponibilité à tout moment et sa flexibilité.

De même, nous remercions notre chargé de cours magistraux et de travaux pratiques du module de « Systèmes et Applications Distribués », respectivement, pour nous avoir permis d'approfondir plusieurs bases importantes à ce projet : Messieurs KOTZINOS Dimitrios et SWAILEH Wassim. Par ailleurs, merci à Monsieur KOTZINOS, pour nous avoir également dispensé le module de « Big Data Analytics » qui nous a permis de d'approfondir notre connaissance sur les données de ce projet mais également pour prendre le temps de lire notre rapport que nous avons pris soin d'écrire, en lui souhaitant une bonne lecture.

Et finalement, nous tenons à remercier toute l'équipe pédagogique et technique de notre université pour le confort de travail et la qualité des cours qu'ils nous ont offert durant toute nos années universitaires mais également en cette période de crise sanitaire.

Nous souhaitons terminer cette section par les remerciements que nous devons à nos familles et amis qui nous ont toujours apporté leur amour et soutien.

Cergy-Pontoise, juin 2020  
L'équipe DATA DIVE.

# Table des matières

Remerciements .....	2
Chapitre 1 Introduction .....	9
1.1. Contexte .....	9
1.2. Objectifs du projet .....	10
1.3. Idée de solution.....	11
1.4. Mise en scénario .....	12
1.5. Organisation du rapport.....	13
Chapitre 2 Présentation et spécification du projet .....	15
2.1. Étude du marché .....	15
2.2. Fonctionnalités attendues.....	16
2.3. Conception globale du projet.....	17
2.4. Problématiques identifiées et solutions envisagées.....	19
2.5. Environnement de travail.....	19
Chapitre 3 Big Data et Analyse de données.....	21
3.1. Analyse de la problématique .....	21
3.2. État de l'art : études des solutions existantes .....	23
3.3. Solution proposée et sa mise en œuvre .....	23
3.3.1. Analyse et pré-traitement de données .....	23
3.3.2. Stockage de données.....	27
3.4. Tests et certifications de la solution .....	27
Chapitre 4 Extraction de pattern .....	29
4.1. Analyse de la problématique .....	29
4.2. État de l'art : études des solutions existantes .....	30
4.2.1. La fouille de données.....	30
4.2.2. Manipulation de dataframes.....	32
4.3. Solution proposée et sa mise en œuvre .....	34
4.4. Tests et certifications de la solution .....	39
Chapitre 5 Géolocalisation.....	41
5.1. Analyse de la problématique .....	41
5.2. État de l'art : études des solutions existantes .....	41
5.2.1. Géolocalisation - Géocodage .....	41
5.2.2. Calcul de distance :.....	43

5.3. Solution proposée et sa mise en œuvre .....	45
5.3.1. Récupération des coordonnées de géolocalisation .....	46
5.3.2. Calcul de la distance - Haversine .....	47
5.3.3. Obtention des restaurants les plus proches.....	47
5.4. Tests et certifications de la solution .....	48
Chapitre 6 Système de recommandations .....	50
6.1. Analyse de la problématique .....	50
6.2. État de l'art : études des solutions existantes .....	50
6.3. Solution proposée et sa mise en œuvre .....	56
6.3.1. Recommandation – Content-Based Filtering .....	57
6.3.2. Recommandation – Collaborative Filtering.....	62
6.3.2.1. Collaborative Filtering – Matrix Factorization (Model Based).....	62
6.3.2.2. Collaborative Filtering – Friendlist Based (Memory).....	67
6.3.2.3. Collaborative Filtering – User Based (Memory) .....	70
6.3.3. Recommandation – Popularity/Trends Based Filtering .....	70
6.3.4. Recommandation – Geolocalisation Decision.....	72
6.3.5. Hybrid System (solution globale) .....	72
6.4. Tests et certifications de la solution .....	73
Chapitre 7 Rendu final .....	86
7.1. Interface utilisateur finale.....	86
7.2. API du système de recommandations .....	89
7.3. Tests utilisateur et certification .....	93
7.4. Autres tests et certifications .....	93
Chapitre 8 Gestion de projet .....	95
8.1. L'organisation de l'équipe .....	95
8.1.1. L'équipe de projet .....	95
8.1.2. Outils de gestion de projet .....	95
8.1.3. Répartition des tâches .....	102
8.1.4. Releases .....	102
8.2. Méthode de gestion .....	104
8.2.1. Adaptation de l'environnement de travail technique pour une meilleure productivité .....	104
8.2.2. Organisation de l'auto-formation des technologies nécessaires pour le projet .....	105

Chapitre 9 Conclusion et perspectives .....	107
9.1. Conclusion .....	107
9.2. Perspectives .....	108
Bibliographie .....	110
Annexe .....	112
 Annexe 1 Yelp Dataset Challenge .....	112
Annexe 2 Analyse complète de données.....	114
Annexe 3 Fichier de configuration Logstash.....	121
Annexe 4 Dashboard Kibana.....	122
Annexe 5 Algorithme Apriori .....	123
Annexe 6 Liste des attributs avec les valeurs possibles .....	123
Annexe 7 Implémentation de l'algorithme de géolocalisation .....	124
Annexe 8 Implémentation de l'algorithme de calcul de la distance haversine .....	124
Annexe 9 Implémentation de la prédiction de ratings de businesses .....	125
Annexe 10 Implémentation de la mesure de proximité des matrices .....	125
Annexe 11 Tableau Trello du projet .....	126

## Liste des figures

Figure 1 Data Dive : un outil de recommandation personnalisé .....	12
Figure 2 Architecture globale du projet.....	17
Figure 3 Architecture logicielle du projet .....	18
Figure 4 TreeMap représentant la répartition des données dans les fichiers .....	22
Figure 5 Nuage de mots basé sur les catégories des business .....	24
Figure 6 Nuage de mots basé sur les noms des business.....	25
Figure 7 Courbe d'apprentissage pour l'algorithme kNN.....	26
Figure 8 Heatmap représentant la corrélation des features .....	28
Figure 9 Les 10 principales tendances de consommations mondiales (2020) .....	29
Figure 10 Rapport sur les données des établissements (business) .....	32
Figure 11 Nombre de restaurants par État .....	35
Figure 12 Nombre d'établissements par catégorie .....	36
Figure 13 Toutes les recherches de tendances temporelles possibles .....	37
Figure 14 Exemple d'affichage des heures d'affluences.....	37
Figure 15 Top populaires catégories (Arizona) .....	39
Figure 16 Correspondances tendances et préférences d'utilisateur (Arizona).....	40
Figure 17 Pourcentage de correspondances .....	40
Figure 18 Résultat de test des heures d'affluences.....	40
Figure 19 Logo ipstack .....	42
Figure 20 Exemple de resultat api d'ipstack .....	42
Figure 21 Illustration de la distance euclidienne .....	43
Figure 22 Formule distance euclidien.....	43

Figure 23 Illustration de la distance d'euclidienne .....	44
Figure 24 Formule distance Manhattan .....	44
Figure 25 Formule haversine .....	45
Figure 26 Code – résultat pour la localisation automatique .....	46
Figure 27 Exemple d'utilisation géocodage .....	47
Figure 28 Exemple de géocodage .....	48
Figure 29 Cartographie représentative des business utilisant la géolocalisation .....	49
Figure 30 Exemple de recommandations personnalisées dans l'usage quotidien.....	51
Figure 31 Approches content-based actuelles .....	52
Figure 32 Illustration de content-based filtering.....	53
Figure 33 User-Item recommandation .....	54
Figure 34 Item-Item recommandation .....	54
Figure 35 Exemple de matrix factorization .....	55
Figure 36 Exemple de recommandation hybride .....	56
Figure 37 Démarche de vectorisation de reviews par Word2Vec .....	58
Figure 38 Architecture Word2Vec .....	59
Figure 39 Les commentaires avant la transformation.....	59
Figure 40 Les commentaires après la transformation .....	59
Figure 41 Formule de calcul de la similarité cosinus .....	60
Figure 42 Exemple de la similarité cosinus entre les mots.....	60
Figure 43 Exemple de recherche mot similaire .....	61
Figure 44 Exemple de recherche mot clé par Content Based recommandation .....	61
Figure 45 Représentation schématique pour Singular Value Decomposition .....	63
Figure 46 Matrice d'users-items de données de ratings .....	64
Figure 47 Matrice de prédition par la technique SVD .....	64
Figure 48 Fonction de coût .....	65
Figure 49 Top 20 de recommandation par la NMF.....	66
Figure 50 Top 20 de recommandation par la SVD .....	67
Figure 51 Recommandations basées sur les amis YELP .....	68
Figure 52 Matrice User x Item en mémoire.....	68
Figure 53 Utilisateurs similaires par utilisateur selon le score de "cosine similarity".....	69
Figure 54 Exemple de résultats de recommandations .....	69
Figure 55 Calcul du score de popularité en fonction de la géolocalisation.....	72
Figure 56 Commande de la recommandation Popularity Based.....	74
Figure 57 Résultat de la recommandation Popularity Based .....	74
Figure 58 Résultat de la recommandation Popularity Based sur le plan .....	75
Figure 59 Commande et résultat de la recommandation Content Based Keyword .....	76
Figure 60 Résultat de la recommandation Content Based Keyword sur le plan .....	77
Figure 61 Commande de la recommandation Content Based Userid.....	77
Figure 62 Informations des businesses déjà évalués par l'utilisateur .....	77
Figure 63 Résultat de la recommandation Content Based Userid .....	78
Figure 64 Informations d'input_business_id évalué par l'utilisateur .....	78
Figure 65 Tuning de nombre de facteurs SVD .....	79
Figure 66 RMSE de SVD sur la cross validation 5 folds .....	80

Figure 67 Recommandations pour l'utilisateur testé .....	81
Figure 68 Utilisateurs les plus similaires à l'utilisateur testé.....	81
Figure 69 Comparaison utilisateur – utilisateur top1 .....	82
Figure 70 Comparaison utilisateur – utilisateur top2 .....	82
Figure 71 Comparaison utilisateur – utilisateur top5 .....	82
Figure 72 Catégories préférées de l'utilisateur .....	84
Figure 73 Attributs préférés de l'utilisateur .....	85
Figure 74 Page d'accueil de l'application.....	86
Figure 75 Page de profil de l'application .....	87
Figure 76 Page de détail.....	88
Figure 77 Architecture d'API de système de recommandation .....	89
Figure 78 Réponse de la requête de géolocalisation.....	90
Figure 79 Réponse de la requête de la recommandation Popularity Based .....	91
Figure 80 Réponse de la requête Content Based Keyword .....	91
Figure 81 Réponse de la requête Content Based Userid (1) .....	92
Figure 82 Réponse de la requête Content Based Userid (2) .....	92
Figure 83 Réponse de la requête Collaborative Filtering (1).....	92
Figure 84 Réponse de la requête Collaborative Filtering (2).....	92
Figure 85 Résultat du test de validité des pages de l'application web.....	94
Figure 86 Résultat du test de validité des feuilles de style de l'application web .....	94
Figure 87 Logo Jupyter .....	95
Figure 88 Logo GitLab .....	95
Figure 89 Applications installées sur le serveur hébergé sur Alibaba Cloud .....	96
Figure 90 Applications installées sur le serveur hébergé sur Google Cloud .....	97
Figure 91 Page d'accueil sur la gestion de multiples hôtes de Terminus .....	98
Figure 92 Accès au serveur via SSH sur Terminus .....	99
Figure 93 Racine de l'interface web Jupyter Notebook .....	100
Figure 94 Dossier principal sur OneDrive .....	100
Figure 95 Logo Microsoft OneDrive.....	101
Figure 96 Logo Microsoft Word .....	101
Figure 97 Liste des thèmes du projet .....	101
Figure 98 Logo Skype .....	101
Figure 99 Planification des releases.....	103
Figure 100 Classement des langages de programmation pour la Data Science.....	105

## Liste des équations

Équation 1 Support d'un item.....	31
Équation 2 Support d'une règle d'association .....	31
Équation 3 Confiance d'une règle d'association.....	31
Équation 4 Calcul du score de chaque business par rapport à l'utilisateur .....	69

## Liste des tableaux

Tableau 1 Outils utilisés pour les différents aspects du projet .....	20
Tableau 2 Comparatif des solutions pour le stockage des données .....	23
Tableau 3 Méthodes descriptives de Data Mining .....	30
Tableau 4 Données pour les tendances habituelles .....	34
Tableau 5 Données pour les tendances temporelles .....	34
Tableau 6 Résultat de test en utilisant l'API Google Maps et Nominatim .....	43
Tableau 7 Résultat de la recherche recommandation business avec la géolocalisation .	48
Tableau 8 Répartition des tâches .....	102

# Chapitre 1 Introduction

## 1.1. Contexte

Actuellement, internet est devenu une source de données gigantesque, chaque seconde, 29 000 giga-octets (Go) d'informations sont publiés dans le monde, soit 2,5 exaoctets par jour soit 912,5 exaoctets par an [1 données publiées]. Cette abondance de données est une source d'informations très précieuse, le “big data” est en croissance fulgurante dans le domaine de l'informatique. De ce fait, cela pousse de plus en plus les entreprises à entreprendre dans ce domaine et se préparer aux défis que celui-ci pose, notamment en termes de nouvelle façon de sauvegarder les données, de les analyser, d'y accéder, etc [2 data]. Les sociétés génèrent de plus en plus de données chaque jour à travers des tableurs, des CRM, bases de données, etc. Face à ce déluge de données, elles sont débordées et peuvent être amenées à ne pas les exploiter pleinement. De ce fait, elles ont besoin de bien gérer leurs données pour mieux gérer leurs activités.

Il y a également la technique d'aide à la décision « business » : le Data Mining, ou fouille de données. Celui-ci sert à mieux comprendre sa clientèle, comprendre son comportement à partir de ses caractéristiques, constituer des panels représentatifs de clients, découvrir des niches inconnues mais rentables, adapter sa politique de fidélisation ; ou bien optimiser l'adéquation de son offre à la demande adapter sa politique commerciale et sa tarification aux différents segments de clientèle, adapter ses canaux de distribution et/ou ses forces de ventes à ces segments, optimiser l'impact et la rentabilité des offres promotionnelles ; ou même donner un ordre de priorité à ses actions de marketing et/ou de vente mieux cibler ses campagnes de marketing direct, évaluer la propension d'un prospect ou client à acheter un produit nouveau.

Difficile de parler de Big Data sans mentionner le Data Mining, ou fouille de données : cette science consiste à faire ressortir des corrélations entre des phénomènes en apparence distincts afin de prédire des tendances. Autrement dit, le Data Mining consiste à détecter dans une masse de données des patterns que l'humain n'est pas capable d'observer. Cela ouvre de nouvelles possibilités, notamment pour les entreprises, où les décisions à prendre sont cruciales. Les informations de ces entreprises peuvent être converties en savoir à propos de patterns historiques ou des tendances futures. Par exemple, l'information sur les ventes au détail d'un supermarché peut être analysée dans le cadre d'efforts promotionnels, pour connaître des informations au sujet des comportements d'acheteurs. Ainsi, un producteur ou un vendeur peut déterminer quels produits doivent faire l'objet d'une promotion à l'aide du Data Mining mais également en utilisant l'analyse de données. La fouille de données permet de mieux comprendre sa clientèle, comprendre son comportement à partir de ses caractéristiques, constituer des panels représentatifs de clients, adapter sa politique de fidélisation. Elle permet d'optimiser l'adéquation de son offre à la demande, optimiser l'impact et la rentabilité des offres promotionnelles, ou même donner un ordre de priorité à ses actions de marketing et/ou de vente mieux cibler ses campagnes de marketing direct, évaluer la propension d'un prospect ou client à acheter un produit nouveau, etc. Les perspectives sont nombreuses.

Évidemment, les domaines d'applications du Big Data ne se limitent pas qu'aux entreprises, de nombreux progrès ont été effectués par les chercheurs scientifiques (météorologie, aéronautique, santé, énergie, etc.) grâce à l'analyse de ces données massives : la recherche biomédicale, par exemple, se sert de l'énorme banque de données pour évaluer et prédire les meilleurs traitements, diagnostiquer les épidémies, surveiller les agents pathogènes, etc. On retrouve aussi l'analyse de données dans de grands secteurs industriels comme l'agriculture, où le besoin d'utiliser le Big Data est né pour optimiser les moyens utilisés (utiliser moins d'eau, des produits non nocifs...) afin de satisfaire une exigence économique et écologique exponentielle.

Il y a aussi de plus en plus d'engouement envers les applications intelligentes, permettant d'obtenir de l'aide dans les situations de la vie quotidienne ; l'utilisateur moderne est assez paresseux et apprécie l'aide de ces nouveaux services comme Google Maps pour chercher des types de restaurants à proximité ou se repérer dans une ville, ou bien l'intelligence artificielle Siri qui répond à ses requêtes comme "joue-moi de la musique Rock". En particulier, les systèmes de recommandations sont actuellement présents sur de nombreux sites Internet et applications et viennent enrichir l'expérience de l'utilisateur en lui mettant en avant des suggestions d'articles, vidéos, produits susceptibles (avec une bonne probabilité) de lui plaire. Ces systèmes ont été popularisés par les sites de E-commerce comme Amazon mais se sont répandus sur d'autres plateformes comme les sites d'informations ou les plateformes de streaming et fonctionnent grâce à la collecte de données sur les utilisateurs. Le système de recommandations de Netflix, par exemple, jouit d'une bonne réputation auprès des abonnés et contribue grandement au succès de la plateforme. Les utilisateurs restent cependant méfiants/réticents en ce qui concerne la collecte de leurs données, notamment à cause d'entreprises comme Facebook qui du fait de leur étendue sur le web créent une sensation d'espionnage.

## 1.2. Objectifs du projet

Notre projet consiste en la conception et mise en œuvre d'un logiciel capable de réaliser des recommandations à chacun de ses utilisateurs de façon très personnalisée. En effet, de nos jours, nous cherchons par tous les moyens à gagner du temps dans chaque geste et action de notre quotidien. Parmi ces actions quotidiennes, nous pouvons souligner le fait de manger et cela à n'importe quel moment de notre journée (le petit-déjeuner, le déjeuner, la collation et le dîner par exemple). De toute évidence, si nous décidons de manger dehors, il est nécessaire parfois de se faire aider dans le choix de la nourriture que nous allons consommer ainsi que du lieu Figure 1.

YELP est une multinationale qui recueille les avis laissés par ses utilisateurs sur les différents établissements (restaurants, salons de beauté, etc) qu'ils ont fréquenté. L'entreprise propose chaque année de mettre à disposition une partie de leurs données aux développeurs et récompense ceux qui aboutissent à des résultats intéressants comme l'extraction de tendances inédites (Annexe 1). Nous avons donc, suite à la suggestion de notre tutrice, décidé d'utiliser ce jeu de données qu'on peut qualifier de "Big Data" pour construire notre système de recommandations.

Pour la conception de notre système, nous avons distingué cinq grandes étapes à réaliser/exploré :

- **Pré-processing des données** : cela englobe les étapes de Data Cleaning, Data Transformation, Data Integration, Data Reduction, etc.
- **Pattern mining / trends extraction** : il paraît compliqué au vu des données disponibles sur les utilisateurs de trouver des tendances par rapport à leur profil, mais nous pouvons essayer d'observer des différences de comportement vis-à-vis des businesses (tendances spatiales, temporelles...). Pour cela, nous utiliserons des algorithmes de Data Mining et des techniques d'agrégations.
- **Géolocalisation de l'utilisateur** : il s'agira de récupérer les coordonnées de l'utilisateur en respectant la confidentialité des données (ex : geo-indistinguishability) et d'évaluer les distances des businesses à proximité en respectant la cartographie de sa position (on ne peut pas utiliser la distance à vol d'oiseau si un fleuve sépare l'utilisateur de l'établissement). Les données fournies par YELP sont malheureusement restreintes sur le continent Nord-américain, aussi nous ne pourrons que recommander que des utilisateurs dont les coordonnées figurent dans cette zone.
- **Système hybride de recommandations** : nous utiliserons diverses techniques de recommandations existantes comme le collaborative filtering user-based (matrice User x Item en mémoire, modèle SVD, ...), le content-based filtering (avec/sans mot-clé), ainsi que des techniques personnalisées comme la recommandation par popularité des businesses aux alentours ou la recommandation par proximité. Ces techniques seront regroupées dans un système hybride qui pourra jongler entre chacune et/ou combiner les résultats selon différents cas de figures.
- **IHM / API** : nous voulons proposer une interface interactive et ludique pour les utilisateurs, dans une optique commerciale de pouvoir en accueillir de nouveaux. Il est aussi envisagé de créer un web service / API pour pouvoir requêter facilement le système de recommandations.

### 1.3. Idée de solution

La solution proposée est une application web en Python permettant de faire des recherches de restaurants via plusieurs critères comme la position GPS ou des mots clés renseignés, mais aussi de proposer une liste de businesses recommandés basés sur les préférences de l'utilisateur ou des autres consommateurs ayant les mêmes goûts. De ce fait, il y aura la possibilité d'afficher les tendances en fonctions de la saison pour aider l'utilisateur dans sa prise de décision. Tout cela implique l'utilisation de différentes techniques telles que le data mining sur les données de YELP pour trouver des tendances, diverses techniques dont le collaborative filtering et content-based filtering pour le système de recommandation et enfin la géolocalisation afin de faire des recherches pour les restaurants à proximité ou dans des villes choisies par l'utilisateur.

Par ailleurs, nous mettrons à disposition une API écrit avec Flask. L'API permet l'interface web d'utilisateur d'envoyer des requêtes HTTP et de répondre au navigateur de contenu de recommandation pour l'utilisateur final navigué sur le site. On désigne des endpoints différents pour les différents types de recommandation dans le cadre de ce projet.

## 1.4. Mise en scénario

Actuellement, nous pouvons facilement obtenir la note globale d'un restaurant en tapant son nom sur les nombreuses plateformes d'avis comme TripAdvisor ou Yelp. Cependant, chaque individu diffère et les avis de la masse ne convergent pas forcément vers ceux de l'utilisateur, et le procédé de taper le nom du restaurant puis lire les avis est fastidieux.

Prenons en exemple Alice. À 30 ans, elle habite à New York et elle est passionnée par la gastronomie chinoise. Un soir, elle se rend sur l'application pour trouver sa prochaine destination gustative. Après s'être géolocalisée, elle renseigne le style de nourriture qu'elle souhaite manger et peut ensuite choisir le restaurant. Notre application va à partir d'un système de score sur les utilisateurs similaires, déterminer les restaurants qu'elle est susceptible d'apprécier. Grâce à la géolocalisation, elle lui proposera alors les lieux les plus pertinents à proximité, et elle n'aura plus qu'à choisir celui qui lui paraît être le plus attrayant.

Nous aurons ainsi épargné au client le processus de recherche, de réflexion, et ainsi économisé son temps. Il aura de plus eu la possibilité de découvrir des lieux qu'il ne connaissait pas mais concordant avec ces envies. Cette mise en scénario peut être résumé à l'aide du schéma de la Figure 1.

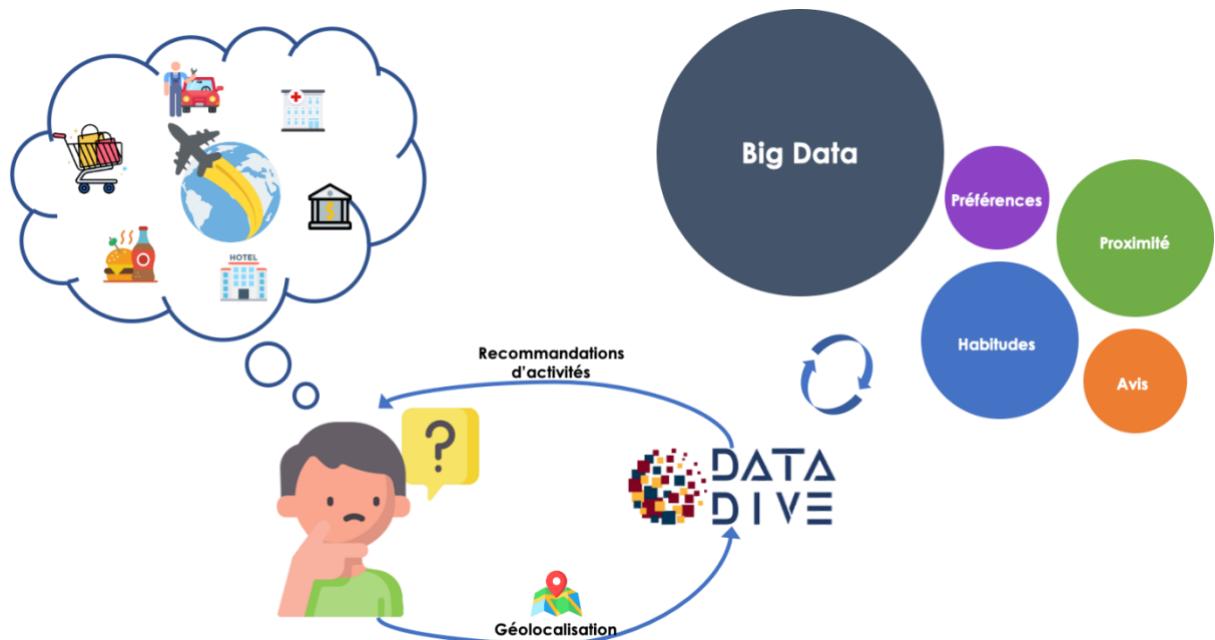


Figure 1 Data Dive : un outil de recommandation personnalisé

## **1.5. Organisation du rapport**

### **Chapitre 2**

Après avoir introduit notre projet, nous allons nous intéresser au chapitre 2 pour présenter et spécifier notre plateforme. Dans ce chapitre, nous allons parler des fonctionnalités globales de notre projet que nous développerons à la suite plus en détail. Ensuite nous allons faire une brève présentation de la conception globale du projet avant de soumettre les différentes problématiques qui se posent avec les solutions que nous proposons. Enfin, nous allons terminer par lister les outils et environnements de travail.

### **Chapitre 3**

Dans le chapitre 3, nous proposons de présenter l'une des parties techniques de notre projet : il s'agit de la Data Science. Ce chapitre nous permettra de présenter nos données et les différentes analyses que nous y avons effectuées dessus. Pour cela, nous commencerons par faire une analyse de la problématique qui se pose lorsque nous sommes amenés à manipuler une très grande quantité de données. Ensuite, d'une part, nous allons étudier les solutions existantes sur le marché. Puis, nous détaillerons la mise en œuvre des solutions que nous proposons ainsi que les tests qui y sont liés.

### **Chapitre 4**

Le quatrième chapitre sera consacré à la partie d'extraction de pattern de notre projet. Nous commençons par l'analyse de la problématique qu'il illustre les besoins de l'utilisateur. Ensuite, nous présentons les solutions existantes liées à cette problématique. Après cela, nous montrons notre approche pour la résolution de ce problème tout en expliquant le procédé que nous avons suivi pour y parvenir et les tests qui ont été effectué dessus.

### **Chapitre 5**

Cette partie est dédiée au système de recommandation basé sur la géolocalisation. Nous commençons par l'analyse de la problématique liée à la géolocalisation, ensuite l'analyse des différentes solutions existantes et le choix de la solution retenue. Pour finir nous expliquerons en détails comment nous avons mis en place le système recommandation basé sur la géolocalisation avec la solution retenue.

### **Chapitre 6**

Ce chapitre est consacré à toutes les spécificités techniques du système de recommandations hybride, à savoir la manipulation de Big Data, gestion de la mémoire, le choix et implémentation des diverses techniques utilisées, la combinaison des résultats, etc. Nous faisant l'état de l'art des systèmes hybrides ayant fait leurs preuves puis expliquons le fonctionnement du nôtre avec pour finir des certifications de la solution proposée.

## **Chapitre 7**

Dans le chapitre 7, nous proposons de vous exposer notre rendu final notamment avec la présentation des fonctionnalités, qui ont été implémenté, en s'appuyant sur l'interface utilisateur. Et nous nous attarderons ensuite sur différents tests en mettant en avant la façon dont ils ont été certifiés.

## **Chapitre 8**

Dans le chapitre 8, la gestion de projet que l'équipe s'est efforcé de respecter tout au long du projet est présentée. Nous nous concentrerons dans un premier temps sur ce qui concerne l'équipe avant de s'intéresser à toute la méthodologie mise en place.

## **Chapitre 9**

Nous allons dans le chapitre 9 conclure notre rapport en guise de synthèse des chapitres précédents avant de proposer quelques extensions possibles qui pourront être fait sur ce projet.

## **Chapitre 2 Présentation et spécification du projet**

S'avoir satisfaire et optimiser le temps l'utilisateur fait partie des préoccupations des systèmes de recommandations. Comment ces systèmes sont-ils pour s'y retrouver dans toutes ses données en tenant compte de chaque utilisateur ?

### **2.1. Étude du marché**

De nos jours, il est très facile d'obtenir un renseignement. Une grande majorité des métropolitains possède un smartphone capable d'accéder à Internet à l'aide des réseaux de données mobiles. En particulier, lorsque l'on souhaite trouver un restaurant, on peut demander à Google ou Siri de nous trouver des restaurants à proximité : l'outil va alors rechercher à l'aide de notre géolocalisation les lieux pertinents dans notre zone. Cependant, notre position géographique n'est pas le seul facteur : Google Maps, par exemple, a publié une mise à jour le 26 Juin 2018 pour son application mobile, qui indique à l'avance si l'utilisateur va aimer le restaurant ou non. En effet, les développeurs ont établi un algorithme capable de calculer son score d'affinité avec le restaurant ciblé, cela à partir des informations que l'application a récolté lors des sollicitations du user.

#### **Étude des besoins**

D'après une étude du cabinet Wavestone [3 études sur la consommation], plus de la moitié des consommateurs utilise des sites d'avis en ligne et les réseaux sociaux, en leur faisant confiance pour trouver des conseils sur des lieux où manger, qu'ils soient chez eux ou en vacances. De plus, des études récentes montrent également que la grande majorité des restaurateurs estiment que la gestion de leur réputation en ligne est devenue une étape essentielle de leur activité. Les consommateurs sont souvent dubitatifs avant de se rendre dans un restaurant. Ce doute est dû à la méconnaissance de la qualité du restaurant mais aussi de la fourchette du prix, qui est parfois trop élevé.

#### **Concurrence**

Notre perspective est donc de s'introduire dans ce marché en proposant notre système de recommandations personnalisées intelligent. Notre principal concurrent est la plateforme Tripadvisor qui est une entreprise américaine située à Newton aux États-Unis. Ce concurrent est détenu par une dizaine d'actionnaires à travers le monde et génère un chiffre d'affaire de 1.5 Milliard \$ en 2019. Ce concurrent a fait l'acquisition de plusieurs autres plateformes notamment LaFourchette en mai 2014 mais aussi plus récemment, en 2019, SinglePlatform.

Tripadvisor est une société d'environ 4 194 employés en 2019 et certaines de leurs filiales ont des clients comme Facebook, Google ou Yelp. GateGuru, une de leur filiale spécialisée dans le suivi de vols et le trafic aérien, a fermé récemment. La société, qui se déclare « le plus grand site de voyage au monde », est présente dans 45 pays dont la Chine sous le nom de Daodao. Elle accueille plus de 315 millions de visiteurs uniques chaque mois et recueille plus de 500 millions d'avis et d'opinions.

Cependant comme le décrit un support member de TripAdvisor, ces derniers ne proposent pas de recommandation intelligente : “*Trip Advisor does not recommend anything at all. It is a huge data base that allows businesses to list their properties.*” [4 Trip Advisor Recommendation]. Nous proposons donc un service qui se veut inédit pour ces millions d’utilisateurs réguliers.

## 2.2. Fonctionnalités attendues

Notre application web est un service de recherche intelligent qui propose à une clientèle de restaurants ou autres commerces (activités, hôtels, vols, services, etc...) des recommandations pertinentes, à l'aide de fouille de données dans les données de YELP, de diverses techniques de recommandation comme le collaborative filtering, le content-based et d'un traitement judicieux d'autres paramètres (géolocalisation de l'utilisateur, tendances, etc....). L'utilisateur aura la possibilité d'effectuer une recherche et de voir en détails les propriétés d'un restaurant qui lui est recommandé.

L'utilisateur est beaucoup influencé par les tendances, qu'elles soient temporelles, comportementales comme les habitudes de l'utilisateur. Ces dernières peuvent être extraites des données que nous avons à disposition afin d'apporter les informations en plus, non visibles directement à travers les données. Grâce à nos techniques de fouilles de données, l'utilisateur sera influencé dans le choix du prochain commerce qu'il visitera. Par exemple, il préférera déjeuner à des heures de non affluence, évitant ainsi les heures de pointes d'un restaurant, par exemple. De plus, il peut être attiré par des restaurants avec terrasse durant la saison estivale ou bien se réfugier au chaud dans un chalet dans hiver glacial.

Une des fonctionnalités majeures du projet est le système de recommandation grâce auquel des commerces pertinents pour ce dernier sont affichés sur l'interface graphique. Cette fonctionnalité permet, grâce à divers paramètres tels que les goûts et la géolocalisation, de retrouver les commerces les plus appropriés aux envies de l'utilisateur. En effet, ayant choisi de faire un système de recommandation hybride combinant plusieurs sortes de recommandation comme la recommandation objet ou content based (analyse des préférences de l'utilisateur vis-à-vis des qualités ou propriétés de l'objet ou du contenu) et la recommandation sociale ou collaborative filtering, quant à elle fondée sur le comportement des utilisateurs similaires ; notre application offre une recommandation assez complète afin de proposer à l'utilisateur du contenu susceptible de lui plaire.

La possibilité pour l'utilisateur de se géolocaliser est primordiale. En effet, prenons l'exemple d'une personne qui, lors d'un soir à Paris, ne connaissant pas l'adresse où elle se situe, souhaite se rendre dans un restaurant conforme à ses goûts et à proximité. Il lui suffira de se géolocaliser afin de s'y rendre, lui évitant ainsi la possibilité de rentrer une mauvaise adresse suggérée mais surtout de faire des économies de temps, de trajet et donc des économies de carburant. Cette fonctionnalité a pour but de simplifier la recherche de l'utilisateur en délimitant la recherche de commerces à une zone à proximité.

## 2.3. Conception globale du projet

### Vue pour l'utilisateur

L'utilisateur aura une interface assez simple d'utilisation constituée de plusieurs blocs. En effet, elle sera composée de :

- Un menu de navigation en haut de la page permettant d'aller d'une page à une autre
- Une barre de recherche pour rechercher un type de restaurant
- Une liste de restaurants recommandés pour l'utilisateur
- Une liste de restaurants à proximité de l'utilisateur
- Une carte interactive regroupant les différents restaurants à proximité

Le détail concernant un restaurant sera visible après avoir cliqué sur ce dernier. Suite à cela, une page s'ouvrira dans laquelle diverses informations seront présentées telles que les horaires d'ouverture, les commentaires du restaurant en question, les heures d'affluences et bien d'autres.

### Architecture technique

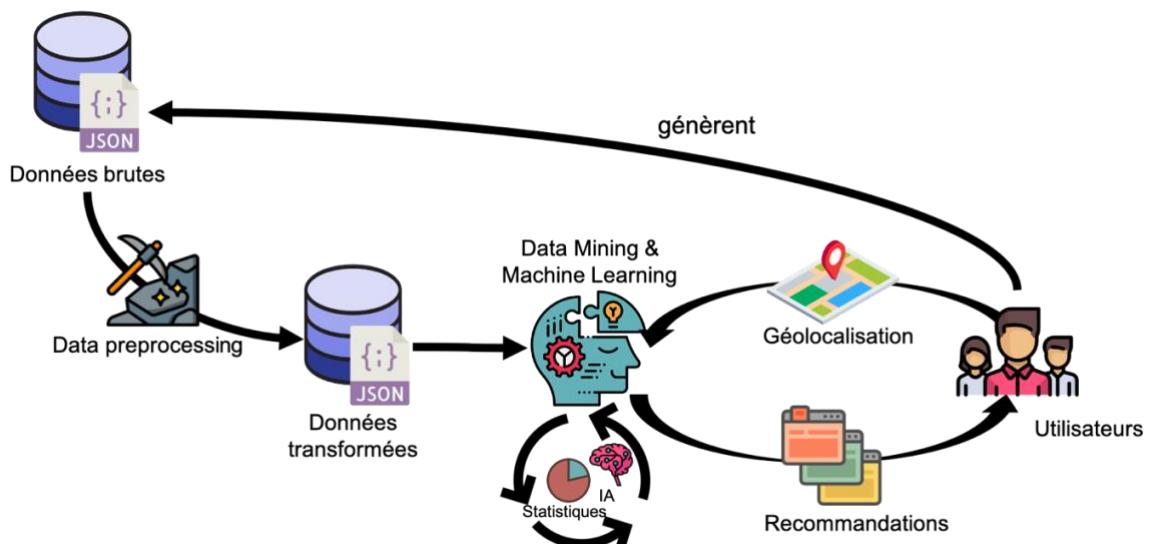


Figure 2 Architecture globale du projet

Afin de présenter l'architecture de notre projet, nous nous basons sur la Figure 2 qui présente les principaux modules de notre système, à savoir le Data Mining, le Machine Learning et la Géolocalisation.

Notre base de données est composée de 6 fichiers json (voir Chapitre 3) qui sont chacun comparable à une table. Ces données ont été pour une partie générées suite aux activités des utilisateurs, qui ont un fichier dédié dans la base de données.

Nous partons de ces données pour en obtenir de nouvelles grâce au processus de prétraitement. Ce processus est une phase très importante pour le bon fonctionnement de notre système. Les données qui vont être obtenues à la fin sont également sous forme de

fichiers json mais elles seront filtrées et beaucoup plus exploitable pour la suite. Ainsi, c'est sur ces nouvelles données, dites transformées que le reste des composants va se baser.

En effet, ensuite nous appliquons de la fouille de données sur ces données afin d'extraire des postulats utiles à l'apprentissage de notre système. Par la suite, nous pouvons commencer à entraîner notre système sur les postulats qui en sont ressortis.

Enfin, les utilisateurs, existants ou non dans la base de données, peuvent alors demander à être recommandé. Si aucune entrée automatique ou manuelle quant à la localisation a été réalisée par l'utilisateur, le système de recommandation affiche des résultats par défaut.

Afin d'obtenir des recommandations, l'utilisateur peut saisir des mots clés concernant le nom d'un restaurant, la catégorie, ou même la ville des restaurants qui pourrait potentiellement l'intéresser. Par ailleurs, s'il le souhaite, il peut également grâce à la géolocalisation obtenir des recommandations plus poussées.

## Architecture logicielle

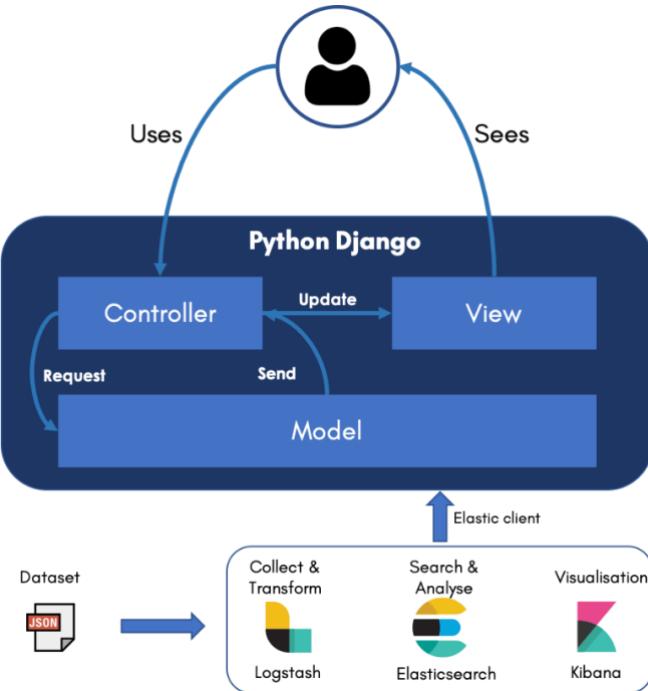


Figure 3 Architecture logicielle du projet

L'architecture logicielle (Figure 3) est une architecture classique 3-tiers : client – serveur – base de données. On dispose des données au format JSON qu'on envoie sur un moteur de recherche distribué, Elasticsearch, à l'aide de l'outil Logstash. À l'aide de Kibana, on peut visualiser les données. À ce moment, rentre en jeu l'application qui va se connecter aux données à l'aide d'un client Elastic. On utilise un modèle MVC, architecture logicielle sur lequel se base l'application et qui est très populaire dans le monde des applications web. L'utilisateur utilise le controller qui va requérir le modèle afin d'avoir des réponses au format JSON une fois de plus et va mettre à jour les vues pour l'utilisateur. Tout cela est effectué à l'aide de Django, un framework web Python très populaire.

