Easy Fridge

Systèmes à recommandations

INFO-F-308 : Printemps des Sciences

HUBAUT Youri, RIGAS Théofanis, VANSPOUWEN Tristan

Année académique : 2015 - 2016

11 avril 2016

Table des matières

Chapitre 1

Introduction

1.1 Plan

Dans ce rapport, nous présentons le travail que nous avons effectué pour le *Printemps des Sciences*. Nous commencerons par établir la problématique que nous étudierons, nous donnerons ensuite un bref aperçu des diverses technologies qui permettent de répondre à celle-ci. Nous continuerons sur la technique que nous avons décidé d'employer et de développer. Enfin, nous terminerons par une analyse de notre application au travers de tests que nous tenterons de critiquer et d'interpréter ainsi que par proposer d'autres procédés afin d'aller plus loin.

1.2 Problématique

Le thème de ce *Printemps des Sciences* est l'alimentation, nous avons donc choisi un sujet en accord, celui du gaspillage alimentaire. Cette problématique a pris de plus en plus d'ampleur ces dernières années et il est important de lutter contre ses répercussions. En effet, ce sujet de préoccupation a beaucoup de conséquences néfastes sur la société en général et son mode de consommation. Alors que des personnes continuent de mourir de faim dans les pays en voie de développement, nous sommes parfois obligés de jeter de la nourriture à la poubelle ! L'industrie agroalimentaire ne sait pas toujours comment récupérer ses déchets afin de les retransformer à des fins utiles comme de l'engrais ou de l'énergie. L'économie et notre portefeuille sont également touchés par ces biens qui ont perdu toute leur valeur et ne contribue donc plus au cycle dans la société.

1.3 Easy Fridge

Nous avons donc développé une application pour téléphone portable afin de lutter contre le gaspillage alimentaire domestique. Nous essayons d'endiguer les principales causes de ce phénomène que sont :

- La mauvaise gestion du réfrigérateur, par le biais d'une classification erronée des produits selon leur date de péremption.
- Le manque d'idées quant à l'utilisation des denrées alimentaires.
- Les achats inutiles que nous effectuons parce que nous avons succombé à des promotions.

Pour cela, nous proposons d'avoir un suivi des dates de péremption et de recevoir des notifications lorsque les produits approchent de leur terme. Nous mettons également à disposition la possibilité de chercher des recettes sur base du contenu de notre réfrigérateur afin de limiter le nombre d'achats à effectuer. Enfin, nous avons également une option pour proposer des aliments qui sont gustativement associés entre eux. [1].

Outre ces aspects ayant attrait à la problématique que nous souhaitons traiter, notre application propose également de pouvoir scanner les barre-codes des produits afin de les insérer plus facilement à notre système ou de synchroniser le contenu du réfrigérateur au travers des différents membres d'une même "famille".

Afin d'effectuer des propositions de recettes ou d'aliments gustativement associés, nous allons étudier un vaste champ de l'informatique, celui des systèmes de recommandations. Nous présenterons un bref aperçu des différents aspects qui entrent en jeu dans cette branche récente et nous nous attarderons sur un système en particulier, celui que nous avons mis en place afin de résoudre le problème de recherche des ingrédients qui ont des affinités pour s'associer. Nous présenterons le contexte global de notre démarche et nous ne présenterons que succinctement le cas que nous avons traité afin de laisser au lecteur libre court à son imagination.

1. Si nous introduisons le mot "poulet", nous pourrions obtenir comme proposition l'épice "curry"

Chapitre 2

État de l'Art

2.1 Introduction

Les systèmes de recommandations sont devenus tellement courants ces dernières années dans notre consommation de la toile qu'on ne cesse de les utiliser sans même s'en apercevoir. Que serait Internet sans les moteurs de recherche, les sites de ventes en ligne ou les plateformes multimedias ? Toutes ces pages ont pour point commun de suggérer une liste de propositions basées sur des informations introduites, potentiellement passivement, par l'utilisateur. L'abondance d'information en ligne ainsi que leur caractère dynamique et leur hétérogénéité a rapidement augmenté la difficulté de trouver ce que nous voulons, quand nous en avons besoin et en accord avec nos attentes. Les systèmes de recommandations étudient le profil de l'internaute et tentent de déterminer quelles seront les suggestions les plus pertinentes à lui fournir en fonction du contexte afin d'influencer ces choix. Ils servent de guides personnalisés aux utilisateurs pour les aider à trouver ce qui pourrait les intéresser dans un large choix qui s'offre à eux. Il existe trois grandes familles de techniques pour répondre à ce problème :

La première se base sur une étude du contenu d'un ensemble discret d'éléments. En étudiant les consommations précédentes et les liens qui les unissent, elle tente de proposer de nouveaux objets partageant certaines propriétés.

