

Projet de TextMining

*Classification supervisée
des recettes sur Marmiton*



AIT ICHOU Omar

Master Mathématiques Appliquées Statistique
Parcours Science des Données

Année 2018-2019

Table des matières

Introduction.....	3
1. Traitements préliminaires	4
1.1 La Collecte des Données :	4
1.2 Construction du DataFrame Panda :	5
2. Classification Supervisée	9
2.1 Algorithmes utilisés	9
2.2 Applications et résultats.....	12

Introduction

Marmiton, leader des sites culinaires sur le web français comptabilise plus de 10 millions de visiteurs uniques par mois. Ce site ou cette communauté de gourmands (tel est son slogan) compte plus de 70986 recettes, 5412 vidéos et 165658 discussions entre amateurs et professionnels, permettant d'offrir une richesse d'échange entre les passionnés.

Ayant un grand amour pour la cuisine, c'est tout naturellement que m'est venue l'idée de travailler sur ces données, et d'appliquer ce que j'ai appris en cours de text mining. Le choix du sujet fut compliqué, car des nombreuses classifications peuvent être effectuées à partir des données de marmiton. En effet, il peut être intéressant de prédire le niveau de difficulté d'une recette, ou encore la satisfaction des internautes en fonction des avis rédigés sur le site web. J'ai cependant conservé la problématique qui m'apparaissait la plus pertinente, à savoir prédire le type de plat, en fonction du texte de la recette.

Les ingrédients d'un dessert en général sont très différents de ceux d'une entrée ou d'un plat ce qui peut conduire à un bon classement de cette catégorie, contrairement aux ingrédients d'un plat ou d'une entrée qui peuvent se ressembler. On va par la suite découvrir via ce projet les similitudes entre les catégories.

1. Traitements préliminaires

1.1 La Collecte des Données :

Dans un premier temps j'ai créé mon jeu de données à l'aide de technique de Web Scraping. Cette étape a été réalisée en Python avec le package BeautifulSoup. En effet, j'ai récupéré des recettes d'entrées, plats et desserts, présentes sur le site Marmiton via les liens suivants :

<https://www.marmiton.org/recettes/?type=entree>

<https://www.marmiton.org/recettes/?type=platprincipal>

<https://www.marmiton.org/recettes/?type=dessert>

Rechercher une recette

Une recette

(je cherche une recette en fonction des ingrédients que j'aime ou que je n'aime pas)

Je lance ma recherche

Dans mon frigo

(je cherche une recette en fonction des ingrédients que j'ai dans mon frigo)

Je lance ma recherche

Les recettes les plus recherchées par les internautes par type de plat

Entrées

Plats

Desserts

Amuses bouches

Sauces

Accompagnements

Boissons





Figure 1 Interface du site Marmiton

Pour chaque catégorie, j'ai extrait les Url de chaque recette, permettant ainsi de récupérer pour chaque recette :

- Le titre de la recette
- Les ingrédients
- Les nombre d'étape de préparation
- Les instructions de préparation

1.2 Construction du DataFrame Panda :

J'ai créé un dataframe contenant comme variables : le titre de la recette, les ingrédients, les instructions de préparation, le nombre d'étape de préparation, le type de plat. Le jeu de données comporte 2683 recettes.

Mon étude utilisera par la suite le dataframe construit de la manière suivante :