## 2.4. Problématiques identifiées et solutions envisagées

Apporter toutes ces fonctionnalités couplées à de gros volumes de données implique certaines problématiques. L'une des problématiques majeures est issue du Big Data, il faudrait pouvoir faire des recherches rapides dans la base de données à l'aide de base de données NoSQL ou de moteur de recherche distribué. De plus, effectuer une analyse de gros volumes de données nécessite l'utilisation de technologies puissantes et efficace donc l'optimisation des algorithmes et le choix des bonnes librairies est primordial notamment lors des processus de traitement de données. En effet, la puissance de calcul étant limité sur nos ordinateurs, les algorithmes de lecture classique de fichier sont fortement déconseillés car ils ne sont pas optimisés pour les gros volumes de données et prendrait beaucoup de temps à s'exécuter. De ce fait, cela impacte aussi le système de recommandation et les algorithmes de Machine Learning.

Le système de recommandation doit pouvoir jongler entre diverses méthodes car chacune possède des avantages et des inconvénients, certaines sont plus adaptées par exemple pour des utilisateurs dont on connaît beaucoup d'informations tandis que d'autres sont plus générales mais fonctionnent mieux sur un plus grand jeu de données.

Les algorithmes de Data Mining / Machine Learning dépendent de la qualité de données traitées lors des phases de prétraitement des données. Une fois les données prédisposées, il est possible de faire :

- De l'extraction de patterns, trends à partir d'algorithmes de fouille de données, chercher à obtenir des informations utiles non évidentes pour l'humain mais justifiables par les données existantes ou calculées. Ces informations peuvent être spatiales, temporelles, fréquentielles...

Ex : les utilisateurs de la ville "Las Vegas" privilégient les restaurants ayant une terrasse pour les saisons Printemps / Été.

- De l'analyse textuelle ou NLP (Natural Language Processing), afin de faire ressortir de contenu texte brut des informations plus pertinentes, concises pour "feed" des algorithmes de Machine Learning, ou bien faire ressortir des informations invisibles (sentiments contenus dans le texte – "Sentimental Analysis", analyse du vocabulaire pour déterminer l'ethnie, l'âge, le sexe...).
- De l'intégration de données, en entraînant des modèles à compléter les données manquantes à partir des données existantes.

Notre jeu de données étant principalement situé sur le continent Nord-Américain, il nous est impossible de proposer un service de recommandation à des utilisateurs hors de ce périmètre. Nous devons donc imaginer un système capable de fonctionner pour n'importe quel utilisateur sur place, mais devrons simuler leur présence lors des phases de tests.

## 2.5. Environnement de travail

Pour la réalisation et la gestion de notre projet, nous avons utilisé différents outils, que nous avons plus ou moins réussi à maîtriser lors de notre cursus informatique. Nous avons listé ces outils dans le Tableau 1.

Aspect	Outils
<b>Système d'exploitation</b>	Windows macOs Linux
<b>Programmation</b>	Python Django
<b>Environnement de développement intégré</b>	PyCharm Jupyter
<b>Système de gestion de base de données</b>	Elasticsearch
<b>Test unitaires</b>	Pytest
<b>Conception/Modélisation</b>	Lucidchart
<b>Rédaction</b>	Microsoft Word / LateX

Tableau 1 Outils utilisés pour les différents aspects du projet

## Chapitre 3 Big Data et Analyse de données

Après avoir présenté et spécifié les différents aspects de notre plateforme, la question de l'analyse de données se pose. Une quantité importante de données doit être stockée dans une base de données et doit être exploitable par l'utilisateur.

### 3.1. Analyse de la problématique

Tout travail de Big Data commence par une longue exploration des données, et cela perdure tout le long du projet avec différentes étapes. De la collection de données jusqu'au stockage et l'exploitation de ces dernières avec de la datavisualisation, en passant par le processus de nettoyage de données, le workflow du traitement des données est un long processus qui nécessite une analyse constante. L'analyse vaut pour chaque étape du processus ; elle permet d'obtenir une compréhension fine des éléments qui composent les données pour envisager une approche analytique la plus productive possible.

Dans notre cas, nous sommes face à une grande quantité de données qui nécessite plusieurs étapes de traitement de par la masse de données mais aussi la qualité que contient cette dernière. Les différentes étapes nous permettront d'aboutir à des données plus propres et plus compréhensibles par l'utilisateur

La première étape consiste à collecter les données (aussi appelée Data Collection) depuis différentes sources et sous différents formats. Dans notre cas, les données proviennent du site YELP [5 YELP data], plateforme qui met à disposition un dataset, à propos de divers commerces, que nous allons détailler à présent.

Notre dataset, composé de 6 fichiers au format JSON, comportent des données spécifiques à l'Amérique du Nord et étalées sur 9 ans (2010 à 2018). Afin de mieux comprendre ces données, nous vous proposons de vous appuyer sur le modèle conceptuel de données (Annexe 2).

Sur la Figure 4, nous pouvons voir les différents fichiers que nous avons avec le nombre de données qu'ils comportent ainsi que la taille de ces derniers :

- **Business** : Ce sont les établissements. C'est le fichier principal de notre base de données. Chaque établissement possède un nom, une liste de catégories dans lesquelles il peut être classé, le note et le nombre d'avis laissés par les utilisateurs. Nous pouvons également trouver des informations sur sa localisation : la longitude, la latitude, l'adresse, la ville, le code postal et l'Etat. Il y a aussi les heures d'ouvertures de l'établissement ainsi qu'une variable qui précise si celui-ci est toujours ouvert ou fermé définitivement.
- **Check-in** : Ce sont les visites que les établissements ont reçues. Une visite correspond à une liste de dates de visites pour chaque établissement.
- **Photo** : Ces sont les photos liées aux établissements. Les photos sont de type JPG et sont présents dans un dossier photos/ grâce à leur identifiant. Chaque photo, possédant une légende, est liée à un établissement et à une catégorie.
- **User** : Ce sont les profils utilisateurs, appelés souvent les yelpers. Un utilisateur peut être qualifié d'élite ou pas. Chaque utilisateur possède un nom et une liste

d'amis. Nous pouvons également obtenir, entre autres, sa date d'inscription au plateforme YELP, le nombre de personnes qui le suivent, le nombre d'avis et la moyenne des notes qu'il a laissé jusqu'à présent. Chaque utilisateur est également noté par d'autres utilisateurs sur certains critères : drôle, cool et utile.

- **Review** : Ce sont les avis laissés par les utilisateurs sur les différents établissements. Chaque avis d'utilisateur, sous forme de texte et d'une note (étoile), est daté et évalué par d'autres utilisateurs en se basant sur les mêmes critères que pour les utilisateurs ci-dessus.
- **Tip** : Ce sont les conseils donnés par les utilisateurs sur les différents établissements. Ces conseils, sous la forme d'un texte, sont datés.

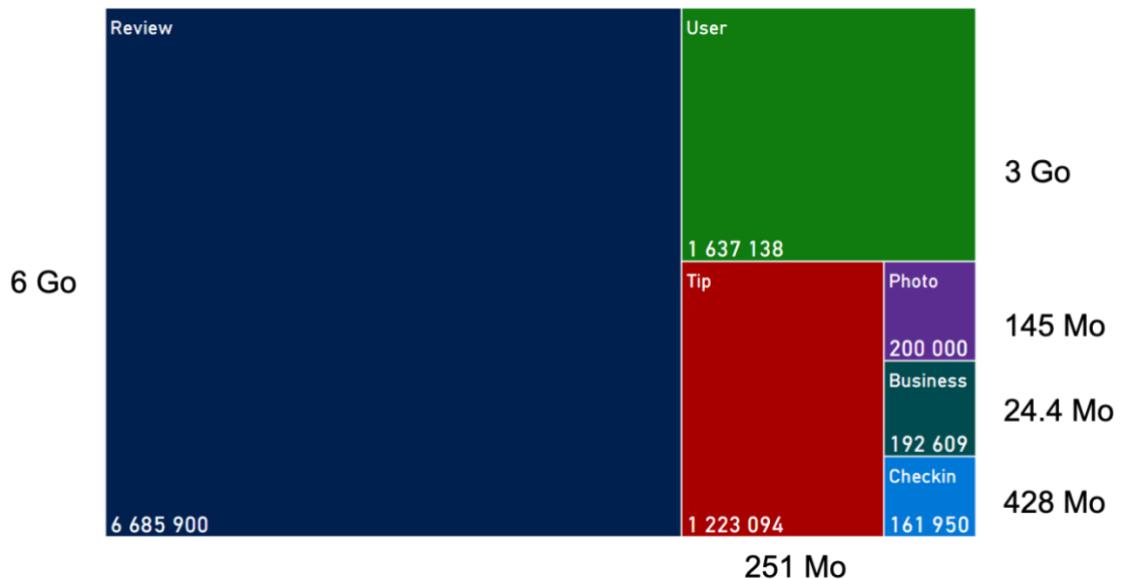


Figure 4 TreeMap représentant la répartition des données dans les fichiers

Après la collecte des données suit l'étape de préparation des données (aussi appelé étape de prétraitement ou Data Preprocessing). C'est l'étape pendant laquelle les données brutes sont nettoyées en vue de l'étape suivante du traitement des données. Initialement, les données ne sont pas forcément propres dans le monde informatique. En effet, la donnée brute est vérifiée avec soin mais elle peut parfois être :

- Incomplète, c'est-à-dire qu'il manque certaines valeurs à certains endroits.
- Contenant des erreurs.
- Manquante (absence de certaines colonnes)

La qualité de la donnée est extrêmement importante pour la suite des étapes. Pour pallier à ce problème, on peut appliquer différentes solutions, qu'elles soient personnalisées ou utilisant du Machine Learning, afin de compléter les données manquantes, changer la structure de certains champs ou créer de nouvelles colonnes, par exemple. Cette étape peut s'avérer coûteuse en temps et possiblement onéreuse si l'on doit avoir recours à des services comme Amazon Web Services, dites AWS [6 aws], ou des APIs payantes comme celles de Google. Afin d'optimiser les temps de calcul, nous avons utilisé des librairies spécifiques à l'analyse de données volumineuse.

La dernière étape du traitement des données est le stockage. Lorsque toutes les données ont été nettoyées, elles sont stockées pour une utilisation ultérieure (certaines données peuvent être utilisées immédiatement).

Nous évoluons dans un monde où l'on a besoin de stocker de plus en plus d'informations chaque jour mais également de pouvoir rechercher des informations de façon toujours plus rapide et efficace. Une recherche dans, par exemple, une base de données relationnelle classique comme MySQL peut paraître simple et rapide si la requête est bien construite avec les jointures nécessaires entre les tables et si les données sont peu volumineuses. Mais plus la volumétrie de cette base de données va grossir, plus les requêtes seront longues à exécuter notamment s'il y a des recherches utilisant des jointures entre les différentes tables.

### 3.2. État de l'art : études des solutions existantes

Aujourd'hui, il n'existe pas d'outils tout-en-un, un outil qui permettrait de faire les différentes étapes de traitement de données et assurer le stockage. Pour cela, on divise le processus en deux parties :

- L'analyse et le pré-traitement de données
- Le stockage de données

En Python, il existe diverses librairies nous facilitant l'analyse de données comme Pandas ou Spark, qui sont assez populaires dans le monde analytique de Python.

En ce qui concerne le stockage de gros volumes de données, plusieurs solutions existent comme par exemple les bases de données NoSQL comme MongoDB, déjà utilisé lors des projets de synthèse de M1, ou les moteurs de recherche et d'analyse distribués comme Elasticsearch. Grâce au Tableau 2, nous pouvons réaliser un comparatif de ces solutions afin de pouvoir choisir, dans la section suivante, la plus adaptée à notre projet (voir 3.3.2).

	MongoDB	Elasticsearch
Prix	Payant	Gratuit
Taille	Limité à 500 Mo pour la version gratuite	Illimité selon la taille du disque
Classement DB <sub>1</sub>	5	7
Type	Base NoSQL	Moteur de recherche basé sur Lucène
Schéma de données	Orienté Document	Orienté Document et Index Inversé

Tableau 2 Comparatif des solutions pour le stockage des données

### 3.3. Solution proposée et sa mise en œuvre

#### 3.3.1. Analyse et pré-traitement de données

Nous avons mis en œuvre des scripts Python sur Jupyter Notebook en utilisant des librairies d'analyse de données comme Pandas et Spark ainsi que la librairie de Machine Learning la plus connue qui est scikit-learn [7 scikit-learn].

<sup>1</sup> D'après <https://db-engines.com/en/ranking>

Pour la préparation des données, nous les avons d'abord visualisés afin d'avoir une vue d'ensemble sur les différents champs et ainsi faire un profilage de colonnes. L'analyse complète est disponible en annexe (Annexe 2). De ce fait, nous avons remarqué que certaines colonnes contenaient des champs vides. Par exemple, dans le fichier *Business*, le champ *categories* contenait des valeurs vides, le champ *hours* contenait 25% de valeurs nulles, et certaines entrées n'avaient pas de champ *attributes* tandis que d'autres pour des champs liés à la géolocalisation (*address*, *city* et *postal\_code*). Ainsi pour les remplir, nous avons utilisé plusieurs techniques :

- Nos propres algorithmes
  - K-Nearest Neighbours (kNN)
  - La bibliothèque Python geopy
  - Le remplissage manuel lorsque c'était nécessaire

## Implémentation d'algorithmes

Pour le remplissage des données manquantes des champs *categories* et *attributes*, nous avons implémenté nos propres fonctions. Le champ *categories* est une liste de catégories liés au *Business* et le champ *attributes* est un dictionnaire avec comme clé l'attribut et comme valeur un booléen (True/False).

D'une part, pour remplir les 482 valeurs vides du champ *attributes*, nous avons dû d'abord remplacer ces valeurs non renseignées par la valeur « Unknown » pour pouvoir les rendre exploitables par la suite. Ensuite, grâce au logiciel Power BI, qui est une solution rapide et esthétique visuellement, nous avons généré une liste de mots qui apparaissent le plus dans les *categories* (Figure 5) et les *names* (Figure 6) des Business. Ces mots ont été classés dans 11 listes selon leur thème : Food/Restaurant - Hair, Beauty and Spas - Automotive - Bank - Hotel, Travel - Health - Shopping - Activity/Entertainment - Animals - Education - Services.



Figure 5 Nuage de mots basé sur les catégories des business



Figure 6 Nuage de mots basé sur les noms des business

Notre fonction consiste à catégoriser les *Business* selon leur *categories* dans un premier temps. En effet, si un mot se trouvant dans son champ *categories* est parmi les mots classés dans les listes, le *Business* est catégorisé par le nom de la liste concernée sinon par “Others”. Et dans un second temps, pour ceux qui n’ont pas pu être catégoriser de le faire selon leur *name* de la même manière. À partir de cette catégorisation, nous avons créé un nouveau champ *categories\_label*.

Enfin, nous avons rempli pour les *Business* dont les *attributes* sont manquantes avec les *attributes* d’un *Business*, choisi aléatoirement, qui possède la même valeur pour le champ *categories\_label*.

D’autre part, pour remplir les 28 836 valeurs vides du champ *categories*, nous nous sommes basés sur les champs *categories\_label* et *stars*. Pour ces derniers, après avoir remplacé les valeurs non renseignées par « N/A », nous avons procédé de 3 façons différents selon les cas :

- Si aucun *Business* ayant le même *categorie\_label* et le même nombre de *stars* n’a de champ *attributes* rempli, nous avons choisi de le remplir de manière aléatoire, par les *attributes* d’un *Business* qui a soit le même nombre de *stars*, soit le même *categories\_label*.
- Sinon, si aucun *Business* ayant le même *categorie\_label* n’a jamais le champ *attributes* rempli, nous avons choisi de remplir par les *attributes* d’un *Business* qui a le même nombre de *stars*.
- Sinon nous le remplissons par les *attributes* d’un *Business* qui a le même *categories\_label* et le même nombre de *stars*.

## Algorithme de Machine Learning

Pour le remplissage des données manquantes du champ *hours*, nous avons utilisé une partie des algorithmes de Machine Learning, notamment k-Nearest Neighbours (kNN), est

une approche de classification supervisée intuitive. Le champ *hour* est un dictionnaire avec comme clés les jours de la semaine et comme valeur l'heure d'ouverture pour le jour concerné.

Pour débuter, nous avons divisé le champ *hours* dans 7 nouveaux champs temporaires, un par jour de la semaine. Ensuite, nous avons modifié les données qui étaient non renseignée par la valeur « Closed ». Et enfin, en prenant pour target chaque jour, nous avons lancé kNN sur les données qui n'ont pas de valeur manquante pour ce champ avec 4 features : *city*, *stars*, *review\_count* et *categories\_label*. Bien évidemment, pour les champs non-numériques *city* et *categories\_label*, nous les avons d'abord encodés avec la technique du Label Encoding. Puis en ce qui concerne l'hyper-paramètre *k* (le nombre de voisins le plus proche), nous avons choisi *k*=3 après avoir tracé la courbe d'apprentissage (Figure 7), ainsi nous avons eu une précision de 78% environ.

Ainsi afin de remplir les 7 champs de jour, après avoir entraîné notre modèle, pour chacun d'eux nous l'avons appliqué aux Business n'ayant pas de valeur pour *hours*. Finalement, nous avons pu reporter de nouveau ces 7 champs, avant de les effacer, dans un dictionnaire que nous avons attribué au champ *hours*.

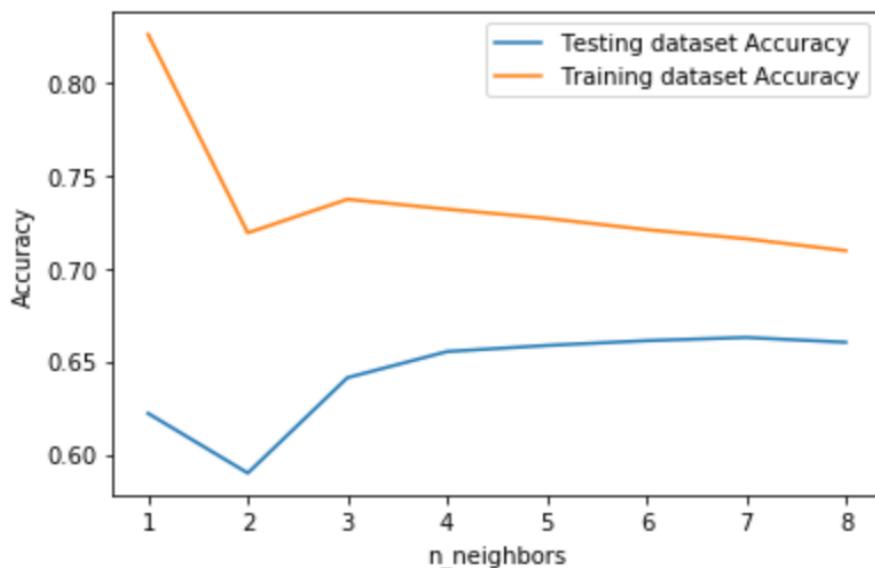


Figure 7 Courbe d'apprentissage pour l'algorithme kNN

## Bibliothèque Python

Pour les champs, liés à la géolocalisation, possédant des valeurs manquantes, nous avons utilisé une bibliothèque Python, geopy. Ces champs vides ont été plus difficile à identifier car contrairement aux autres champs traités précédemment, le champ était bien renseigné mais d'une chaîne de caractère vide. Les fonctions basiques en python `isna()` (appartenant à la bibliothèque numpy) et `isnull()` nous ne permettaient pas de les identifier.

Ainsi, nous avons pu voir qu'il manquait 7682 valeurs pour le champ *address*, 1 pour *city* et 659 pour *postal\_code*. Ensuite, nous avons pu récupérer très facilement les valeurs

manquantes grâce à cette bibliothèque en renseignant la *latitude* et la *longitude* du *Business* et en les utilisant avec du reverse geocoding.

### Technique manuelle

Cependant, n'ayant pas pu obtenir 10 valeurs pour le champ *postal\_code* pour quelques commerces, nous avons été amenés à les remplir manuellement.

### 3.3.2. Stockage de données

Pour le stockage de données, nous avons retenu la solution d'Elasticsearch car elle est gratuite, illimité en termes de stockage et permet de faire des recherches rapides sur plusieurs millions de données. L'utilisation d'Elasticsearch se résume en trois étapes, qui forme ce qu'on appelle la suite ELK (Elastic, Logstash et Kibana) :

- Tout d'abord Logstash qui est l'outil de collecte et d'analyse. Il prend les logs en entrée, ici des entrées JSON, afin de les transformer éventuellement et les parser pour ensuite les stocker dans Elasticsearch à l'aide d'un fichier de configuration. Ce fichier est composé de trois fonctions L correspond à l'entrée. C'est les fichiers de logs que l'on donne à Logstash. Le filter lui va lire les fichiers de l, les filtrer et parser les logs. L va mettre les résultats du filter dans la base de données, Elasticsearch (voir Annexe 3).
- Une fois que les données sont envoyées sur Elasticsearch, on peut les requêter avec une API Restful ou un client Python, en spécifiant plusieurs paramètres comme le nombre de résultats à retourner ou des filtres sur les données similaires à la clause WHERE du langage SQL.
- Enfin, Kibana qui est l'outil de visualisation des données à partir duquel on peut créer des tableaux de bord ou dashboard (voir Annexe 4)

### 3.4. Tests et certifications de la solution

Afin de pouvoir tester notre solution, nous avons commencé à implémenter nos différentes méthodes sur des échantillons de données de différentes tailles. Nous avons pu, par la suite réalisé des vérifications rapides, à main levé sur la cohérence des résultats,

Par ailleurs, pour l'algorithme kNN, nous avons pu visualiser graphiquement la corrélation des champs entre elles grâce au heatmap de la Figure 8. Cette fonction appartient à la librairie seaborn [8 seaborn heatmap]. Le choix des features s'est basé sur ce heatmap et à la fin nous avons pu nous baser sur un rapport de classification, qui présentait notamment une précision d'environ 80%, pour certifier notre modèle.

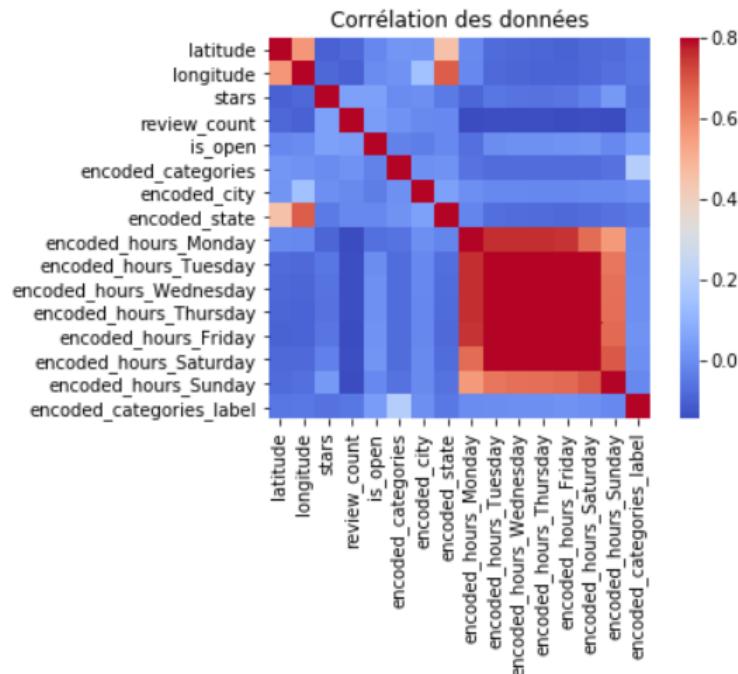


Figure 8 Heatmap représentant la corrélation des features

## Chapitre 4 Extraction de pattern

Finalement, après une longue phase de préparation de données qui nous a permis de mieux connaître nos données, est venue la question d'extraction de pattern (modèles) ressortir les comportements et le(s) phénomène(s) sous-jacent(s). Cette étape est assez importante pour projet car les postulats qui en ressortent seront les fondations de l'apprentissage de notre machine.

### 4.1. Analyse de la problématique

D'après une des études du marché Euromonitor International [9 étude de tendances], 10 principales tendances de consommation mondiales ont pu être identifiées pour cette année 2020. Parmi ces tendances présentées dans la Figure 9, deux nous ont paru intéressantes : « Catch Me in seconds » et « Private Personalisation ».



Figure 9 Les 10 principales tendances de consommation mondiales (2020)

D'une part, celle intitulée « Catch Me in seconds » explique que le consommateur souhaite recevoir plus de contenu en moins de temps. Et d'autre part, celle intitulée « Private Personalisation » explique que ce dernier désire également avoir une expérience sur mesure. En effet, beaucoup de consommateurs commencent à remettre en question la manière d'agir collective que leur impose la société comme l'a dit De Paulo Coelho, un écrivain brésilien : « Chaque être humain est unique, il a ses propres qualités, ses instincts, ses formes de plaisir. Cependant la société impose une manière d'agir collective, et les gens ne cessent de se demander pourquoi ils doivent se comporter ainsi. » [10 citation de paulo]. Afin de répondre à ces deux attentes, il faut pouvoir proposer des résultats pertinents et ciblés.

Dans notre cas, nous avons 1 637 168 utilisateurs différents comme nous l'avons montré sur le Tree Map (Figure 4) du Chapitre 3 ci-dessus. Un utilisateur possède 22 champs dont un user\_id par lesquels il est identifié. Il est également doté d'autres champs très pertinents tels que la liste de ses amis, le nombre de fans, etc. Après le fichier comportant les avis, le fichier des utilisateurs est le plus volumineux. Pour chaque utilisateur, il est question ici de faire ressortir ses tendances habituelles quant à sa fréquentation des différents restaurants, mais également sur les caractéristiques des restaurants fréquentés. De ce fait, chaque utilisateur pourra ainsi recevoir une recommandation personnalisée.

Cependant, il est nécessaire également de présenter d'autres éléments susceptibles de l'intéresser en dehors de ses habitudes. C'est pourquoi, nous prenons également en compte qu'un utilisateur donné peut être amené à faire des choix selon sa localisation, c'est-à-dire sa ville, ou même son état. Dans nos données, nous avons 1 203 villes différents, de par leur nom, pour lesquels il faut générer des tendances temporelles.

## 4.2. État de l'art : études des solutions existantes

### 4.2.1. La fouille de données

La fouille des données, plus communément appelée Data Mining, est la combinaison de quatre différents domaines : les statistiques, l'intelligence artificielle (IA), la visualisation des données (DataViz ou datavisualisation) ainsi que la gestion des données. En Data Mining, il existe de nombreuses méthodes séparées en deux types : prédictives et descriptives.

Les méthodes de type prédictif (Arbre de décision et Réseaux de neurones) permettent d'extraire des modèles tandis que celles de type descriptif (Tableau 3), des patterns.

Méthodes	Algorithmes	Description
Clustering	K-means, CAH	Identification des clusters <sup>2</sup> intégrés dans les données
Règles d'association	Apriori, FP-Growth, Eclat, SSDM, kDCI	Découverte des associations ou des relations cachées dans les grandes bases de données
Séquence mining	GSP, SPADE	Création d'un ensemble de fonctionnalités <sup>3</sup> basé sur les données d'origine

Tableau 3 Méthodes descriptives de Data Mining

Souhaitant extraire des patterns, ce sont les analyses non supervisées qui sont adaptées à notre cas. Comme nous montre le Tableau 3, il existe 3 méthodes possibles pour ce type d'analyse et pour chacun d'eux, il existe différents algorithmes. Le plus intéressant dans notre cas serait l'utilisation de méthode de règles d'association étant donné que nous souhaitons connaître les habitudes.

Les règles associatives sont des règles qui sont extraites d'une base de données transactionnelles (item-set) et qui décrivent des associations entre certains éléments. Cette technique permet de faire ressortir les associations entre les choix des utilisateurs. Par exemple, si un utilisateur donné a visité un restaurant de type A, il est très probable qu'il visite

<sup>2</sup> Cluster : collection d'objets de données similaires en quelque sorte les uns aux autres.

<sup>3</sup> Fonctionnalité : combinaison d'attributs qui présentent beaucoup de caractéristiques similaires

un restaurant de type B. Grâce à ces règles, nous pouvons accroître les chances de prédire des résultats très pertinents pour l'utilisateur. Ces algorithmes permettent de résoudre des problèmes dits de Frequent Set Counting (FSC).

Les algorithmes de règles d'association peuvent être séparés en 2 catégories selon le type de parcours des données qu'ils effectuent : parcours en largeur (ex Apriori, kDCI) et parcours en profondeur (ex FP-Growth, Eclat).

Ces différents algorithmes utilisent, tous, les notions de support (Équation 2) et de confiance (Équation 3), qui se basent sur le support d'un item (Équation 1) pour déterminer la pertinence d'une règle d'association. Une règle d'association est une application de la forme  $X \rightarrow Y$  où  $X$  et  $Y$  sont des ensembles d'items disjoints.

$$Support, \sigma(X) = Card(\{t_i | X \subseteq t_i, t_i \in T\})$$

où  $Card(A)$  représente le cardinal de l'ensemble  $A$ .

Équation 1 Support d'un item

$$Support, s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N}$$

Équation 2 Support d'une règle d'association

$$Confiance, c(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

Équation 3 Confiance d'une règle d'association

Les règles associatives générées sont retenues seulement si elles ont un support supérieur à  $minSupp$  et une confiance supérieure à  $minConf$  qui sont deux constantes à définir au début.

Concernant nos données, il serait intéressant d'utiliser un algorithme classique de règles d'association notamment Apriori (Annexe 5) qui est un algorithme publié en 1993 par Agrawall, Imielinski et Swami.

L'algorithme Apriori démarre avec la liste des produits les plus fréquents dans la base de données respectant les seuils de support. Un ensemble de règles (des candidats) est généré à partir de cette liste. Les candidats sont testés sur la base de données et les candidats ne respectant pas  $minSupp$  et  $minConf$  sont retirées. Tant que l'algorithme ne trouve pas de règles pertinentes, l'itération de ce processus a lieu en augmentant à chaque fois la taille des candidats d'une unité. À la fin, les ensembles de règles trouvés sont fusionnés.

La génération des candidats se fait en deux étapes :

- 1) La jointure : jointure d'un ensemble de règles à k éléments sur lui-même qui aboutit à la génération d'un ensemble de candidats à k+1 éléments.
- 2) L'élagage : suppression les candidats dont au moins une des sous-chaîne à k éléments n'est pas présente dans l'ensemble des règles à k éléments.

### 4.2.2. Manipulation de dataframes

Il existe une autre solution, intéressante, pour l'extraction de pattern qui est la manipulation de dataframe qui va nous permettre de visualiser les tendances plus facilement. À partir de nos données, nous pouvons appliquer quelques fonctions pour obtenir de nouvelles features qui représenteront différentes tendances : spécifiques à une période de temps, à un utilisateur, etc.

En effet, nous pouvons utiliser la librairie python Pandas, qui possède une documentation complète [11 librairie pandas], afin de générer de nouvelles données pertinentes. Cette dernière, rapide, puissante, flexible et simple d'utilisation permet de manipuler et analyser des données. Pour la manipulation, les données sont stockées sous forme de tableaux avec des lignes (rows) qui représentent une observation et des colonnes (features) qui sont les différents attributs d'une observation. Ces tableaux sont appelés des dataframes et peuvent être chargé à travers la lecture de fichiers csv, json, etc. Mais la création de nouveaux fichiers de ces types est aussi possible grâce à elles. Nous pouvons, par ailleurs, visualiser ces données sous forme de graphique avec une autre librairie python, matplotlib.

Avec la librairie Pandas, paquetage chargé au préalable avec *import pandas*, le chargement est très rapide avec les différentes fonctions de l'écriture que celle-ci propose. Notre base de données étant composée de fichiers json, c'est la fonction *read\_json()* qu'il faut utiliser. À la suite, il est possible d'obtenir des informations très basiques sur nos données comme les nombres de colonnes et de lignes avec la fonction *shape()* ou même un rapport statistique (Figure 10) grâce à la fonction *describe()*. Il faut noter que certains indicateurs statistiques ne sont valables que pour les variables numériques (ex. moyenne, min, etc.), et inversement pour les non-numériques (ex. top, fréquence, etc.) mais dans tous les cas, cette fonction est très intéressante pour la bonne compréhension et manipulation des données.

	business_id	name	address	city	state	postal_code	latitude	longitude	stars	review_count	is_open	attributes	categories	categories_label
count	192609	192609	192609	192609	192609	192609	192609.000000	192609.000000	192609.000000	192609.000000	192609.000000	192609	192609	192609
unique	192609	145046	153770	1203	36	17604	NaN	NaN	NaN	NaN	NaN	67875	93386	12
top	m8hXQJQ29znUly4NFlvHQ	Starbucks	Enhanced Municipal Services District Boundary, Suite 1400, North 4th Street, Roosevelt Point, Central City, Phoenix, Maricopa County	Las Vegas	AZ	89109	NaN	NaN	NaN	NaN	NaN	{'BusinessAcceptsCreditCards': 'True'}	Restaurants, Pizza	Food, Restaurant
freq	1	1066	239	29370	56686	3196	NaN	NaN	NaN	NaN	NaN	20362	1043	76358
mean	NaN	NaN	NaN	NaN	NaN	NaN	38.541803	-97.594785	3.585627	33.538962	0.823040	NaN	NaN	NaN
std	NaN	NaN	NaN	NaN	NaN	NaN	4.941964	16.697725	1.018458	110.135224	0.381635	NaN	NaN	NaN
min	NaN	NaN	NaN	NaN	NaN	NaN	33.204642	-115.493471	1.000000	3.000000	0.000000	NaN	NaN	NaN
25%	NaN	NaN	NaN	NaN	NaN	NaN	33.637408	-112.274677	3.000000	4.000000	1.000000	NaN	NaN	NaN
50%	NaN	NaN	NaN	NaN	NaN	NaN	36.144815	-111.759323	3.500000	9.000000	1.000000	NaN	NaN	NaN
75%	NaN	NaN	NaN	NaN	NaN	NaN	43.602989	-79.983614	4.500000	25.000000	1.000000	NaN	NaN	NaN
max	NaN	NaN	NaN	NaN	NaN	NaN	51.299943	-72.911982	5.000000	8348.000000	1.000000	NaN	NaN	NaN

Figure 10 Rapport sur les données des établissements (business)

De plus, cette librairie nous permet d'accéder aux valeurs très facilement de deux manières :

- En parcourant le dataframe avec les *iterrows()* et *itertuples()*,
- Ou bien en y accédant directement avec les crochets (`['column_name']`) et un point (ex. `column_name`).

Par ailleurs, en ce qui concerne l'ajout ou la modification d'une colonne, nous avons également diverses méthodes :

- La déclaration d'une nouvelle liste comme colonne. Cette liste doit comporter autant d'éléments que de données (rows) dans le dataframe.  
⇒ `dataframe['column_name'] = list_of_values`
- L'utilisation de la fonction *insert()* qui prend en paramètres la position à laquelle la colonne doit être insérer dans le dataframe, le nom de cette colonne, et la liste de valeurs qui encore ne doit comporter autant d'éléments que de données dans le dataframe.  
⇒ `dataframe.insert(position, 'column_name', list_of_values)`
- L'utilisation de la fonction *assign()* qui prend en paramètres le nom de la colonne et la liste des éléments, toujours de la même taille. Cette fonction permet également de générer un nouveau dataframe avec pour données cette nouvelle colonne ajoutant aux colonnes de l'ancien dataframe.  
⇒ `dataframe = dataframe.assign('column_name' = list_of_values)`  
⇒ `new_dataframe = dataframe.assign('column_name' = list_of_values)`
- La déclaration de dictionnaire qui aura pour clés les valeurs d'une colonne existante et pour valeurs les valeurs de la nouvelle colonne à ajouter.  
⇒ `dict = {existing_v1 : new_v1 ; ... ; existing_vn : new_vn}`  
`dataframe['column_name'] = dict`
- L'utilisation de la fonction *apply()* qui permet de changer les valeurs d'une colonne rapidement ou bien de créer une nouvelles colonnes. Pur ces deux cas, nous pouvons nous baser sur des colonnes existantes. Si nous souhaitons utiliser qu'une seule colonne, nous pouvons appliquer la fonction après avoir sélectionner cette colonne, sinon on l'applique directement sur la dataframe. Ensuite, nous pouvons implémenter directement, à l'intérieur de la fonction, les actions à effectuer pour obtenir la nouvelle valeur ou bien passer par une autre fonction préalablement définie.  
⇒ `dataframe['column_name']= daframe['column'].apply(lambda x : funct)`  
⇒ `dataframe['column_name']= daframe.apply(lambda row : funct)`

Tout comme la modification ou l'ajout de colonnes, la suppression de colonne est simple avec la fonction *drop()*. De ce fait, cela nous encourage à travailler avec des colonnes temporaires. Ces colonnes temporaires sont un atout pour ne pas se perdre dans nos variables. Le type valeurs des colonnes qu'elles soient temporaires ou pas, peuvent être changé à tout moment grâce à la fonction *apply()* vu ci-dessus et les fonctions de typages telles que *str()*, *list()*, *dict()*, etc.

Bien que nous pouvons utiliser les fonctions *len()* et *count()* pour travailler sur les tendances, la librairie Pandas propose la fonction *value\_count()* qui nous permet de d'obtenir directement le nombre d'apparition de chaque valeur possible pour une colonne donnée. Cette dernière est très intéressante pour l'extraction d'item set.

Enfin, nous pouvons utiliser la fonction *sort\_values* pour trier nos listes et ne garder que les meilleurs éléments à prédire en premier. Et par-dessus tout, travailler avec les dataframe Pandas nous permet d'avoir un affichage agréable.

### 4.3. Solution proposée et sa mise en œuvre

Avant toute chose, nous avons jugé nécessaire de séparer les tendances par ville. En effet, les goûts et habitudes changent très souvent d'une région à l'autre. Il se peut que les restaurants d'une catégorie donnée soient vraiment populaires dans une ville alors que dans une ville voisine, cette catégorie de restaurants n'est pas fréquentée du tout.

Pour extraire nos tendances habituelles, nous nous sommes basés sur les avis laissés par les utilisateurs sur les 3 dernières années, c'est-à-dire de 2016 à 2018, pour les établissements catégorisés « Food/Restaurant » dans le chapitre précédemment. Seuls les utilisateurs ayant laissés des avis sur les 3 dernières années sont considérés comme actifs. Et par ailleurs, prendre les avis récents semble plus pertinent car les goûts et envies des utilisateurs peuvent évoluer dans le temps donc il est nécessaire d'avoir d'analyser des données récentes. De ce fait, notre travail s'est effectué sur une quantité réduite mais toujours aussi conséquente (Tableau 4).

