Table des matières

Introduction

La fouille de textes est une sous partie de la fouille de données. Elle fait partie du domaine de l'Intelligence Artificielle et a pour objectif d'extraire automatiquement des informations à partir de grandes quantités de textes. Elle est composée de quatre tâches fondamentales : Recherche d'information, Classification, Annotation et Extraction d'information. Pour ce devoir, nous nous intéresserons plus particulièrement à la tâche de Classification. Cette dernière comporte deux grandes approches possibles, la première consiste à écrire un programme à la main, mais cette option est fastidieuse et coûteuse en temps. C'est pourquoi nous allons utiliser la seconde approche : l'apprentissage automatique. Il s'agit d'un programme qui s'améliore avec l'expérience. Le programme prend en entrée des textes possédant déjà une classification, c'est la phase d'apprentissage. Il va ensuite se servir de ces exemples pour classifier les nouvelles données.

Avec le logiciel Weka nous utiliserons différents programmes d'apprentissage automatique, dans le but de réaliser une tâche de classification de textes. Afin de tester ces différents programmes de classification, nous avons choisi de nous intéresser aux recettes de cuisine. Nous avons constitué un corpus composé de trois classes : Entrée, Plat, Dessert. Dans chacune des différentes classes, nous avons respectivement récolté vingt recettes. Nous avons ainsi obtenu un corpus composé de soixante recettes. Puis nous avons transformé ce corpus en données pour Weka. Enfin nous avons procédé à des expérimentations dans le but de définir la meilleure combinaison possible entre l'une des représentations des textes et l'un des classifieurs.

Constitution du corpus

Les textes qui constituent notre corpus sont des recettes de cuisine. Dans les livres ou les sites qui en répertorient, nous avons remarqué que ces recettes étaient toujours classées en Entrée, Plat et Dessert. Il existe d'autres classes (Amuse-bouche, Cocktail, Sauce, ...) mais elles sont secondaires et nous ne nous y intéresserons pas dans cette étude. Le critère de classification de nos textes correspond donc aux trois classes principales : Entrée, Plat, Dessert.

Pour constituer notre corpus, nous avons choisi des recettes classiques, en privilégiant la simplicité, notamment en évitant les recettes dites « sucré-salé ». Nous avons récupéré ces recettes sur différents sites internet. Certains sont entièrement consacrés à la cuisine (Marmiton, CuisineAZ, 750g), un autre est spécialisé (Recette-dessert) et les derniers sont plus généraux (Notrefamille, Auféminin, Doctissimo). La plupart de ces sites sont participatifs. En effet, les internautes postent leurs propres recettes. Cela est justement intéressant pour notre projet de classification ; dans le cas d'une nouvelle recette postée sans classification (Entrée/Plat/Dessert), le classifieur pourrait lui attribuer une classe automatiquement à partir de l'apprentissage qu'il aura expérimenté grâce à notre corpus-exemple de recettes déjà classées.

	CuisineAZ	Marmiton	750g	Notre famille	Au féminin	Recette - dessert	Doctissimo	Total
Entrée	7	3	5	3	2	0	0	20
Plat	6	7	3	2	2	0	0	20
Dessert	2	5	5	4	0	3	1	20
Total	15	15	13	9	4	3	1	60

Nous avons sélectionné vingt recettes pour chacune des classes, afin qu'elles soient représentées de manière équilibrée. Les sites revenant le plus régulièrement sont Marmiton, 750r, et CuisineAZ.

Le contenu textuel a été légèrement modifié afin de pouvoir isoler les mots (ajout d'espaces, suppression de tirets ou de slashs) mais aucun mot n'a été supprimé. Chaque recette a été enregistrée dans un fichier texte, puis placée dans un dossier en fonction de sa classe.