titre	Ingrédients	préparation	Nombre d'étapes	catégorie
Flan de courgettes	['1 kg de courgette ', '100 g de gruyère râpé (60 g de gruyère + 40 g de parmesan) ', '8 cuillère de lait ', '6 oeuf ', '20 feuille de basilic ']	" Couper les courgettes en très fines rondelles. Les faire revenir dans de l'huile d'olive. Saler, poivrer, cuire environ 10 mm à petit feu. Battre les oeufs et ajouter le fromage + sel + poivre + basilic + lait. Jeter la préparation sur les courgettes. Remuer. NE PAS FAIRE PRENDRE L'OMELETTE. Mettre au four 180°C pendant 35 mn. SERVIR FROID OU CHAUD avec une sauce mayonnaise + ketchup hot. "	5	entrée
Mousse au chocolat facile	['3 oeuf ', '100 g de chocolat (noir ou au lait) ', '1 sachet de sucre vanillé ']	Séparer les blancs des jaunes d'oeufs. Faire ramollir le chocolat dans une casserole au bain-marie. Hors du feu, incorporer les jaunes et le sucre. Battre les blancs en neige ferme et les ajouter délicatement au mélange à l'aide d'une spatule. Verser dans une terrine ou des verrines et mettre au frais 1 heure ou 2 minimum. Décorer de cacao ou de chocolat râpé Déguster	9	dessert

Les données recueillies sont réparties entre les différentes classes de la manière suivante :

classe	effectif	pourcentage
Entrée	897	0.34
Plat	897	0.34
Dessert	889	0.32

On constate, que les recettes sont réparties de manière à peu près équitable au sein de mes trois catégories. Cela va me permettre d'éviter le cas où une catégorie est plus présente que les autres au sein de ma base de données. En effet, lorsque c'est le cas, mon algorithme peut prédire à tort la catégorie dominante mais donner un bon taux de classement ; le score ne reflètera alors pas la qualité de prédiction de l'algorithme.

Avant d'effectuer la classification, je me suis intéressé à la répartition du nombre d'étapes nécessaire à la réalisation d'une recette, au sein de chaque catégorie.

classe	min	moyenne	médiane	max
Entrée	1	6.39	6	25
Plat	1	7.42	7	55
Dessert	1	7.66	7	37

On constate que le nombre d'étape ne varie pas beaucoup en fonction de la catégorie. Cependant, au sein d'une même catégorie, on constate que le nombre d'étape à réaliser est très hétérogène. En effet, cela peut dépendre de la difficulté de la recette, du temps de préparation ou encore du degré de détail de la recette, car certains internautes expliquent plus en détails que d'autres la recette.

Afin d'avoir une idée plus précise des recettes en fonction de chaque catégorie, j'ai réalisé un nettoyage avant de représenter les nuages de mots des bigrammes apparaissant plus de 5 fois. J'ai décidé de dessiner un nuage de mots par catégorie afin d'avoir une meilleure idée des mots qui reviennent le plus souvent dans chaque label.

2. Classification Supervisée

2.1 Algorithmes utilisés

Au vu de mes données et de mon objectif (prédiction de recettes à l'appartenance de classes), j'ai décidé de réaliser une classification supervisée. Pour cela, j'ai comparé les résultats de trois algorithmes de Machine Learning : Régression Logistique, Random Forest et SVM (Support Vector Machine). Je cherche à savoir quel est l'algorithme le plus performant pour mes données, c'est-à-dire quel algorithme va minimiser le risque moyen sur les données de validation. Avant toute chose, j'explique en quoi consistent ces algorithmes :

Régression Logistique :

La régression logistique est un cas particulier de modèle linéaire généralisé. Il existe différents types de régression logistique : la régression logistique binaire et la régression logistique multi-classes.

La première est utilisée pour expliquer une variable binomiale ou une variable binaire (0, 1) de Bernoulli, à l'aide de variables explicatives. Cette méthode est fortement utilisée dans le domaine médical où l'on souhaite souvent prédire la présence ou non d'une maladie en fonction de différentes variables ; ou bien dans le domaine du marketing, si l'on souhaite prédire « bon » ou « mauvais » client. Ici, ce type de régression pourra être utilisée pour définir si la recette est facile ou difficile. L'équation du modèle variera en fonction du type de variable explicative :