	Données de base	Données utilisées
Villes	1203	812
Avis	6 685 900	235 861
Établissements	192 609	62 760
Utilisateurs	1 637 138	1 122 987

Tableau 4 Données pour les tendances habituelles

En ce qui concerne les tendances temporelles, nous nous sommes basés sur les visites que les différents établissements ont reçues. Tout comme pour les tendances habituelles, nous nous sommes restreints à travailler sur les établissements catégorisés « Food/Restaurant ». Ainsi, seules les villes possédant au moins un restaurant de ce type ont été pris en compte, ce qui y a réduit la quantité de nos données comme le montre le Tableau 5. Données pour les tendances temporelles

	Données de base	Données utilisées
Villes	1203	838
Établissements	192 609	73 356
Visites	161 950	73 356

Tableau 5 Données pour les tendances temporelles

Tout d'abord, nous avons tenté d'appliquer l'algorithme de fouille de données décrit dans la section précédente (4.2.1). Afin de vérifier si les résultats seraient pertinents, nous avons souhaité appliquer l'algorithme pour l'État possédant le plus de restaurants. Ainsi, nous avons

pu obtenir le graphique de la Figure 11 qui nous a permis de conclure qu'il s'agissait de « AZ », c'est-à-dire l'état « Arizona ».

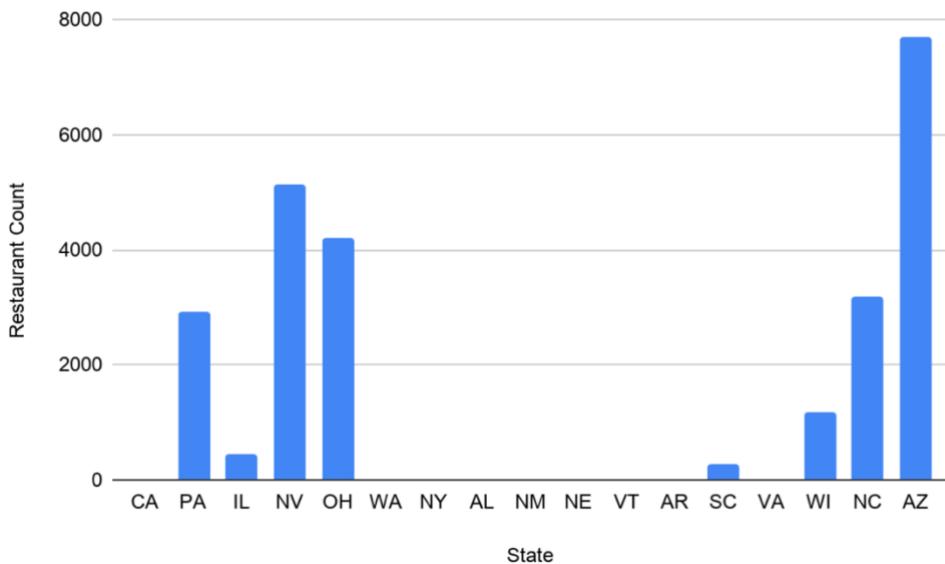


Figure 11 Nombre de restaurants par État

Par la suite, nous avons souhaité faire des tests sur un petit échantillon de données donc nous n'avons pris que 1 505 avis, plus précisément tous les avis de la dernière année disponible (2018). Avec ses avis, nous avons pu obtenir une liste de restaurants. Ainsi, pour chaque mois de l'année, nous avons lancé Apriori sur les catégories ces restaurants.

Cependant, les résultats obtenus ne nous ont pas semblés cohérents et ils étaient difficiles à vérifier. Chaque utilisateur présente des caractéristiques différentes et dans la plupart des cas, nous n'avons pu obtenir pas plus d'un itemset. Nous avons donc débuté notre travail d'extraction en manipulant les dataframes.

Bien que les consommateurs, appelés « les utilisateurs » dans notre cas, ayant des caractéristiques communes (âge, région, etc.) peuvent présenter des similarités en matière de goût, mais ce n'est pas toujours le cas. Souhaitant offrir à nos utilisateurs une expérience unique et personnalisée, nous avons décidé d'extraire des patterns en se basant sur leurs habitudes. Par ailleurs, la recommandation basée sur la liste d'amis de l'utilisateur sera traitée plus loin (voir section 6.3.2.2)

N'ayant pas les informations concernant les visites (Checkin) des utilisateurs (User), nous nous sommes basés sur les avis (Review) laissés par chacun d'eux. Grâce à ces avis, pour chaque utilisateur, nous avons pu obtenir les éléments suivants :

- Une liste des restaurants sur lesquels l'utilisateur a donné son avis, qui sont potentiellement ceux qu'il aurait visité. Cette liste est triée selon le nombre de visites ainsi que les notes, sous forme de nombre d'étoiles, qu'il a attribué.
- Une liste des catégories de restaurants qu'il a l'habitude de fréquenter. Cette liste est triée du plus fréquent au moins fréquent.

- Une liste des attributs de restaurants qu'il a l'habitude de fréquenter. Cette liste est triée également du plus fréquent au moins fréquent.

Par ailleurs, il existe toujours des utilisateurs, plus aventureux, qui aiment tester de nouvelles ou juste suivre la tendance. Pour répondre aux attentes de ces derniers, il nous est évident d'extraire également des tendances temporelles, par ville, en se basant maintenant sur les visites (Check-in). Pour chaque ville, nous avons donc recensé par mois, par jour de la semaine, par intervalle d'heures :

- La liste des restaurants visités triée par le nombre de visites
- La liste des catégories des restaurants visités triée également par le nombre de visites. Nous avions au total 975 catégories mais en se basant sur la Figure 12, nous avons éliminé les catégories « trop commun » qui sont présents dans les catégories d'énormément de restaurants et qui ne seront pas pertinents pour la suite, tels que « Restaurants », « Shopping » et « Food ».

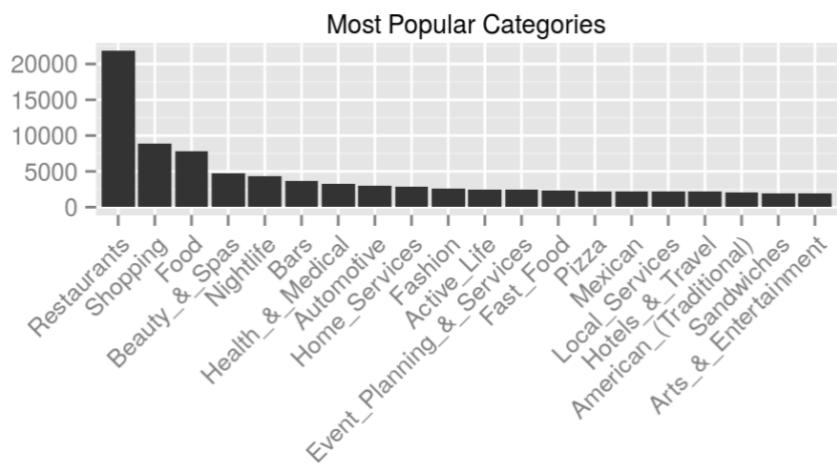


Figure 12 Nombre d'établissements par catégorie

- La liste des attributs avec leur valeur des restaurants visités trié par la fréquence d'apparition du couple attribut-valeur. Nous avons au total 38 attributs distinctes dont 6 possédants des listes d'attributs. La liste de tous les attributs et valeurs possible est disponible en annexe (Annexe 6). En effet, un attribut peut prendre différents types de valeurs que nous allons présenter un peu plus loin.

Pour chaque ville, nous pouvons donc rechercher les tendances très généralement avec la colonne des mois ou au contraire, nous pouvons aller en détails en recherchant pour un intervalle d'heure donnée un jour de la semaine d'un mois précis. Nous avons donc 2 035 façons de rechercher (Figure 13).

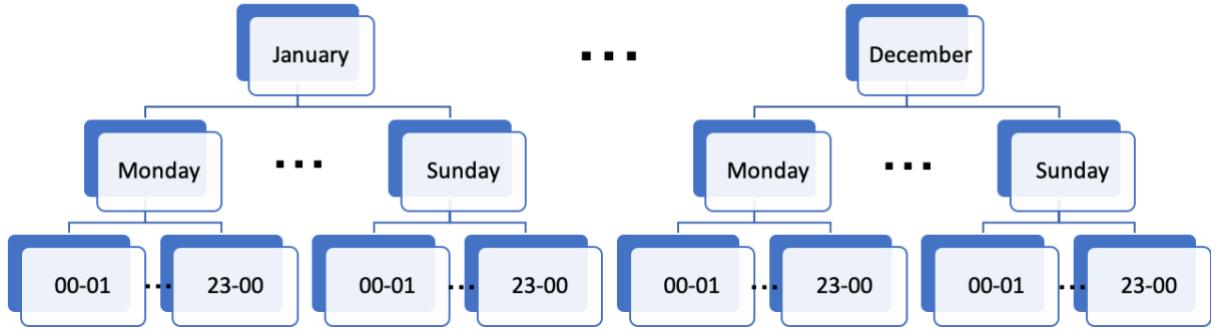


Figure 13 Toutes les recherches de tendances temporelles possibles

À la suite de ces extractions de tendances, nous avons pu obtenir le nombre de visites par heure de chaque restaurant. Cela nous a également permis d'ajouter une nouvelle fonctionnalité pour l'utilisateur : les heures d'affluences pour ces restaurants, lorsque ces dernières sont disponibles, comme nous pouvons le voir sur la Figure 14. Ces heures d'affluences seront affichées dans les détails de chaque établissement sur l'interface utilisateur (voir Chapitre 7). Ensuite, le nombre de visites par mois permet à l'utilisateur d'avoir la visibilité si restaurant est bon à être fréquenter durant le mois ou saison courant. Et enfin, avoir les visites par années nous permet tout simplement de visualiser l'évolution d'un établissement.

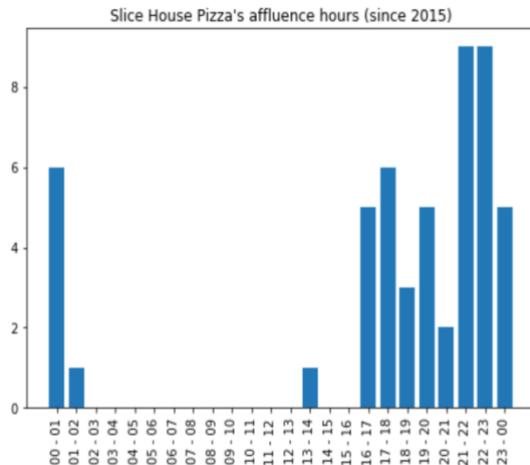


Figure 14 Exemple d'affichage des heures d'affluences

La manipulation des attributs que cela soit dans le cas des habitudes des utilisateurs ou dans le cas des habitudes temporelles, est une tâche difficile. La colonne « attributes » représentant les attributs dans chaque business est sous forme de dictionnaire, c'est-à-dire sous forme de clé-valeur. Les clés qui sont les appellations des différents attributs sont des chaînes de caractères, tandis que les valeurs sont de différents types :

- Booléen
- Textuel
- Numérique
- Dictionnaire

### Valeurs booléennes

Les valeurs booléennes ne sont que de deux types : True (Vrai) ou False (Faux).

### Valeurs textuelles

Pour les valeurs textuelles, nous en avons aussi de plusieurs types :

- Du texte entre guillemets : par exemple, « casual »,
- Du texte avec la lettre « u » devant un texte entre guillemets : par exemple « u'casual' » que nous avons rendu similaire au premier type cité ci-dessus en remplaçant la lettre « u » par un caractère vide
- Et du texte avec la valeur « None » qui correspondrait également aux textes avec la valeur « no » ou « none ».

### Valeurs numériques

En ce qui concerne les valeurs numériques, elles sont discrètes. Ces dernières, souvent pour un même attribut qui est le rang de l'établissement « RestaurantsPriceRange2 », varient de 1 à 4.

### Valeurs sous forme de dictionnaires

Lorsque qu'un élément du dictionnaire est un dictionnaire, il est appelé un objet imbriqué (nested object en anglais). La manipulation des données devient complexe dans ce cas.

## 4.4. Tests et certifications de la solution

Afin de tester notre solution, nous avons effectué utiliser différents échantillons de données : les 100 premières données et les 100 dernières données. Ensuite nous avons prélevé 25% des données, parmi ces échantillons de 100 valeurs et nous avons vérifié si les résultats étaient plus ou moins cohérents.

De plus, nous pouvons tester et certifier notre solution en combinant les différentes parties de celle-ci : les tendances habituelles et temporelles. Dans la logique, les utilisateurs contribuent aux tendances temporelles. Nous pouvons nous attendre à ce que les goûts de plus de 75% d'utilisateurs correspondent aux tendances temporelles.

De ce fait, pour les catégories et attributs des restaurants, nous avons réalisé des tests sur les données pour Arizona (AZ), Nevada, (NV) et Ohio (OH), les 3 Etats comportant le plus d'établissements comme montrer dans la section précédente (Figure 11).

Prenons l'exemple de Arizona pour expliquer le procédé. Nous commençons par récolter toutes les catégories des restaurants qui ont reçu plus de 2 500 visites, de l'Etat en question, dans une liste (Figure 15). Il y avait au total, 9 restaurants au total.

```
Categories list of top restaurant in AZ :  
- American (New)  
- Mexican  
- Bakeries  
- Nightlife  
- Bars  
- Coffee & Tea  
- Breakfast & Brunch  
- Sandwiches  
- American (Traditional)  
- Cafes  
- Food Trucks  
- Farmers Market  
- Burgers  
- Specialty Food  
- Breweries  
- Pizza  
- Pubs  
- Cheese Shops  
- Sushi Bars  
- Desserts  
- Poke  
- Soup  
- Seafood  
- Hawaiian  
- Caterers  
- Event Planning & Services  
- Barbeque  
- Salad  
- Sports Bars  
- Gastropubs
```

Figure 15 Top populaires catégories (Arizona)

Par la suite, nous avons récupéré tous les utilisateurs qui ont laissé leur avis sur les 9 restaurants que nous avons trouvé juste avant. Puis nous avons vérifié (Figure 16), pour chaque utilisateur, si sa liste de catégories préférés contient au moins un élément de la liste précédente générée précédemment.

Entrée [50]:	<pre> 1 check = df_az.top_fav_categorie.apply(lambda x : any(item for item in cat_list if item in x)) 2 df_az['check'] = check 3 df_az </pre>		
Out[50]:	user_id	top_fav_categorie	check
	0 bLbSNKLggFnqwNNzzq-ljw	Cafes, Meat Shops, Restaurants, Event Planning & Services, Venues & Event Spaces, Food, Specialty FoodWholesale Stores, Food, Grocery, Department Stores, Shopping, Fashion	True
	1 PcvbBOCOcs6_suRDH7TSTg	Food, Coffee & Tea, Coffee RoasteriesIndian, Food, Food Trucks, Fast Food, Mexican, Restaurants	True
	4 ic-tyi1jEIL_umxZVh8KNA	Restaurants, Thai, Noodles, Vietnamese, Japanese, Asian Fusion	False
	7 DK57YibC5ShBmqQI97CKog	Ice Cream & Frozen Yogurt, Restaurants, Fast Food, Burgers, FoodCustom Cakes, Cupcakes, Food, Bakeries	True
	8 3nluSCZk5f_2WWYMLN7h3w	Food, Restaurants, Shopping, Arts & Entertainment, Coffee & Tea, Art GalleriesDeli's, Restaurants, Sandwiches, Fast Food	True
	...	...	...
292591	O-gBVPO6CIN_ynVHMcl5Zg	Restaurants, Chinese	False
292592	m53bM64d0w-9YmTUCANIMQ	Greek, Restaurants, MediterraneanRestaurants, Pizza	True
292598	m5gS7wocsXgrglAblSj2GQ	Restaurants, Barbeque, Juice Bars & Smoothies, Food, Salad, Sandwiches	True
292600	NzvPf7IrJ7T8WnwjyR68oA	Chinese, Restaurants	False
292612	XaXPMxGq4UamzgLxDL_wug	Mexican, RestaurantsRestaurants, Sandwiches, Canadian (New), Seafood, Steakhouses	True

96825 rows × 3 columns

Figure 16 Correspondances tendances et préférences d'utilisateur (Arizona)

Et enfin, on vérifie le pourcentage de différence. Ce pourcentage est de 22% environ pour Arizona, comme c'est visible sur la Figure 17. Nous avons alors au moins 78% des utilisateurs suivent les tendances, 3% de plus qu'espérer.

```

Entrée [51]: 1 df = df_az.groupby('check')['check'].count().reset_index(name='count_check')
2
3 number_true = df[df.check==True]['count_check'].iloc[0]
4 number_false = df[df.check==False]['count_check'].iloc[0]
5
6 print('{}% of users trends are not matching to temprel trends'
7     .format(int(number_false/number_true *100)))

```

22% of users trends are not matching to temprel trends

Figure 17 Pourcentage de correspondances

Par ailleurs, pour s'assurer de la cohérence des horaires d'affluences, nous avons vérifié si l'intervalle de ces horaires étaient bien l'intervalle d'ouverture de chaque établissement (Figure 18).

```

QXAEGFB4oINsVuTFxEYKFQ tested -> ok
gnKjwl_1w79qoiV3IC_xQQ tested -> ok
HhyxOkGAM07SRYtlQ4wMFQ tested -> ok
1Dfx3zM-rW4n-31KeC8sJg tested -> ok
fweCYi8FmbJXHCqLnwuk8w tested -> ok
-K4gAv8_vjx8-2BxkVeRkA tested -> ok
PZ-LZzSlhSe9utkqYU8pFg tested -> ok
1RHY4K3BD22FK7Cfftn8Mg tested -> ok
tstimHoMcYbkSC4eBA1wEq tested -> ok
C9oCPomVP0mtKa8z99E3gg tested -> ok
NDuUMJfrWk52RA-H-0trpA tested -> ok
SP_YXIEwkFPPL_9anCYmpQ tested -> ok
irft4YkdNsww4DNf_Aftew tested -> ok
BvYU3jvGd0TJ7IyZdfiN2Q tested -> ok
NBn4hgfGtNz91k3VsDZlmw tested -> ok
e_EMySqP0uwlvZfd8mRaaQ tested -> ok
GW087Y-IqlL54_Ijx6hTYAQ tested -> ok

```

Figure 18 Résultat de test des heures d'affluences

# Chapitre 5 Géolocalisation

La géolocalisation est un élément primordial dans notre système de recommandation des businesses. Il nous permet d'assainir la recommandation finale en combinant avec les autres éléments de notre système. Sur ceux il est important de savoir comment nous avons mise en place le système de géolocalisation.

## 5.1. Analyse de la problématique

Traditionnellement, en tant que particulier, la géolocalisation est utilisée par une enseigne pour nous permettre de trouver le magasin le plus proche.

La géolocalisation est aussi utile aux entreprises pour nous soumettre de la publicité susceptible de nous intéresser suivant notre position géographique. Elle est devenue l'un des enjeux majeurs en entreprise et présente des avantages de gestion des déplacements, mais également en matière de sécurité. En effet, connaître la localisation d'un équipement, suivre le déplacement d'un véhicule ou encore identifier la position d'un outil permet d'optimiser le partage des ressources et d'améliorer les processus opérationnels. Les entreprises envoyant par exemple leurs salariés sur des sites à risque peuvent, en cas d'urgence, les retrouver rapidement, où qu'ils soient.

Si l'on ne s'intéresse maintenant qu'à notre projet, l'idée est de mettre en place un système automatique permettant de localiser les utilisateurs et leur proposer des businesses à proximité.

Disposant d'un jeu de données contenant plusieurs informations sur des businesses des villes différentes se trouvant aux USA. Disons par exemple qu'un utilisateur se trouvant à Las Vegas et désirant connaitre les businesses qui sont proches de son emplacement. Il est important de savoir sa position mais également faire une restriction que sur les businesses se trouvant dans le périmètre de Las Vegas, cela implique également la propriété de distance entre les businesses se trouvant à Las Vegas et la position de l'utilisateur.

## 5.2. État de l'art : études des solutions existantes

Pour le système de recommandation basé sur la géolocalisation, il est reparti en 3 parties : géolocalisation ou géocodage, calcul de la distance et recommander les businesses les plus proches. Nous avons exploité plusieurs solutions pour répondre à un système de recommandation de géolocalisation rassemblant ces 3 parties :

### 5.2.1. Géolocalisation - Géocodage :

La géolocalisation désigne de façon très large un ensemble de techniques employées pour localiser, sur un plan ou une carte numérique, un objet ou un individu en fonction de ses coordonnées géographiques. Nous avons plusieurs techniques de géolocalisation : par satellite, Wi-Fi, GSM, Bluetooth ou encore l'adresse IP, etc.

Le géocodage est le processus qui consiste à transformer la description d'un emplacement, telle qu'une paire de coordonnées, une adresse ou un nom de lieu, en emplacement à la surface du globe. Nous effectuons le géocodage entrant un nom de ville ou une adresse. Les emplacements obtenus en sortie sont des entités géographiques avec des

attributs comme longitude/latitude, que nous utilisons dans notre système de recommandation. Il est une des techniques de géolocalisation.

Afin de faire un choix de solution proposant la géolocalisation, nous avons étudié certaines solutions existantes ou apis comme : Google maps, Nominatim, Api ipstack.



Figure 19 Logo ipstack

L'outil ipstack [12 ipstack] offre une puissante API IP de géolocalisation en temps réel capable de rechercher des données de localisation précises et d'évaluer les menaces de sécurité provenant d'adresses IP risquées. Les résultats sont livrés en quelques millisecondes au format JSON (par défaut) ou XML. À l'aide de l'API ipstack, nous pourrons localiser automatiquement les utilisateurs au premier coup d'œil.

A screenshot of a web-based API results interface. At the top, there is a search bar containing the IP address "46.193.64.37" and a blue "LOOK UP" button. Below the search bar, the results are displayed in a JSON-like format:

```
ip: "46.193.64.37"
type: "ipv4"
continent_code: "EU"
continent_name: "Europe"
country_code: "FR"
country_name: "France"
region_code: "HDF"
region_name: "Hauts-de-France"
city: "Ronchin"
zip: 59155
# latitude: 50.602779388427734
# longitude: 3.0697200298309326
location: Object {}
time_zone: Object{}
```

The results include geographical coordinates (latitude and longitude), a city name ("Ronchin"), a zip code ("59155"), and two nested objects for location and time zone.

Figure 20 Exemple de résultat api d'ipstack

Il fournit un service d'API plus rapide, plus avancé et plus évolutif. Et, ils ont déjà plus de 100 000 clients. Le service de géolocalisation d'ipstack reste entièrement gratuit pour jusqu'à 10 000 demandes [13 utilisation de ipstack].

Nous avons effectué un test avec 65 entités géolocalisées en utilisant l'api de google maps et Nominatim, dont les résultats sont visibles dans le Tableau 6. Sur ce dernier, la colonne "Bon" signifie que l'API a trouvé la bonne géolocalisation, "Moyen" signifie que l'API n'est pas sûre et a retourné plusieurs résultats qu'il pense être juste. Enfin "Mauvais", signifie que l'API n'a pas trouvé de géolocalisation. On peut voir que Google maps ressort grand vainqueur, il arrive à géolocaliser la quasi-totalité des adresses. Google Maps propose non seulement le service de géocodage mais aussi celui de localisation automatique.

	Bon	Moyen	Mauvais
Google maps - Géocodage	63	1	1
Nominatim	14	31	20

Tableau 6 Résultat de test en utilisant l'API Google Maps et Nominatim

Nous avons également remarqué que Nominatim avait du mal à trouver les adresses exactes. Cela était dû au fait que les adresses pouvaient contenir des fautes d'orthographies ou trop d'informations, comme le nom de la résidence ou le nom du quartier.

Malheureusement, les services de Google Maps sont limités et payants. Du coup, nous avons décidé de garder ipstak pour la localisation automatique et Nominatim pour le géocodage, car il propose des services Open-Source et gratuit.

### 5.2.2. Calcul de distance :

Nous avons testé plusieurs formules de calcul de distance, à savoir :

- **La distance euclidienne (ou métrique euclidienne)** : est l'utilisation la plus courante de la distance. Elle est la distance "ordinaire" (c'est-à-dire en ligne droite) entre deux points dans l'espace euclidien.

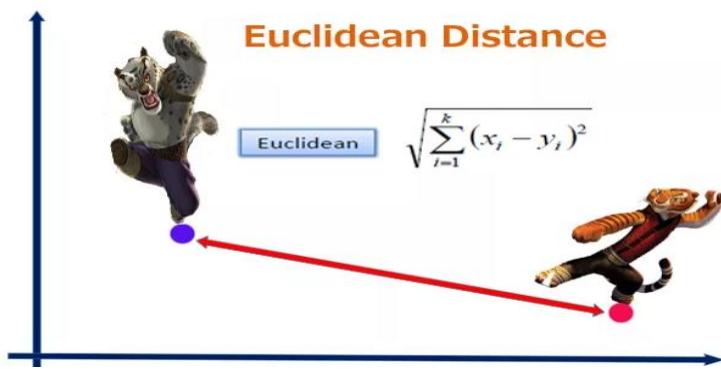


Figure 21 Illustration de la distance euclidienne

Avec cette distance, l'espace euclidien devient un espace métrique. La norme associée est appelée la norme euclidienne.

$$D_e = \left( \sum_{i=1}^n (p_i - q_i)^2 \right)^{1/2}$$

Figure 22 Formule distance euclidien

- ◆  $n$  = nombre de dimensions
- ◆  $p_i, q_i$  = points de données

L'euclidien est souvent la distance « par défaut » utilisée par exemple dans K-nearest neighbors ou K-means pour trouver les « k points les plus proches » d'un point

d'échantillonnage particulier. Un autre exemple marquant est le clustering hiérarchique, le clustering agglomératif (liaison complète et unique) où vous voulez trouver la distance entre les clusters.

- **La distance de Manhattan :** est la distance entre deux points parcourus par un taxi lorsqu'il se déplace dans une ville où les rues sont agencées selon un réseau ou quadrillage.

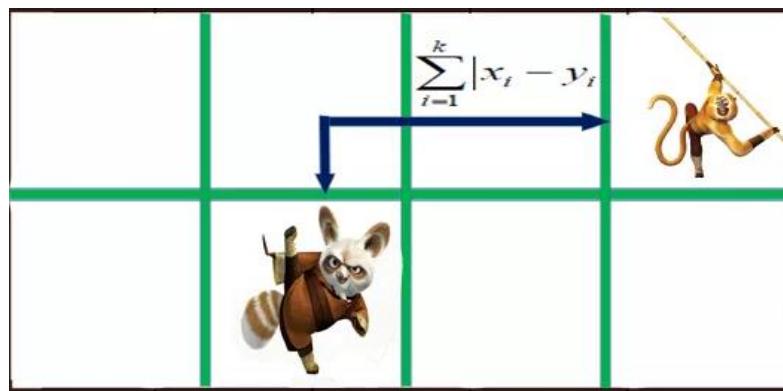


Figure 23 Illustration de la distance d'euclidienne

Elle est une métrique dans laquelle la distance entre deux points est la somme des différences absolues de leurs coordonnées cartésiennes. D'une manière simple, c'est la somme totale de la différence entre les coordonnées x et les coordonnées y.

$$D_m = \sum_{i=1}^n |p_i - q_i|$$

Figure 24 Formule distance Manhattan

- ◆  $n$  = nombre de données
- ◆  $p_i, q_i$  = points de données

Supposons que nous ayons deux points A et B si nous voulons trouver la distance de Manhattan entre eux, juste nous avons, pour résumer, la variation absolue de l'axe x et de l'axe y signifie que nous devons trouver comment ces deux points A et B sont variant dans les axes X et Y. D'une manière plus mathématique, dire la distance de Manhattan entre deux points mesurés le long des axes à angle droit.

- **La distance haversine :** calcule la distance la plus courte entre deux points sur une sphère en utilisant leurs latitudes et longitudes mesurées le long de la surface.

$$d_{\text{haversine}} = 2 \cdot \sin^{-1} \left( \sqrt{\alpha} \right) \cdot R_{\text{Earth}}$$

$$\alpha = \sin^2 \left( \frac{\Delta \text{Lat}}{2} \right) + \cos(\text{Lat}_1) \cdot \cos(\text{Lat}_2) \cdot \sin^2 \left( \frac{\Delta \text{Lon}}{2} \right)$$

Figure 25 Formule haversine

- ◆  **$\Delta \text{Lat}$**  =  $\text{Lat1} - \text{Lat2}$  : coordonnées latitude du point 1 et 2
- ◆  **$\Delta \text{Lon}$**  =  $\text{Lon1} - \text{Lon2}$  : coordonnées longitude du point 1 et 2
- ◆  **$R_{\text{Earth}}$**  : rayon de terre en kilomètres

Haversine (Figure 25) est une formule très populaire et fréquemment utilisée lors du développement d'une application SIG (système d'information géographique) ou de l'analyse de chemins et de champs [14 formule haversine].

**En résumé :** Il n'y a pas de formule parfaite, car la forme réelle de la terre est trop complexe pour être exprimée par une formule. De plus, la forme de la terre change en raison des événements climatiques [15 nasa gov] et change également au fil du temps en raison de la rotation de la terre.

Pour notre projet, nous avons décidé d'utiliser la formule haversine car la terre étant presque sphérique, la formule haversine fournit une bonne approximation de la distance entre deux points de la surface de la terre, avec une erreur inférieure à 1% en moyenne [16 scikit-learn].

### 5.3. Solution proposée et sa mise en œuvre

Nous avons décidé de retenir comme solution l'algorithme k-Nearest Neighbors en utilisant les données de géolocalisation et la distance haversine. Donc dans cette section, nous allons développer chaque élément de notre algorithme.

Notre algorithme est divisé en 3 étapes :

- Récupération des coordonnées de géolocalisation
- Calcul de la distance – Haversine
- Obtention des businesses les plus proches

### 5.3.1. Récupération des coordonnées de géolocalisation

Pour la récupération des coordonnées de l'emplacement de l'utilisateur, nous avons mis en place deux possibilités :

#### A) Localisation automatique d'un utilisateur :

Nous récupérons automatiquement les coordonnées de l'utilisateur une fois connecté sur notre application avec l'Api ipstack (décris dans la section 5.2)

- Pour commencer, nous avons créé un compte sur ipstack afin d'avoir une clé de l'api gratuite.
- Ensuite nous effectuons une requête via l'adresse URL d'ipstack<sup>4</sup> avec la clé de l'api obtenue.
- Avec le module des requêtes, nous effectuons une requête HTTP pour obtenir la réponse retournée par l'URL et la convertir en tableau JSON (Figure 26). Nous extrayons ensuite que les champs : latitude, longitude et city.

```
{'ip': '212.195.70.113',
 'type': 'ipv4',
 'continent_code': 'EU',
 'continent_name': 'Europe',
 'country_code': 'FR',
 'country_name': 'France',
 'region_code': 'IDF',
 'region_name': 'Île-de-France',
 'city': 'Nanterre',
 'zip': '92000',
 'latitude': 48.890010833740234,
 'longitude': 2.1970200538635254,
 'location': {'geoname_id': 2990970,
 'capital': 'Paris',
 'languages': [{'code': 'fr', 'name': 'French', 'native': 'Français'}],
 'country_flag': 'http://assets.ipstack.com/flags/fr.svg',
 'country_flag_emoji': '🇫🇷',
 'country_flag_emoji_unicode': 'U+1F1EB U+1F1F7',
 'calling_code': '33',
 'is_eu': True}}
```

Figure 26 Code – résultat pour la localisation automatique

L'un des avantages d'ipstack est qu'il nous aide à détecter plus efficacement les menaces provenant d'adresses IP à risque. Quand une adresse indésirable visite notre application web, l'identification rapide d'ipstack aide à la protéger.

<sup>4</sup> [http://api.ipstack.com/check?access\\_key=YOUR-API-KEY-HERE](http://api.ipstack.com/check?access_key=YOUR-API-KEY-HERE)"

## B) Géocodage avec GeoPy et Nominatim :

Pour le géocodage, nous avons utilisé la librairie geopy qui nous fournit la fonctionnalité "Nominatim" qui est un géocodeur pour les données OpenStreetMap.

Son utilisation demande des exigences suivantes :

- Pas d'utilisation intensive (un maximum absolu de 1 requête par seconde).
- Fournir un référent HTTP ou un agent utilisateur valide identifiant l'application (les agents utilisateur en stock définis par les bibliothèques http ne le feront pas).

**Exemple :** Nominatim(user\_agent="Data\_Dive\_Test")

- Affichez clairement l'attribution comme convenant à votre support.
- Les données sont fournies sous la licence ODbL

```
geoCode_address("628 w craig Rd, Las Vegas NV 89032")
```

```
{'latitude': 36.2388451, 'longitude': -115.207537, 'city': 'Las Vegas'}
```

Figure 27 Exemple d'utilisation géocodage

### 5.3.2. Calcul de la distance - Haversine

Une fois les données de localisation de l'utilisateur récupérées, nous effectuons un calcul de distance avec celles-ci et les données de localisation des businesses se trouvant en base de données. Pour cela, nous utilisons la formule distance haversine (décrivit dans la section 5.2) en écrivant notre propre fonction qui prend en paramètre les coordonnées (latitude et longitude) de l'utilisateur et celle du business. L'implémentation de cela est proposée en annexe (Annexe 8).

### 5.3.3. Obtention des restaurants les plus proches

Pour obtenir les restaurants les plus proches :

- Nous récupérons les données de localisation en utilisant notre fonction de géolocalisation préparée ci-dessus (étape 1)
- Nous faisons également une restriction pour n'avoir que les données des businesses se trouvant dans la ville de l'utilisateur qui devient notre jeu de données d'entraînement.
- Ensuit nous calculons la distance entre les coordonnées de l'utilisateur (longitude/latitude) et les coordonnées de chaque businesse se trouvant dans notre donnée d'entraînement retourner à l'étape précédent.
- Une fois les distances calculées, nous trions les données en fonction de leur distance. Nous le faisons en gardant une trace de la distance pour chaque businesse dans l'ensemble de données en tant que tuple. Ensuite nous trions (dans l'ordre croissant) cette liste des tuples par la distance, puis nous récupérons les voisins.

L'implémentation de cela est proposée en annexe (

Annexe 7).

## 5.4. Tests et certifications de la solution

Nous testons la solution avec un exemple de l'adresse d'un utilisateur.

Exemple adresse : **628 w craig Rd, Las Vegas NV 89032**

- Géocodage de l'adresse :

```
geoCode_address("628 w craig Rd, Las Vegas NV 89032")
```

```
{'latitude': 36.2388451, 'longitude': -115.207537, 'city': 'Las Vegas'}
```

Figure 28 Exemple de géocodage

- Sur le Tableau 7, nous avons les 10 businesses, les plus proches de l'adresse de l'utilisateur triés par ordre croissant.

		name	address	city	latitude	longitude	stars	distance
0	Original Tommy's Hamburgers		7079 W Craig Rd	Las Vegas	36.239890	-115.248651	3.5	0.1423
1	Vapeco	7083 W Craig Rd, Ste 195		Las Vegas	36.239509	-115.248505	5.0	0.1591
2	At Your Service	7083 W Craig Rd, Ste 175		Las Vegas	36.239562	-115.248650	5.0	0.1642
3	Kneaders Bakery & Cafe		7100 W Craig Rd	Las Vegas	36.240983	-115.249473	3.5	0.2007
4	Silver State Smiles	7101 W Craig Rd, Ste 102		Las Vegas	36.240045	-115.249513	3.5	0.2067
5	Starbucks	7101 West Craig Road, 1		Las Vegas	36.240146	-115.249588	3.5	0.2104
6	Brother's Pizza	7121 West Craig Rd 101		Las Vegas	36.239486	-115.249321	3.0	0.2171
7	Sushi-ko		7101 W Craig Rd	Las Vegas	36.239954	-115.249611	3.5	0.2180
8	China Hot Wok	7121 W Craig Rd, Ste 112		Las Vegas	36.239263	-115.249271	4.0	0.2281
9	Nail Club	7121 Craig Rd, Ste 106		Las Vegas	36.239683	-115.249629	3.5	0.2306

Tableau 7 Résultat de la recherche recommandation business avec la géolocalisation

- La cartographie représentative des 10 business :
  - Marqueur rouge : position de l'utilisateur
  - Marqueur bleu : les 5 premiers restaurants les plus proches de l'utilisateur
  - Marqueur vert : les restaurants éloignés de l'emplacement de l'utilisateur

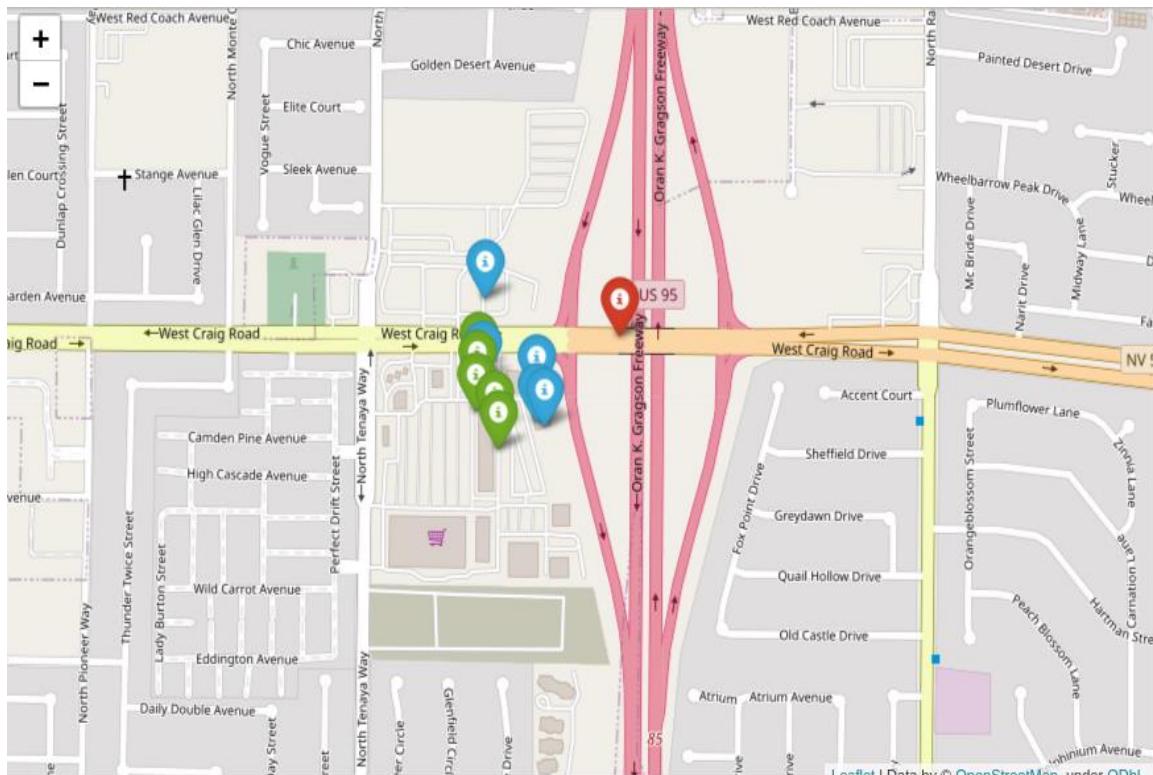


Figure 29 Cartographie représentative des business utilisant la géolocalisation

Nous pouvons donc confirmer le fonctionnement la recommandation des businesses les plus proches de la géolocalisation de l'utilisateur.

# Chapitre 6 Système de recommandations

Dans les chapitres précédents, nous avons expliqué les diverses étapes d'analyse et de pre-processing des données, nos implémentations de pattern/trends extraction, et l'implémentation de la géolocalisation dans un objectif de mise en production. Nous arrivons maintenant au chapitre le plus technique et complexe de notre projet "Smart Restaurant Recommendation" qui comme son nom l'indique, se veut de proposer un service intelligent et personnalisé de recommandations pour un utilisateur (nous avons décidé d'élargir les recommandations sur tous les businesses proposés par les données de YELP plutôt que de se restreindre aux restaurants). Les travaux réalisés précédemment vont servir à la conception, mais il reste encore beaucoup à faire.