Transformation des textes en données pour Weka

Variation de la représentation des textes

Nous avons tout d'abord lancé le script vectorisation.py sur les textes de notre corpus avec un fichier de mots vides vide. Nous avons testé plusieurs classifieurs sur le corpus obtenu. Nous nous sommes rendues compte que les mots vides apparaissaient trop fréquemment dans les critères de classification. De notre point de vue, ces mots non-porteurs de sens ne permettaient pas de discriminer les classes. De plus, ces mots peuvent varier en fonction de l'auteur, et donc du site internet (il peut y avoir plusieurs auteurs par site étant donné que la plupart de nos sources sont des sites participatifs). Nous avons donc pris la décision de constituer un fichier de mots vides, pour cela nous nous sommes servies du fichier mots_vides.txt mis à notre disposition. Cependant il n'était pas d'emblée totalement adapté à notre corpus. Nous l'avons donc amélioré après avoir lu les recettes, et après avoir utilisé les différents classifieurs.

Nous remarquons aussi que certains mots apparaissent plusieurs fois sous différentes formes (« l'oeuf », « d'oeuf », « œuf, », ...). Et certains caractères de ponctuation sont comptabilisés comme étant des mots par Weka car ils apparaissent entre deux espaces « ; » (ils ont été segmentés comme tels par vectorisation.py).

Nous avons donc fait varier la représentation des textes pour améliorer les résultats des classifieurs. Dans un premier temps, nous avons préparé les textes ; d'abord « à la main » avec la fonction « rechercher-remplacer » d'un éditeur de texte pour remplacer les apostrophes par des espaces. Ensuite nous avons modifié le script de vectorisation.py pour ne pas prendre en compte les caractères de ponctuation collés aux mots comme les virgules ou les points par exemple. Enfin, nous avons ajouté à la liste des mots vides les caractères de ponctuation qui peuvent apparaître seuls comme les « ; » ou les « ! » ou encore les « | » par exemple.

Après ces pré-traitements, nous avons créé un corpus à l'aide des textes préalablement recueillis et rangés dans trois sous-dossiers d'un dossier nommé « Corpus », portant chacun le nom d'une classe. Dans ce but, nous avons utilisé le script vectorisation.py pour obtenir un fichier .arff contenant un corpus exploitable par Weka à partir des textes recueillis et du fichier de mots vides.

Après cela, quelques problèmes subsistent lors des expérimentations. D'une part, on remarque que les chiffres sont très présents dans les recettes (quantité, poids, temps de cuisson...) et que certains classifieurs les utilisent dans leurs règles pour classer les textes. Cela pourrait être une bonne idée. En effet, de grandes quantités, ou un temps de cuisson élevé, pourraient être associés aux Plats par exemple. Cependant, ils ne sont pas réellement exploitables sous la forme actuelle du corpus ; les textes ne venant pas tous du même site, le modèle d'écriture des recettes varie et parfois on ne sait pas à quelle unité de mesure correspond un chiffre (1,5 kg vs 1500 gr). Cela altère donc la pertinence des résultats. C'est pourquoi nous avons décidé de supprimer les chiffres à l'aide d'une expression régulière (.*[0-9].*) dans la fenêtre « preprocess » de Weka.

D'autre part, le mot "moyen" nous a posé quelques difficultés, car il apparaît dans différents contextes : pour les Plats on le retrouve 11 fois dans le contexte "coût : moyen" ou "difficulté : moyen" et 5 fois dans le contexte "à feu moyen". Pour les Entrées : "feu moyen" apparaît 1 fois, et pour les Desserts, on retrouve 1 fois "œufs moyens", 2 fois "difficulté : moyen" et 1 fois "feu moyen". Nous nous sommes donc demandées si nous devions supprimer ce mot. Après l'avoir fait, nous nous sommes rendues compte que les résultats étaient égaux ou moins bons aux précédents pour les différents classifieurs testés : Jrip, NaiveBayes, J48, SMO. Nous avons donc décidé de le conserver.