La deuxième propose une approche fort différente. En effet, un profil est construit en fonction du passé de l'utilisateur mais celui-ci est alors comparé à ceux d'autres consommateurs ayant les mêmes habitudes. L'idée est davantage de créer des classes d'individus qui partagent de mêmes passions.

La troisième met en jeu les deux précédentes techniques en les combinant afin de profiter au maximum des avantages de chacun de ces systèmes. La fusion des deux approches peut s'effectuer de plusieurs manières et offre donc généralement plus de libertés dans la réalisation de l'objectif.

Cet état de l'art est largement inspiré par le livre *Recommender Systems Handbook*, écrit par Francesco Ricci, Lior Rokach, Bracha Shapira et Paul B. Kantor [1]. Ceux-ci proposent une vision très étendue de tous les aspects qui rentrent en compte dans ce vaste champ de l'informatique. Ainsi que du livre *Recommender Systems : An Introduction*, écrit par Dietmar Jannach, Markus Zanker, Alexander Felfernig et Gerhard Friedrich [2] faisant preuve de beaucoup de pédagogie.

2.2 Data mining

2.2.1 Introduction

Avant de donner un bref aperçu des différentes techniques utilisées pour réaliser des systèmes de recommandations, il nous a semblé essentiel d'aborder certaines notions liées au *Data Mining* [3]. En effet, ce champ de l'informatique s'intéresse au traitement des données en créant des modèles au travers d'outils tels que les statistiques ou l'intelligence artificielle. L'exploration de données consiste généralement en trois étapes : *Data Processing*, *Data Analysis* et *Result Interpretation* [4].

2.2.2 Traitement de l'information

Nous commencerons par définir ce que nous entendons par données comme étant un ensemble d'objets possédant certains attributs. La première étape consiste à nettoyer les données initiales du problème afin de ne garder que l'essentiel. Nous nous intéresserons aux grandes techniques utilisées pour les systèmes de recommandations.

Mesures des distances

L'une des notions primordiales afin de déterminer si des objets peuvent être considérés comme étant liés est celle de distance ou de proximité. Outre les notions usuelles de normes L^p ($\|\overrightarrow{AB}\|_p = (\sum_{k=1}^{n}(b_k - a_k)^p)^{\frac{1}{p}}$), ont été introduites la distance Mahalanobis ($D_M(\overrightarrow{AB}) = \sqrt{(b-a)\sigma^{-1}(b-a)^\intercal}$ où σ est la matrice des données [5] ou encore la distance Bhattacharyya ($D_B(\overrightarrow{AB}) = -\log \sum_{x \in X} \sqrt{p(x)q(x)}$) [6]. Seulement, ces distances peuvent être difficiles à calculer et pour des vecteurs épars, nous pouvons utiliser la notion de similarité cosinus avec la définition euclidienne du produit scalaire ($\cos\theta = \frac{A.B}{\|A\|\|B\|}$). L'avantage de cette similarité est qu'elle se combine très bien avec le coefficient de corrélation de Pearson ($Pearson(a,b) = \frac{\sum(a,b)}{\sigma_a \sigma_b}$) et possède d'autres avantages qui seront vus par la suite [7].

Échantillonage

Collecter des données est une chose mais si cette population devient trop grande, il faut l'échantillonner. L'autre avantage de cette technique est qu'il est plus facile d'entraîner le modèle, au travers d'un choix de paramètres adéquats ou de petites personnalisations dans les algorithmes. Ou encore, tester un ensemble de données afin de s'assurer de l'efficacité du système. La difficulté réside essentiellement dans la manière de sélectionner les éléments représentatifs. Plusieurs heuristiques ont été proposées et généralement la communauté utilise *le principe de Pareto* qui prétend que 80% des données sont utilisées comme entraînement et 20% comme test. Bien sûr, cette méthode pourrait entraîner des optimums locaux et nous chercherons donc à faire une validation croisée [8].