## 6.1. Analyse de la problématique

Tout d'abord, nous devons déterminer parmi les données que nous possédons celles qui peuvent présenter une information sur les goûts d'un utilisateur. Il est aussi intéressant de pouvoir différencier des utilisateurs/businesses selon leurs caractéristiques propres, si l'on veut déterminer une similarité ou appliquer des techniques de clustering/classifications. Nous devons aussi respecter la position d'un utilisateur, et donc prendre en compte la position de chaque business pour garder une pertinence lors des recommandations.

Une fois toutes ces données organisées, il va falloir faire l'état de l'art des systèmes et technique de recommandations, puis déterminer les techniques qui sont implantables au vu de nos données. Du fait de nos recherches, nous savons qu'il existe une pléthora de techniques et que les systèmes en utilisent souvent une combinaison. Aussi, nous envisageons un système qui utilisera un assemblage de diverses techniques pour "scorer" les restaurants à proximité et proposera ceux ayant obtenu les meilleurs scores. Cependant, il y a aussi une problématique de gestion de Big Data : des techniques de recommandations classiques fonctionnent sur des petits jeux de données mais il n'est pas garanti qu'elles puissent être reproduisibles sur notre jeu de données qui est beaucoup plus volumineux. Notre mémoire n'est pas exceptionnelle, il faudra faire des concessions et des choix intelligents. Nous n'avons pas la possibilité et les connaissances pour utiliser un système distribué, aussi notre système devra intégralement fonctionner sur la mémoire de notre serveur.

Enfin, une fois le système mis au point, il faudra réfléchir à comment faire la liaison avec l'interface utilisateur. Pour cela, nous avons envisagé de créer un web service et de proposer les recommandations par résultats de requêtes.

## 6.2. État de l'art : études des solutions existantes

Il existe plusieurs catégories de techniques de recommandations intelligentes et on distingue en particulier les suivantes listées ci-dessous.

## Recommandation personnalisée

La recommandation personnalisée, pour laquelle on cherche à analyser le comportement de l'utilisateur (historique, temps passé, nombre de clics, ...) afin de prédire ce qu'il pourrait être susceptible d'acheter par la suite. On retrouve cette technique dans beaucoup de systèmes de recommandations très efficaces comme celui de Netflix ou Amazon (Figure 30), il nécessite cependant de traiter des flux de données conséquents (nécessité d'algorithme comme DGIM) et un tracking en temps réel des activités de chaque utilisateur. Bien que très intéressante, il nous est impossible d'implémenter cette technique car nous ne pouvons pas récolter ces données.



Figure 30 Exemple de recommandations personnalisées dans l'usage quotidien

## Recommandation objet

La recommandation objet, ou “content-based filtering”, où l'on va chercher à analyser le contenu (avec une notion d'attributs spécifique au type d'items recommandés) des items et les comparer aux antécédents connus de l'utilisateur. Le contenu de l'item est représenté généralement par un vecteur de descripteurs, sous la forme de mots-clés, et le profil de l'utilisateur comporte les préférences connues de l'utilisateur sur ces différents descripteurs. Les divers algorithmes déterminent ensuite les combinaisons “Utilisateur x Item” les plus pertinentes à recommander en calculant des scores de similarité pour chacune.

C'est une technique qui est généralement utilisée dans les systèmes hybrides pour pallier les faiblesses de la recommandation sociale qui, pour recommander un item, doit disposer d'un ensemble d'avis suffisamment large sur ce dernier, tandis que la recommandation objet ne s'intéresse qu'aux préférences connues de l'utilisateur sur le contenu des items et peut donc intégrer de nouveaux items facilement. Des études ont montré que les approches content-based, bien que moins populaires que les approches collaborative-based, peuvent être plus efficaces dans certains domaines d'applications de la vie courante (recommandation sur l'internet mobile [17 internet], recommandation personnalisée d'articles Forbes [18 Forbes], recommandation basée sur le click-behavior [19 click-behavior]).

Par ailleurs, certains aspects de la recommandation personnalisée ne peuvent être efficacement implémentés que si l'on considère le contenu d'un objet : diversification de la

liste de recommandation, justifications des recommandations, création de modèle utilisateur logique... Enfin, il y a de plus en plus de nouvelles sources de données pouvant servir à décrire le contenu d'un item, certaines sont structurées comme celles de DBpedia, semi-structurées comme celles de Wikipédia, et enfin il y a des données non-structurées comme les avis textes laissés par un utilisateur pour accompagner sa note et qui représentent une énorme mine d'informations pour décrire les items, ce qui ne peut qu'améliorer la performance des recommandations objet.

Les approches pures content-based ont cependant un inconvénient fréquent qui est que la qualité de l'item est rarement prise en compte : on peut donc recommander un item similaire à ceux que l'utilisateur connaît mais sans pouvoir garantir que l'item est du même niveau de qualité ou médiocre en comparaison. Comme expliqué précédemment, la disponibilité de nouvelles données, dont du contenu généré par l'utilisateur comme ses reviews, a permis l'arrivée de nouvelles approches content-based pour mieux prendre en compte ces informations et améliorer les prédictions.

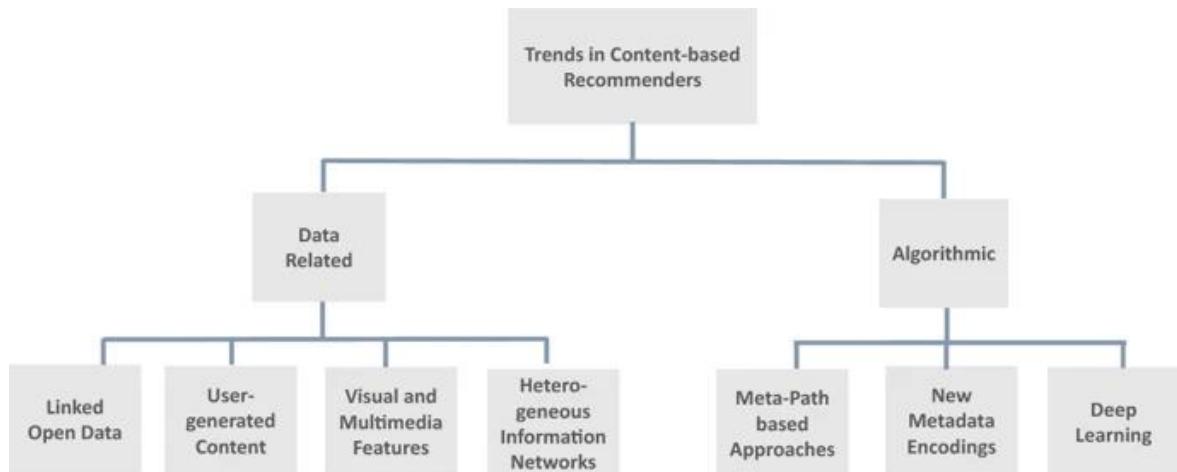


Figure 31 Approches content-based actuelles

Pour revenir au fonctionnement concret d'une méthode content-based, la plupart des techniques s'accordent et visent à construire un profil pour chaque item, pour cela, l'algorithme le plus populaire est la représentation TF-IDF qui représente dans l'espace les mots-clés par fréquence. Ensuite, le système construit un profil pour l'utilisateur basé sur un vecteur d'attributs d'items dont les poids sont en fonction de l'historique connu de l'utilisateur (un poids élevé pour un attribut indique une importance de cet attribut pour l'utilisateur). Pour calculer ces poids, c'est là où les techniques vont se diversifier : des approches simples utilisent des moyennes de chaque vecteur d'items connus par l'utilisateur, tandis que des approches plus poussées ont recours à des techniques de Machine Learning comme les "Bayesian Classifiers", l'analyse par clusters, les arbres de décisions, ou les réseaux de neurones artificiels pour estimer la probabilité qu'un utilisateur apprécie l'item. [20 content-based filtering]

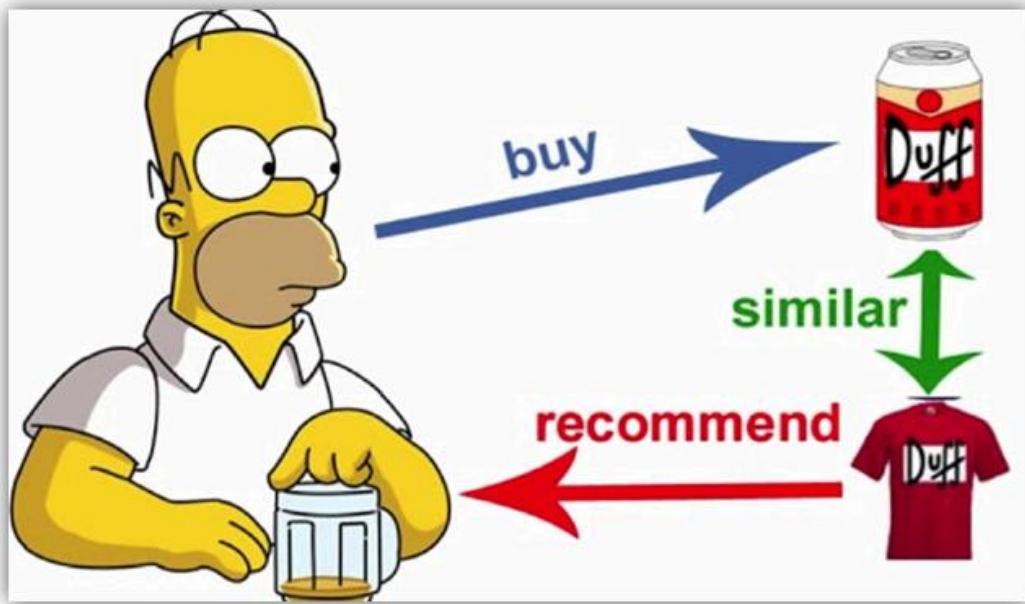


Figure 32 Illustration de content-based filtering

## Recommandation sociale

La recommandation sociale, ou “collaborative filtering”, qui quant à lui effectuera des propositions en se basant sur les choix que d’autres utilisateurs ont émis. On se base sur le profil d’un utilisateur, créé par l’inscription de ce dernier ou construit avec des données récoltées et on cherche à déterminer des groupes aux avis similaires. La supposition ici est que si un utilisateur A a la même opinion que l’utilisateur B sur un item, alors il est plus probable qu’il partage son opinion sur un deuxième item que celle d’un autre utilisateur choisi au hasard. On distingue plusieurs approches de collaborative filtering :

- **Memory-based** : cette approche utilise les “ratings” laissés par l’utilisateur pour calculer la similarité entre des paires d’utilisateurs ou d’objets. Pour cela, on construit une énorme matrice avec comme lignes/colonnes les différents utilisateurs/objets et à l’intérieur de la matrice les “ratings” (cette étape est logiquement accompagnée d’une normalisation des “ratings” pour mettre au même niveau les “nice-raters” et “tough-raters”), puis on utilise des techniques mathématiques comme la “cosine similarity” ou la “Pearson correlation similarity” pour obtenir les scores de similarités, la recommandation a ensuite lieu à l’aide des paires les plus semblables. On distingue ainsi deux façons d’aborder le problème :
  - ⇒ **User-Item** (Figure 33Error! Reference source not found.) : “les utilisateurs semblables à vous ont aussi aimé ...”. Pour ce cas d’utilisation, on donne plus d’importance au clustering des utilisateurs. On cherche des utilisateurs qui notent de la même manière que notre utilisateur. Cette méthode a l’inconvénient de nécessiter beaucoup de calculs car il y a généralement plus d’utilisateurs que d’items (idéalement) et les utilisateurs ont une probabilité de changer d’opinions assez élevée. Le modèle est donc instable et doit être reconstruit fréquemment.

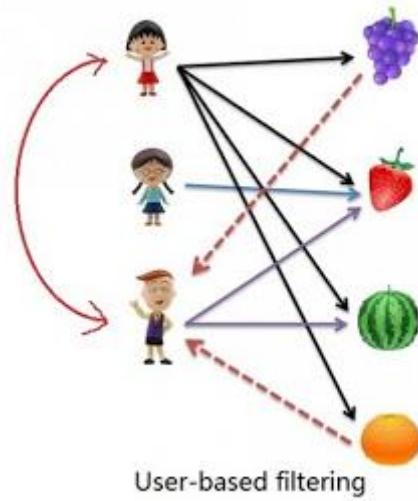


Figure 33 User-Item recommandation

⇒ Item-Item (Figure 34) : "les utilisateurs ayant aimé l'objet aimé par l'utilisateur ont aussi aimé ...". Ici, on privilégie le goût de l'utilisateur, on cherche une corrélation sur les items qu'il a aimé. C'est une approche qui a été inventée et popularisée par Amazon car dans le cas où le nombre d'utilisateurs est exponentiellement plus grand que le nombre d'items, les items auront tendance à avoir plus de "ratings" que les utilisateurs, donc la variance du "rating average" sera minime. Cela garantit une stabilité du modèle et permet donc d'éviter des "model rebuilds" coûteux en termes de puissance de calcul.

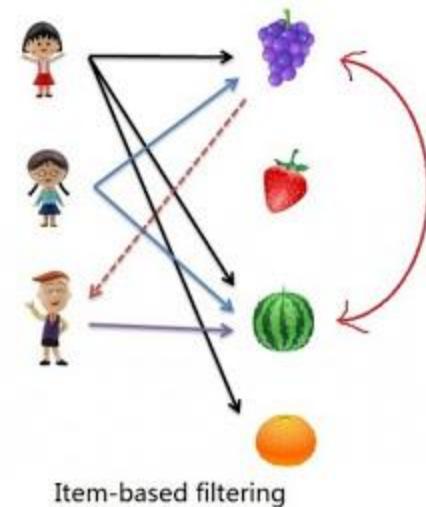


Figure 34 Item-Item recommandation

L'approche memory-based a l'avantage d'apporter une justification de la recommandation à l'utilisateur, permet d'incorporer de nouveaux utilisateurs assez facilement, et n'a pas besoin d'analyser le contenu de l'item. Elle souffre cependant sur le problème du "scaling" car plus les nombres d'items et d'utilisateurs grandissent, plus la matrice devient creuse ("sparse matrix") et la performance en pâtit.

- **Model-based** : les approches "model-based" se basent sur différentes techniques de Data Mining et Machine Learning pour prédire l'opinion d'un utilisateur sur des items non évalués. Il y a beaucoup d'algorithmes "model-based collaborative filtering": "Bayesian networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, multiple multiplicative factors, latent Dirichlet allocation and Markov decision process-based models."

Dans cette approche, les méthodes de réduction de dimensions sont souvent utilisées comme moyens pour améliorer la robustesse et la précision des approches "memory-based". Des méthodes comme SVD ou PCA servent à compresser la matrice utilisateur-item en une représentation à faible dimension en termes de facteurs latents. Au lieu d'avoir une énorme matrice avec beaucoup de "ratings" absents, on travaille sur une plus petite matrice à dimension réduite qui peut par la suite être utilisée par des algorithmes de voisinage ("neighborhood algorithms") basés sur l'utilisateur ou l'item. Cette méthode vient palier les inconvénients de l'approche "memory-based" et permet de travailler sur des matrices creuses et est beaucoup plus "scalable" avec des jeux de données plus gros.

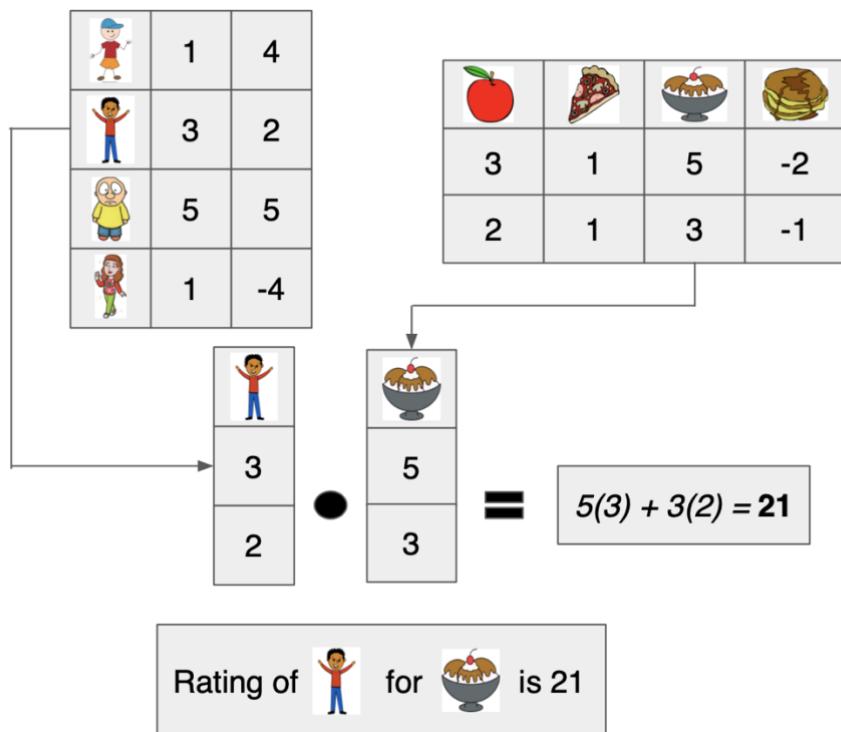


Figure 35 Exemple de matrix factorization

- **Deep Learning** : dans les dernières années, des techniques de “deep learning” ont émergées pour divers scénarios (“context-aware, sequence-aware, social tagging, etc”) mais son utilisation dans un scénario de recommandations collaboratives simple doit encore faire ses preuves.

Pour conclure, la recommandation collaborative est pertinente mais elle présente tout de même des inconvénients comme le “cold start” (besoin d'avoir suffisamment de ratings par utilisateur/item), la “scalability”, les “gray sheeps” (utilisateurs dont les avis sont tellement singuliers qu'il est impossible de trouver des utilisateurs suffisamment similaires) et enfin ont tendance à privilégier les items ayant le plus de “ratings” (“popularity bias”).

### Recommandation hybride

La recommandation hybride, qui est une combinaison de toutes les techniques de recommandations listées, ainsi que d'autres techniques particulières dont nous ne ferons pas usage. Par hybride, on entend que le système est capable de choisir parmi ces techniques les plus pertinentes selon l'utilisateur donné : il peut donc choisir de combiner le résultat de plusieurs techniques, en écarter certaines... la règle est qu'il faut que le résultat final soit le plus pertinent possible.

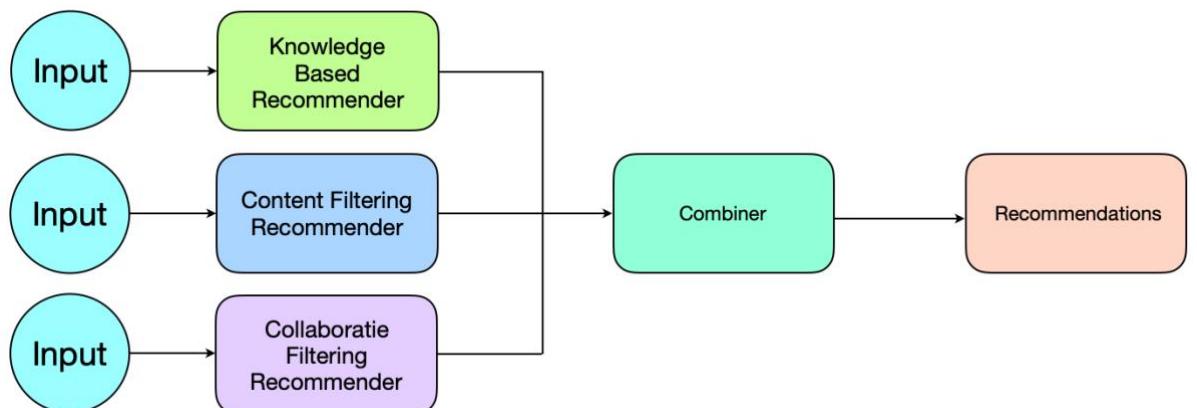


Figure 36 Exemple de recommandation hybride

En général, la plupart des systèmes de recommandations intelligents proposent une solution hybride regroupant ces techniques, ainsi que d'autres paramètres plus spécifiques.

### 6.3. Solution proposée et sa mise en œuvre

Comme l'histoire l'a prouvé (Amazon, Netflix, Linkedin, Pandora...), et par pure logique, les systèmes de recommandations hybrides sont de toute évidence les plus performants, nous allons donc en concevoir un avec comme piliers les recommandations objet et sociale.

Notre objet, restaurant ou “business”, possède divers attributs :

- Des avis laissés par les consommateurs, reviews, note + texte
- Des Tips (conseils) optionnellement laissés par les utilisateurs
- Des photos avec légendes, plus rares, mais sans informations de la provenance

- Des catégories (grill, Japonais, continental...)
- Des attributs (wifi, parking, niveau de prix...)
- Une position géographique (latitude, longitude, ville, quartier, état)

L'utilisateur, est quant à lui, assez anonymisé, on ne connaît que son nom et sa liste d'amis YELP. Il aurait été envisageable d'utiliser des réseaux sociaux comme Facebook pour obtenir d'avantage d'informations sur l'utilisateur mais nous avons pris la décision de ne pas explorer cette piste du fait de la complexité du système de recommandations que nous envisagions et du manque de temps.

Avec toutes ces données à notre disposition, nous avons suffisamment pour établir un système hybride utilisant les techniques suivantes :

- **Recommandation objet / content-based filtering** : nous avons suffisamment d'informations sur le contenu des businesses pour concevoir un algorithme content-based performant pour une recherche avec mots-clés ou une recommandation basée sur l'historique des restaurants aimés par l'utilisateur.
- **Recommandation sociale / collaborative filtering** : bien que nous disposions seulement des "ratings" laissés par les utilisateurs, l'approche collaborative est toujours abordable. Les approches "memory-based" et "model-based" seront implémentées mais le jeu de données est très large et nous serons limités par notre mémoire.
- **Recommandation par popularité / "trends"** : dans le cas d'un nouvel utilisateur sans historique, nous opterons sur une technique plus neutre (il est évidemment impossible de recommander parfaitement un utilisateur dont on ne connaît pas assez les goûts, c'est un échec que nous tolérons). Sinon, ces résultats serviront à consolider d'autres recommandations.
- **Recommandation par géolocalisation** : nous tenons à accorder une importance à la proximité du business recommandé. Là aussi, nous aurions idéalement utilisé un API comme Google Maps pour efficacement estimer la distance entre l'utilisateur et les businesses mais afin d'éviter de perdre du temps sur un détail technique et pour ne pas payer le service, nous avons implanté notre propre méthode d'estimation des distances.

### 6.3.1. Recommandation – Content-Based Filtering

On veut utiliser NLP pour faire la recommandation Content-Based. La similarité Item-Item est calculée pour ce type de problème. Par exemple, quand on regarde un film sur une plateforme de streaming, il y a souvent une section sur la page qui propose des films similaires basés sur la description du film actuel. Ces éléments ont généralement le même genre ou la même description. On parle donc de content-based.

La recommandation par Content-Based pourra prendre le profil de l'utilisateur et ses dernières évaluations pour générer les nouvelles recommandations ciblées pour lui. Cependant, cette approche présente le problème de cold start où on a aucune information sur un nouvel utilisateur. Par conséquent, la technique de Content-Based ne marchera pas s'il

Il y a un manque de données. Afin de surmonter le problème du cold-start, on a conçu la recommandation NLP basée sur le mot-clé saisi par le nouvel utilisateur plutôt que sur le profil d'utilisateur. Ainsi, le nouvel utilisateur arrive sur la page et saisit un mot arbitraire dans la barre de recherche. Prenez le mot-clé "tacos" (spécialité mexicaine) par exemple, on est capable de recommander une liste de restaurants Mexicain sans avoir à compter sur le champ de catégorie "Mexicain" dans les données. Le modèle serait capable de comprendre la nature du mot saisi et ensuite de répondre à la demande de recommandations par les businesses dont la catégorie est similaire ou inclut le mot clé cherché.

Afin de concevoir un système tel quel, on a décidé de fouiller les commentaires d'utilisateur pour chaque restaurant. Cela nous permet d'extraire les spécialités, par exemple de savoir les plats proposés dans le menu d'un restaurant. Cela en sachant que ce type d'information n'est pas disponible dans l'attribut "Catégorie" d'un business ou bien n'est pas aussi précis. Le modèle aura donc la capacité de fouiller et de comprendre les commentaires d'utilisateurs. On a choisi l'implémentation de Word2Vec pour ce fait.

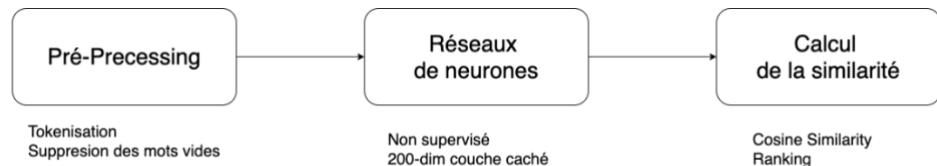


Figure 37 Démarche de vectorisation de reviews par Word2Vec

## Word2Vec

Word2Vec repose sur des réseaux de neurones à deux couches et cherche à apprendre les représentations vectorielles des mots composant un texte, de telle sorte que les mots qui partagent des contextes similaires soient représentés par des vecteurs numériques proches. Word2Vec prend à l'entrée une collection de texte et il produit un espace de vecteur de généralement 300 mots/descripteurs (200 dans notre cas). Chaque mot unique dans la collection est assigné au vecteur correspondant dans l'espace de vecteur.

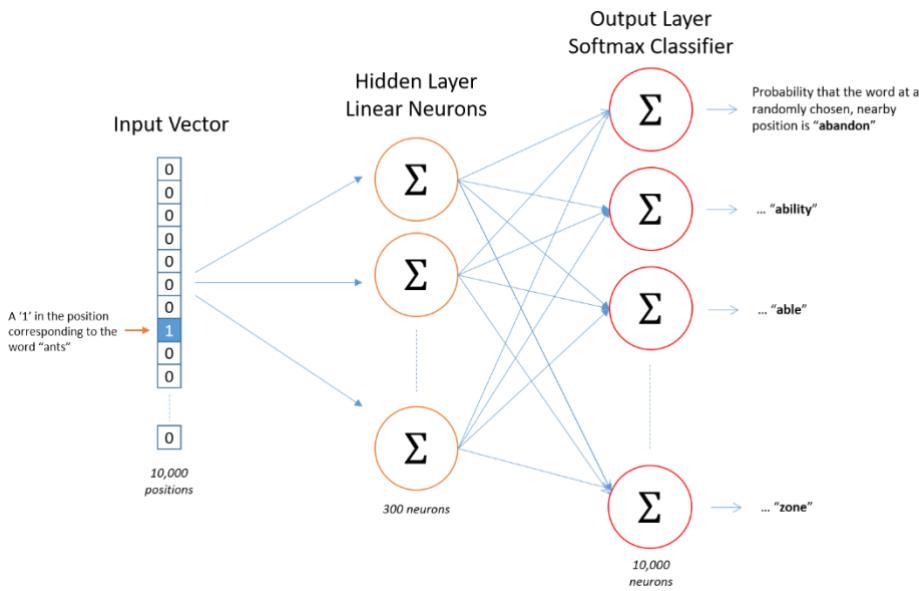


Figure 38 Architecture Word2Vec

Commençons par le pre-processing de données, on regroupe tous les commentaires écrits par les utilisateurs pour chaque business. Ensuite, nous effectuons la tokenisation et nous supprimons les “stop-words” (prépositions, articles...). Word2Vec combine chaque mot trouvé dans le commentaire (nettoyé) dans un vecteur de dimension 200 (200 features). Nous trouvons dans l'espace de vecteur que les mots similaires sont proches alors que les mots qui n'ont aucun rapport sont loin.

	<b>business_id</b>	<b>text</b>
0	--1UhMGODdWsrMastO9DZw	[last, review, mention, get, charge, extra, ,...]
1	--6MefnULPED_I942VcFNA	[decent, food, decent, price, ., standard, chi...
2	--7zmmkVg-IMGaXbuVd0SQ	[recent, tour, lake, norman, area, brewery, ,...]
3	--8LPVSo5i0Oo61X01sV9A	[dr., purcell, good, thorough, ., office, staf...
4	--9QQLMTbFzLJ_oT-ON3Xw	[ever, believe, check, time, ., always, 20, mi...

Figure 39 Les commentaires avant la transformation

	<b>business_id</b>	<b>text_vec</b>
0	--1UhMGODdWsrMastO9DZw	[0.45983207, -0.15746975, 0.42093047, 0.176013...]
1	--6MefnULPED_I942VcFNA	[0.50584954, -0.24407916, 0.3510201, 0.2930915...]
2	--7zmmkVg-IMGaXbuVd0SQ	[0.47147155, -0.09576023, 0.2647474, 0.4898245...]
3	--8LPVSo5i0Oo61X01sV9A	[0.94687724, -1.0266593, 0.50560087, 0.9978179...]
4	--9QQLMTbFzLJ_oT-ON3Xw	[0.9763589, -0.7084046, 0.7299864, 0.85709083,...]

Figure 40 Les commentaires après la transformation

Une fois, les mots transformés en vecteur, nous pouvons déterminer le taux de similarité entre les deux mots ou les deux documents en fonction de la similarité cosinus.

## Cosine Similarity

La similarité cosinus est le produit scalaire entre les deux vecteurs divisés par la production de la grandeur de ces deux vecteurs. Celle-ci calcule l'angle entre les deux vecteurs. Un petit angle signifie une similarité faible et un grand angle signifie une forte corrélation. La valeur peut varier de -1 à 1.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Figure 41 Formule de calcul de la similarité cosinus

## Recherche des mots similaires par le modèle de Word2Vec

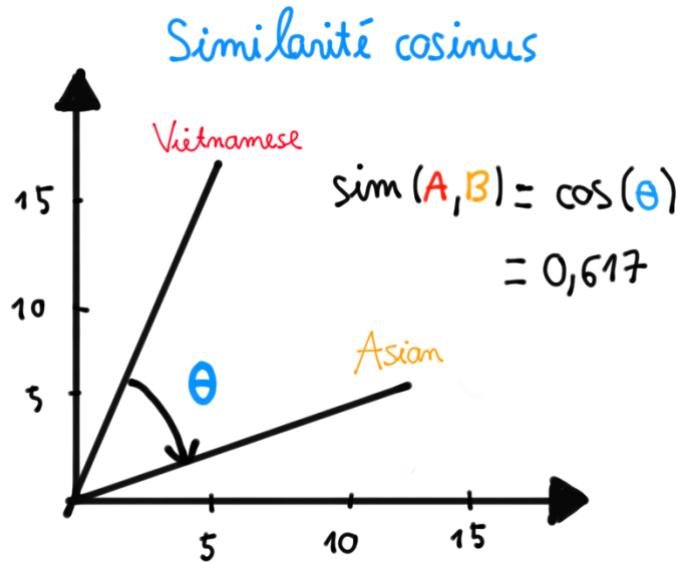


Figure 42 Exemple de la similarité cosinus entre les mots

Dans la Figure 43, nous essayons de chercher les cuisines similaires à la cuisine vietnamienne et nous pouvons voir que la plupart des réponses sont de nature asiatique, ce qui est très logique. Nous avons 3 exceptions de cuisines d'Amérique du sud qui possèdent quelques bases de la cuisine asiatique.

```
model.wv.most_similar("vietnamese")
```

```
[('chinese', 0.7128885388374329),  
 ('taiwanese', 0.7126409411430359),  
 ('cantonese', 0.7020187377929688),  
 ('viet', 0.687890887260437),  
 ('filipino', 0.662496030330658),  
 ('asian', 0.6172739267349243),  
 ('korean', 0.6125739812850952),  
 ('cuban', 0.6119341850280762),  
 ('ethiopian', 0.6098645925521851),  
 ('colombian', 0.6047247648239136)]
```

Figure 43 Exemple de recherche mot similaire

## Recommandation Mots-Clés

Dans la Figure 44, nous avons la recommandation pour le mot clé chicken cheese. Ici, deux facteurs sont pris en compte, nous voulons obtenir les restaurants qui proposent les plats avec du poulet ou du fromage, dans le meilleur des cas un plat qui a les deux ingrédients. Dans nos résultats, nous nous attendons à avoir les restaurants qui proposent, par exemple, des burgers dans leurs catégories puisque c'est assez connu que le burger contient du poulet avec le fromage cheddar.

```
c = keyword_recommend("chicken cheese", review_by_business, model, 10)  
  
d = business_details(business_df, c)  
  
d[["name", "categories", "city", "score"]]
```

	name	categories	city	score
0	Arby's	Sandwiches, Fast Food, Burgers, Restaurants	Amherst	0.691516
1	Swiss Chalet Rotisserie & Grill	American (Traditional), Restaurants, Barbeque,...	Mississauga	0.744961
2	KFC	Chicken Wings, Fast Food, Chicken Shop, Restau...	South Euclid	0.702790
3	Seaboot	Soul Food, Restaurants, Southern	Champaign	0.712854
4	Superpumper	Gas Stations, Automotive	Phoenix	0.691757
5	Nana's Soul Food Kitchen	Breakfast & Brunch, Restaurants, Cajun/Creole,...	Matthews	0.690423
6	Ollie's Pizza	Restaurants, Pizza	Canonsburg	0.758181
7	The Family Fry Guy	Street Vendors, Food	Calgary	0.706818
8	Ming Moon	Restaurants, Chinese	Cleveland	0.709589
9	Domino's Pizza	Pizza, Chicken Wings, Restaurants, Sandwiches	Kannapolis	0.719900

Figure 44 Exemple de recherche mot clé par Content Based recommandation

Nous pouvons voir dans le résultat qu'il y a différentes catégories de restaurants : on a quelques restaurants proposant des burgers comme nous l'attendions puis, nous avons des catégories de soul food qui sont des plats avec comme ingrédient principal le poulet. Les

autres résultats proposent des sandwiches, chicken wings, pizza, etc. Ces résultats sont évidents parce ce que ces plats sont composés soit de poulet soit de fromage.

### 6.3.2. Recommandation – Collaborative Filtering

Le véritable défi de la recommandation sociale est de trouver un moyen efficace, tout en restant pertinent, d'utiliser l'énorme jeu de données pour établir des propositions. Nous disposons d'une mémoire très limitée et ne pouvons évidemment pas rivaliser avec les moyens employés par des géants comme Amazon ou Netflix qui ont des milliers de machines dédiées à leurs calculs. L'implémentation des algorithmes est une étape à franchir mais nous avons souvent été confrontés à nos limitations de mémoire et avons dû improviser pour trouver des compromis, optimiser et reconstruire nos techniques de recommandations.

La technique de Collaborative Filtering vise à exploiter les informations des autres utilisateurs (ses items préférés avec ses évaluations correspondantes) complétées avec l'historique d'évaluation de l'utilisateur en question pour prédire l'évaluation sur une collection d'item que l'utilisateur ciblé n'a pas encore évalué. L'hypothèse derrière cette approche est que si l'utilisateur X avait le même avis que l'utilisateur Y sur une collection de items, X a la tendance d'avoir la même préférence que Y sur les autres items choisis de manière arbitrairement. Ceci fait la différence par rapport à la technique de Content Based dans le sens que l'item lui-même ne sera pas pris en compte dans la démarche de la recommandation.

#### 6.3.2.1. Collaborative Filtering – Matrix Factorization (Model Based)

Pour la partie Model Based, on utilise principalement la factorisation de matrices. C'est une méthode beaucoup utilisée et appréciée par les services qui s'occupent souvent de la recommandation pour un nombreux de l'utilisateur et des produits afin d'accélérer la recherche de recommandations de contenu pour les utilisateurs. Parmi les techniques de la factorisation de matrices on va découvrir les deux méthodes les plus connues : SVD (Singular Value Decomposition) et NMF (Non Negative Matrix Factorization)

##### SVD (Singular Value Decomposition)

SVD est une technique de factorisation de matrice permettant de réduire le nombre de features dans le jeu de données en compressant la dimension de  $M \times N$  ou  $M < N$ . Pourtant, dans le contexte du système de recommandation, on ne s'intéresse pas à la partie de la réduction dimensionnelle. Ce qui nous intéresse c'est de prédire la note d'évaluation sur les items que l'utilisateur n'a pas encore évalué. Autrement dit, on essaie de remplir les valeurs 0 dans la matrice d'user-item qui est très "sparse". D'ailleurs, la factorisation de matrice est effectuée sur la matrice d'user-item rating. Malgré qu'on ait des évaluations d'utilisateurs pour les businesses, la préférence des utilisateurs se cachent sous-jacent les données. Ces attributs ne sont pas visibles ou ne sont pas évidents dans les données en raison d'attributs ou de la tendance cachée. Par contre, ces facteurs cachés peuvent être détectés par le model-based de CF d'où la matrice de factorisation.

L'idée de SVD est de trouver 3 matrices factorisés depuis la matrice originale dans lesquelles la production de ces 3 matrices se produisent un résultat plus proche que la matrice originale que possible avec un minimum de perte. La formule de SVD est comme suit :

$$\mathcal{X} = \mathcal{U} \times \mathcal{S} \times \mathcal{V}_{(T)}$$

avec  $\mathcal{X}$  la matrice  $m \times n$

$\mathcal{U}$  la matrice orthogonale  $m \times m$

$\mathcal{S}$  la matrice diagonale

$\mathcal{V}_{(T)}$  la matrice orthogonale  $r \times n$

La matrice  $\mathcal{X}$  est factorisée à  $\mathcal{U}, \mathcal{S}$  et  $\mathcal{V}$

$\mathcal{U}$  représente le vecteur de features correspondant aux users dans l'espace de features cachés  
 $\mathcal{V}$  représente le vecteur de features correspondant à items dans l'espace de features cachés

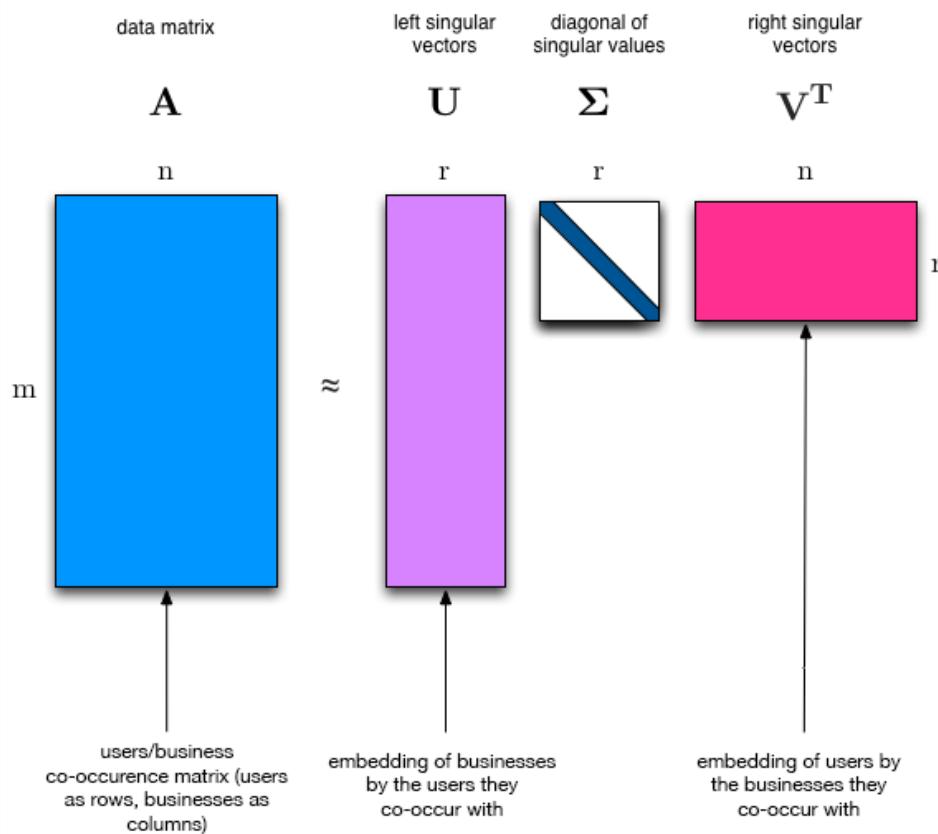


Figure 45 Représentation schématique pour Singular Value Decomposition

Finalement, nous faisons la prédiction par la production de  $\mathcal{U}, \mathcal{S}$  et  $\mathcal{V}_{(T)}$ .