Enfin, en testant le classifieur JRip , nous nous sommes rendues compte que le mot « cuisineaz.com » était utilisé par le classifieur pour classer les recettes comme Plat. Cependant, en regardant le tableau des sources, on remarque que cuisineaz comptabilise plus d'Entrées que de Plats et qu'il ne s'agit pas de la source comportant le plus de Plat (Marmiton). En outre, CuisineAZ est la source la plus importante de nos recettes (ex-aequo avec Marmiton, mais il est le premier par ordre alphabétique). La présence du nom de ce site ne semble donc pas être un critère de classification très pertinent selon nous. De plus, inclure ce mot dans la liste des mots vides nous a permis d'obtenir de meilleurs résultats.

Expérimentations

Il y a plusieurs méthodes de guider la phase d'apprentissage pour les classifieurs. Nous avons d'abord testé « training set » avec plusieurs classifieurs mais cette configuration consiste à apprendre et tester un classifieur sur les mêmes données. Les résultats seront bons pour le corpus, mais cela s'avérera plus difficile avec de nouvelles données à classer. Nous utiliserons donc pour nos expérimentations : « cross validation » (Folds 10). En effet, la validation croisée nous paraît plus appropriée ; elle consiste à apprendre un classifieur à partir des 9 premiers blocs puis à faire un test sur le 10ème bloc, et cela à plusieurs reprises pour faire jouer le rôle du test à chaque bloc à tour de rôle. Avec un petit corpus, nous aurions pu diminuer le nombre de blocs, mais le notre étant plutôt long, cette configuration nous paraît être la meilleure.

Nous avons ensuite testé tous les classifieurs et nous avons trouvé très intéressant de voir comment chacun d'entre eux définit les limites entre les classes. Certains ont un « raisonnement » plutôt accessible pour un humain, d'autres sont plus abstraits et mathématiques.

ComplementNaiveBayes

La famille des classifieurs Bayes est fondée sur un calcul de probabilités appelés « probabilités Bayesiennes ». Ces algorithmes utilisent le Théorème de Bayes :

$$p(A \mid B) = \frac{p(B \mid A).p(A)}{p(B)}$$

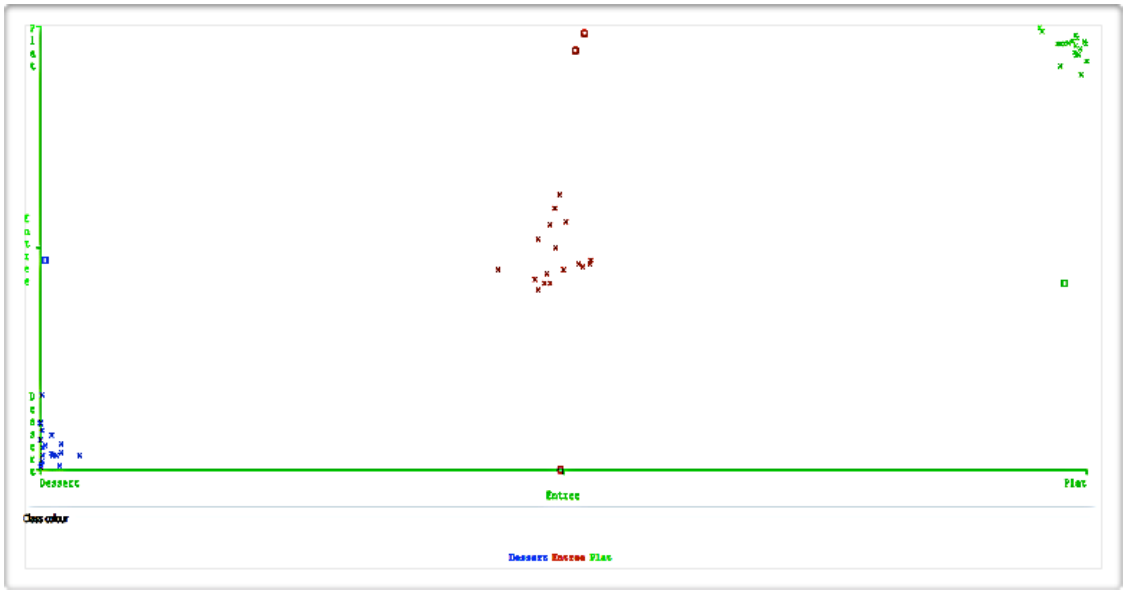
Nous avons testé différents classifieurs de cette famille, et celui qui a donné le résultats les plus probants est ComplementNaiveBayes. Voici ses résultats d'après nos critères d'évaluation :