- Dans le cas où la variable cible est expliquée par une seule variable quantitative nommée x , l'équation du modèle sera la suivante :

$$\text{logit } p_{\beta}(x) = \beta_0 + \beta_1 x.$$

Avec β_0 et β_1 les coefficients du modèle

- Dans le cas où la variable cible est expliquée par une seule variable qualitative à 2 modalités (ex : le sexe -> 2 modalités possibles : Homme ou Femme), l'équation du modèle sera la suivante :

$$\text{logit } p_{\alpha}(x) = \alpha_0 + \alpha_F \mathbf{1}_F(x) + \alpha_H \mathbf{1}_H(x).$$

Avec α_0 , α_F et α_H les coefficients du modèle, tel que $\alpha_F = 0$ pour l'identifiabilité du modèle

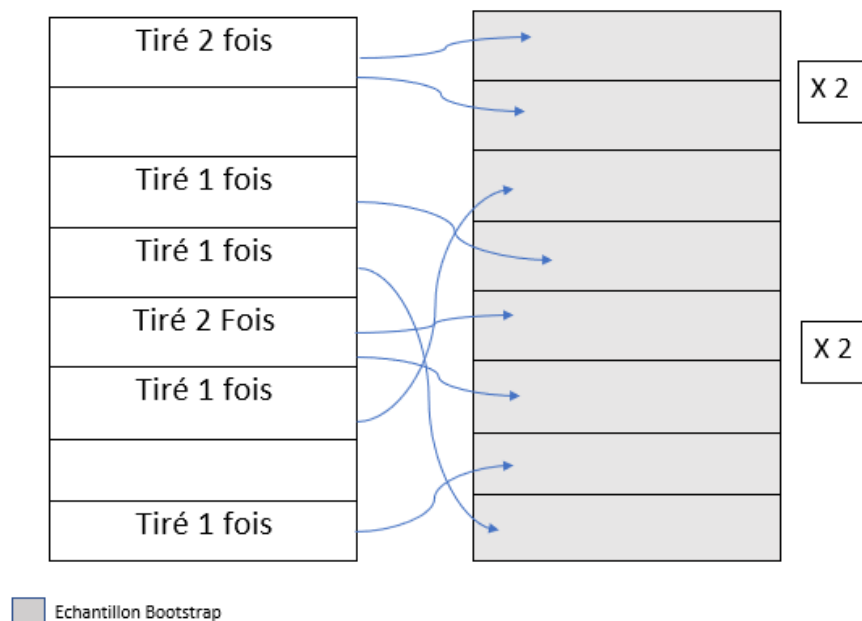
Lorsque l'on cherche à expliquer une variable qualitative qui comporte plus de deux modalités, on utilise un modèle logistique multi-classes. On distingue deux cas de régression logistique multi-classes :

- Soit les modalités de la variable cible sont ordonnées ; il existe une hiérarchie naturelle entre elles. Par exemple, on cherche à expliquer dans le domaine du marketing, le degré de satisfaction à un produit (pas satisfait, peu satisfait, satisfait ou très satisfait) ou bien dans le domaine médical, la taille d'une tumeur (absence de tumeur, bénigne ou maligne). Ainsi, on parle de modèle polytomique ordonné.
- Soit il n'existe pas d'ordre sur les modalités de la variable à prédire, ainsi elle est purement nominale : genre de la recette : (entrée, plat, dessert etc ...). On parle dans ce cas de modèle polytomique multinominal.

Random Forest :

La Random Forest ou Forêt aléatoire est un algorithme de Machine learning qui peut selon les cas être très utile pour optimiser des problèmes de prédiction.

Pour construire un algorithme de prédiction du type Random Forest, il faut tout d'abord comprendre ce qu'est un échantillon Bootstrap.