En analysant les données de ratings, nous pouvons constater que la matrice de users-items entre user\_id et business\_id est vraiment sparse. Les valeurs dans la matrice correspondent aux ratings des utilisateurs pour les businesses. En réalité, nous avons rarement une ligne bien remplie par des valeurs, c'est-à-dire un utilisateur donne des ratings pour la plupart des businesses existants dans la base. La matrice de users-items de données de ratings de Yelp de 2016-2018 est comme sur la Figure 46.

```

1 for i, row in user_active_df.iterrows():
2     ratings_mat[row["user_id_matrix"], row["business_id_matrix"]] = row["stars"]

1 ratings_mat
<7024x90744 sparse matrix of type '<class 'numpy.float64'>'>
with 519838 stored elements in List of Lists format>

```

Figure 46 Matrice d'users-items de données de ratings

Nous pouvons voir que la matrice a une taille importante de 7 024x90 744 et que seuls 519 838 champs sont remplis. Pour faire la recommandation pour un utilisateur, il est nécessaire de connaître les ratings potentiellement donnés par l'utilisateur, c'est-à-dire de prédire la valeur des champs vides dans la matrice. On applique donc la technique de SVD pour la prédition de ratings de businesses qui n'ont pas encore été évalués. On choisit la librairie SVD de numpy [21 numpy svd] pour implémenter l'algorithme. La démarche est comme suit :

- Chercher le mean() de la matrice creuse
- Faire la soustraction de la matrice creuse par le mean
- Appliquer l'algorithme SVD sur la nouvelle matrice pour obtenir 3 matrices U, sigma et Vt
- Construire une nouvelle matrice prédite par la production de 3 matrices factorisées par SVD en rajoutant le mean dans la première étape

L'implémentation en Python se trouve en Annexe 9.

A noter que dans l'étape d'application de SVD, il y a un hyper paramètre correspondant au nombre de facteurs, pour lequel nous attribuons la valeur à 20 après le tuning par cross validation. Finalement, nous avons une matrice toute remplie par la production de 3 matrices factorisées de SVD. La Figure 47 représente la matrice après l'utilisation de SVD.

```

1 all_user_predicted_ratings.shape
(7024, 90744)

1 all_user_predicted_ratings
matrix([[ 2.80572478e-05, -3.06430879e-04, -7.62462615e-04, ...,
         -8.92782792e-04, -1.19342250e-03,  2.50393074e-04], ...,
         [-4.05617430e-04, -1.57598950e-03,  1.14617587e-03, ...,
          -6.23662393e-04, -1.35499455e-03, -4.97264856e-04], ...,
         [-7.10023843e-05, -2.94977877e-05,  4.15602000e-06, ...,
          2.98501478e-03, -5.45721588e-05, -1.07575741e-03], ...,
         ..., ...,
         [ 1.28020963e-04, -7.67523658e-05, -8.58576604e-04, ...,
          -2.37807115e-03, -8.00979523e-04,  9.76637589e-04], ...,
         [ 1.64032159e-03,  2.01158188e-02,  5.88522429e-04, ...,
          1.62551568e-03,  1.43704570e-04,  1.19472070e-03], ...,
         [ 4.72845026e-04,  2.85724191e-02, -6.24569761e-04, ...,
          3.34162209e-04,  1.90163201e-04,  3.64881264e-04]])
```

Figure 47 Matrice de prédition par la technique SVD

Pour la recommandation d'un utilisateur, il nous reste plus qu'à récupérer toutes les valeurs dans la ligne correspondant à l'utilisateur, puis trier ces valeurs en ordre décroissant avant de retourner les businesses correspondant aux meilleurs scores.

## NMF (Non Negative Matrix Factorization)

La NMF est une technique de réduction de dimension adaptée aux matrices creuses contenant des données positives, largement utilisée en data-mining, par exemple des occurrences ou dénombremens de mots, de pannes, etc. La méthode est donc plus adaptée à certaines situations que la SVD mais cela a un prix, la complexité algorithmique de la SVD est polynomiale de l'ordre du produit  $n \times p$  des dimensions de la matrice. La complexité de la NMF est un problème non-deterministic polynomial, l'existence d'un algorithme de complexité polynomiale est inconnue.

Le principe de la NMF est comme suit :

Soit  $X$  une matrice de taille  $M \times N$ , ne contenant que des valeurs non négatives et sans ligne ou colonne ne comportant que des 0.

La factorisation non négative de la matrice  $X$  est la recherche de deux matrices  $W$  (de taille  $M \times K$ ) et  $H$  (de taille  $K \times N$ ) telles que :

- Le rang  $K$  est plus petit que  $\min(M, N)$
- $W$  et  $H$  ne contiennent que des valeurs positives ou nulles
- $X \approx WH$ .

La dernière condition indique que le produit  $WH$  approxime bien la matrice  $X$ . La qualité de cette approximation est mesurée par une fonction de coût  $L(X, W, H)$ . Dans ce projet, nous décidons d'utiliser la norme de Frobenius pour mesurer la proximité entre matrices. La fonction de coût est donc définie par Figure 48:

$$\begin{aligned} L(\mathbf{X}, \mathbf{W}, \mathbf{H}) &= \text{tr}((\mathbf{X} - \mathbf{WH})(\mathbf{X} - \mathbf{WH})^t) \\ &= \sum_{m,n} (X_{m,n} - (WH)_{m,n})^2 \end{aligned}$$

Figure 48 Fonction de coût

où  $X_{m,n}$  est l'élément de la  $m^{\text{ème}}$  ligne et  $n^{\text{ème}}$  colonne de  $X$ .

Pour mettre cette technique en pratique, on utilise la librairie NMF de Scikit-Learn. On prend la même matrice creuse qu'on utilise dans SVD. La démarche est comme ci-dessus :

- Appliquer l'algorithme NMF sur la matrice creuse pour obtenir 2 matrices  $W$  (features) et  $H$  (features weight).
- Construire une nouvelle matrice prédite par la production de 2 matrices factorisées par NMF

L'implémentation en Python se trouve en Annexe 10.

## Comparaison de performance entre SVD et NMF

On teste la performance de SVD contre le NMF, on prend le même nombre de facteur de matrice factorisation, mis à 20 et le test s'effectuera sur le même user identifié « AyjqBovADgbskmLrIBOMIQ ». On récupère les valeurs dans le vecteur d'user-businesses, trié par la valeur (score) et enfin comparable l'ordre d'évaluation sur les businesses entre les deux algorithmes.

Le résultatat de la prédiction de NMF est décrit dans Figure 49.

	<b>nmf_score</b>	<b>business_id</b>
<b>0</b>	2.126611	eaNenRk_liZBERFFLCXqqQ
<b>1</b>	2.026556	uBdYMY6a6A7FyxzTSwOiDg
<b>2</b>	2.017999	iFCz-xI7CV98fcab4Chh3g
<b>3</b>	1.987034	L1-1P3acJc4gEFvWwjXcNQ
<b>4</b>	1.985794	i9D9xPBV0gR1Ja9kbY4NCw
<b>5</b>	1.821615	JPfi_QJAaRzmfh5aOyFEw
<b>6</b>	1.754412	YILyHegzhy1vlc_LNVfObw
<b>7</b>	1.672821	2iTsrqUsPGRH1li1WVRvKQ
<b>8</b>	1.654355	jaSowNITPRRCYpPb3_pjdA
<b>9</b>	1.590757	igHYkXZMLAc9UdV5VnR_AA
<b>10</b>	1.501722	IIDgEB1HnKAq03H1fpjzLQ
<b>11</b>	1.490843	4JNXUY8wbaaDmk3BPzlWw
<b>12</b>	1.488517	CoyeXg8FBsS_d20QzNly-A
<b>13</b>	1.450209	beuVp5CZxCdNvQIIPBS2rw
<b>14</b>	1.351684	33Tr0eRki1Yamzleu4GMdw
<b>15</b>	1.330192	Ec9CBmL3285XkeHaNp-bSQ

Figure 49 Top 20 de recommandation par la NMF

Le résultatat de SVD est décrit dans Figure 50.

	<b>svd_score</b>	<b>business_id</b>
<b>0</b>	2.643354	eaNenRk_liZBERFFLCXqqQ
<b>1</b>	2.426658	uBdYMY6a6A7FyxzTSwOiDg
<b>2</b>	2.320469	igHYkXZMLAc9UdV5VnR_AA
<b>3</b>	2.284643	i9D9xPBV0gR1Ja9kbY4NCw
<b>4</b>	2.250377	JPfi__QJAaRzmfh5aOyFEw
<b>5</b>	2.116713	YILyHegzhy1vlc_LNVfObw
<b>6</b>	2.079236	2iTsRqUsPGRH1li1WVRvKQ
<b>7</b>	1.961197	iFCz-xI7CV98fcaB4Chh3g
<b>8</b>	1.796299	L1-1P3acJc4gEFvWwjXcNQ
<b>9</b>	1.771679	4JNXUYY8wbaaDmk3BPzlWw
<b>10</b>	1.711478	umXvdus9LbC6oxtLdXelFQ
<b>11</b>	1.694789	faPVqws-x-5k2CQKDNTHxw
<b>12</b>	1.663054	beuVp5CZxCdNvQIIPBS2rw
<b>13</b>	1.608685	Ec9CBmL3285XkeHaNp-bSQ
<b>14</b>	1.600479	IIDgEB1HnKAq03H1fpjzLQ
<b>15</b>	1.485448	jaSowNITPRRCYpPb3_pjdA

Figure 50 Top 20 de recommandation par la SVD

On observe que malgré que les résultats des deux algorithmes ne soient pas exactement identiques, la recommandation sur le top 20 est considérablement similaire entre les deux, spécialement ils se mettent d'accord sur l'importance de deux premiers businesses, il y a un décalage dans l'ordre de recommandation pour certaines businesses mais le résultat dans l'ensemble est vraiment similaire, surtout au niveau de business en commun dans le top. Pour la suite de ce projet, on décide d'utiliser SVD

#### 6.3.2.2. Collaborative Filtering – Friendlist Based (Memory)

Puisque nous disposons de la liste d'amis d'un utilisateur, nous sommes partis de la supposition suivante : nous avons d'avantage confiance en les recommandations de nos amis que les recommandations d'étrangers. Nous avons donc utilisé l'approche "memory-based user-item" pour déterminer parmi les amis d'un utilisateur ceux dont il est le plus proche et émettre des recommandations.

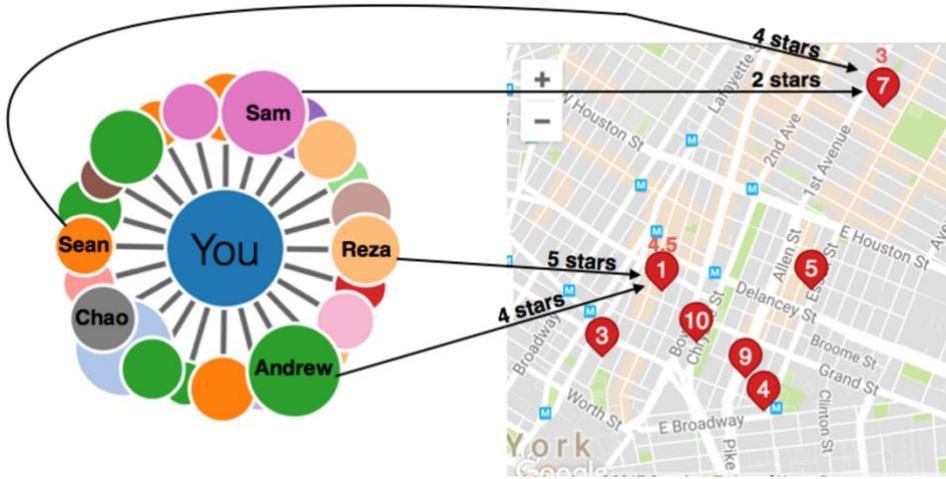


Figure 51 Recommandations basées sur les amis YELP

Étapes :

- **Data gathering / pre-processing** : on commence par recueillir toutes les reviews de l'utilisateur et de ses amis, en faisant une jointure avec les informations des businesses correspondants, puis on normalise les notes de chaque utilisateur en soustrayant la moyenne de leurs notes à chacune de leur note (« mean-zero »).
- **Pivot table** : on crée la matrice user x item et on remplit les valeurs manquantes par la moyenne de chaque business (choix arbitraire, on aurait pu utiliser la moyenne des utilisateurs).

```

final = pd.pivot_table(Rating_avg,values='adg_rating',index='user_id',columns='business_id')
nrows = final.shape[0]
# Replacing NaN by business Average
final_business = final.fillna(final.mean(axis=0))
final_business.head()

```

Out[13]:

business_id	-95mbLJsa0CxXhpaNL4LvA	lWsoxH7mLJTpU5MmWY4w	wCtRhzwJ40Z4F8mmg7kWg	OCB7YB1qRSWLQvMbHw3Fmw	0EpnzIpeFvh!
user_id					
3uq5Qyl28UF-UGD9LKfAmw	1.120301	0.857143	1.120301	1.120301	
NPLCLcsJbxpwjphfOoZTpQ	1.120301	0.857143	1.120301	1.120301	
ae4odTLq2CMvrKKUpVFZSw	1.120301	0.857143	1.120301	1.120301	
cQrn1xjgVBhjAc0Jttw48aw	1.120301	0.857143	1.120301	1.120301	
r9CIB1N837hn2mgqXPEn2A	1.120301	0.857143	1.120301	1.120301	

5 rows x 181 columns

Figure 52 Matrice User x Item en mémoire

- **Finding similar users** : on détermine ensuite les utilisateurs les plus similaires à l'utilisateur en calculant la “cosine similarity”.

# top 30 neighbours for each user					
Out[6]:	top1	top2	top3	top4	top5
user_id					
-DQtUcNR10Ke4TvuTnFt9w	kDKH0fcOsquMKT-ErBQ97Q	WwuiXySQN8t2hwqH_yWurA	LxS4wqYlkL2SuQx-MQ	VAb-	BZ86ul3lmKRhtHNaMR1JPw
3qmm61AgcDefmyRfmmADA	kDKH0fcOsquMKT-ErBQ97Q	WwuiXySQN8t2hwqH_yWurA	LxS4wqYlkL2SuQx-MQ	VAb-	BZ86ul3lmKRhtHNaMR1JPw
73Ze4XoO5xurQFw2SR3qGw	kDKH0fcOsquMKT-ErBQ97Q	WwuiXySQN8t2hwqH_yWurA	LxS4wqYlkL2SuQx-MQ	VAb-	BZ86ul3lmKRhtHNaMR1JPw
7xDfMMf8qh0ExxybmKnd0Q	kDKH0fcOsquMKT-ErBQ97Q	WwuiXySQN8t2hwqH_yWurA	LxS4wqYlkL2SuQx-MQ	VAb-	BZ86ul3lmKRhtHNaMR1JPw
AHRrG3T1gJpHvtpZ-K0G_g	kDKH0fcOsquMKT-ErBQ97Q	WwuiXySQN8t2hwqH_yWurA	LxS4wqYlkL2SuQx-MQ	VAb-	BZ86ul3lmKRhtHNaMR1JPw

Figure 53 Utilisateurs similaires par utilisateur selon le score de "cosine similarity"

**Computing unseen businesses scores** : enfin, en se basant seulement sur les avis de ces utilisateurs similaires, pour chaque business non évalué par l'utilisateur, on détermine le score de chacun avec la formule de la [21 numpy svd]

<https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html>

- 22 formule score] en prenant soin de filtrer les businesses fermés et dans une ville différente.

where k is a normalizing factor defined as  $k = 1 / \sum_{u' \in U} |\text{simil}(u, u')|$ , and

$$r_{u,i} = \bar{r}_u + k \sum_{u' \in U} \text{simil}(u, u') (r_{u',i} - \bar{r}_{u'})$$

where  $\bar{r}_u$  is the average rating of user  $u$  for all the items rated by  $u$ .

Équation 4 Calcul du score de chaque business par rapport à l'utilisateur

```
In [18]: recommend("AyjqBovADgbskmLrIB0MlQ") #Op-cYcN7IJJiJfx1IVjAwA
The Recommendations for user AyjqBovADgbskmLrIB0MlQ
['dp3Gs7uWo9GkNviwdR2_3w', 6.252293577981652, 'Las Vegas', ['Men's Hair Salons', 'Hair Stylists', 'Hair Salons', 'Hair Extensions', 'Blow Dry/Out Services', 'Hair Removal', 'Waxing', 'Beauty & Spas'], 1]
['kyGiC1tbPuW0Ux7eV0k7zQ', 5.585626911314985, 'Henderson', ['Auto Repair', 'Auto Parts & Supplies', 'Tires', 'Oil Change Station', 'Automotive'], 1]
['OcbLmXz_kLeizUige0lmu', 5.585626911314985, 'Las Vegas', ['Automotive', 'Auto Glass Services', 'Windshield Installation & Repair', 'Body Shops'], 1]
['7KjXE5UCy1Fy8oQNzFVhLw', 4.752293577981652, 'Las Vegas', ['Automotive', 'Auto Repair'], 1]
['mGOI83lIiDfc4cE0tmZl0', 4.752293577981652, 'Las Vegas', ['Chinese', 'Pan Asian', 'Restaurants'], 1]
```

Figure 54 Exemple de résultats de recommandations

En exploitant cet attribut de l'utilisateur qui est la liste d'amis, nous avons pu trouver un compromis à l'approche brute qui aurait été de prendre la liste complète des utilisateurs et de chercher les utilisateurs les plus similaires. Le cercle d'amis n'indique pas forcément une similarité mais il est tout de même assez pertinent pour émettre une recommandation.

On gagne de plus l'avantage de pouvoir utiliser l'intégralité des "reviews" laissés par chaque utilisateur car on sait que la matrice ne sera pas énorme, ce qui améliore la recommandation.

### 6.3.2.3. Collaborative Filtering – User Based (Memory)

C'est l'approche la plus commune pour expliquer le fonctionnement d'une recommandation de type "collaborative filtering" : on dispose d'un utilisateur dont on connaît suffisamment d'avis sur les items à recommander, on souhaite donc trouver des utilisateurs qui ont noté dans le "même ordre" (en effet, des utilisateurs peuvent donner des notes drastiquement différentes au même business, mais après normalisation ("zero-mean") des notes, on se rend compte que leur façon de noter diffère simplement d'une rigueur différente et que leurs notes normalisées sont identiques) afin de créer un cluster d'utilisateurs similaires. On peut ainsi se baser simplement sur ce cercle de confiance pour déterminer une note plausible sur des businesses que l'utilisateur n'a pas encore découvert.

Le problème de cette approche est que bien qu'elle soit efficace en théorie, elle est difficilement "scalable". Nous disposons de millions de reviews, millions d'utilisateurs, centaines de milliers de businesses, construire une matrice user x item à cette échelle demande une mémoire exceptionnelle, il nous a donc fallu réfléchir à des compromis pour rendre cette solution simplement "exécutable" :

- Tout d'abord, nous avons réduit le champ d'utilisateurs recommandables à ceux dont le review\_count était au moins de 30.
- Nous avons ensuite restreint le champ des reviews aux années 2016-2018 qui étaient les plus récentes du jeu de données et les plus remplies.
- Nous avons aussi exclu tous les business qui n'étaient pas dans le 90th percentile des businesses classés par review\_count. Cela induit un biais de popularité du business mais augmentera la pertinence des recommandations car les similarités des utilisateurs seront plus précises.
- Enfin, nous avons stocké toutes ces informations dans un fichier JSON prêt à être chargé dans un dataframe panda, pour gagner en temps d'exécution.

Nous avons ensuite utilisé la même procédure que pour la recommandation "Friendlist-Based" et avons pu obtenir nos recommandations. Cependant, notre serveur n'est pas assez puissant et le temps d'exécution reste trop long (environ 1min), aussi, nous avons écarté cette méthode du système hybride pour privilégier l'approche model-based plus performante.

### 6.3.3. Recommandation – Popularity/Trends Based Filtering

Il est compliqué pour les techniques de recommandations sociale et objet de recommander un utilisateur ayant laissé peu de reviews car l'échantillon est trop faible et irrégulier pour déterminer une similarité avec d'autres utilisateurs/items. Des systèmes de recommandation comme celui de Twitter ou Linkedin impose à l'utilisateur de renseigner ses goûts/avis afin de pouvoir construire un profil suffisamment conséquent pour établir des recommandations acceptables ; pour notre projet, il aurait été envisageable de simuler le même processus mais les tâches étant déjà nombreuses, nous avons décidé d'écartier cette

solution. C'est pour cela que nous avons choisi d'utiliser des recommandations plus neutres, ne se basant pas sur l'utilisateur mais plutôt sur les tendances actuelles qui puisque ce sont des tendances, ont une probabilité assez élevée de produire des recommandations qu'un utilisateur pris au hasard pourrait juger correctes.

Pour déterminer la popularité d'un établissement, il faut en prendre en compte la somme des notes laissées par chaque utilisateur plutôt que la moyenne des notes, car les établissements avec peu de reviews sont susceptibles d'avoir une moyenne "inflated" et qui est difficilement comparable à la moyenne pondérée d'un établissement avec des milliers de reviews qui ne pourra jamais atteindre un score parfait. Ainsi, ce classement des établissements par popularité privilégie les établissements présents sur le long terme qui ont mis d'accord une majorité de Yelpers.

Nous travaillons, pour raisons de mémoire, sur les reviews les plus récentes du jeu de données, à savoir 2016-2018. Cependant, la moyenne des ratings et le nombre de reviews d'un business, qui sont renseignés dans son index sont quelque peu faussés pour plusieurs raisons :

- Ces chiffres sont calculés sur l'intervalle total : 2008-2018
- Ces chiffres ne matchent pas avec l'index review : si par exemple un business indique un nombre de reviews de 24, l'index review ne comporte pas nécessairement les 24 reviews.

Nous avons donc pris tout cela en compte pour calculer la popularité d'un business donné. Nous avons identifié deux facteurs connotant la popularité d'un business :

- Le "star rating" normalisé : pour un business donné, nous regroupons toutes les reviews dont nous disposons entre 2016 et 2018, puis nous calculons la somme des "stars" et le nombre de reviews. On calcule ainsi le star rating normalisé en calculant la moyenne puis en appliquant la normalisation min-max pour avoir un score entre 0 et 1.
- On a cependant le problème de star rating "inflated", pour contrebalancer cela, le membre de l'équipe qui travaille sur les trends a proposé de fournir pour chaque business le nombre de checks-ins sur l'intervalle 2016-2018, cela met en valeur la popularité sur le long terme d'un établissement et punit ainsi ces businesses dont le star rating était "inflated". Nous appliquons la normalisation min-max à cette somme pour avoir nos deux facteurs sur la même échelle.

Il reste ainsi à sommer ces deux scores puis de nouveau normaliser pour obtenir le "**pop\_score**". Afin de gagner en performance et par logique, nous limitons ces calculs aux businesses à proximité de l'utilisateur, en utilisant la distance business-utilisateur normalisée comme facteur pour calculer le "**geo\_score**".

Pour déterminer la recommandation la plus impartiale et logique possible, nous appliquons les poids 0.5 aux deux scores et faisons la somme pour obtenir le score final de chaque business :

$$score = 0.5 \times \text{pop\_score} + 0.5 \times \text{geo\_score}$$

	<b>business_id</b>	<b>pop_score</b>	<b>distance</b>	<b>geo_score</b>	<b>score</b>
<b>38</b>	fY0RCsymg465GQ7tmxLYog	0.492647	0.4559	0.456399	0.949046
<b>6</b>	ff-bUaqzSnyAOzW_Qtd65A	0.500000	0.5594	0.446003	0.946003
<b>9</b>	8gYkL76weP9dmd-W8NOeqw	0.500000	0.5915	0.442779	0.942779
<b>27</b>	Olc8vVH1MTmObFeeoc0LFA	0.500000	0.6687	0.435025	0.935025
<b>14</b>	ovFnrmsT5rUpbAQE1r-V0Q	0.500000	0.6738	0.434513	0.934513
<b>20</b>	LklurWnK8agxFp6G-v1CMg	0.500000	0.7311	0.428758	0.928758

Figure 55 Calcul du score de popularité en fonction de la géolocalisation

Nous avons considéré d'autres idées de trends comme utiliser l'algorithme "Apriori" pour déterminer des tendances sur les businesses ou les catégories mais cela s'est avéré infructueux. Nous disposions aussi de tendances temporelles mais les résultats étaient trop "niche" pour vraiment influencer les résultats de recommandations et auraient demandé davantage de temps pour implémenter les résultats dans le système. Par manque de temps, nous avons donc décidé de simplement afficher les tendances détectées sur l'interface final et de laisser le système hybride indépendant à part pour les check-ins.

### 6.3.4. Recommandation – Geolocalisation Decision

Pour compléter notre système de recommandations, il reste un facteur important qui est la distance géographique entre l'utilisateur et les businesses recommandés. En effet, si deux recommandations obtiennent un score similaire, mais que le premier business est à 50m de l'utilisateur tandis que le second est à 3km, il est logique de privilégier le premier résultat même si son score est légèrement inférieur au second. Aussi, nous avons décidé de normaliser la distance des businesses à proximité de l'utilisateur afin d'obtenir un score géographique, que nous pourrons utiliser pour davantage affiner la recommandation finale. Pour cela, nous avons utilisé les travaux du Chapitre 5.

### 6.3.5. Hybrid System (solution globale)

Comme expliqué précédemment, le système hybride consiste à regrouper et combiner efficacement les résultats des différentes techniques implémentées jusqu'ici. Notre système hybride peut décider d'utiliser ou non certaines techniques selon notre intérêt. Nous avons distingué plusieurs cas possibles :

- L'utilisateur ne figure pas dans notre base de données et donc nous ne pouvons pas établir de recommandations sociale/objet pertinentes (utilisateur peu actif, ou utilisateur anonyme), dans ce cas nous utiliserons le **popularity/trend based recommender (PTB)** et la **geolocalisation decision (GD)** et déterminerons le score des businesses évalués avec la formule suivante :

$$Score_{restaurant} = 0.5 \times \textcolor{red}{PTB} + 0.5 \times \textcolor{green}{GD}$$

- L'utilisateur n'a pas utilisé la recherche par mots-clés, dans ce cas nous utiliserons le **friendlist based recommander (FB)**, le **Singular Value Decomposition (SVD)** et le **content-based recommander without input (CB)** avec les poids suivants (le score de geoloc étant déjà appliqué dans les scores de chaque autre technique) :

$$Score_{restaurant} = 0.6 \times CB + 0.2 \times FB + 0.2 \times SVD$$

- L'utilisateur a renseigné des mots-clés caractérisant ce qu'il recherche, dans ce cas il est convenable d'utiliser le **content-based recommander with key-word (CB)** et la **geolocalisation decision (GD)** et d'évaluer le score des businesses évalués comme ceci :

$$Score_{restaurant} = 0.8 \times CB + 0.2 \times GD$$

- Pour établir une recommandation purement sociale on appliquera le **friendlist based recommander (FB)**, le **Singular Value Decomposition (SVD)** et d'évaluer le score des businesses évalués comme ceci :

$$Score_{restaurant} = 0.5 \times FB + 0.4 \times SVD + 0.1 \times GD$$

- On souhaite afficher une recommandation purement objet sur l'interface utilisateur (pour justifier la recommandation), dans ce cas nous utiliserons le **content-based recommander with user\_id (CB)** et la **geolocalisation decision (GD)** et d'évaluer le score des businesses évalués comme ceci :

$$Score_{restaurant} = 0.8 \times CB + 0.2 \times GD$$

Une fois que les recommandations ont été obtenues, nous appliquons un filtre pour seulement garder les businesses à proximité de l'utilisateur (périmètre de 5 km) afin de rester pertinent dans nos recommandations ; nous ne voulons pas recommander des businesses de Las Vegas à un utilisateur situé à New York, ça n'aurait pas de sens. Il est envisageable qu'aucune des recommandations ne passent le filtre des businesses près de l'utilisateur, dans ce cas nous préviendrons l'utilisateur que notre système ne peut trouver de recommandations.

## 6.4. Tests et certifications de la solution

### Popularity Based

La recommandation Popularity Based est dédié aux nouveaux utilisateurs. Ceci permet l'utilisateurs de voir rapidement les businesses populaires au moment l'utilisateur arrive sur la page d'accueil.

On peut configurer deux paramètres pour ce système de recommandation : la localisation de l'utilisateur et la catégorie de business que l'utilisateur veut chercher. Si on ne met pas la valeur pour la catégorie, le système considère tout type de business. Ci-dessus la commande pour la recommandation des hôtels populaires à Las Vegas dans le périmètre de 5km

```

1 filters={}
2 filters["nearby"] = business_nearby
3 filters["categories"] = "Hotels"

```

```

1 start = time.time()
2 rec_pop = engine_pop.predict(filters=filters, top_n=50)
3 print("Done popularity recommendation in %s seconds." % (time.time() - st

```

Total reviews retrieved: 18844  
Done popularity recommendation in 2.0517847537994385 seconds.

Figure 56 Commande de la recommandation Popularity Based

La recommandation est effectuée dans 2 secondes pour 18844 commentaires

On obtient le résultat de top 50 businesses par Popularity Based comme suit :

```

1 rec_pop = business_details(business_df, rec_pop)
1 rec_pop[["name", "categories", "city", "distance", "pop_score", "geo_scor

```

	name	categories	city	distance	pop_score	geo_score	score
0	Bridger Inn	Hotels, Event Planning & Services, Hotels & Tr...	Las Vegas	0.2102	0.312500	0.481077	0.793577
1	Segway Las Vegas	Tours, Motorcycle Rental, Hotels & Travel, Act...	Las Vegas	0.6738	0.500000	0.434513	0.934513
2	Enterprise Rent-A-Car	Hotels & Travel, Car Rental	Las Vegas	0.6005	0.269231	0.441875	0.711106
3	Cowboy Trail Rides	Hotels & Travel, Tours	Las Vegas	2.9530	0.500000	0.205590	0.705590
4	Las Vegas Club Hotel & Casino	Casinos, Hotels & Travel, Event Planning & Ser...	Las Vegas	0.5930	0.312500	0.442629	0.755129
5	Garden Court Buffet	Restaurants, Hotels, Event Planning & Services...	Las Vegas	0.8199	0.306383	0.419839	0.726222

Figure 57 Résultat de la recommandation Popularity Based

On trouve dans la photo le nom trois colonnes de score : pop\_score, geo\_score et score. Le score est la somme de geo\_score et pop\_score avec le poids de 0.5 pour chacun.

A noter que le score de la popularité pop\_score est calculé avec la somme de ratings des commentaires divisé par le nombre de commentaires et avec la somme des check-ins sur les années 2016-2018 puis normalisé. Par ailleurs le score de la localisation est aussi normalisé entre 0 et 1. Plus la distance est faible plus le score est élevé.

Au final, on voit bien les businesses de catégorie Hotels localisé à Las Vegas. Ils sont classés par la distance et par la note d'évaluation des autres utilisateurs.

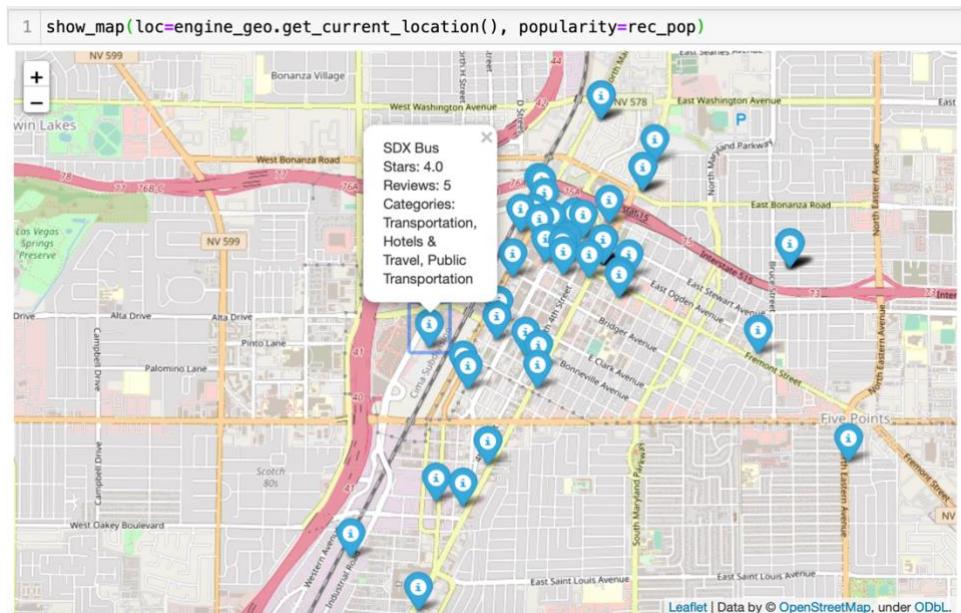


Figure 58 Résultat de la recommandation Popularity Based sur le plan

## Content Based

Pour le système de recommandation par Content Based, on est capable d'effectuer deux types de la recherche :

- Content Based Keyword: C'est réservé pour les nouveaux utilisateurs, le système de recommandation propose aux utilisateurs les businesses similaires au mot saisi dans la barre de recherche par l'utilisateur
- Content Based User\_id: Il est utilisable uniquement pour les utilisateurs connus, c'est à dire qu'on a déjà certaines connaissances sur l'utilisateur grâce à leurs commentaires. Le système recommande les businesses similaires à ce qu'il a évalué récemment.

### Test de Content Based Keyword

Supposant que l'utilisateur saisie dans la barre de recherche le mot "tacos" et il se trouve à Las Vegas, le résultat de la recommandation est comme ci-dessous :

```

1 filters={}
2 filters["nearby"] = business_nearby
3
1 start = time.time()
2 rec_cb = engine_cb.keyword_recommend(input_str="tacos", top_n=20, filters=filters)
3 print("Done keyword recommendation in %s seconds." % (time.time() - start))

```

Done keyword recommendation in 1.045820951461792 seconds.

```

1 rec_cb = business_details(business_df, rec_cb)
1 rec_cb[["name", "categories", "city", "distance", "content_score", "geo_score", "score"]]

```

	name	categories	city	distance	content_score	geo_score	score
0	Carnitas y Tortas Ahogadas Guadalajara 2	Mexican, Restaurants	Las Vegas	1.7347	0.462409	0.655913	0.501110
1	Dos Brothers	Food Trucks, Restaurants, Food, Mexican, Event...	Henderson	1.2748	0.403980	0.748298	0.472844
2	Taqueria El Buen Pastor	Mexican, Hot Dogs, Food, Restaurants, Street V...	Las Vegas	0.6559	0.378987	0.872622	0.477714
3	Taqueria El Pastorcito	Food Trucks, Street Vendors, Food	Las Vegas	1.9574	0.415857	0.611177	0.454921
4	Wana Taco	Mexican, Restaurants, Tacos	Las Vegas	0.5976	0.451652	0.884333	0.538188
5	Bajamar Seafood & Tacos	Dive Bars, Restaurants, Bars, Tacos, Fast Food...	Las Vegas	1.7502	0.418699	0.652799	0.465519
6	Taqueria El Buen Pastor	Street Vendors, Mexican, Food, Food Trucks, Re...	Las Vegas	1.3128	0.410758	0.740664	0.476739
7	La Onda Banquet Hall 2	Event Planning & Services, Venues & Event Spaces	Las Vegas	4.3253	0.467790	0.135514	0.401335

Figure 59 Commande et résultat de la recommandation Content Based Keyword

On voit dans le résultat les businesses proposés sont pour la plupart les restaurants Mexicain indiqués dans la catégorie. Il n'a pas précisé parfois explicitement Tacos dans la catégorie mais il est logique de conclure les restaurants Mexicain propose les tacos parmi leurs plats.

La recommandation est faite sous 2 secondes. D'ailleurs, on trouve bien un business à Henderson et cette ville se situe à côté de Las Vegas, cela signifie que notre système de géolocalisation marche bien.

Le résultat est trié par la combinaison du score de la recommandation elle-même et du score de la distance. Le poids de score final est de :

$$0.8 \times \text{score\_content\_based} + 0.2 \times \text{score\_geo\_localisation}$$

Sur la Figure 60, nous pouvons voir les businesses recommandés sur le plan. On a limité sur le top 20 de business et dans le périmètre de 5km.

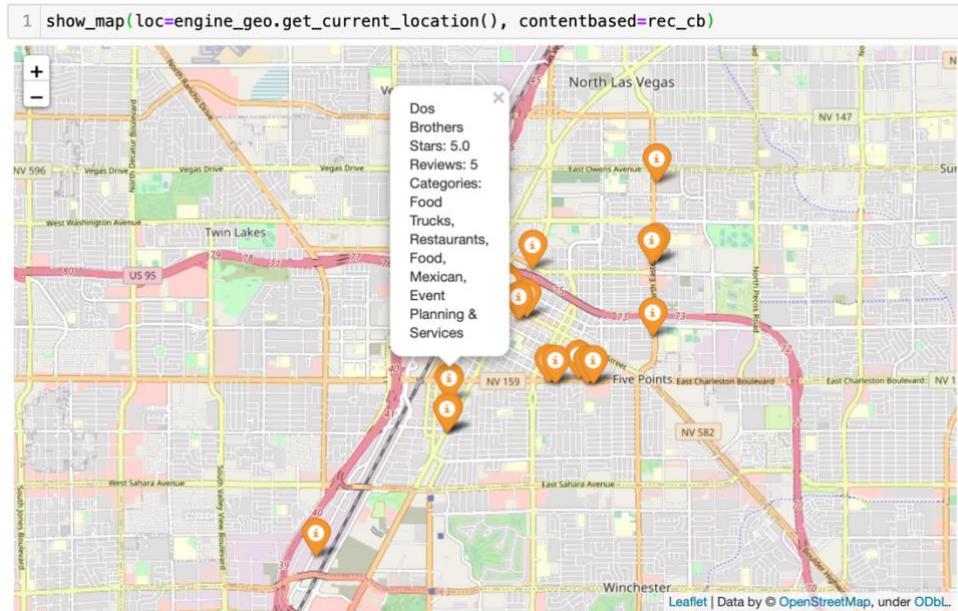


Figure 60 Résultat de la recommandation Content Based Keyword sur le plan

### Test de Content Based Userid

On lance la recommandation pour l'utilisateur EC5nxNCWCmjHg1F14Wr1xQ à Las Vegas, la commande dans Jupyter Notebook est comme suit :

```
1 start = time.time()
2 reviewed_df, rec_cb = engine_cb.content_recommend(user_id="EC5nxNCWCmjHg1F14Wr1xQ", top_n=20,
3 print("Done Content Based Userid in %s seconds." % (time.time() - start))

INFO:elasticsearch:POST http://47.91.72.40:9200/yelp-review*/_search?_source_excludes=text%2C4
0 timestamp%2C40version%2Ccool%2Cuseful%2Cfunny&_source_includes=&scroll=1m&size=2000
[status:200 request:0.016s]
INFO:elasticsearch:POST http://47.91.72.40:9200/_search/scroll?scroll=1m [status:200 request:0.
012s]

Total reviews retrieved: 44
Mask size : 5202
Done Content Based Userid in 5.0554587841033936 seconds.
```

Figure 61 Commande de la recommandation Content Based Userid

En effet, on va récupérer les 5 derniers commentaires d'utilisateur pour faire la recommandation. On s'intéresse qu'aux commentaires avec une note d'évaluation élevée (supérieure que 3). Pour chaque business entre les 5 évalués par l'utilisateur, on propose à l'utilisateur les top 20 businesses similaires. On vérifie, tout d'abord, les businesses évalués par l'utilisateur. La Figure 62 décrit les 5 derniers businesses visités récemment par l'utilisateur.