Précision	0,916
Rappel	0,917
F-mesure	0,916
Temps de calcul	0,05 s.
Lisibilité	Difficile à interpréter

MATRICE DE CONFUSION			
a	b	c	< – classified as
19	1	0	a = Dessert
1	**17**	2	b= Entrée
0	1	**19**	c = Plat

→ On observe qu'il classe très bien les Desserts et les Plats.

Visualisation des erreurs de classification (erreurs = carrés) :



SMO

Nous avons aussi testé SMO. Il s'agit d'une variante de SMV (Séparateur à Vaste Marge). Cet algorithme utilise l'astuce du noyau, il calcule la distance entre les points dans un espace de plus grande dimension que l'espace initial. Le résultat obtenu est une combinaison linéaire de tous les attributs. Comme seuls les ordinateurs peuvent faire ce calcul, il est difficilement interprétable et compréhensible pour les humains.

Précision	0,845
Rappel	0,817
F-mesure	0,82
Temps de calcul	0,19 s.
Lisibilité	Difficile à interpréter

MATRICE DE CONFUSION			
a	b	c	< – classified as
15	4	1	a = Dessert
0	**18**	2	b= Entrée
0	4	**16**	c = Plat

→ il classe très bien les Entrées, mais moins bien le reste.

JRip

Cet algorithme fait partie de la famille Rules. Il classe les recettes selon ces règles :

(sucre >= 1) => classe=Dessert (20.0/2.0)
(chocolat >= 10) => classe=Dessert (2.0/0.0)
(oignons >= 1) => classe=Plat (12.0/0.0)
(four >= 2) => classe=Plat (3.0/0.0)
(moyen >= 1) => classe=Plat (3.0/0.0)
=> classe=Entree (20.0/1.0)

Précision	0,769
Rappel	0,767
F-mesure	0,76
Temps de calcul	0,2 s.
Lisibilité	Facile à interpréter / Paraît logique

MATRICE DE CONFUSION			
a	b	c	< – classified as
19	0	1	a = Dessert
3	**15**	2	b= Entrée
2	6	**12**	c = Plat

→ Il classe très bien les Desserts, mais moins bien le reste.

ZeroR

Il s'agit d'un algorithme de la classe majoritaire. Il classe les nouvelles recettes dans la classe la plus nombreuse. Ici il classe toutes les recettes en Desserts. On suppose qu'il classe toutes les recettes dans la première classe qu'il rencontre (certainement par ordre alphabétique, étant donné qu'elles possèdent le même nombre de recettes chacune).

Précision	0,111
Rappel	0,333
F-mesure	0,167
Temps de calcul	0 s.
Lisibilité	Facile à interpréter

MATRICE DE CONFUSION			
a	b	c	< – classified as
20	0	0	a = Dessert
20	**0**	0	b= Entrée
20	0	**0**	c = Plat

FT / J48

Ces classifieurs font partie de la famille des arbres. Ils construisent un arbre de décision dont les nœuds sont des critères liés à des attributs. La lecture d'un arbre correspond à une suite de tests dont les réponses amènent à des décisions (choix de classe).