Comme le montre ce schéma, un échantillon Bootstrap consiste à tirer aléatoirement et avec remise des individus parmi notre jeu de données. Ce tirage donnera lieu à un nouvel échantillon pouvant posséder plusieurs fois les mêmes lignes et devant être de la taille de nos données de départ. Il s'agira ensuite de construire un arbre de décision sur cet échantillon Bootstrap en tirant aléatoirement un nombre de variables de l'échantillon qui serviront à construire les nœuds de l'arbre. Il suffira ensuite d'appliquer sur les individus non pris en compte dans l'échantillon Bootstrap, (qu'on appelle les Out of Bag) l'arbre ainsi construit afin d'estimer la performance de celui-ci. Le terme de Forêts désigne le caractère itératif du modèle et donc par définition cette étape précédemment décrite est à répéter. Une fois un nombre d'arbres construit, il suffira de moyenner les erreurs estimées de chacun d'entre eux pour obtenir la qualité prédictive de la forêt ainsi construite.

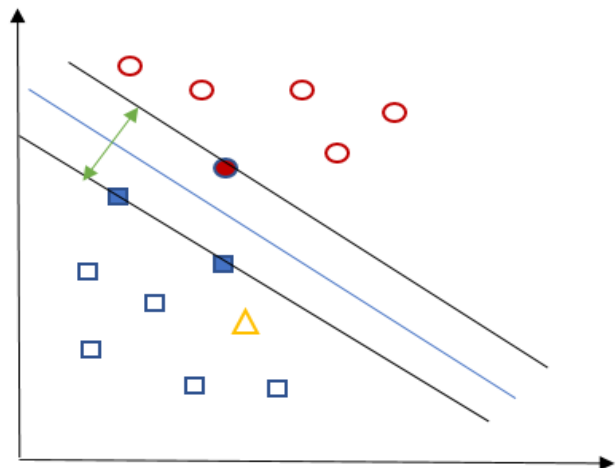
Cette méthode a pour avantage de fournir une bonne qualité de prédiction, d'être facile à utiliser et à paralléliser, ce qui permet de réduire le temps de calcul. Cependant, d'un point de vue métier, il est compliqué d'interpréter facilement les arbres construits (« effets boîtes noires »).

SVM :

La **SVM (Support Vector Machine)** est un algorithme d'apprentissage supervisé qui peut être utilisé aussi bien en discrimination qu'en régression. Le principe de la **SVM** consiste à trouver un hyperplan séparateur (linéaire ou non) de marge maximale qui sépare au mieux les données (afin de réaliser la prédiction pour un nouvel individu).

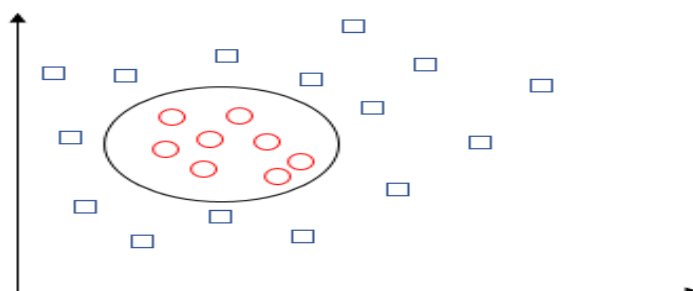
On appelle **vecteurs supports** les points bien classés mais situés sur les frontières qui définissent la marge maximale ou qui se trouvent dans la marge, et les points mal classés. Trouver une frontière séparatrice optimale revient finalement à résoudre un problème d'optimisation sous contrainte.

Il est possible d'optimiser cet algorithme en tolérant des **marges plus souples** qui vont permettre de bien classer les vecteurs qui se trouvent dans la zone définie par la marge, voir les mal classés, grâce à la **constante de tolérance C** qui est un paramètre à calibrer. Cette dernière permettra de juger plus ou moins les mal classés. Plus le paramètre est petit, plus on va être intolérant aux mal classés et inversement.



La figure ci-contre illustre une SVM linéaire en discrimination binaire. La **droite bleue** représente **l'hyperplan optimal**. Il s'agit d'une droite, car nous sommes en dimension 2. La **double flèche verte** correspond à la **marge maximale**. Les trois points coloriés sont les **vecteurs supports**. Le label prédit du nouvel individu (le triangle jaune) est donc un carré.