44 hits				
user_id	business_id	date	descending	stars
> EC5nxNCWCmjHg1F14Wr1xQ	ZRsDVZmMjE8jLqHivluWLA	Jul 13, 2018 @ 22:50:39.000		4
> EC5nxNCWCmjHg1F14Wr1xQ	4byedmWf21X0QQVEIWIA8g	Jun 17, 2018 @ 18:01:19.000		4
> EC5nxNCWCmjHg1F14Wr1xQ	yooVq4aUUthr7DQOiz7yEA	Jun 17, 2018 @ 03:56:33.000		4
> EC5nxNCWCmjHg1F14Wr1xQ	w7ZAphNGA5g41MYMKg0tSg	Feb 4, 2018 @ 11:23:49.000		5
> EC5nxNCWCmjHg1F14Wr1xQ	RrCgc8eAKbHu-2IpQXg6Rw	Jan 17, 2018 @ 02:52:05.000		4

Figure 62 Informations des businesses déjà évalués par l'utilisateur

Nous pouvons voir dans le résultat de recommandation (Figure 63) qu'il y a deux colonnes concernant l'identifiant de business : `input_business_id` et `business_id`. La colonne `input_business_id` contient les identifiants des businesses que l'utilisateur a déjà visité et commenté. Nous en avons 3 dans cet exemple. Pour chacun de ces 3 businesses déjà évalués, nous essayons de recommander à l'utilisateur les 20 businesses les plus similaires.

1	rec_cb[["input_business_id", "business_id", "name", "categories", "distance", "content_score"]]							
	input_business_id	business_id	name	categories	distance	content_score	geo_score	score
w7ZApNGA5g4lMYMKg0tSg	1YxLacCdn4yYQDPUzdye8g	Ted Wiens Tire & Auto		Oil Change Stations, Tires, Automotive, Auto R...	1.8205	0.943604	0.638677	0.882615
ZRsDVZmMjE8jLqHivluWLA	BQIR8VWkWlzv4zzLPp7OuQ	Banich Terence G MD Facs		Doctors, Health & Medical	1.6581	0.941932	0.671300	0.887805
ZRsDVZmMjE8jLqHivluWLA	q0GIGkG6fgJ3Q6iMNPKamQ	Apex Medical Center		Medical Centers, Family Practice, Allergists, ...	1.5304	0.944695	0.696953	0.895146
4byedmWf21X0QQVEIWA8g	q0GIGkG6fgJ3Q6iMNPKamQ	Apex Medical Center		Medical Centers, Family Practice, Allergists, ...	1.5304	0.956182	0.696953	0.904336
ZRsDVZmMjE8jLqHivluWLA	gsHW2yqN4JwwtRaOSVCq8w	Goodell, DO FACOOG Julie		Doctors, Health & Medical, Obstetricians & Gyn...	1.6958	0.945001	0.663727	0.888746

Figure 63 Résultat de la recommandation Content Based Userid

Les autres attributs trouvés dans le résultat sont ceux contenant les informations pour le business recommandé, et non pour le business déjà évalué par l'utilisateur. Nous justifions la relation par les informations entre `input_business_id` et `business_id` séparément

On récupère sur elasticsearch les informations concernant le businesses déjà évalué par l'utilisateur :

3 hits					
	business_id	city	name	stars	categories
>	4byedmWf21X0Q	Henderson	Henderson Hospital	3	Health & Medical, Medical Centers, Hospitals
>	ZRsDVZmMjE8jLqHivluWLA	Henderson	Complete Care OBGYN	4,5	Childbirth Education, Obstetricians & Gynecologists, Doctors, Medical Centers, Education, Health & Medical, Specialty Schools
>	w7ZApNGA5g4lMYMKg0tSg	North Las Vegas	Smog Hut	4	Smog Check Stations, Automotive, Auto Repair

Figure 64 Informations d'`input_business_id` évalué par l'utilisateur

Commençons par l'analyse d'input\_business\_id w7ZAphNGA5g4lMYMKg0tSg qui est une station d'automobile. On trouve dans le résultat, le business avec l'identifiant 1YxLacCdn4yYQDPUzdy8g qui a la même catégorie.

Ensuite, on a l'input\_business\_id 4byedmWf21X0QQVEIWIA8g qui correspond au business avec comme catégorie d'Hôpital, Santé et Médical. Le business id recommandé q0GIGkG6fgJ3Q6iMNPKamQ sur Figure 63 est de même catégorie.

Enfin, on a des businesses très similaires avec le troisième input\_business\_id ZRsDVZmMjE8jLqHivluWLA

### Model Based SVD Collaborative Filtering

Il n'est pas évident de tester l'efficacité de l'algorithme de SVD vu que la recommandation pour un utilisateur dépend la tendance des autres utilisateurs. Ce qu'on peut tester c'est le RMSE (Root Mean Square Error). Cela signifie le taux d'erreur de la matrice reconstruit par la production de 3 matrices factorisées par rapport à la matrice originale.

Tout d'abord, on choisit le meilleur paramètre de facteur pour minimiser le RMSE par la technique cross-validation. On utilise la librairie « surprise » pour réaliser cette tâche. En effet, surprise est une librairie avec les algorithmes conçus pour la recommandation. Cela est une solution out-of-the-box pour le tuning de hyperparamètres du modèle et pour les métriques d'évaluation.

On commence par la préparation de données. Les données contiennent 3 494 121 interactions entre l'utilisateur et le business. Pour éviter le problème de cold start, on enlève les users qui ont moins de 40 commentaires dans le jeu de données. Le filtrage de cold start nous permet aussi de réduire la taille de matrice éventuellement pour la recommandation, étant donné que la taille de mémoire est limitée. La taille de données après le filtrage est de 519 838.

Pour la partie de tuning, on démarre la recherche de nombre de facteur idéal avec 4 valeurs : 20, 30, 50 et 100 par cross-validation 5 folds. Le résultat obtenu est le suivant :

```
1 gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=-1)
2 gs.fit(data)
3 params = gs.best_params['rmse']

/home/hongphuc95/anaconda3/lib/python3.7/site-packages/joblib/externals/loky/process_executor.py:691: UserWarning: A worker stopped while some jobs were given to the executor. This can be caused by a too short worker timeout or by a memory leak.
    "timeout or by a memory leak.", UserWarning

1 print("Best number of factors: %d" % (params['n_factors']))
Best number of factors: 20
```

Figure 65 Tuning de nombre de facteurs SVD

On observe que le facteur optimal pour notre modèle est 20. On continue de calculer la moyenne de RMSE en prenant en compte ce paramètre. La moyenne obtenue est 1.3273. Les détails sont dans la Figure 66.

```

1 svdtuned = SVD(n_factors=params['n_factors'])

2 # Run 5-fold cross-validation and print results.
2 cross_validate(svdtuned, data, measures=['RMSE', 'MAE'], cv=5, n_jobs=-1, verbose=True)

Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

      Fold 1  Fold 2  Fold 3  Fold 4  Fold 5   Mean    Std
RMSE (testset)  1.3276  1.3269  1.3283  1.3280  1.3254  1.3273  0.0010
MAE (testset)   1.0689  1.0678  1.0689  1.0685  1.0667  1.0682  0.0008
Fit time        70.20   72.28   70.61   71.76   70.22   71.02   0.85
Test time       11.69   8.21    8.37    8.51    8.08    8.97    1.37

{'test_rmse': array([1.32763919, 1.32694807, 1.32827826, 1.32802855, 1.3254026 ]),
 'test_mae': array([1.06891823, 1.06779658, 1.06889518, 1.06854169, 1.06672802]),
 'fit_time': (70.20356464385986,
 72.27842116355896,
70.61429762840271,
71.76304912567139,
70.21897721290588),
'test_time': (11.692693948745728,
8.206737279891968,
8.372921705245972,
8.507348775863647,
8.079296350479126)}

```

Figure 66 RMSE de SVD sur la cross validation 5 folds

## Friendlist Based Collaborative Filtering

Pour tester un algorithme de collaborative filtering, il est difficile de justifier les recommandations : en effet, si les goûts d'un utilisateur viennent s'accorder avec ceux d'autres utilisateurs, par exemple un intérêt commun pour des restaurants asiatiques, la recommandation peut très bien proposer un restaurant d'une autre catégorie et il est dès lors difficile de justifier ce choix à partir des goûts connus de l'utilisateur car ce n'est pas forcément une catégorie "phare" de l'utilisateur.

On peut cependant déjà montrer que l'algorithme trouve bien des utilisateurs similaires en comparant les avis sur le même restaurant de l'utilisateur avec ces K voisins les plus similaires.

Pour ce test, nous utilisons un utilisateur avec 506 amis et 196 reviews (user\_id : "AyjqBovADgbskmLrlBOMIQ").

Après normalisation des scores, on obtient les prédictions suivantes :

In [16]: 1 test

Out[16]:

		business_id	score
1907	L1-1P3acJc4gEFvWwjXcNQ	1.000000	
1944	beuVp5CZxCdNvQlIPBS2rw	0.964212	
1053	sZIVzaaEBp_HiYutZ2IWag	0.959333	
1103	po0p6NIro0cDrmKkcyPy0w	0.941904	
1428	YILyHegzhy1vlc_LNVfObw	0.904806	
...	...	...	...
628	MU1PQ5CWuV0OKKOel7jx6Q	0.000374	
179	RX8Q4_nu3VnAwXtHdgAKCg	0.000070	
1391	VAD-Faox-xQdE1WdYn8_5Q	0.000000	
639	vl2lZrNJEA8npSjqXbdwxw	0.000000	
1113	BjrKNWhtQkedHw8hP_0Bjg	0.000000	

2035 rows × 2 columns

Figure 67 Recommandations pour l'utilisateur testé

Par ailleurs, nous pouvons retracer les 30 voisins qui ont permis d'établir ces recommandations :

```
In [20]: 1 user_neighbors = neighbors[(neighbors.index.get_level_values('user_id') == "AyjqBovADgbskmLrIBOMlQ")]
2 user_neighbors
```

Out[20]:

	top1	top2	top3	top4	top5	
user_id						
AyjqBovADgbkskmLrlBOMIQ	60skTN6p8SdQ2sGNKL0vRQ	bLbSNkLggFnqwNNzq-jlw	FQbVI3UyKIL-HfQgCRF8aA	ZCLDdUi4px9mys8vVUelkg	TprC8suzj8MkuomrqUSlw	FcWu03lLuGY

1 rows x 30 columns

Figure 68 Utilisateurs les plus similaires à l'utilisateur testé

Nous pouvons à l'aide d'une jointure des reviews sur le "business\_id" comparer les avis laissés sur les mêmes businesses par l'utilisateur et chacun de ses voisins :

```
In [5]: 1 a = get_user_similar_restaurants("AyjqBovADgbskmLrIBOMlQ","60skTN6p85dQ2sGNKL0vRQ")
2 a = a.loc[ :, ['stars_x','stars_y','categories_x']]
3 a.head()
4
5 # 60skTN6p85dQ2sGNKL0vRQ--->bLbSNkLggFnqwNNzzq-Ijw---FQbVI3UyKLL-HfQGcRF8aA---ZCLDdUi4px9my:
```

Out[5]:

	stars_x	stars_y	categories_x
0	3.0	4.0	[Coffee & Tea, Breakfast & Brunch, Restaurants...]
1	5.0	4.0	[Cafes, Restaurants, Japanese, Soup, Noodles]
2	4.0	4.0	[Restaurants, American (New)]
3	4.0	3.0	[Restaurants, Cafes, Coffee & Tea, Desserts, F...
4	4.0	4.0	[Asian Fusion, Japanese, Sushi Bars, Restaurants]

Figure 69 Comparaison utilisateur – utilisateur top1

```
In [7]: 1 a = get_user_similar_restaurants("AyjqBovADgbskmLrIBOMlQ","bLbSNkLggFnqwNNzzq-Ijw")
2 a = a.loc[ :, ['stars_x','stars_y','categories_x']]
3 a.head()
4
5 # 60skTN6p85dQ2sGNKL0vRQ--->bLbSNkLggFnqwNNzzq-Ijw---FQbVI3UyKLL-HfQGcRF8aA---ZCLDdUi4px9my:
```

Out[7]:

	stars_x	stars_y	categories_x
0	5.0	3.0	[Ethnic Food, American (New), Burgers, Food, R...
1	4.0	3.0	[American (Traditional), Tapas/Small Plates, B...
2	3.0	4.0	[Coffee & Tea, Breakfast & Brunch, Restaurants...]
3	5.0	4.0	[Cafes, Restaurants, Japanese, Soup, Noodles]
4	5.0	5.0	[Hotels & Travel, Shopping, Arts & Entertainme...

Figure 70 Comparaison utilisateur – utilisateur top2

```
In [8]: 1 a = get_user_similar_restaurants("AyjqBovADgbskmLrIBOMlQ","TprC8sujz8MkuomrqUSiw")
2 a = a.loc[ :, ['stars_x','stars_y','categories_x']]
3 a.head()
4
5 # 60skTN6p85dQ2sGNKL0vRQ--->bLbSNkLggFnqwNNzzq-Ijw---FQbVI3UyKLL-HfQGcRF8aA---ZCLDdUi4px9my:
```

Out[8]:

	stars_x	stars_y	categories_x
0	5.0	3.0	[Cafes, Restaurants, Japanese, Soup, Noodles]
1	5.0	3.0	[Cafes, Restaurants, Japanese, Soup, Noodles]
2	4.0	4.0	[Restaurants, Sushi Bars, Seafood, Japanese]
3	4.0	4.0	[Restaurants, Sushi Bars, Seafood, Japanese]
4	4.0	4.0	[Beer, Wine & Spirits, Restaurants, Gastropubs...]

Figure 71 Comparaison utilisateur – utilisateur top5

On observe effectivement que les utilisateurs notent bien de la même manière que l'utilisateur qu'on cherche à recommander et que plus le voisin s'éloigne de l'utilisateur, plus la différence des notes s'accroît.

Intéressons-nous maintenant à la pertinence du business ayant obtenu le meilleur score : une rapide requête sur Kibana nous indique que ses catégories sont “Greek, Restaurants, Mediterranean”, et qu'il possède notamment les attributs suivants (nous omettons volontairement les attributs dont les valeurs sont False pour une lecture plus simple) :

- BusinessAcceptsCreditCards\_True
- RestaurantsPriceRange2\_2
- Ambience\_casual\_True
- RestaurantsTakeOut\_True
- WiFi\_free
- Alcohol\_beer\_and\_wine
- GoodForKids\_True
- GoodForMeal\_desert\_True
- GoodForMeal\_lunch\_True
- GoodForMeal\_dinner\_True
- HasTV\_True
- WheelchairAccessible\_True
- RestaurantsGoodForGroups\_True
- RestaurantsAttire\_casual
- BikeParking\_True
- NoiseLevel\_average
- Caters\_True

Nous pouvons par ailleurs estimer les goûts de l'utilisateur à l'aide d'une fonction parcourant les différentes reviews laissées par ce dernier pour compter chaque élément en appliquant un poids (1,-1) selon son star rating : supérieur ou inférieur à sa moyenne.

```
In [26]: 1 categories, attributes = getUserTaste("AyjqBovADgbskmLrIBOMlQ")
4.228155339805825
```

---

```
In [27]: 1 categories
```

```
Out[27]: [('Restaurants', 50),
('Bars', 40),
('Food', 28),
('Nightlife', 19),
('American', 18),
('Japanese', 12),
('Arts', 11),
('New', 11),
('Services', 10),
('Cocktail', 9),
('Entertainment', 9),
('Tapas', 9),
('Breakfast', 8),
('Brunch', 8),
('Wine', 8),
('Asian', 7),
('Fusion', 7),
('Health', 7),
('Sushi', 7),
('Meditteranean', 3)]
```

Figure 72 Catégories préférées de l'utilisateur

Nous remarquons que l'utilisateur ne semble pas avoir de catégories de restaurant fétiches, et qu'il est plutôt généreux avec une moyenne de star rating d'environ 4.2. Notre algorithme a cependant réussi à recommander un restaurant parmi tous les autres businesses que Yelp propose (magasins, bars, stations de lavage...) et c'est la catégorie principale de l'utilisateur avec un score de 50 (Seafood obtient le score de 4 et Meditteranean 3). Nous pouvons donc considérer que l'algorithme a bien respecté les goûts de l'utilisateur en ce qui concerne les catégories d'un business.

```
In [28]: 1 attributes
Out[28]: [('BusinessAcceptsCreditCards_True', 71),
           ('BusinessParking_validated_False', 57),
           ('RestaurantsGoodForGroups_True', 51),
           ('Ambience_divey_False', 50),
           ('Ambience_touristy_False', 49),
           ('Ambience_romantic_False', 48),
           ('Ambience_intimate_False', 46),
           ('BusinessParking_valet_False', 46),
           ('BusinessParking_street_False', 45),
           ('Ambience_upscale_False', 44),
           ('GoodForMeal_breakfast_False', 44),
           ('GoodForMeal_dinner_True', 43),
           ('RestaurantsAttire_casual', 43),
           ('Ambience_classy_False', 42),
           ('GoodForKids_True', 42),
           ('RestaurantsTakeOut_True', 42),
           ('Ambience_casual_True', 41),
           ('Ambience_hipster_False', 41),
           ('GoodForMeal_latenight_False', 40),
```

Figure 73 Attributs préférés de l'utilisateur

En ce qui concerne les attributs, nous retrouvons beaucoup de ceux que l'utilisateur affectionne, comme ambience/attire casual, good for kids, good for dinner, option to takeout, accepts credit cards, etc. Nous pouvons donc aussi considérer que la recommandation a bien respecté les goûts de l'utilisateur.

Enfin, le temps d'exécution est tout à fait correct puisque nous travaillons sur une petite matrice, inférieur à 1 sec. Nous pouvons donc conclure après ce test que la recommandation friendlist fonctionne dans sa logique collaborative. Il reste cependant subjectif d'apprécier ou non ces recommandations en tant qu'utilisateur.

# Chapitre 7 Rendu final

Dans les chapitres précédents, nous avons développé les parties techniques concernant les différentes problématiques de notre projet. Maintenant, nous allons nous intéresser au rendu final notamment en présentant l'interface finale et les différents tests qui y sont liés.

## 7.1. Interface utilisateur finale

Notre interface utilisateur se divise en plusieurs pages accessibles grâce au menu situé tout en haut de la page. Chacune de ces pages vont être détaillées par la suite.

### Accueil

En arrivant sur notre application web, l'utilisateur se trouve sur la page d'accueil (Figure 74).

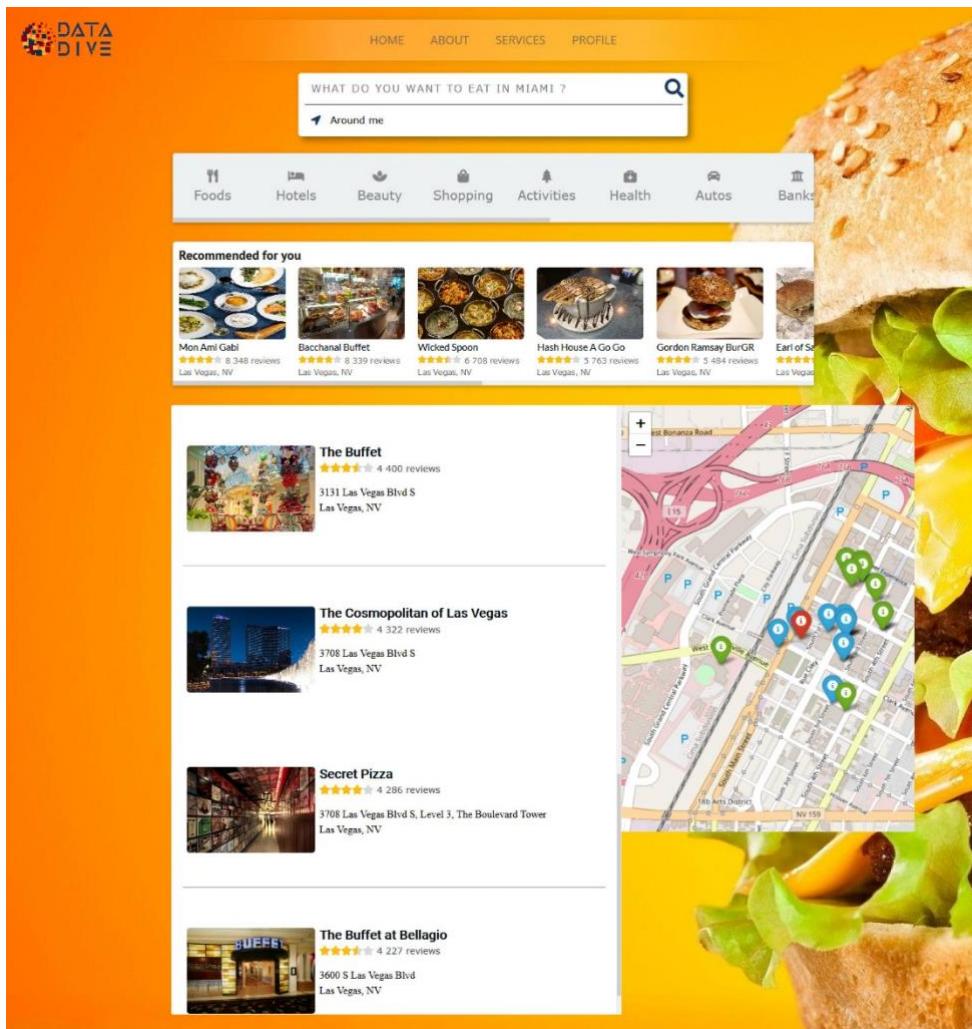


Figure 74 Page d'accueil de l'application

Nous pouvons distinguer différentes parties sur celle-ci :

- Une barre de navigation pour accéder aux différentes pages de l'application.
- Une barre de recherche à l'aide de laquelle l'utilisateur fait une recherche en tapant des mots clé ou bien en se géolocalisant.
- Un menu de catégories déroulant horizontalement avec lequel l'utilisateur peut réduire le périmètre de ses recherches.
- Une liste de restaurants recommandés en fonction des goûts de l'utilisateur et de sa géolocalisation.
- Une section avec des restaurants à proximité de l'endroit où se trouve l'utilisateur.
- Une carte géographique résumant les restaurants aux alentours.

Les images des différents commerces ont été récupérés grâce à l'API YELP qui, malheureusement, ralentit la recherche de business. En effet, après avoir récupéré les business, il faut requêter l'API YELP pour chacun d'entre eux. Étant limité à 5 000 requêtes par jour et possédant plus de 190 000 business en plus du fait d'avoir une recommandation personnalisée, il est impossible de stocker images de ces derniers.

## Profil

Cette page de profil (Figure 75) remplace la page d'authentification habituelle afin de faciliter la démonstration. Il suffit de s'authentifier en rentrant l'identifiant d'utilisateur, aucun mot de passe est nécessaire. Après avoir sélectionné un utilisateur parmi la liste, il faut valider afin d'avoir des informations sur ce dernier et l'enregistrer en mémoire pour lui faire des recommandations. Si l'utilisateur est nouveau, il faut sélectionner *nouveau utilisateur*.

Ensuite, une fois les informations affichées, il faut renseigner une adresse afin de la géolocaliser et soumettre le formulaire. De ce fait nous avons un contexte pour recommander l'utilisateur : son *user\_id* et sa *localisation*.

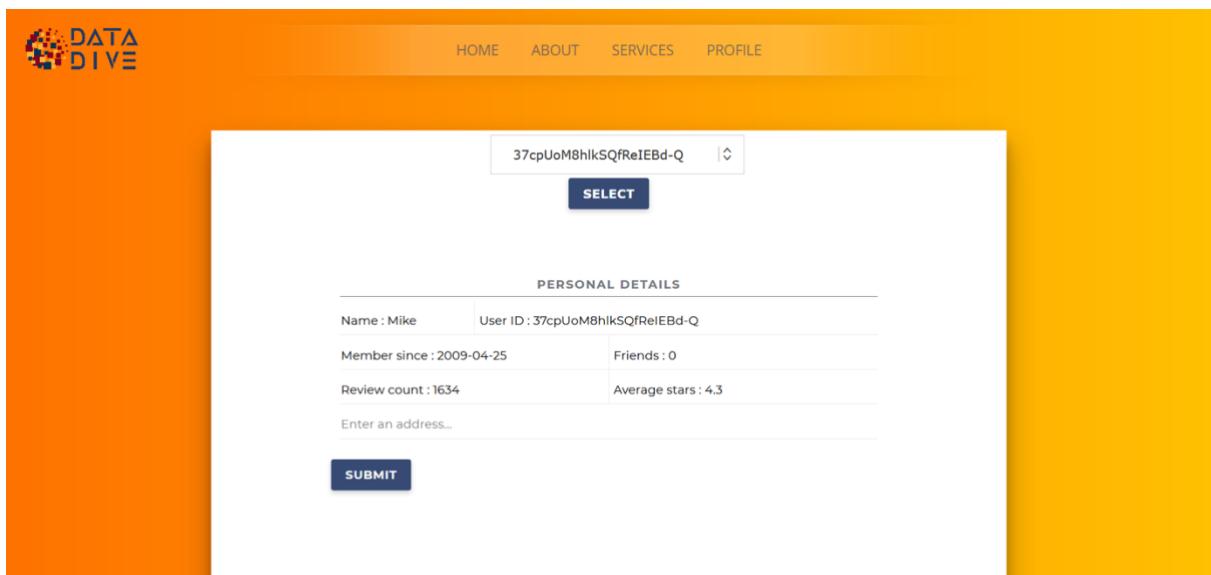


Figure 75 Page de profil de l'application

## Détails

Lorsqu'on clique sur un business (un restaurant par exemple), une page de détail (Figure 76) concernant ce restaurant vient s'afficher. On peut y voir diverses informations :

- Une section contenant des détails sur le business comme l'image du business, le nom et les coordonnées géographiques du business, sa notation ainsi que le nombre de commentaires sur ce dernier, le type de business et enfin les horaires de ce dernier.
- Une partie à gauche à propos des caractéristiques du restaurants.
- Une partie à droite concernant différentes statistiques sur ce dernier. En effet, on y voit deux onglets : un onglet pour les affluences quotidienne et hebdomadaires et un autre onglet pour les affluences mensuelles et annuelles.
- Et enfin, nous avons une section dédiée aux commentaires laissés par les utilisateurs à propos du business. On y retrouve le nom, la note, la date de visite ainsi que le commentaire.

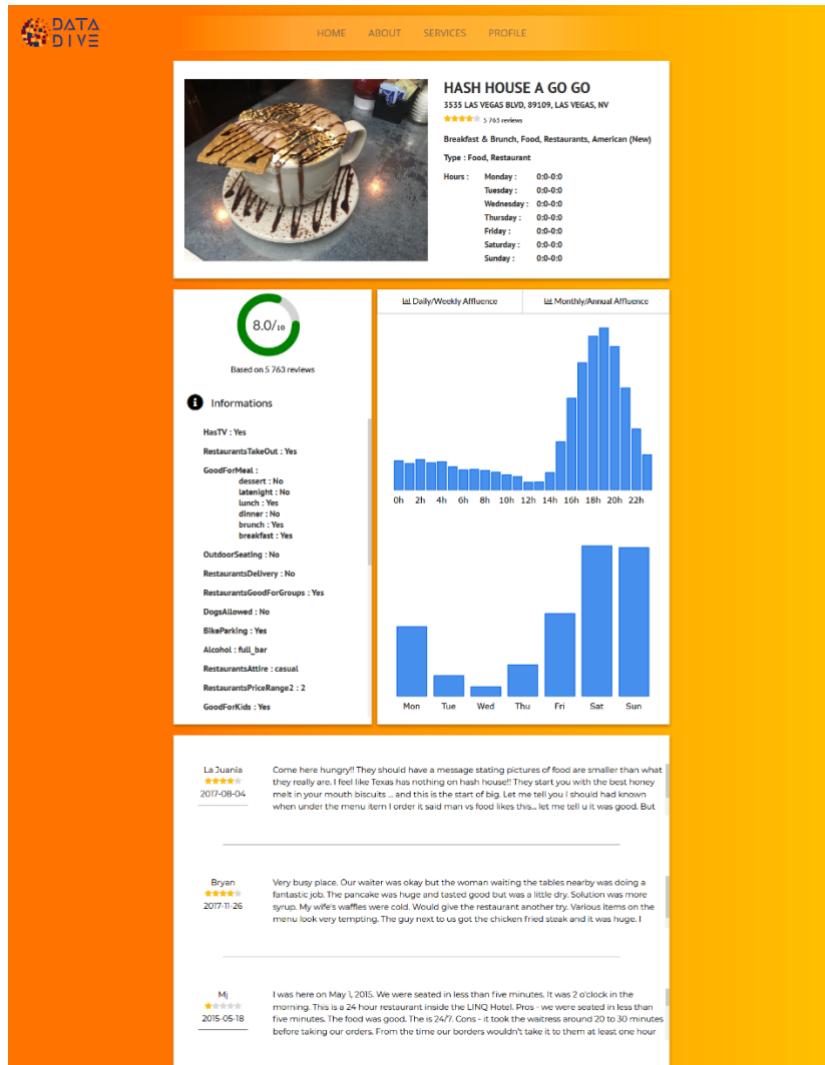


Figure 76 Page de détail

## 7.2. API du système de recommandations

Pour que l'interface web puisse communiquer au système de recommandations, nous avons écrit un API en Flask. Flask est un Framework Python facilitant le développement web. Il peut être notamment utilisé pour la création d'API. Cet API est capable de répondre aux demandes de l'utilisateur concernant la recommandation de business en fonction du profil d'utilisateur.

### Architecture

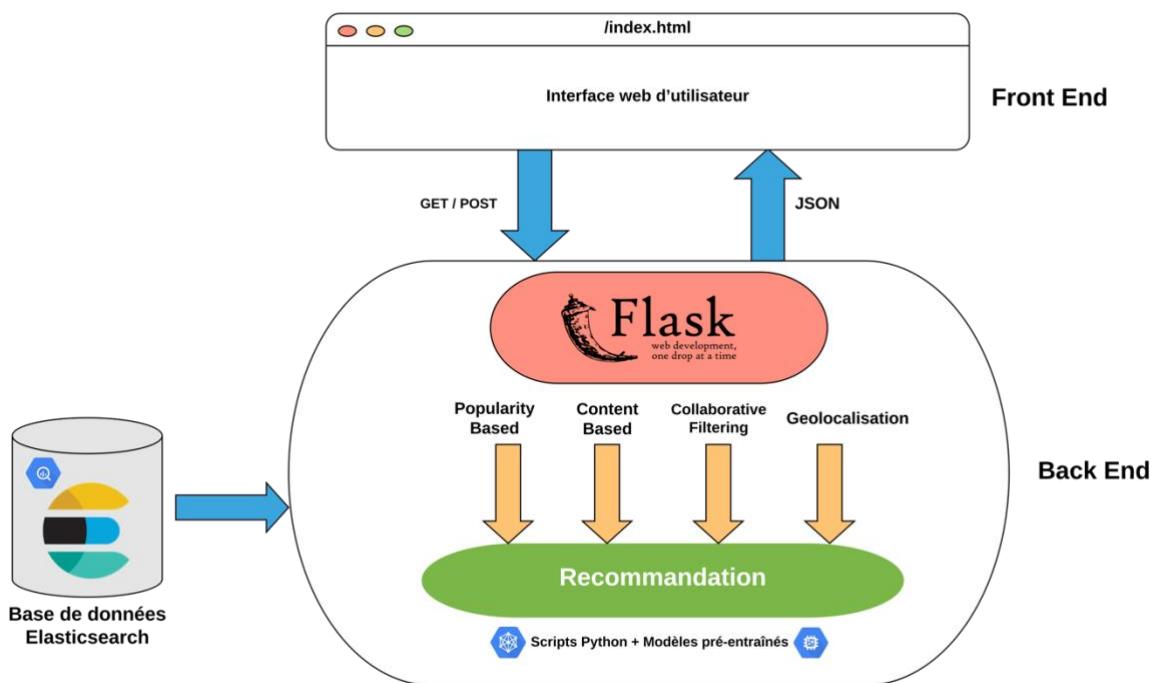


Figure 77 Architecture d'API de système de recommandation

### Comment ça marche ?

Une fois que l'utilisateur met la géolocalisation, l'interface web envoie une requête qui permet à l'API d'initialiser le moteur de recommandation en cherchant les businesses dans le périmètre de l'utilisateur. Par la suite, en fonction de profil d'utilisateur, nouveau ou existant dans la base de données, l'interface web envoie les requêtes correspondantes.

L'API possède les mêmes fonctionnalités que les points décrits dans la partie de système de recommandation qui sont les suivantes :

- Géolocalisation
- Popularity Based
- Content Based Keyword & Content Based Userid
- Collaborative Filtering Model Based (SVD)
- Collaborative Filtering Memory Based (Friendlist)

Les formats de la requête et de la réponse respectent la convention suivante :

- Requête : <http://34.76.62.206:5000/{methods}/recommend>
- Réponse : en JSON avec deux champs :
  - **rated** qui contient la liste des identifiants de business que l'utilisateur a évalué
  - **recommend** qui contient une liste imbriquée des businesses recommandés, on trouve le identifiant de business avec le score de recommandation

```
"rated": [ {business_id}],  
"recommended": [  
[ {  
    "business_id": {business_id},  
    "score": {score}  
} ... ]  
]
```

## Fonctionnalités & Appels de requêtes

### Geolocalisation:

Set la localisation de l'utilisateur

- Endpoint : http://34.76.62.206:5000/geo/set
- Paramètres :
  - userloc: <lat, lon> ou l'adresse d'utilisateur
- Réponse :

The screenshot shows the Postman interface for a POST request to <http://34.76.62.206:5000/geo/set>. The 'Body' tab is selected, showing form-data with two fields: 'userloc' (value: 36.1672559, -115.1485163) and 'userloc' (value: Las Vegas). The 'Headers' tab shows 9 headers. The 'Tests' tab contains the code: `1 Done retrieving businesses informations in the area by users address`. The status bar at the bottom indicates: Status: 200 OK | Time: 8.47 s | Size: 220 B | Save Response.

Figure 78 Réponse de la requête de géolocalisation

## Popularity Based

Rechercher les businesses les plus populaires

- Endpoint : <http://34.76.62.206:5000/pop/recommend>
- Paramètres :
  - categories : (optionel) la catégorie de business recommandé
  - topn : le nombre de résultats de la recommandation
- Réponse :
  - business\_id : l'identifiant du business recommandé
  - score : le score du business recommandé
  - distance : la distance entre le business et l'utilisateur

```
{  
    "rated": [],  
    "recommended": [  
        {  
            "business_id": "Vyadl8RsxaFaAFjm98lNTQ",  
            "count": 2,  
            "distance": 0.0218,  
            "geo_score": 0.5,  
            "pop_score": 0.5,  
            "ratings_avg": 5.0,  
            "score": 1.0,  
            "sum": 10.0  
        },  
        {  
            "business_id": "Nu0GjN3HeFomQ5ArsJackQ",  
            "content_score": 0.5255619287490845,  
            "distance": 0.6898,  
            "geo_score": 0.8658122576886764,  
            "score": 0.5936119945370029  
        }  
    ]  
}
```

Figure 79 Réponse de la requête de la recommandation Popularity Based

## Content Based

La recommandation par Content Based

- Endpoint : <http://34.76.62.206:5000/cb/recommend>
- Paramètres :
  - keyword : la recherche par mot clé pour les nouveaux utilisateurs
  - userid : l'identifiant de l'utilisateur à recommander
  - topn : le nombre de résultat de la recommandation
- Réponse de la recommandation par mot clé (Figure 80) :
  - business\_id: l'identifiant de business recommandé
  - score : le score du business recommandé
  - distance : la distance entre le business et l'utilisateur

```
{  
    "rated": [],  
    "recommended": [  
        {  
            "business_id": "Nu0GjN3HeFomQ5ArsJackQ",  
            "content_score": 0.5255619287490845,  
            "distance": 0.6898,  
            "geo_score": 0.8658122576886764,  
            "score": 0.5936119945370029  
        }  
    ]  
}
```

Figure 80 Réponse de la requête Content Based Keyword

- Réponse de la recommandation par userid (Figure 81 et Figure 82) :
  - rated : liste des identifiants business déjà évalué par l'utilisateur
  - business\_id : identifiant de business recommandé
  - score : le score du business recommandé
  - distance : la distance entre le business et l'utilisateur
  - input\_business\_id : l'identifiant de business sur lequel la recommandation est basé

```
{
  "rated": [
    "ZRsdVZmMjE8jLqHivluWLA",
    "4byedmwf21X0QQVEIWIA8g",
    "yooVq4aUthr7D0Qiz7yEA",
    "w7ZAphNGA5g4lMYMKg0tSg",
    "RrCgc8eAKbHu-2IpQXg6Rw",
    ...
  ]
}
```

Figure 81 Réponse de la requête Content Based Userid (1)

```
],
  "recommended": [
    [
      {
        "business_id": "nkMAcmooVzvBuLqywYYHwg",
        "content_score": 0.9524773359298706,
        "distance": 0.9404,
        "geo_score": 0.8154717663365542,
        "input_business_id": "ZRsdVZmMjE8jLqHivluWLA",
        "score": 0.9250762220112073
      },
      ...
    ]
  ]
}
```

Figure 82 Réponse de la requête Content Based Userid (2)

## Collaborative Filtering

### La recommandation par Collaborative Filtering

- Endpoint : <http://34.76.62.206:5000/cf/recommend>
- Paramètres :
  - userid : l'identifiant de l'utilisateur à recommander
  - topn : le nombre de résultat de la recommandation
- Réponse :
  - business\_id : id de business recommandé
  - score : le score de recommandé

```
{
  "rated": [
    "-3zffZUHoY8b0jGfPSoBKQ",
    "-C0okjildrY7UzezXcdEBw",
    "-FcZY7a7qqxTULtvuyJnq",
    "-NR4KqSGlhseVvJ-GFzfMA",
    "-Wz4zGE6zIbhAq3CKLTtuQ"
  ]
}
```

Figure 83 Réponse de la requête Collaborative Filtering (1)

```
],
  "recommended": [
    [
      {
        "business_id": "ighYKXZMLAc9UdV5VnR_AA",
        "score": 3.549854412143262
      },
      {
        "business_id": "oRbrnRAXGQetQDbkDhJp8w",
        "score": 2.5969174780080633
      }
    ]
  ]
}
```

Figure 84 Réponse de la requête Collaborative Filtering (2)

### 7.3. Tests utilisateur et certification

Des tests utilisateurs ont été réalisés afin d'avoir un point de vue externe. Ayant étant en confinement lorsque l'implémentation de l'interface a commencé, seuls les membres de nos familles respectives ont pu tester cette solution. Par la suite, des critiques constructives et des conseils nous ont été donnés pour l'ergonomie et la prise en main de l'application.

De plus, nous pouvons certifier notre solution car notre interface a été comprise de tous. En effet, ils ont eu une facilité à la prise en main lors de l'utilisation, avec très peu d'indication de notre.

Par ailleurs, nous avons également utilisé Sélénum<sup>5</sup>s qui est un outil d'automatisation de navigateur web. Ce dernier permet de réaliser des tests de surface. Dans un premier temps nous avons écrit des scénarios types tels que « lancer la page et faire une recherche », « choisir un business et vérifier ses heures », etc. Par la suite, il nous a suffi de lancer les tests pour que les scénarios s'exécutent de façon automatiquement avec des messages de logs, en cas de succès ou d'échec.