• FT :

Class 0 : *-2.05 +* *[chocolat] * 0.16 +* *[cuisson] * 0.59 +* *[heures] * 0.61 +* *[poivre] * -4.27 +* *[poivrez] * -0.86 +* *[préparation] * 0.56 +* *[revenir] * -0.54 +* *[sucre] * 1.73*	*Class 1 :* *-1.23 +* *[coupez] * 1.13 +* *[dés] * 0.67 +* *[eau] * -0.84 +* *[entrée] * 3.11 +* *[flamber] * -0.82 +* *[imprimer] * 1.32 +* *[légumes] * 0.39 +* *[plat] * -0.85 +* *[rondelles] * 0.74 +* *[soupe] * 0.17 +* *[vinaigrette] * 0.62*	*Class 2 :* *-4.28 +* *[bouillante] * 1.54 +* *[cuire] * 0.46 +* *[eau] * 0.41 +* *[kg] * 1.27 +* *[moyen] * 1.49 +* *[oignons] * 1.47 +* *[parmesan] * 0.39 +* *[plat] * 0.89 +* *[poivre] * 0.65 +* *[principal] * 1.39*

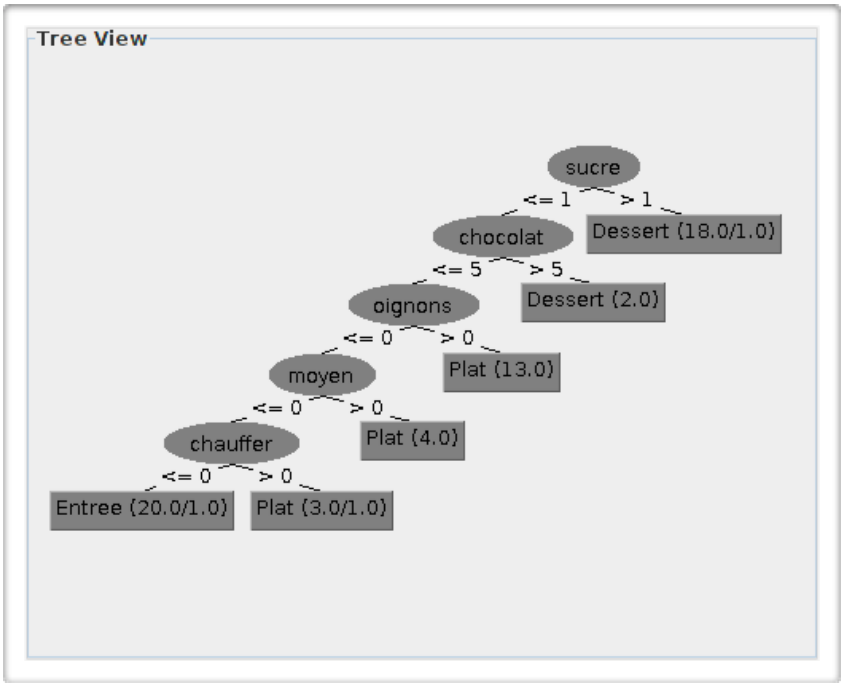
Précision	0,952
Rappel	0,95
F-mesure	0,95
Temps de calcul	0,62 s.
Lisibilité	Assez facile à interpréter, sauf pour la visualisation de l'arbre.

MATRICE DE CONFUSION			
a	b	c	< – classified as
20	0	0	a = Dessert
2	**18**	0	b= Entrée
0	1	**19**	c = Plat

→ Il classe tout très bien.

• J48 :

Visualisation de l'arbre de décisions :



Précision	0,803
Rappel	0,8
F-mesure	0,799
Temps de calcul	0,06 s.
Lisibilité	Très facile à interpréter

MATRICE DE CONFUSION			
a	b	c	< – classified as
18	1	1	a = Dessert
2	**16**	2	b= Entrée
1	5	**14**	c = Plat

→ Il classe très bien les Desserts et un peu moins bien le reste.

REMARQUE

Certains classifieurs se basent sur la présence (ou l'absence) d'un seul ingrédient pour classer les recettes. Par exemple, OneR utilise le poivre et DecisionStump utilise le sucre. Ces classifieurs donnent de bons résultats pour les Desserts mais pas pour le reste. Cela ne nous intéresse donc pas.