Dans le cas où les données ne sont pas linéairement séparables, comme on peut le voir sur la figure ci-dessous. La SVM linéaire donnera de très mauvais résultat avec un nombre de vecteurs supports qui sera très important. Une solution serait d'utiliser l'astuce du noyau.



On a quelques noyaux classiques qui conviennent très bien à ce genre de problème : le **noyau radial** (ou gaussien), le **noyau polynomial** ou encore le **noyau laplacien**. Le noyau radial est le plus utilisé. Pour chacun des noyaux, il y a des paramètres supplémentaires à optimiser. Par exemple pour le noyau polynomial on a en plus de la constante de tolérance C, le degré du polynôme à calibrer.

Le gros avantage de cet algorithme est qu'il permet de traiter des problèmes qui ne sont pas forcément linéaires avec un nombre très important de variables. C'est aussi un algorithme qui est **robuste** par rapport aux points atypiques. Cependant, l'algorithme est très coûteux en termes de temps et le choix du noyau est plus difficile lorsqu'on ne peut pas visualiser les données.

2.2 Applications et résultats

Tout d'abord, comme expliqué dans la première partie du dossier, j'ai décidé de classer mes recettes en trois catégories : entrée, plat, dessert. Pour ce faire j'ai expérimenté les 3 types d'algorithmes présentés précédemment. Cependant, j'ai effectué un nettoyage de texte au préalable, afin d'avoir des vecteurs de mots comme variables explicatives qui ne soient pas des mots vides. Pour cela j'ai utilisé la fonction stopwords du package nltk, qui renvoie une liste de mots vides. J'ai également rajouté les lettres de l'alphabet, pour les fautes de frappe. J'obtiens donc la liste des mots vides suivante :

```
In [92]: print(stop_words)
['aurez', 'fussiez', 'sommes', 's', 'ces', 'fût', 'mes', 'vos', 'vous', 'n', 'aurait', 'ayez', 't', 'ai',
'eus', 'avons', 'soient', 'sera', 'eût', 'tes', 'ton', 'étiez', 'la', 'de', 'suis', 'étants', 'avons',
'aies', 'auras', 'ayons', 'fus', 'aux', 'moi', 'étantes', 'ayante', 'un', 'aie', 'le', 'd', 'seraient',
'leur', 'étante', 'furent', 'fussions', 'étions', 'ses', 'aient', 'aurai', 'serez', 'notre', 'étée',
'eussions', 'sur', 'fusses', 'du', 'me', 'elle', 'serions', 'fûmes', 'aurais', 'eussent', 'soyez', 'sont',
'ou', 'es', 'eussiez', 'par', 'eûmes', 'ce', 'qui', 'son', 'est', 'auraient', 'avais', 'on', 'étés', 'se',
'étais', 'eut', 'que', 'à', 'y', 'pour', 'êtes', 'nos', 'été', 'étaient', 'm', 'serait', 'ayants',
'auriez', 'fussent', 'avec', 'as', 'eu', 'serai', 'te', 'nous', 'sois', 'aurions', 'une', 'eue', 'fut',
'eusse', 'ta', 'soit', 'qu', 'sa', 'toi', 'et', 'en', 'l', 'aurons', 'avait', 'eûtes', 'lui', 'fusse',
'au', 'eusses', 'étant', 'eux', 'seras', 'était', 'ayant', 'auront', 'des', 'votre', 'c', 'aviez',
'fûtes', 'ma', 'ayantes', 'dans', 'ont', 'avaient', 'ait', 'eurent', 'mon', 'étées', 'serais', 'j',
'soyons', 'seront', 'avez', 'eues', 'aura', 'même', 'serons', 'seriez', 'il', 'je', 'tu', 'ne', 'a', 'b',
'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'les', 'être', 'avoir', 'faire', 'qu', 'qu'il']
```

Une fois ces mots éliminés, j'ai utilisé la méthode TF-IDF pour créer ma matrice de travail.