Et enfin, nous avons utilisé Postman afin de tester notre API. Le logiciel Postman permet de construire et d'exécuter des requêtes HTPP, de les stocker dans un historique afin de pouvoir les exécuter à nouveau. Il possède différent dont l'onglet « Test » que nous avons utilisé. L'historique est stockée est stocké dans ce qu'on appelle une collection Postman. Ainsi, nous avons regroupé différentes requêtes au sein d'une collection puis nous les avons lancé à chaque modification importante apportée à notre API.

### 7.4. Autres tests et certifications

Tout d'abord, l'intégration continue est une pratique consistant à intégrer de façon continue les changements apportés à un projet et ainsi détecter les erreurs rapidement pour pouvoir les corriger. Ainsi ayant utilisé cette pratique tout au long de notre, grâce aux différents tests et certifications réalisés dans les chapitres précédents, nous pouvons également certifier ce rendu final.

Par ailleurs, le W3C [23 w3c], World Wide Web Consortium, est un organisme international qui développe des standards pour le Web afin que les gens puissent communiquer efficacement à travers Internet. Cet organisme propose plusieurs services en ligne notamment 3 qui sont intéressants :

- Le **Markup Validator**<sup>6</sup> qui permet de vérifier si les pages sont écrites correctement
- Le **CSS Validator**<sup>7</sup> qui permet de vérifier les feuilles de style
- Le **Link Checker**<sup>8</sup> qui permet de vérifier la validité et l'accessibilité des différents liens présents sur la page

<sup>5</sup> <https://www.selenium.dev/>

<sup>6</sup> <https://validator.w3.org/>

<sup>7</sup> <https://jigsaw.w3.org/css-validator/>

<sup>8</sup> <https://validator.w3.org/checklink>

D'une part, après avoir lancé notre application web, nous avons récupéré le code de source de chaque page pour en vérifier la validité grâce au « Markup Validator » (Figure 85). Ainsi, ce service, utilisé également en tant qu'outil de débogage, nous permet d'assurer la qualité de notre application, aujourd'hui mais également dans le temps. D'autre part, à l'aide du « CSS Validator », nous avons pu obtenir la vérification de notre feuille de style (Figure 86) qui nous permet de certifier quant à la convivialité de notre application. Mais, le service Link Checker n'a pas pu être utilisé car notre application n'a pas été hébergé.

### Document checking completed. No errors or warnings to show.

Figure 85 Résultat du test de validité des pages de l'application web

### Résultats de la validation W3C CSS de TextArea (CSS niveau 3 + SVG)

#### Félicitations ! Aucune erreur trouvée.

Ce document est valide conformément à la recommandation [CSS niveau 3 + SVG](#) !

Figure 86 Résultat du test de validité des feuilles de style de l'application web

Enfin, il existe une théorie nommée « la règle des 3 clics » (3-click rules, en anglais), très connue, qui stipule qu'un internaute quitte une page web s'il ne trouve pas, en 3 clics maximum, l'information recherché. Bien qu'il existe toujours débat sur la véracité de cette règle de conception web, nous avons pensé qu'une recherche fastidieuse pouvait sûrement avoir un impact sur nos utilisateurs. Nous avons pu compter le nombre de clics que les différents utilisateurs de notre application réalisée, et nous pouvons certifier que notre solution respecte bien cette règle.

# Chapitre 8 Gestion de projet

Dans le chapitre précédent, nous nous sommes intéressés au rendu final en présentant l'interface utilisateur et les divers tests qui ont été effectués. Dans ce chapitre, nous allons nous focaliser sur la façon dont le projet a été géré du début jusqu'à la fin.

## 8.1. L'organisation de l'équipe

### 8.1.1. L'équipe de projet

Avant toute chose, pour une bonne gestion de projet, nous avons constitué notre équipe composée des personnes suivantes : Ashraf Ali, Bayo Mohamed II, Fagot Benoît, Saleem-Ahamed Mina et Vu Hong-Phuc. En qualité de chef de projet, nous avons désigné Ashraf Ali. Tous les membres de cette équipe sont issus de la promotion Systèmes Intelligents et Distribués (SID).

### 8.1.2. Outils de gestion de projet

Différents outils ont été utilisés durant la réalisation de notre projet, afin d'assurer l'efficacité de la communication au sein de l'équipe, pour les différents aspects de celui-ci : la programmation et la rédaction.



Figure 87 Logo Jupyter



Figure 88 Logo GitLab

D'une part, pour la programmation, tout au long du projet, nous avons utilisé des serveurs différents pour réaliser ce projet (Figure 87). Chaque serveur, dont nous allons vous faire la description par la suite, a un rôle différent dans l'implémentation globale de ce projet..

## Serveur de base de données Elasticsearch

Utilisation : base de données d'Elasticsearch et son interface web Kibana

Hébergement : Alibaba Cloud

Adresse IP publique accessible :

- 47.91.72.40 via port 9200 (Elasticsearch),
- 5601 (Kibana)
- et 22 (SSH)

Caractéristiques :

- 2 cores de vCPU,
- 4GB de RAM
- et 40 GB de stockage HDD

Services : La base de données Elasticsearch et l'interface web Kibana sont configurées comme des services lancés en permanence. Ces deux services sont accessibles depuis n'importe quel adresse IP.

Applications installées :

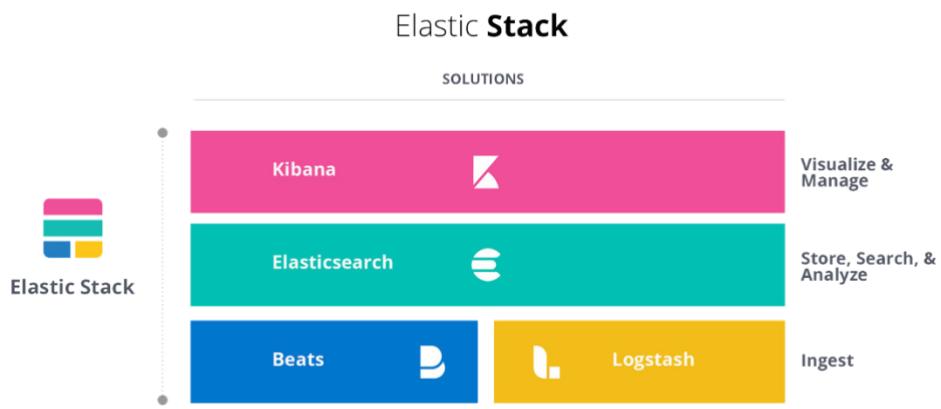


Figure 89 Applications installées sur le serveur hébergé sur Alibaba Cloud

## Serveur d'API et de Jupyter Notebook

Utilisation : API de l'application et également l'environnement de travail Notebook pour tester les différentes solutions et implémentations du système de recommandations. Nous stockons également les fichiers originaux de Yelp dataset sur ce serveur.

Hébergement : Google Cloud

Adresse IP publique accessible :

- 34.76.62.206 via port 8888 (Jupyter Notebook),
- 5000 (API)
- et 22 (SSH)

Caractéristiques :

- 4 cores de vCPU,
- 25GB de RAM
- et 50GB de stockage HDD

Services : L'environnement de travail Jupyter Notebook est démarré comme un service donc tout le monde peut y accéder par l'interface pour coder et tester les implémentations. API Flask est enregistré comme un service et fonctionne en permanence pour répondre aux requêtes d'API de l'extérieur.

Applications installées :



Figure 90 Applications installées sur le serveur hébergé sur Google Cloud

## Accès aux environnements de travail

En fonction du besoin, l'accès aux serveurs se fait de deux façons : par SSH ou par le navigateur web.

### **SSH**

La communication par ce protocole nous permet de modifier ou installer les composants et applications sur les serveurs. Le transfert de données peut aussi être effectué via ce protocole, ouvert sur le port 22. Pour faciliter la gestion SSH entre les différents serveurs, nous avons décidé d'utiliser le client SSH Terminus (Figure 91). Cette application nous permet d'accéder aux serveurs depuis n'importe quel appareil (PC, Mac, iPhone) via SSH et elle gère également le transfert de données via SFTP.

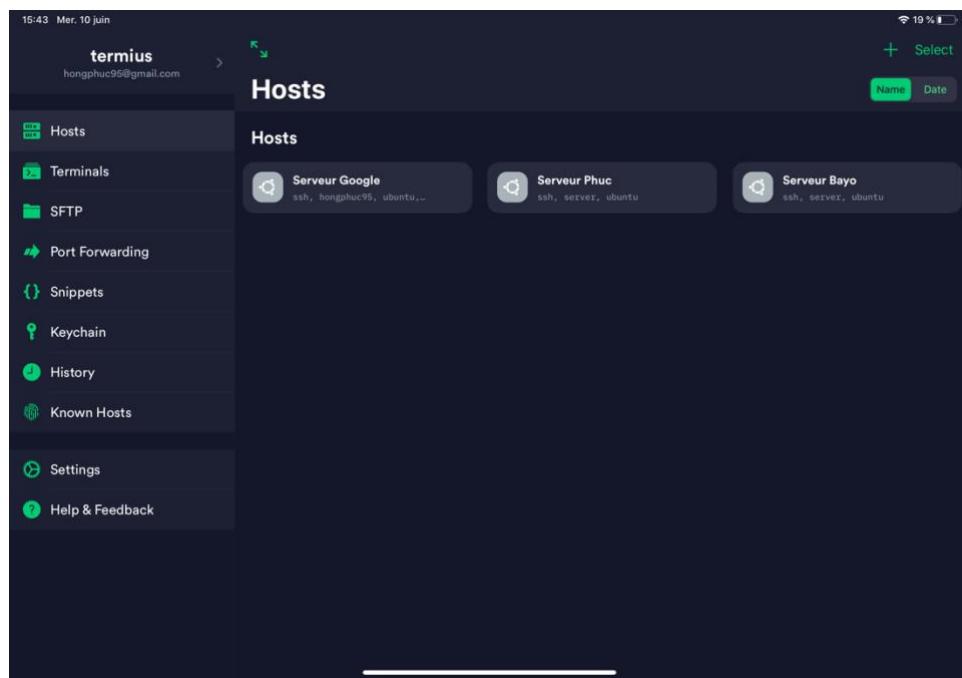


Figure 91 Page d'accueil sur la gestion de multiples hôtes de Terminus

```

Serveur Google ...
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 5.3.0-1020-gcp x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Wed Jun 10 13:43:56 UTC 2020

System load: 0.0          Processes:           136
Usage of /: 66.1% of 48.29GB  Users logged in:   1
Memory usage: 17%          IP address for ens4: 10.132.0.2
Swap usage:  0B

* MicroK8s gets a native Windows installer and command-line integration.
  https://ubuntu.com/blog/microk8s-installers-windows-and-macos

* Canonical Livepatch is available for installation.
  - Reduce system reboots and improve kernel security. Activate at:
    https://ubuntu.com/livepatch

39 packages can be updated.
0 updates are security updates.

*** System restart required ***
Last login: Tue Jun  9 17:06:40 2020 from 91.171.159.70
hongphuc95@instance-1:~$ cat /proc/sys/vm/swappiness
00
hongphuc95@instance-1:~$ cat /proc/cpuinfo
display all 196 possibilities? (y or n)
hongphuc95@instance-1:~$ cat /proc/cpuinfo
processor       : 0
vendor_id      : GenuineIntel
cpu family     : 6
model          : 63
model name    : Intel(R) Xeon(R) CPU @ 2.30GHz
stepping        : 1
microcode      : 0x1
cpu MHz        : 2300.000
cache size     : 46080 KB
physical id    : 0
siblings        : 4
core id        : 0
cpu cores      : 2
apicid         : 0
initial apicid : 0
fpu             : yes

```

Figure 92 Accès au serveur via SSH sur Terminus

## Navigateur web

L'accès par navigateur web est une alternative si nous ne voulons pas modifier les fonctionnalités principales des serveurs, nous pouvons passer par l'interface web de Jupyter Notebook accessible depuis 34.76.62.206:8888. Cette interface nous permet de téléverser ou télécharger le jeu de données et notebooks dans lesquels nous faisons du codage. Chaque personne dans le groupe a un propre dépôt pour travailler et collaborer ensemble. Voici la photo de page d'accueil par défaut quand on se connecte à l'interface via le navigateur Safari :

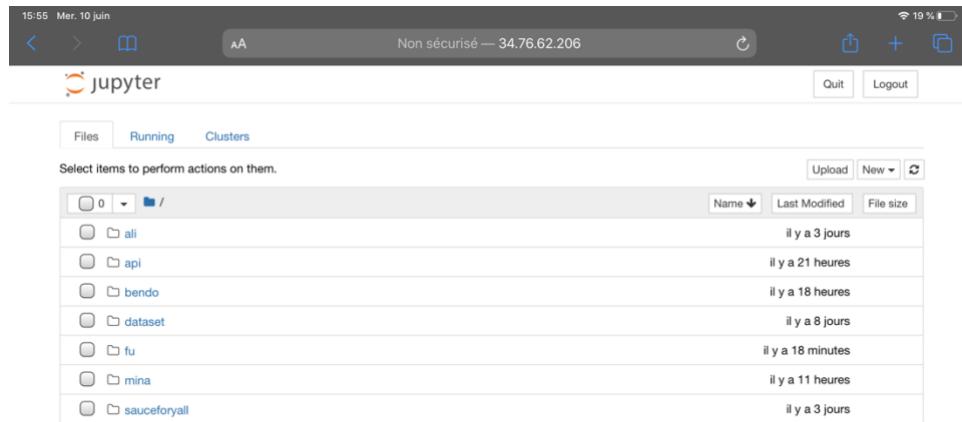


Figure 93 Racine de l'interface web Jupyter Notebook

Par ailleurs, pour la solution finale, nous avons déposé le code de notre projet sur le logiciel GitLab (Figure 88), pour le rendre accessible à tous.

D'autre part, pour toute la documentation, nous avons décidé d'utiliser Microsoft OneDrive (Figure 95). En effet, nos documents comme ce rapport, le slide de soutenances, le compte rendu des réunions et divers documents ont été partagés à toute l'équipe sur cette espace de stockage pour pallier aux contraintes de conflits (Figure 94).

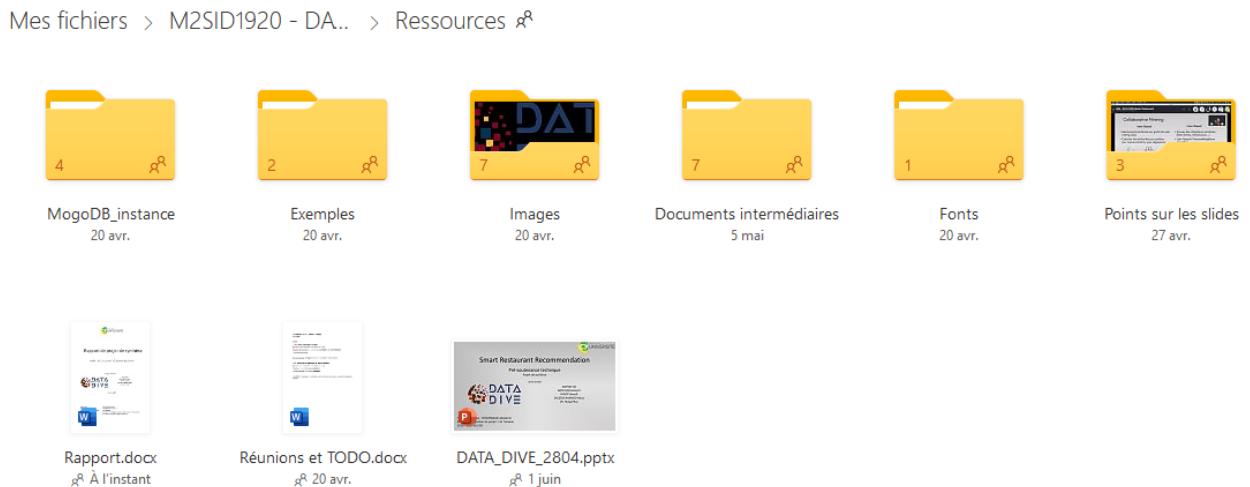


Figure 94 Dossier principal sur OneDrive

En ce qui la rédaction de notre rapport, nous avons souhaité utiliser LaTex grâce au plateforme Overleaf, un éditeur en ligne. Mais la version gratuite de cette plateforme ne permet la collaboration que de deux personnes à la fois. De ce fait, nous avons choisi d'utiliser Microsoft Word (Figure 96).



Figure 95 Logo Microsoft OneDrive



Figure 96 Logo Microsoft Word

Ensuite, pour une bonne organisation du travail et avoir une vue d'ensemble sur l'avancement de notre projet à tout moment, nous avons répertorié toutes les tâches de notre projet sur Trello (Annexe 11). Nous avons utilisé diverses étiquettes (Figure 97) pour classer nos tâches par thème(s), ainsi faciliter la lecture. Grâce à ce tableau, la répartition des tâches est également visible. Les tâches sont placées dans différentes colonnes selon leur progression :

- « TODO » comporte les tâches à effectuer.
- « IN PROGRESS » comporte les tâches en cours de progression.
- « TEST » comporte les tâches réalisées mais qui doivent être en cours de test par une personne qui n'a pas participé à sa réalisation.
- « DONE » comporte les tâches terminées.
- « Appointment » comporte la liste des réunions passées et à venir.



Figure 97 Liste des thèmes du projet

Et enfin, nous avons eu recours au pair-programming pour l'élaboration des fonctionnalités un peu plus compliquées. À partir de mi-mars, ces séances de pair-programming ont pu quand même être effectuées à distance grâce au logiciel Skype (Figure 98).



Figure 98 Logo Skype

### 8.1.3. Répartition des tâches

Afin de débuter au mieux notre projet, nous avons réalisé un planning qui sera présenter dans la section suivante (8.1.4), en prenant bien en considération les dates butoirs suivantes :

- 22 octobre 2019 : choix de l'équipe et du projet
- 17 novembre 2019 : rendu du cahier des charges
- 28 avril 2020 : pré-soutenance technique
- 03 juin 20120 : rendu du rapport bêta 1
- 10 juin 20120 : rendu du rapport bêta 2
- 11 juin 20120 : rendu du rapport final
- 18 juin 2020 : soutenance finale
- 19 juin 2020 : démonstration finale

Début février, étant en pleine phase de conception, nous avons réalisé une répartition des tâches (Tableau 8) plus en détail afin de débuter proprement la phase de programmation.

	Technique(s)	Domaine(s)	Responsable(s)
Data	Sentiment analysis	Natural Language Processing	Ali & Phuc
	Clustering	Machine Learning	Ali & Phuc
	Remplir les champs vides	Machine Learning & divers	Ali & Mina
	Pattern mining	Datamining Mining & divers	Mina
IHM	Schéma + Prototype	-	Ali & Mina
	Démo	-	Ali
Algorithmes	Système de recommandations	Collaborative filtering, Content-based filtering, Hybrid System, API	Benoit & Phuc
Géolocalisation	Simulation	-	Mohamed
	Système de recommandations basées sur la géolocalisation	-	Mohamed

Tableau 8 Répartition des tâches

### 8.1.4. Releases

Nous avons découpé notre projet en 3 releases distinctes avec une importante phase de conception qui nous a pris un plus de 2 mois. Par ailleurs, la livraison de notre première release a connu un léger retard à cause de la crise sanitaire survenue en plein milieu. Une courte release 2 a été programmé par la suite pour être sûr d'atteindre l'objectif attendu. Et enfin, une dernière release, en guise du rendu final, a été prévue.

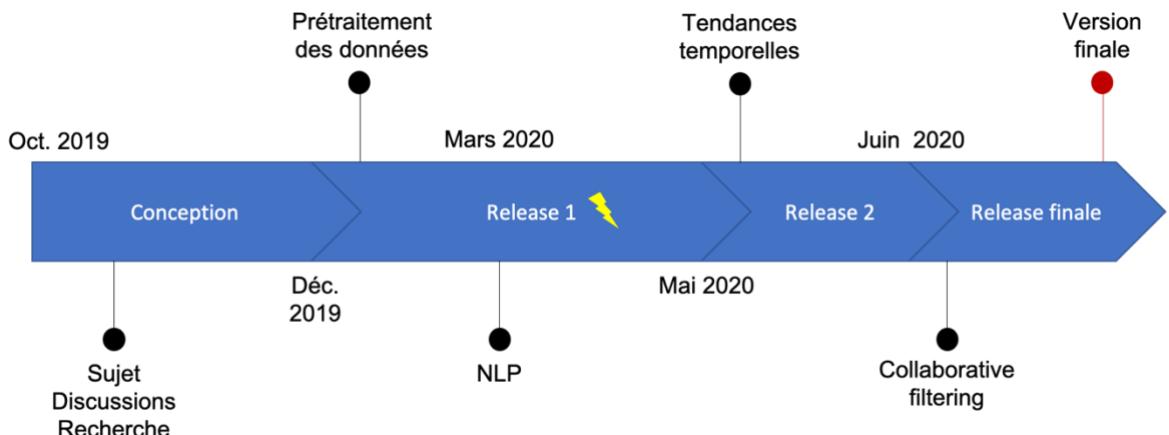


Figure 99 Planification des releases

## Conception

Cette phase, qu'on pourrait appeler release 0, était une étape importante à notre projet. Après avoir choisi le sujet de notre projet, nous avons rédigé les objectifs du projet et choisi notre environnement de travail. Nous avons créé ensuite, grâce à nos données, le **Modèle Conceptuel de Données** (Annexe 2) pour mieux comprendre et analyser nos données. Par la suite, de nombreux réunions et recherches ont été réalisé.

## Release 1

Durant cette release, une grande partie de notre projet a été réalisé. Le processus de prétraitement des données s'est divisé en plusieurs parties. Nous avons dû analyser nos données plus en profondeur afin de pouvoir les prétraiter. À la fin de ce processus, nous avons pu obtenir un dataset plus complet.

C'est également durant cette release, nous avons commencé à nous intéresser au traitement automatique du langage naturel, **Natural Language Processing** en anglais.

## Release 2

Cette deuxième release consistait en grande partie à la mise en place de l'architecture applicative du projet. Nous avons pu implémenter la structure du projet avec le Framework Django, la liaison avec le moteur de recherche Elastic et un début d'interface graphique.

Du côté des données, nous avons extrait les tendances temporelles des données afin de les utiliser dans les divers algorithmes.

## Release finale

Durant cette release, nous sommes concentrés sur l'interface graphique notamment en créant notre propre feuille de style. Nous avons implémenté le header et la barre de recherche ainsi que le menu de navigation. De plus, nous avons utilisé l'API YELP afin d'afficher les images des différents restaurants.

Par ailleurs, le système de recommandation était à un stade avancé donc nous avons commencé à l'intégrer à l'interface utilisateur.

## 8.2. Méthode de gestion

### 8.2.1. Adaptation de l'environnement de travail technique pour une meilleure productivité

Le projet Data Dive est un projet purement logiciel, utilisant des données qualifiables de “Big Data”. Cela requiert donc la mise en place d’un environnement de travail particulier et adapté à la manipulation de ces données.

Les données sont fournies par YELP sous la forme de fichiers volumineux au format JSON, elles sont donc simplement accessibles localement par les utilisateurs. La première étape est donc de mutualiser les données pour chaque membre du projet, en faisant l’étude des serveurs de gestions de base de données disponibles pour élire le plus adéquat à la structure et au volume des données. Nous avons opté pour un serveur ElasticSearch, qui utilise la bibliothèque open source Lucene afin d’indexer et rechercher les informations issues de collections de données sous le format JSON, ElasticSearch restant performant sur des hauts volumes de données. Ce choix a été appuyé par l’expérience de deux de nos membres qui ont déjà recours à ce SGBD dans leur environnement professionnel. Nous pouvons désormais requérir sur des données communes, et les ajouts/modifications de chacun sont pris en compte.

En tant qu’amateurs “Data Scientist”, nous sommes habitués à développer sur un environnement Jupyter Notebook, qui nous permet d’observer les données avec beaucoup plus de confort. Aussi, afin de pouvoir travailler sur une plateforme commune, nous avons loué un serveur et installé un répertoire Jupyter commun à tous les membres. Chacun peut donc avoir accès aux travaux des autres en temps réel, avec cependant le compromis de travailler sur une mémoire commune. La mémoire du serveur initial était de 4GB, nous avons vite pris la décision d’utiliser un serveur plus puissant et avons “upgradé” pour une mémoire de 25GB, mais nous avons parfois été limités du fait de la puissance de calcul que nos algorithmes requiraient. Lorsque ce cas de figure survenait, les autres membres devaient libérer la mémoire et stopper leur travaux le temps des tests du membre en question, ou basculer en local. Malgré ce détail, cette plateforme commune a nettement boosté notre productivité, car elle permettait de facilement basculer entre les fichiers de chacun et de récupérer des bouts de codes importants très rapidement, résoudre des erreurs de collègues, surveiller l’avancement global, etc. Un autre avantage non négligeable de la plateforme commune est que nous n’avons plus eu à nous soucier de soucis d’installations de modules et librairies qui peut créer des conflits selon le système d’exploitation utilisé (on pense notamment à Spark).

L’année 2020 a été frappée par un imprévu significatif qui est le coronavirus, cela a bousculé l’organisation et l’environnement de travail de beaucoup d’équipes. Cependant, notre projet étant purement logiciel, nous étions parmi les plus disposés à s’adapter à cette nouvelle contrainte. Nous avons utilisé différents outils de communications, comme Skype qui nous a été introduit par M. LIU, proposant un service de messagerie instantané directement disponible sur l’ordinateur, et qui permet de garder un suivi des conversations, et puis Zoom, qui est plus “One Shot”, mais qui a l’avantage de proposer le contrôle d’écran à distance en plus du partage, cela nous a extrêmement facilité la tâche lorsqu’il s’agissait par

exemple de configurer des logiciels comme Postman (qui permet de requêter des APIs) ou Kibana (interface de gestion et requêtage pour ElasticSearch) pour les membres novices.

## 8.2.2. Organisation de l'auto-formation des technologies nécessaires pour le projet

### Le langage Python et ses librairies

En Data Science, on privilégie les langages non verbeux et disposant d'une communauté active (frameworks, librairies, documentation...). Python est lors de l'écriture de ce rapport toujours leader des langages pour les adeptes de Big Data.

PROGRAMMING LANGUAGES FOR DATA SCIENCE

Platform	2019 % share	2018 % share	% change
Python	65.8%	65.6%	0.2%
R Language	46.6%	48.5%	-4.0%
SQL Language	32.8%	39.6%	-17.2%
Java	12.4%	15.1%	-17.7%
Unix shell/awk	7.9%	9.2%	-13.4%
C/C++	7.1%	6.8%	3.7%
Other programming and data languages	6.8%	6.9%	-17.1%
Scala	3.5%	5.9%	-41.0%
Julia	1.7%	0.7%	150.4%
Perl	1.3%	1.0%	25.2%
Lisp	0.4%	0.3%	46.1%
Javascript	6.8%	na	na

Jelvix

Source: KDnuggets

jelvix.com

Figure 100 Classement des langages de programmation pour la Data Science

Bien que des langages comme Scala augmentent en popularité pour utiliser Hadoop-Spark, Python reste le langage le plus intuitif et polyvalent, proposant de nombreuses libraries Data Science et AI, et dans le monde du travail, les entreprises mettent de plus en plus de

projets “Data” en production, là où surgit la limite de R qui est plus un langage d’analyse qu’un langage “purpose”.

La plupart des membres de l’équipe ont découvert Python en première année de Master, utilisé pour le cours de PSSR (Probabilités et Statistiques pour le Signal et les Réseaux), d’autres s’étant auto-formés dans le passé. Le langage a l’avantage d’être très facile à apprendre une fois qu’on a rencontré des langages comme Java ou C, qui sont beaucoup plus rigoureux dans la syntaxe. L’équipe est d’accord pour considérer Python comme un langage agréable à utiliser, simple et intuitif. Il a cependant une faiblesse qui est la mauvaise gestion de la mémoire.

Pour ce projet, nous disposions de connaissances basiques sur la librairie de Machine Learning principale (sci-kit), de gestion de datasets (Pandas, Spark), d’opérations de matrices (Numpy) grâce aux cours de Machine Learning et de Systèmes et Applications Distribués 1 et 2. Cependant, nous manquions beaucoup de pratique, et avons dû utiliser beaucoup de documentation externe pour nous former sur la gestion de cas beaucoup plus complexes que ceux vus en cours.

## Elastic

Pour la majorité des membres, travailler avec Elastic était une première. Les membres avec de l’expérience ont donc dédié de leur temps pour expliquer le fonctionnement concret aux autres. Il a de plus été émis l’idée de créer des classes de requêtage par l’un d’entre eux (*Hong Phuc Vu*) afin de faciliter la transition “Requête SGBD classique” au format “Requête par nested arrays” d’Elastic. Celui-ci a donc écrit une classe Python avec des requêtes similaires aux Getters d’une classe Java pour chaque cas de figure récurrent. Il a aussi pensé à des cas de figures plus généraux et complexes, une fois que les membres se seraient un peu plus familiarisés avec le format.

En sacrifiant de son temps de travail, ce membre a permis aux autres de gagner en productivité et de créer du gain pour l’équipe sur le long terme.

## Le Framework Django

Le projet requiert des technologies que nous ne maîtrisons pas dès le début, notamment le Framework Django et le moteur de recherche Elastic avec sa suite ELK. Des recherches ont été faites dès le début du projet afin d’avoir un aperçu de l’utilisation de ces dernières, c’est-à-dire leur manière de fonctionner, évaluer la difficulté des requêtes pour Elastic, etc... Avec le temps et la pratique, accompagné d’heures de visionnages de tutoriels, aujourd’hui nous maîtrisons ces technologies de manière à pouvoir créer une application web stable.

# Chapitre 9 Conclusion et perspectives

## 9.1. Conclusion

L'objectif de ce projet était de créer un système de recommandations de restaurants utilisant diverses méthodes et technologies, et de concevoir et développer une application web afin de proposer ce système aux utilisateurs. Cette solution vise à faciliter la recherche de l'utilisateur et lui proposer des choix personnalisés. Elle permet de concilier une base de données importantes en volume et une facilité de recherche.

Nous savons que la réussite d'un projet nécessite une base solide en connaissance théoriques et techniques mais il faut faire en sorte qu'aucune étape ne soit négligée. C'est pourquoi nous avons consacré énormément d'importance à la conception du projet mais aussi au processus de pré-traitement des données. En effet, avant de débuter la programmation, nous avons analysé les données afin d'avoir une vue d'ensemble pour l'exposer aux membres de l'équipe. Suite à cela, nous avons pu poser les problématiques principales de notre projet qui reposent sur les domaines de la Data Science et sur les systèmes de recommandations. Bien que nous ayons scindé l'équipe afin de pouvoir bien répondre à chacune des problématiques, chaque membre de l'équipe a participé de près ou de loin à chaque fonctionnalité du projet.

### Big data

En premier lieu, pour la partie Big Data, nous avons collectés les données du site YELP. Suite à nos analyses, nous étions conscients que nous serons amenés à compléter les données grâce à diverses techniques. Étant une application de recommandation, les données doivent être complète afin de les exploiter pleinement et pouvoir recommander l'utilisateur final de la meilleure des manières.

### Solution finale

En second lieu, nous avons entamé notre travail par une étude sur les différentes solutions de recommandation et les différentes solutions algorithmiques qui peuvent répondre à notre problématique. Puis, nous avons analysé et spécifié les besoins nécessaires pour le développement de cette fonctionnalité, pour ensuite attaquer sa conception générale et son implémentation finale. Enfin, nous avons montré les résultats obtenus par notre application sous forme d'interface graphique. Ces résultats répondent aux besoins de l'utilisateur de par les la géolocalisation mais aussi des goûts de ce dernier.

### Acquisitions personnelles

Tout au long du projet nous avons pu mettre en pratique nos connaissances théoriques que nous avons acquises durant notre parcours universitaire, de la première année de licence à la fin du master 2. Ainsi, nous avons pu mettre en avant les domaines de la Data, de l'intelligence artificielle et de l'algorithmique. Par ailleurs, nous pouvons lister divers apports personnels quant à la réalisation de ce projet. Tout d'abord, ce projet nous a permis de manipuler les données d'une nouvelle manière et en plus grande quantité. En ce qui concerne l'intelligence artificielle et l'algorithmique, nous sommes en mesure de considérer cette

partie comme étant un vrai challenge. Et enfin, de par ce projet, nous avons essayé de nous rapprocher le plus possible du monde des applications de recommandations, en nous intéressant notamment à l'interface graphique et à la restitution des données pour l'utilisateur.

### **Nouvelles technologies**

Par ailleurs, ce projet nous a donné la possibilité de découvrir de nouveaux Framework comme Django, qui est utilisé par les plus grandes société comme Instagram, de nouvelles manières de stocker et requêter les données à l'aide du moteur de recherche distribué Elastic.

## **9.2. Perspectives**

Comme énormément de solutions, il en va de soi que notre application web peut être sujette à des améliorations suite à de nouveaux besoins. Mais de nouvelles fonctionnalités, que nous n'avons pas pu implémenter dans les temps, peuvent également y être ajouter.

### **Politiques de confidentialité**

Par exemple, le système de géolocalisation pourrait respecter certaines politiques de confidentialité. De ce fait, implémenter le “geo-indistinguishability” permettrait à l'utilisateur de ne pas transmettre ses coordonnées géospatiales précises à l'application mais plutôt une zone avec un périmètre.

### **Multi-plateformes**

L'application étant actuellement en version web, une autre amélioration est de convertir en application mobile. En effet, une version mobilité permet d'améliorer l'accessibilité de notre application tout en créant une proximité au quotidien avec les utilisateurs.

### **Mise à jour des données**

De plus l'intégration de nouvelles données est une chose inévitable dans le cas d'un système de recommandations car les goûts et envies des utilisateurs évoluent avec le temps. Nous pouvons donc envisager d'essayer de rendre notre application plus flexible afin d'intégrer plus facilement l'arrivée de nouvelles données. Ainsi, la mise à jour de tendances et de patterns, grâce aux actions des utilisateurs sur notre site web, devra être possible.

## **Changement de contexte**

Dans notre cas, nous avons choisi de réaliser un système de recommandation d'établissements comme des restaurants, des hôtels, des banques, des hôpitaux, etc. Cependant, un axe d'amélioration d'essayer d'adapter ce système afin qu'il fasse d'autres types de recommandation : par exemple, des films, des livres, des chansons, etc.

## **Système de pagination**

Actuellement, sur l'interface utilisateur, seules les 20 meilleures recommandations sont affichées sur une unique page. L'utilisateur pourrait alors ne pas forcément trouver le résultat recherché. Nous pouvons donc considérer l'ajout d'un système de pagination afin d'une part, obtenir plus de résultats, et d'autre part, bien organiser ces derniers.