SYNTHÈSE

Nos critères d'évaluation des classifieurs sont le rappel, la précision, la F-Mesure, la lisibilité et le temps de calcul. Ils sont variables en fonctions des classifieurs. Nous allons donc privilégier de bons résultats pour le rappel, la précision et la F-Mesure, tout en gardant un temps de calcul raisonnable et un minimum de lisibilité.

Les classifieurs ont chacun leur particularité. Certains, comme ZeroR, sont très rapides mais ne donnent pas de bons résultats. Cela permet de dégrossir le travail, mais ce n'est pas intéressant pour ce projet. En revanche, le temps de calcul de la phase d'apprentissage est trop long pour d'autres classifieurs. Certains le sont tellement que nous n'avons pas laissé tourner le classifieur jusqu'au bout (DTNB, NBTree).

Le meilleur programme de classification de Weka pour notre corpus est FT. Ses résultats en terme de Précision, de Rappel et de F-Mesure sont meilleurs que les autres. Cependant, il prend plus de temps. Pour notre corpus, cela ne pose pas réellement de problème mais cela pourrait être le cas pour un corpus beaucoup plus important. Il est facilement interprétable au niveau des règles, mais nous n'avons pas pu accéder à un arbre compréhensible.

ComplementNaiveBayes nous a donné de bons résultats également en terme de Précision, Rappel et F-Mesure. Il est néanmoins légèrement moins bon que FT, mais il est plus rapide. Nous l'aurions privilégié si notre corpus avait été plus long.

SMO a de moins bons résultats que ComplementNaiveBayes mais meilleurs que J48. En revanche, il est difficilement interprétable par les humains alors que J48 est très clair. Ils sont tous les deux rapides. Toutefois, ils se trompent différemment : SMO classe bien les Entrées et J48 classe bien les Desserts.

JRip ne nous a pas donné de résultats satisfaisants. Cependant ses règles de classifications nous paraissent très claires et logiques.

De manière générale, on remarque que la classe Dessert se distingue des deux autres classes. Les Entrées et les Plats peuvent partager les mêmes ingrédients, ce n'est pas le cas avec les Desserts, qui comportent plus de sucre et moins de sel ou de poivre en général. Les classifieurs ont donc tendance à plus confondre les Entrées avec les Plats, que les Entrées / Plats avec les Desserts.

Nous nous attendions à des résultats différents de ceux obtenus. Nous pensions obtenir les meilleurs résultats avec SMO. Finalement pour notre corpus, FT et ComplementNaiveBayes se sont révélés être les classifieurs les plus adaptés.

Conclusion

Il est théoriquement prouvé qu'il est impossible qu'un algorithme de classification soit le meilleur dans tous les cas possibles. C'est pour cela que Weka garde un grand choix de programmes de classification. En effet, on ne peut pas connaître à l'avance le meilleur classifieur pour un corpus.

De même, une certaine représentation d'un texte ne sera pas la meilleure dans tous les cas possibles. Par exemple la présence des mots vides peut donner des indices sur des sujets qui ne sont pas pertinents pour notre analyse (auteur d'un texte, ...). La présence des chiffres aurait aussi pu nous être utile dans un autre contexte (si toutes les recettes provenaient du même site internet et que les mesures étaient normalisées).

Nous avons donc tenté de trouver la meilleure combinaison entre une représentation des textes et un classifieur. Nous avons déduit de nos expériences que la meilleure classification était obtenue avec le classifieur FT sur notre corpus délesté des mots vides, des chiffres et du nom du site «cuisineaz.com».

Ressources

- 750g : http://www.750g.com
- Auféminin : http://recette-de-cuisine.aufeminin.com
- CuisineAZ : http://www.cuisineaz.com
- Doctissimo : http://recettes.doctissimo.fr
- Marmiton : http://www.marmiton.org
- Notrefamille : http://cuisine.notrefamille.com
- Recette-dessert : http://www.recette-dessert.com