TF-IDF (term frequency - inverse document frequency) est une méthode de pondération qui permet d'évaluer l'importance d'un terme contenu dans un document, relativement à une collection. Le poids augmente proportionnellement au nombre d'occurrences du mot dans le document et varie en fonction de la fréquence du mot dans la collection.

Le coefficient TF-IDF d'un terme t_i dans un document d_j , se calcule de cette manière :

$$tfidf_{ij} = tf_{ij} * \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

Avec :

- tf_{ij} : le nombre d'occurrence du terme t_i dans le document d_j
- $|D|$: le nombre total de documents dans la collection
- $|\{d_j : t_i \in d_j\}|$: le nombre de documents où le terme t_i apparaît.

Ainsi, ma matrice de travail aura autant de lignes qu'il y a de recettes dans mon jeu de données, et autant de colonnes qu'il y a de termes apparaissant plus de cinq fois dans les données. Les coefficients de cette matrice seront les coefficients TF_IDF. Comme les ingrédients apparaissent normalement au moins une fois dans le descriptif de la recette, je n'ai pas utilisé la variable « ingrédients » pour la classification, car cela aurait entraîné de la « redondance » d'information inutile.

Afin d'évaluer les modèles prédictifs, j'ai séparé mon jeu de données en deux échantillons ; un échantillon d'apprentissage, sur lequel les algorithmes seront construits et un échantillon de validation sur lequel je testerai les algorithmes, calculerai le taux de bon classement et estimerai l'erreur de classification.

Ainsi mon échantillon d'apprentissage est constitué de 2012 recettes et mon échantillon de validation est constitué de 671 recettes. J'ai fait attention à ce que la répartition entre les classes soit respectée au sein de chaque échantillon.

J'ai commencé par expérimenter des algorithmes de régression logistique. Ici, il s'agit d'une régression logistique multi-classes, car il y a trois modalités à prédire : « entrée », « plat » et « dessert ». J'ai appliqué cette méthode sur 3 matrices : une matrice contenant seulement les uni-grammes, puis une autre où j'ai introduit les bi-grammes en plus, puis une autre où j'ai ajouté les trigrammes en plus.

J'ai répété ce processus sur les avis lemmatisés et sur les avis racinisés. J'ai représenté le taux de bon classement de chaque régression logistique afin de bien se rendre compte des différences de performance :

	brut	lemmatisé	racinisé
unigrammes	0.84	0.81	0.75
Uni+bigrammes	0.83	0.84	0.82
Uni+bi+trigrammes	0.83	0.84	0.82

On constate que la meilleure régression logistique classe bien les recettes à hauteur de 84%. Afin de comprendre quels mots contribuent le plus à classer dans chaque catégorie les recettes, j'ai regardé les variables ayant les plus fortes associations négatives et positives avec les différentes classes au sein de la régression.

catégorie	Association positive	Association négative
Entrée	Mixer, cake, vinaigre, salade, poivre, thon, mayonnaise, gruyère, mixer tout, couper	Sucre, chocolat, pâte, plat, pommes, mijoter, poulet, cannelle, riz, fruits
Plat	Viande, poulet, sauce, plat, pâte, mijoter, riz, poêle, oignons, revenir	Sucre, mixer, moule, chocolat, refroidir, œuf, pâte, frais, laisser refroidir, saladier
Dessert	Sucre, chocolat, pâte, fruits, vanille, gateau, cannelle, moule, sirop	Poivre, huile, sauce, saler, sel poivre, ail, revenir, poivrer, fromage, viande

Ainsi, on peut constater, que les mots qui contribuent le plus à classer les recettes dans chaque catégorie, sont des ingrédients et des verbes typiques de chaque type de plats (ex : sucre pour le dessert, mijoter pour le plat etc ...)

J'ai décidé d'expérimenter d'autres algorithmes de classification supervisé sur mes données afin d'essayer d'améliorer le résultat.