# Bibliographie

1 données publiées

<https://www.planetoscope.com/Internet-/1523-informations-publiees-dans-le-monde-sur-le-net-en-gigaoctets-.html>

2 data

<https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data>

3 études sur la consommation

[https://www.wavestone.com/app/uploads/2018/03/Tendance\\_conso\\_FocusA4\\_VF\\_WEB-1.pdf](https://www.wavestone.com/app/uploads/2018/03/Tendance_conso_FocusA4_VF_WEB-1.pdf)

4 Trip Advisor Recommendation

[https://www.tripadvisor.com>ShowTopic-g1-i12104-k11463552-Trip\\_Advisor\\_Recommendation-Help\\_us\\_make\\_Tripadvisor\\_better.html](https://www.tripadvisor.com>ShowTopic-g1-i12104-k11463552-Trip_Advisor_Recommendation-Help_us_make_Tripadvisor_better.html)

5 YELP data

<https://www.yelp.com/dataset>

6 aws

[https://docs.aws.amazon.com/index.html?nc2=h\\_ql\\_doc\\_do\\_v](https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do_v)

7 scikit-learn

<https://scikit-learn.org/stable/>

8 seaborn heatmap

<https://seaborn.pydata.org/generated/seaborn.heatmap.html>

9 étude de tendances

[https://go.euromonitor.com/white-paper-EC-2020-Top-10-Global-Consumer-Trends.html?utm\\_source=partner\\_lecho&utm\\_medium=article&utm\\_campaign=CT\\_WP\\_20\\_01\\_14\\_Top\\_10\\_GCT\\_2020](https://go.euromonitor.com/white-paper-EC-2020-Top-10-Global-Consumer-Trends.html?utm_source=partner_lecho&utm_medium=article&utm_campaign=CT_WP_20_01_14_Top_10_GCT_2020)

10 citation de paulo

Paulo Coelho (1998). Veronika décide de mourir (éd Anne Carrière, traduit par Françoise Marchand-Sauvagnargues). Paris

11 librairie pandas

<https://pandas.pydata.org/pandas-docs/stable/reference/frame.html>

12 ipstack

<https://ipstack.com/>

13 utilisation de ipstack

<https://www.programmableweb.com/news/how-to-use-ipstack-api-free-geolocation-and-determine-timezone-language-and-currency/sponsored-content/2018/08/15>

14 formule haversine

<https://www.igismap.com/haversine-formula-calculate-geographic-distance-earth/>

15 nasa gov

<http://www.nasa.gov/centers/goddard/earthandsun/earthshape.html>

16 scikit-learn haversine

[https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.haversine\\_distances.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.haversine_distances.html)

17 internet

Jannach, D., Hegelich, K.: A case study on the effectiveness of recommendations in the mobile internet. In: Proceedings of the 3rd ACM Conference on Recommender Systems, pp. 205–208 (2009)

<https://dl.acm.org/doi/10.1145/1639714.1639749>

18 Forbes

Kirshenbaum, E., Forman, G., Dugan, M.: A live comparison of methods for personalized article recommendation at Forbes.com. In: Machine Learning and Knowledge Discovery in Databases, pp. 51–66. Springer, Berlin (2012)  
[https://link.springer.com/chapter/10.1007/978-3-642-33486-3\\_4](https://link.springer.com/chapter/10.1007/978-3-642-33486-3_4)

19 click-behavior

Liu, J., Dolan, P., Pedersen, E.R.: Personalized news recommendation based on click behavior. In: Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI '10, pp. 31–40 (2010)  
<https://dl.acm.org/doi/10.1145/1719970.1719976>

20 content-based filtering

<https://link.springer.com/article/10.1007/s11257-019-09231-w#article-info>  
[https://en.wikipedia.org/wiki/Recommender\\_system#Content-based\\_filtering](https://en.wikipedia.org/wiki/Recommender_system#Content-based_filtering)

21 numpy svd

<https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html>

22 formule score

[https://en.wikipedia.org/wiki/Collaborative\\_filtering#Memory-based](https://en.wikipedia.org/wiki/Collaborative_filtering#Memory-based)

23 w3c

<http://www.w3.org>

# Annexe

## Annexe 1 Yelp Dataset Challenge

<p><b>Title: Yelp Dataset Challenge</b></p> <p><b>Introduction:</b></p> <p>Yelp connects people to great local businesses. To help people find great local businesses, Yelp engineers have developed an excellent search engine to sift through over 115 million reviews and help consumers find the most relevant businesses for their needs.</p> <p>Yelp is proud to introduce a deep dataset for research minded academics from our wealth of data. If you've been looking for a rich set of data to train your models on and use in publications, this is it. Tired of using the same standard datasets? Want some real world relevance in your research project? This data is for you!</p> <p>How well can you guess a review's rating from its text alone? Can you take all of the reviews of a business and predict when it will be the most busy, or when the business is open? Can you predict if a business is good for kids? Has WiFi? Has Parking? What makes a review useful, funny, or cool? Can you figure out which business a user is likely to review next? How much of a business's success is really just location, location, location? What businesses deserve their own subcategory (i.e., Szechuan or Hunan versus just "Chinese restaurants"), and can you learn this from the review text? What are the differences between the cities in the dataset? There is a myriad of deep, machine learning questions to tackle with this rich dataset.</p> <p>If you are a student and come up with an appealing project, you'll have the opportunity to win one of up to ten Yelp Dataset Challenge awards for \$5,000. Yes, that's \$5,000 for showing us how you use our data in insightful, unique, and compelling ways.</p> <p>Additionally, if you win the challenge and then publish a research paper about your winning research in a peer reviewed academic journal, then you'll be awarded an additional \$1,000 as recognition of your publication. If you are published, Yelp will also contribute up to \$500 to travel expenses to present your research using our data at an academic or industry conference. There is no cost to enter, no purchase required, and you need not be a Yelp user or hold a registered Yelp account to participate.</p> <p><b>Challenge Summary:</b></p> <p>The Yelp Dataset Challenge begins January 24, 2017 and continues through June 30, 2017, however, Yelp may end the contest at any time at its sole discretion.</p> <p>Contest is open only to undergraduate and graduate university students. Participants must be 18 or older, currently enrolled in an accredited degree program in any field, and must be from countries not restricted by the Office of Foreign Assets Control of the U.S. Department of the Treasury.</p>	<ul style="list-style-type: none"><li>• Access to the Yelp Dataset is governed by the separate Dataset Challenge Academic Dataset Agreement.</li><li>• To qualify for a \$5,000 Grand Prize, you must submit your code and a writeup of your project to Yelp via <a href="http://www.yelp.com/dataset_challenge">http://www.yelp.com/dataset_challenge</a> before June 30, 2017. Up to ten individuals or team entrants may receive this Grand Prize. (See below for restrictions on team size and entry).</li><li>• Each Grand Prize winner may qualify for an additional \$1,000 Publication Prize if their work is published by a peer reviewed academic journal, and if published, an additional \$500 prize may be awarded to pay for attendance at an academic or industry conference to present your work.</li><li>• All prize awards are subject to Yelp's verification of your compliance with all stated rules.</li></ul> <p><b>Contest Rules, Terms and Conditions:</b></p> <p><b>Term:</b></p> <p>The Yelp Dataset Challenge ("Contest") entry period commences on January 24, 2017 (00:00:00 UTC) and continues until June 30, 2017 (23:59:59 UTC) (the "Contest Term"). No additional registrations or Contest submissions will be accepted after the close of the Contest Term. Yelp reserves the right to cancel the Contest at any time, in its sole discretion.</p> <p><b>Who can participate?</b></p> <p>Contest Participants may be individuals or teams. There is no restriction on team size. Each Participant consents to be bound by these Rules. For team Participants, one person from the team must be designated as the team leader. A team leader will be the sole contact for all communications with Yelp regarding the contest. Individual team members, however, must each satisfy the contest requirement for individual participants. While individual team members may be members of multiple teams; teams with an identical set of members are not permitted. Yelp may change any publicly displayed team data, including team names, if Yelp in its sole discretion deems it to be inappropriate. All Contest Participants, whether entering the Contest as individuals or team members, must also each agree to the separate Dataset Challenge Academic Dataset Agreement.</p> <p>All contest participants must be at least eighteen (18) years old, and be enrolled as an Undergraduate or Graduate at an accredited university in any field. While professors and other non-student advisers may consult with contest Participants, and may be identified as advisers on any submission made in connection with the Contest, such individuals cannot be contest Participants (or team members) themselves, must not be identified as co-authors on any submission, and are ineligible to receive any Contest prizes.</p> <p>Current and former employees, officers, interns, contractors and agents of Yelp, its subsidiaries, together with members of their immediate families (parent, child, sibling and spouse of each) and those living in their same households, or Yelp and any independent contractors supporting</p>
<p>the Contest, are ineligible to enter and participate in the Contest or be awarded or retain any Contest prize.</p> <p>Contest Participants may not be a national or permanent resident of any of the countries restricted by the Office of Foreign Assets Control of the United States Department of the Treasury. The Contest is void in those places, and where else prohibited or restricted by law. Yelp reserves the right to further limit, or restrict, participation in the Contest to any person at any time for any reason. All entry information and submissions shall be deemed collected, stored, and judged in the United States.</p> <p>It is each Contest Participant's sole responsibility to review, understand and abide by his or her employer's or academic institution's policies, if any, regarding eligibility to participate in the Contest, or receipt and retention of any Contest prizes. If Yelp learns that a Participant is in violation of their school's or employer's policies, they may be disqualified from the Contest and may not be awarded, or allowed to retain, any Contest prizes. Yelp disclaims any and all liability or responsibility for disputes arising between a student or employee and their school or employer related to this Contest.</p> <p>Participants agree that Yelp may contact them regarding potential employment opportunities at Yelp.</p> <p><b>Receipt and Judging of Qualifying Submissions:</b></p> <p>Submissions should reflect creative uses of the Dataset Challenge Academic Dataset. Submissions must be originally developed or implemented, and must not violate or infringe on any applicable law or regulation or third party rights.</p> <p>Participants may submit their project by uploading or submitting a link to a research paper or other document, blog, slide deck or video, or any other medium that sufficiently communicates their use of the Dataset to Yelp.</p> <p>The submission must be in English, with mathematical formulae as necessary. It must be written at a level sufficient for a practitioner in computer science to reproduce the results obtained by the participant, if any. Citations of previous related work should be referenced as appropriate. Submissions must include a writeup of your project. Participants shall provide submissions to Yelp through the following website: <a href="http://yelp.com/dataset_challenge">yelp.com/dataset_challenge</a> before June 30, 2017. Participants warrant that any source code is either fully or substantially developed and functions or will function as represented by the writeup or other description provided.</p> <p>In order for Yelp to verify the submission, each Participant will grant to Yelp (including its affiliates and subsidiaries, employees, agents, and contractors), and any testing partners, an irrevocable, royalty free, fully paid up, worldwide nonexclusive license under the Participants'</p>	<p>copyrights, patents or other intellectual property rights to use, review, assess, test and otherwise analyze the submissions and all their content in connection with this Contest.</p> <p>All participants shall grant to Yelp (including its affiliates and subsidiaries, employees, agents, and contractors), an irrevocable, royalty free, fully paid up, worldwide nonexclusive license under the Participants' copyrights, patents or other intellectual property rights in their submission to reproduce, distribute, display, and create derivative works from the submission and also to make, have made, use, sell, offer for sale, and import products.</p> <p><b>Award of Contest Prizes:</b></p> <p><b>Contest Prizes:</b></p> <ol style="list-style-type: none"><li>1. Grand Prizes: \$5,000 (USD)</li><li>2. Publication Prizes: \$1,000 (USD) (for Grand Prize winners only)</li><li>3. Conference Prizes: Up to \$500 (USD) for travel to, and attendance at, up to one (1) academic or industry conference to present your winning and published project (for Publication Prize winners only)</li></ol> <p>A Participant, whether an individual or a team, is eligible to win and be awarded only one (1) Grand Prize of \$5,000 per team. Teams with multiple members are responsible for splitting the prize amongst themselves. Up to 10 (ten) Participants may be chosen as Grand Prize winners, and Yelp reserves the right to choose fewer than 10 (ten) winners, including none, at its own discretion.</p> <p>Only Grand Prize winners are eligible to receive Publication and Conference Prizes, and each Grand Prize winner is only eligible to receive up to one (1) Publication Prize and up to one (1) Conference Prize per Participant (or in the case of teams, only one (1) per team), even if the Grand Prize winner is published in multiple publications or attends multiple conferences.</p> <p>For teams, any awarded prizes will be delivered to the team leader. We recommend teams determine ahead of time how prizes, if any, would be shared among team members. Participants, including teams, may only represent themselves, and may not represent any other entity (including any employer or academic institution).</p> <p>In the event a potential winner cannot be reasonably notified, verified or confirmed based on two (2) attempts, a Contest Prize is declined, or a potential winner is not in compliance with these Rules or eligible for award of a Contest Prize for whatever reason, Yelp may allow the Contest to remain open and select and verify an alternate potential prize winner in accordance with these Rules, or to not select any winners, at its sole discretion.</p>

Prizes will be distributed as described below; however, Yelp retains the right to change this distribution schedule in its own discretion. In the event of a team Participant winner, any prize will be distributed to the team leader on behalf of the team Participant.

(a) Grand Prizes will be distributed to winning Participants in lump sum payments within 46 weeks after Yelp has declared a winner.

(b) Publication Prizes will be distributed to winning Participants in lump sum payments within 46 weeks after Yelp has received sufficient proof of publication of research in an acceptable peer reviewed academic journal.

(c) Conference Prizes will be distributed to winning Participants in lump sum payments within 46 weeks after Yelp has received sufficient proof of attendance at a qualifying academic or industry conference. Such proof may include evidence of payment of registration, travel, and lodging expenses incurred in connection with the conference.

Any costs or expenses associated with the acceptance and use of any Contest Prize monies are the sole responsibility of the Contest Prize winners. A Contest Prize winner may make no substitutions or assignment of a Contest Prize other than as specified in these Rules.

#### General Contest Terms and Conditions:

**A. Delivery.** A Contest participant is solely responsible for the equipment and Internet access required to participate in the Contest. Yelp will not be responsible for lost, late, incomplete, stolen, misdirected, undelivered, delayed, inaccurate, garbled or illegible materials, entries, email or mail; or for any computer, telephone, cable, network, satellite, electronic or Internet hardware or software malfunctions, failures, connections, or availability, or for garbled or jumbled transmissions, or for service provider/Internet/Web site use net accessibility or availability, traffic congestion, or unauthorized human intervention, or for inaccurate capture of any information, or the failure to capture any such information. Yelp is not responsible for any incorrect or inaccurate information, whether caused by printing errors, Site users, tampering, hacking, or by any of the equipment or programming associated with or utilized in the Contest, and is not responsible for any error, omission, interruption, deletion, defect, delay in operation or transmission, communications line failure, theft or destruction or unauthorized access to the Contest Site or any communications means, or for any other error or failure of any kind in connection with or relating to this Contest, whether caused by any human, typographical, printing, mechanical, computer or other means, including without limitation errors or failures which may occur in connection with the processing or storing of information, the administration of the Contest, the uploading, processing, validating, posting and/or judging of entries, the announcement of the Contest Prizes or in any Contest related materials. Yelp is not responsible for injury or damage to any participants' or to any other person's computer related to or resulting from participating in this Contest or downloading materials from or use of the Site. Yelp shall

have no obligation to accept or consider entries delayed past the submission deadline, or any entries or communications submitted in a language other than English.

**B. Entries.** All activities relating to Participant's participation in the Contest and material submitted are subject to verification and/or auditing for compliance with these Rules and Participants agree to reasonably cooperate with Yelp concerning verification and/or auditing. In the event that Contest verification activity or an audit evidences noncompliance with the Rules or official Contest communications, as determined by Yelp, a Participant's continuing participation in any aspect of the Contest may be suspended or terminated at any time during or after the Contest. Yelp reserves the right, in its sole discretion, to suspend or cancel the Contest. In the event the Contest is cancelled, Yelp may, at its sole discretion, determine winners, if any, from among all eligible entries received up to the time of such cancellation, but it has no obligation to do so.

**C. Participants' Affidavits, Contest Consent and Releases.** As a prior condition of winning the Contest and receipt and acceptance of a Contest Prize, a participant may be required to complete, sign and return a nonexclusive grant of license to Yelp, an Affidavit of Eligibility and Liability Release and where lawful, a Publicity and Submission Release and License ("Publicity Release"). The Publicity Release will include consent by a winning participant to use their name, picture, voice, likeness, and the name, description and Application comprising winner's entry, worldwide in perpetuity, for the purposes of Contest related advertising, promotion and publicity, and additionally including those specific purposes disclosed on the Site, without further compensation, notification or permission, unless prohibited by law. Failure to complete and sign these documents within two (2) weeks may disqualify that entry and additional qualifying entries will be considered.

In addition, winning participants may be required to complete applicable U.S. tax withholding related documentation and, if necessary, provide proof of foreign status, as beneficiaries of U.S.-sourced promotional consideration.

**D. Tax Consequences.** Participation and receipt of benefits from the Contest will have tax consequences. Participants should speak with their tax advisers prior to participation and receipt and acceptance of Contest Prizes to prevent any undesired results. Should there be any local tax liability for participation in the Contest or the receipt of a Contest Prize, or costs or expenses relating to participation in the Contest or for the use and enjoyment of any Contest Prizes, or for any other reason, such taxes and/or expenses are the sole responsibility of the participant. Yelp will issue Contest related tax reporting, as may be required by applicable laws and regulations. As a condition of winning the Contest and receipt and use of any Contest Prize, participants agree to self report to applicable local taxing authorities, as may be required by local laws.

**E. Rules Amendment and Interpretation.** The Contest and its continuing terms and conditions, benefits and participation are offered to participants at the sole discretion of Yelp. Yelp reserves the right to amend or interpret these Rules or any other official Contest

communications and any element or elements arising under or relating to the Contest at any time, upon published notice to participants on the following site: [yelp.com/dataset\\_challenge](http://yelp.com/dataset_challenge). A participant shall be deemed to have notice of any such amendments or interpretations upon publication of the same and shall be deemed to have accepted such amendments or interpretations by virtue of a participant's continuing participation in the Contest. Should a participant not wish to continue to participate in the Contest pursuant to the prevailing Rules, as amended or interpreted, a participant may terminate participation in the Contest after providing a submission by notifying Yelp in writing.

**F. Arbitration, Adjudication and Remedial Process.** Yelp reserves the right to make all decisions relating to the Contest at its sole discretion, and such decisions by Yelp are final and binding. Participants in the Contest agree that:

(1) any and all disputes, claims, and/or causes of action arising out of or connected with the Contest, or any benefits received, or the administration of the Contest, shall be resolved individually, without resort to any form of class action, and exclusively by arbitration, to take place in the City and County of San Francisco in the State of California pursuant to the Rules of the American Arbitration Association, then effective.  
(2) any and all claims, judgments and awards shall be limited to actual out of pocket costs incurred, including costs associated with participating in the Contest and in no event shall Participant be entitled to receive attorneys' fees or other legal costs;  
(3) under no circumstances will a Participant be permitted to obtain awards for and Participants hereby waive all rights to claim punitive, incidental and consequential damages and any other damages, other than for actual out of pocket expenses, and;  
(4) any and all disputes, claims, and causes of action arising out of or relating to the Competition, the Data or any prize awarded must be brought within one (1) year after the occurrence of the event giving rise to such claim (if such claim is not filed, then that claim is permanently barred). Except where prohibited, all issues and questions concerning the construction, validity, interpretation and enforceability of these Rules, or the rights and obligations of Participant(s) and Yelp in connection with the Contest, shall be governed by, and construed in accordance with, the laws of the United States of America and the State of California, without regard for conflicts of law doctrine of any jurisdiction, and subject to resolution through arbitration only in the City and County of San Francisco, State of California, United States of America. By entry and participation in the Contest, Participants consent to personal jurisdiction in the City and County of San Francisco, State of California, United States of America, and agree that the City and County of San Francisco, State of California, United States of America will be the exclusive venue for any dispute resolution. Yelp's failure to exercise or enforce any right or provision under these Rules will not constitute a waiver of such right or provision and will not be deemed a waiver of any further rights hereunder.

**G. NO WARRANTY.** YELP, ITS AFFILIATED COMPANIES AND ALL OF THEIR RESPECTIVE OFFICERS, DIRECTORS, EMPLOYEES, REPRESENTATIVES AND AGENTS (COLLECTIVELY, THE "RELEASED PARTIES") EXPRESSLY DISCLAIM ANY AND ALL WARRANTIES OR CONDITIONS OF ANY KIND (WHETHER EXPRESS, IMPLIED,

STATUTORY OR OTHERWISE), INCLUDING BUT NOT LIMITED TO, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. UNDER NO CIRCUMSTANCES SHALL RELEASED PARTIES BE HELD RESPONSIBLE OR LIABLE FOR A PARTICIPANT'S USE OF THE INFORMATION AND/OR PRODUCTS PROVIDED AND/OR MADE AVAILABLE THROUGH THE CONTEST OR FOR ERRORS OR ANOMALIES RESULTING IN THE UNINTENDED OR ERRONEOUS PARTICIPATION, AWARD OF CONTEST PRIZES OR OTHER BENEFITS UNDER THE CONTEST TO PARTICIPANTS. RELEASED PARTIES OFFER NO ASSURANCES, GUARANTEES OR WARRANTIES OR CONDITIONS THAT THE CONTEST OR CONTEST WEBSITES WILL BE UNINTERRUPTED OR ERROR FREE AND DOES NOT GUARANTEE THE ACCURACY OR RELIABILITY OF ANY INFORMATION OBTAINED THROUGH THE CONTEST. RELEASED PARTIES ASSUME NO RESPONSIBILITY FOR ANY COMPUTER RELATED DAMAGES DUE TO DOWNLOADING MATERIALS. RELEASED PARTIES WILL NOT BE LIABLE AND ARE NOT RESPONSIBLE FOR DAMAGES OF ANY KIND RELATED TO A PARTICIPANT'S PARTICIPATION OR INABILITY TO PARTICIPATE IN THE CONTEST, WHETHER THE DAMAGES ARE DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL. FURTHER, BY PARTICIPATING IN THE CONTEST AND/OR ACCEPTING A CONTEST PRIZE, A PARTICIPANT AGREES THAT RELEASED PARTIES SHALL NOT BE LIABLE FOR, AND WILL BE HELD HARMLESS BY PARTICIPANT AGAINST, ANY LIABILITY FOR ANY DAMAGE, INJURY OR LOSS TO PERSON (INCLUDING DEATH) OR PROPERTY DUE IN WHOLE OR IN PART, DIRECTLY OR INDIRECTLY, TO ACCEPTANCE, POSSESSION, USE OR MISUSE OF A CONTEST PRIZE, PARTICIPATION IN ANY CONTEST PRIZE RELATED ACTIVITY, RELEASED PARTIES' USE OF ANY TENDERED ENTRY OR PARTICIPATION IN THIS CONTEST. SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES; THEREFORE THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY.

**H. Indemnification.** Participants agree to release, defend, indemnify and hold harmless Released Parties from and against, and accept all responsibility of any kind, including but not limited to financial, for any liability, claims, losses, damages or proceedings, including but not limited for death, (including reasonable attorneys' fees) relating to any actions taken by a participant, including any submissions, or anyone else using a participant's email or team password or otherwise purporting to act on participant's behalf in regard to the Contest (whether or not such use occurred with or without your permission). Participant also agrees to release, defend, indemnify and hold harmless the Released Parties from any and all liability, claims, losses, damages or proceedings, including but not limited for death (including reasonable attorneys' fees) relating to Prizes and any other matter in connection with a participant's participation in the Contest or Yelp's or its designees or Yelp's or its designees use of his/her Submission. These rules, terms and conditions specified here comprise the Contest's Official Rules and, together with the Dataset Challenge Academic Dataset Terms of Use and official Contest communications or publications by Yelp, shall govern and apply to all participation and activity arising out of and relating to the Contest and use of the Site. These Rules cannot be modified or superseded except by Yelp, at its sole discretion.

## Annexe 2 Analyse complète de données

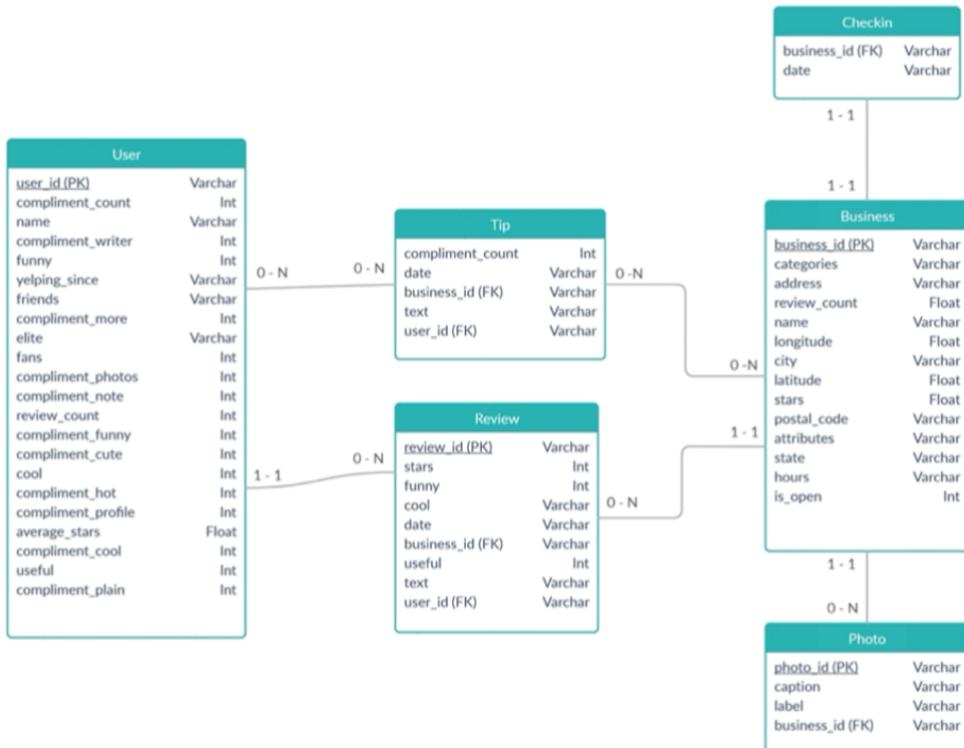
# Analyse de données (Yelp Dataset)

Grâce à la Suite Elastic (Logstash, Elasticsearch, Kibana), les données ont pu être analysées. En utilisant Logstash (type ETL), les données ont été envoyées vers Elasticsearch, qui est un moteur de recherche et d'analyse distribué, dans le but de les visualiser avec l'outil Kibana.

Les données sont composées de 6 fichiers json : business, checkin, photo, review, tip et user.

### I. Modélisation des données

Les données que nous avons à disposition sont reliées entre elles par le biais de clés étrangères. Comme nous pouvons le voir dans le schéma ci-dessous, nous avons différentes cardinalités entre les différentes tables.



## II. Business

Dans cette section, nous analyserons le contenu du fichier business.json (131 Mo). Ce fichier contient les informations à propos des restaurants répertoriés par Yelp.

- Nombre de données : **192 609**
- Nombre de champs : **14**

Champs	Type
categories	Varchar
address	Varchar
review_count	Float
name	Varchar
longitude	Float
city	Varchar
latitude	Float
stars	Float
postal_code	Varchar
attributes	Varchar
state	Varchar
business_id	Varchar
hours	Varchar
is_open	Int

Lors de l'analyse, nous avons remarqué que certains champs contenaient des valeurs vides, notamment le champs *attributes*. Cela nous facilitera la phase de data cleaning lors de laquelle certains champs seront supprimés afin de garder seulement les données pertinentes.

- Data quality

Nous avons alors classé les données par pourcentage de valeurs vides :

Fields	Null Rows	% Null
address	0	0
business_id	0	0
city	0	0
is_open	0	0
latitude	0	0
longitude	0	0
name	0	0
postal_code	0	0
review_count	0	0
stars	0	0
state	0	0
categories	482	0,25
hours.Thursday	46 706	24,25
hours.Friday	47 435	24,63
hours.Wednesday	47 452	24,64
hours.Tuesday	49 181	25,53
hours.Monday	56 842	29,51
hours.Saturday	66 748	34,65
attributes.BusinessAcceptsCreditCards	79 476	41,26
attributes.RestaurantsPriceRange2	84 430	43,83
attributes.BusinessParking	88 896	46,15
hours.Sunday	101 273	52,58
attributes.BikeParking	107 210	55,66
attributes.GoodForKids	126 299	65,57
attributes.RestaurantsTakeOut	130 532	67,77
attributes.OutOfDoorSeating	137 786	71,54
attributes.RestaurantsGoodForGroups	137 891	71,59
attributes.RestaurantsDelivery	140 087	72,73
attributes.RestaurantsReservations	140 322	72,85
attributes.WiFi	142 545	74,01
attributes.RestaurantsAttire	143 970	74,75
attributes.Alcohol	144 146	74,84
attributes.HasTV	144 511	75,03
attributes.Ambience	144 530	75,04
attributes.ByAppointmentOnly	145 756	75,67
attributes.NoiseLevel	148 730	77,22
attributes.Caters	151 981	78,91
attributes.GoodForMeal	162 683	84,46
attributes.WheelchairAccessible	172 650	89,64
attributes.RestaurantsTableService	175 436	91,08
attributes.BusinessAcceptsBitcoin	179 524	93,21
attributes.DogsAllowed	185 174	96,14
attributes.AcceptsInsurance	185 359	96,24
attributes.Music	187 360	97,27
attributes.HappyHour	187 403	97,3
attributes.GoodForDancing	187 867	97,54
attributes.CoatCheck	189 103	98,18
attributes.Smoking	189 111	98,18
attributes.BestNights	189 133	98,2
attributes.DriveThru	189 417	98,34
attributes.BYOB	191 186	99,20
attributes.HairSpecializesIn	191 603	99,48
attributes.Corkage	191 947	99,66
attributes.AgesAllowed	192 486	99,94
attributes.DietaryRestrictions	192 550	99,97
attributes.BYOB	192 581	99,99

### III. Checkin

Dans cette section, nous analyserons le contenu du fichier checkin.json (389 Mo).

- Nombre de données : **161 950**
- Nombre de champs : **2**

Champs	Type
<b>business_id</b>	Varchar
<b>date</b>	Varchar

- Data quality :

Cette table ne contient pas de champs vide.

Fields	Null Row	% Null
business_id	0	0
date	0	0

### IV. Photo

Dans cette section, nous analyserons le contenu du fichier checkin.json (389 Mo).

- Nombre de données : **200 000**
- Nombre de champs : **4**

Champs	Type
<b>caption</b>	Varchar
<b>label</b>	Varchar
<b>business_id</b>	Varchar
<b>photo_id</b>	Varchar

- Data quality :

Ici, seul le champ *caption* contient des valeurs vides :

Fields	Null Row	% Null
caption	107850	53.93
photo_id	0	0
business_id	0	0
label	0	0

## V. Review

Dans cette section, nous analyserons le contenu du fichier review.json (4 970 Mo).

- Nombre de données : **6 685 900**

- Nombre de champs : **9**

Champs	Type
review_id	Varchar
stars	Int
funny	Int
cool	Varchar
date	Varchar
business_id	Varchar
useful	Int
text	Varchar
user_id	Varchar

- Data quality :

Il n'y a pas de champ vide dans ces données-là :

Fields	Null Row	% Null
review_id	0	0
stars	0	0
funny	0	0
cool	0	0
date	0	0
business_id	0	0
useful	0	0
text	0	0
user_id	0	0

## VI. Tip

Dans cette section, nous analyserons le contenu du fichier tip.json (233 Mo).

- Nombre de données : **1 223 094**
- Nombre de champs : **5**

Champs	Type
compliment_count	Int
date	Varchar
business_id	Varchar
text	Varchar
user_id	Varchar

- Data quality :

Il n'y a pas de champ vide dans ce jeu de données :

Fields	Null Row	% Null
compliment_count	0	0
date	0	0
business_id	0	0
text	0	0
user_id	0	0

## VII. User

Dans cette section, nous analyserons le contenu du fichier user.json (2 310 Mo).

- Nombre de données : **1 637 138**
- Nombre de champs : **22**

Champs	Type
compliment_count	Int
name	Varchar
compliment_writer	Int
funny	Int
yelping_since	Varchar
friends	Varchar
compliment_more	Int
elite	Varchar
fans	Int
compliment_photos	Int
compliment_note	Int
review_count	Int
compliment_funny	Int
compliment_cute	Int
cool	Int
compliment_hot	Int
compliment_profile	Int
average_stars	Float
compliment_cool	Int
useful	Int

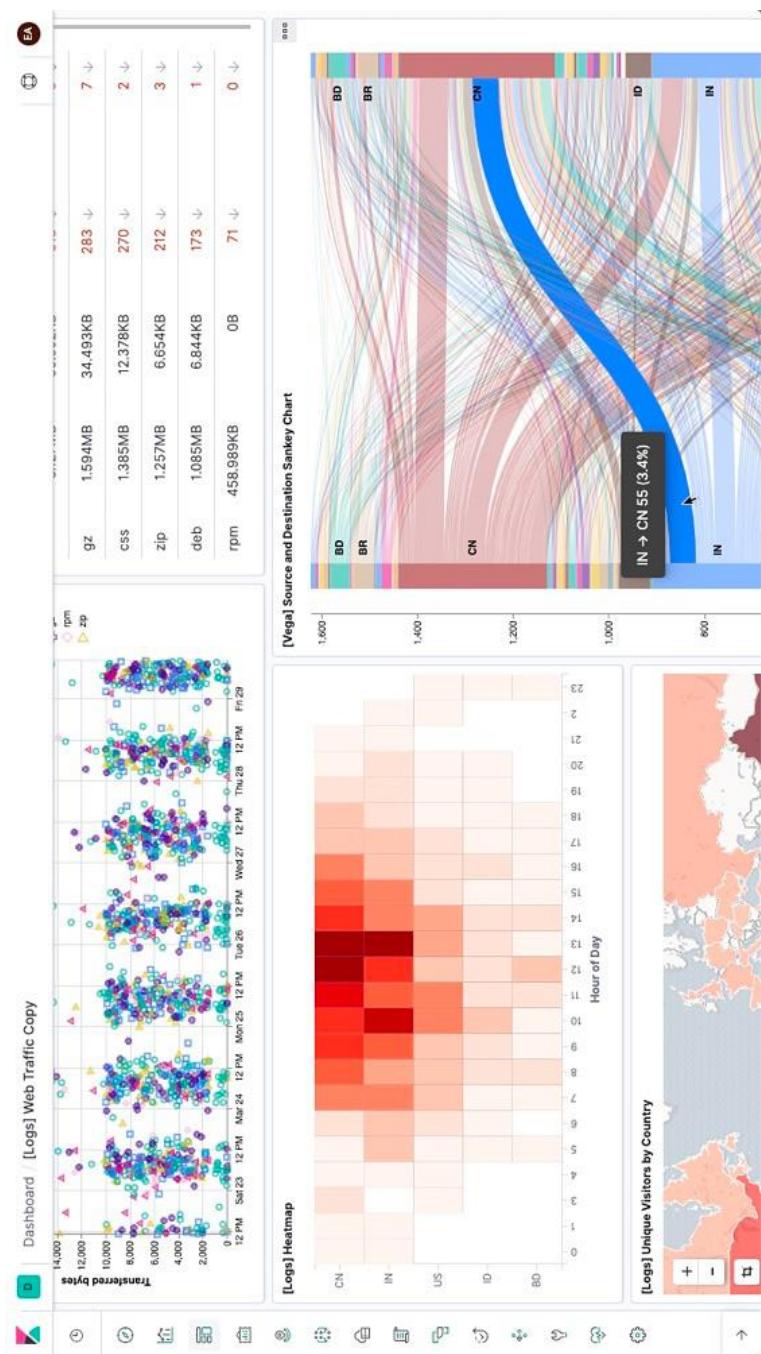
Annexe 3 Fichier de configuration Logstash

```
input {
    file{
        path=>"yelp_dataset/*.json"
        start_position=>"beginning"
        sinedb_path =>"NUL"
    }
}

filter {
    json {
        source => "message"
    }
    grok {
        match => ["path", "yelp_dataset/%{GREEDYDATA:index_name}"]
    }
    mutate {
        remove_field => [ "host", "path", "@timestamp", "@version", "message" ]
    }
}

output {
    stdout {
        codec => rubydebug
    }
    elasticsearch {
        index => "%{index_name}"
        hosts => ["localhost:9200"]
    }
}
```

Annexe 4 Dashboard Kibana



**ENTRÉES:**  $T, \epsilon$ 

$L_1 = \{\text{ensemble de 1-item qui apparaissent dans au moins } \epsilon \text{ transactions}\}$

$k = 2$

**tant que**  $L_{k-1} \neq \emptyset$  faire

$C_k = \text{Generate}(L_{k-1})$  /\* Génère l'ensemble de candidats \*/

**pour**  $t \in T$  faire

$C_t = \text{SousEnsemble}(C_k, t)$  /\* Sélection des candidats de  $C_k$  présents dans  $t$  \*/

**pour**  $c \in C_t$  faire

$\text{count}[c] = \text{count}[c] + 1$

**fin pour**

**fin pour**

$L_k = \{c \in C_k | \text{count}[c] \geq \epsilon\}$

$k = k + 1$

**fin tant que**

**retourner**  $\bigcup_k L_k$

Annexe 6 Liste des attributs avec les valeurs possibles

attribut	possible_values
RestaurantsReservations	['True', 'False', 'None']
GoodForMeal	['None', {}]
GoodForMeal_desert	[False]
GoodForMeal_latenight	[False]
GoodForMeal_lunch	[True]
GoodForMeal_dinner	[True]
GoodForMeal_brunch	[False]
GoodForMeal_breakfast	[False]
BusinessParking	['None', {}]
BusinessParking_garage	[False]
BusinessParking_street	[False]
BusinessParking_validated	[False]
BusinessParking_lot	[True]
BusinessParking_valet	[False]
Caters	['True', 'False', 'None']
NoiseLevel	["'u'loud'", "'u'average'", "'u'quiet'", "'u'average'", "'quiet'", "'u'very_loud'", "'loud'", "'very_loud'", 'None']
RestaurantsTableService	['True', 'False', 'None']
RestaurantsTakeOut	['True', 'False', 'None']
RestaurantsPriceRange2	['2', '1', '3', '4', 'None']
OutdoorSeating	['False', 'True', 'None']
BikeParking	['False', 'True', 'None']
Ambience	['None', {}]
Ambience_touristy	[False]
Ambience_hipster	[False]
Ambience_romantic	[False]
Ambience_divey	[False]
Ambience_intimate	[False]
Ambience_trendy	[False]
Ambience_upscale	[True]
Ambience_classy	[False]
Ambience_casual	[False]
HasTV	['False', 'True', 'None']
WiFi	["'u'no'", "'u'free'", "'free'", "'no'", "'u'paid'", "'paid'", 'None']
GoodForKids	['True', 'False', 'None']
Alcohol	["'u'full_bar'", "'u'beer_and_wine'", "'u'none'", "'none'", "'beer_and_wine'", "'full_bar'", 'None']
RestaurantsAttire	["'u'casual'", "'casual'", "'dressy'", "'u'dressy'", "'u'formal'", 'None', "'formal'"]
RestaurantsGoodForGroups	['True', 'False', 'None']
RestaurantsDelivery	['False', 'True', 'None']
BusinessAcceptsCreditCards	['True', 'False', 'None']
BusinessAcceptsBitcoin	['False', 'True']
ByAppointmentOnly	['True', 'False', 'None']
BYOCorkage	["'u'no'", "'yes_corkage'", "'yes_free'", "'u'no'", "'u'yes_free'", "'u'yes_corkage'", 'None']
DriveThru	['False', 'True', 'None']
HappyHour	['False', 'True', 'None']
WheelchairAccessible	['True', 'False', 'None']
Smoking	["'u'no'", "'u'outdoor'", "'u'yes'", "'no'", 'None', "'yes'", "'outdoor'"]
DogsAllowed	['False', 'True', 'None']
CoatCheck	['False', 'True', 'None']
Music	['None', {}]
BestNights	['None', {}]
BestNights_monday	[False]
BestNights_tuesday	[False]
BestNights_friday	[True]
BestNights_wednesday	[False]
BestNights_thursday	[False]
BestNights_sunday	[False]
BestNights_saturday	[True]
GoodForDancing	['False', 'True', 'None']
Corkage	['True', 'False']
BYOB	['False', 'True']
AgesAllowed	["'u'21plus'", "'u'allages'", "'u'19plus'", "'u'18plus'", 'None']
AcceptsInsurance	['False', 'True', 'None']
DietaryRestrictions	['None', {}]
DietaryRestrictions_dairy-free	[False]
DietaryRestrictions_gluten-free	[True]
DietaryRestrictions_vegan	[True]
DietaryRestrictions_kosher	[False]
DietaryRestrictions_halal	[False]
DietaryRestrictions_soy-free	[False]
DietaryRestrictions_vegetarian	[True]
HairSpecializesIn_straightperms	[False]
HairSpecializesIn_coloring	[True]
HairSpecializesIn_extensions	[True]
HairSpecializesIn_africanamerican	[False]
HairSpecializesIn_curly	[False]
HairSpecializesIn_kids	[False]
HairSpecializesIn_perms	[False]
HairSpecializesIn_asian	[False]
Open24Hours	['False', 'True']
RestaurantsCounterService	['True', 'False']

#### Annexe 7 Implémentation de l'algorithme de géolocalisation

```
def get_neighbors_recommend(address):
    distances = list()
    """
    etape 1 : recuperation des coordonnées de geolocalisation
    """
    if address != "":
        coordGeoloc = get_geocoordinates(address)
    else :
        coordGeoloc = getAutomaticLocation()
    data = get_restaurant(coordGeoloc['city'])
    """
    etape 2 : calcul de la distance
    """
    for train_row in data.itertuples():
        dist = haversine_distance(coordGeoloc['latitude'], coordGeoloc['longitude'],
                                   train_row.latitude, train_row.longitude)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    print("Nombre de données : ", len(distances))
    neighbors = list()
    """
    etape 3 : retourner les restaurants les plus proche
    """
    for i in range(len(distances)):
        neighbors.append(distances[i][0])
    return pd.DataFrame(neighbors)
```

#### Annexe 8 Implémentation de l'algorithme de calcul de la distance haversine

```
[4]: def haversine_distance(lat1, lon1, lat2, lon2):

    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2) ** 2 + cos(lat1) * cos(lat2) * sin(dlon / 2) ** 2
    c = 2 * asin(sqrt(a))
    r = 6371 # Radius of earth in kilometers.
    res = c * r
    return np.round(res, 4)
haversine_distance(36.195615, -115.040529, 36.238596, -115.233331)
```

[4]: 17.9445

Annexe 9 Implémentation de la prédiction de ratings de businesses

```
from scipy.sparse.linalg import svds

NUMBER_OF_FACTORS_MF = 20
U, sigma, Vt = svds(ratings_mat_demeaned, k = NUMBER_OF_FACTORS_MF)
sigma = np.diag(sigma)
all_user_predicted_ratings = np.dot(np.dot(U, sigma), Vt) \
    + user_ratings_mean.reshape(-1, 1)
```

Annexe 10 Implémentation de la mesure de proximité des matrices

```
from sklearn.decomposition import NMF

NUMBER_OF_FACTORS_MF = 20
nmf = NMF(n_components=NUMBER_OF_FACTORS_MF)
nmf.fit(ratings_mat)

#Features
W = nmf.transform(ratings_mat)

#Features Weight
H = nmf.components_
#Reconstructed matrix
ratings_mat_fitted = W.dot(H)
```

Annexe 11 Tableau Trello du projet

The screenshot shows a Trello board with the following structure:

- Board Title:** projet synthèse M2
- Columns:**
  - To Do:** Contains a card for "Geo indistinguishability" under "GEO-LOCALISATION".
  - In Progress:** Contains cards for "Technical Architecture" under "ARCHITECTURE", "Recommendation System (Collaborative filtering)" under "ALGO", "Pattern mining" under "DATA", and "Cross validation test" under "MACHINE LEARNING".
  - Done:** Contains cards for "Sentiment Analysis" under "NLP", "Geolocalisation : récupérer les coordonnées de localisation" under "GEO-LOCALISATION", "Project Creation Enterprise" under "OTHERS", "Gantt diagram" under "DATA", "Elastic query limit / API" under "DATA", "Data gathering" under "DATA", "Data Analysis" under "DATA", and "Upcoming Appointment" under "OTHERS".
- Team Members:** SAM, BF, HV, M, AU
- Labels:** Q, Free, & Visible par les membres d'une équipe
- Buttons:** Inviter, Butler, Deadlines, ... Afficher le menu, + Ajoutez une au