Ainsi j'ai appliqué des algorithmes de Forêts aléatoires. Le paramètre à calibrer dans la Random Forest est le nombre d'itérations. J'ai donc essayé cet algorithme avec différents paramètres, sur les avis non lemmatisés et non racinisés.

La forêt aléatoire qui obtient le meilleur score (77.8%) est celle prenant en compte les termes composés de 1 et 2 mots ; avec 1000 itérations. J'aurais pu pousser le nombre d'itérations, cependant cette méthode statistique est très couteuse en termes de temps sur ma machine personnelle. Ainsi, le manque de temps m'a contraint à abandonner cette piste.

J'ai également décidé d'expérimenter des modèles SVM (Support Vector Machine). Ma matrice étant en grande dimension, il n'est pas possible de visualiser les données. Ainsi, il est plus compliqué de choisir le type de noyau. Après de nombreux essais, il s'est avéré que le noyau linéaire était le plus adapté à mon problème. Cependant, les résultats de mon algorithme SVM sont moins concluants que ma régression logistique, le taux de bon classement est sensiblement inférieur.

Voici les scores obtenus avec l'algorithme SVM :

	Unigramme	+ bigrammes	+ trigrammes
entrée	0.76	0.79	0.81
plat	0.69	0.69	0.66
dessert	0.95	0.93	0.93
total	0.8	0.8	0.8

Ainsi, on obtient un taux de bon classement de 80%. On constate que la catégorie des desserts, est sans surprise la catégorie la mieux prédite, du fait des ingrédients typique de cette catégorie (sucre, chocolat) et du vocabulaire (pétrir etc. ...). On remarque également que les plats et les entrées sont plus difficiles à différencier : en général les ingrédients sont dans les

deux catégories, salés. Ce qui peut différencier ces deux catégories, c'est le temps de cuisson ainsi que la quantité des ingrédients.

Perspectives d'amélioration

En prenant du recul par rapport à mon travail, j'ai bien conscience que celui-ci n'est pas parfait, et que certaines améliorations pourraient être apportées.

Tout d'abord, j'ai pensé, un peu tardivement, à créer une fonction qui corrige les fautes d'orthographe et les fautes de frappe, avec une méthode de mesure de similarité entre deux chaînes de caractères. Cependant, comme je viens de le préciser, j'ai pensé à cette méthode, trop tard pour la mettre en place.

Il pourrait également être envisageable de créer un algorithme pour chaque type de difficulté, permettant ainsi d'affiner l'analyse.

Enfin, il pourrait être intéressant d'utiliser des synonymes pour des termes, ou des ingrédients apparaissant trop peu fréquemment dans le corpus, afin qu'ils soient pris en compte dans l'analyse (car si « litchi » apparaît 2 fois dans le corpus, il n'est pas pris en compte dans l'analyse)

Conclusion

Ce projet m'a permis d'appréhender plus en détail les méthodes de fouille de données textuelles. En me confrontant à cette tâche, de la constitution du jeu de données jusqu'à la classification des recettes, en passant par le nettoyage du corpus, j'ai beaucoup appris. Cela m'a permis d'appliquer l'ensemble des méthodes vues en cours, mais également de chercher une problématique pertinente utilisant des données textuelles.

La problématique que j'ai traitée, n'est pas la seule qui découle des données du site marmiton, bien au contraire. En effet, un algorithme pourrait être créé afin de détecter les recettes « loufoques ». On pourrait encore effectuer des suggestions de recettes en fonction de ce que la personne a dans son frigo, basé sur de la recommandation.

Références

Fromont M.- “Apprentissage Statistique - Partie I - Master MAS, parcours Science des Données Université Rennes 2, 2017-2018”.

Wikistat Machines à vecteurs supports. Disponible sur : <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-svm.pdf>

Rouviere L. – « Régression logistique avec R », Université Rennes 2

Efraim O. – « Analyse de sentiments sur des tweets »

Wikipédia, la méthode TF_IDF :