Réduction de dimensionalité

Il est courant que l'espace dans lequel les systèmes de recommandations travaillent soit épars. De plus, les notions de distance et de densité, pourtant critiques aux notions de proximité, deviennent moins significatives avec les dimensions d'ordre supérieur au

travers du phénomène de *Curse of Dimensionality* [9]. Il existe deux grandes familles pour réduire en dimensionalité : Principal Component Analysis (PCA) et Singular Value Decomposition (SVD) [10].

L'analyse de la composante principale consiste à supprimer les termes ayant le moins d'influence sur la variance tout en essayant de rester un maximum significatif. Cependant, cette technique possède de fortes limitations, notamment le fait que l'ensemble des données s'exprime dans une certaine base située au barycentre et que cette distribution des données soit à peu près gaussienne [11].

La décomposition en valeur singulière est plus récente et consiste à obtenir une matrice contenant toutes les valeurs singulières en décomposant la matrice de données. Il suffit alors de tronquer cette matrice pour diminuer en dimensionnalité. Elle a l'avantage qu'il existe des algorithmes permettant de calculer la décomposition approximative de manière itérative et qu'elle conserve les angles nécessaires à la simularité cosinus [12].

2.2.3 Analyse des données

Maintenant que les bases ont été posées, il ne reste plus que le plus dur à réaliser ! Nous appellerons *classificateur* ce qui permet d'attribuer une classe ou une catégorie à chaque objet (ou individu) à classer, en se basant sur des données statistiques. Et les différentes techniques permettant une classification font partie de la branche de *Machine Learning* appellée *Pattern Recognition*. Nous parlerons de *régression* lors qu'il s'agit du problème inverse, de déterminer si des variables mises ensemble sont liées entre-elles. Il existe énormément de techniques pour résoudre ces problèmes et nous ne présenterons un choix non exhaustif [13].

Les k plus proches voisins

Les k-voisins les plus proches ou k-NN est un *Instance-based*, c'est à dire qu'il travaille en stockant des données d'entraînement et les utilise pour déterminer la classe des nouveaux éléments. Comme son nom l'indique, à partir d'un point, l'algorithme va effectuer ses recherches dans son voisinage direct. Il va alors chercher la meilleure classe correspondant à ces caractéristiques. L'idée de l'algorithme consiste à trouver un sous-ensemble de la base d'entraînement qui minimise la distance par rapport au point [14]. La réelle question est combien de voisins faut-il prendre en compte ? Un trop petit nombre et le classificateur sera sensible aux points extrêmes, un trop grand nombre et les voisins pourraient appartenir à trop de classes différentes. Comme il ne construit pas de modèle explicitement, il est qualifié de *lazy learner*. C'est à la fois un avantage et un inconvénient en ce sens que l'idée est très proche du collaborative filtering et qu'il n'a pas de structures à maintenir, ce qui permet de s'adapter rapidement au changement mais au prix de devoir recalculer le voisinage et la matrix de similarité [15].

Arbre de décisions

Comme leur nom l'indique, ce sont des arbres où chaque nœud représente une décision et les feuilles les catégories. Il existe plusieurs algorithmes qui s'appliquent sur ces structures de données : C4.5, CART, ID3, ... [16]. L'idée consiste généralement à classifier les attributs des données en fonction de leur apport en information au travers de la notion d'entropie, de l'erreur de mauvaise classification ou du coefficient de Gini. Ensuite, nous ajoutons un nœud qui permet de séparer les éléments avec ce même

attribut, puis nous répétons cette action sur ces sous-ensembles jusqu'à obtenir des différences tellement faibles que nous les fusionnons toutes dans une même catégorie. L'avantage de cette technique est que l'arbre est très simple à construire et que classifier des éléments inconnus est très rapide. Malheureusement, déterminer un seuil d'arrêt est critique et si trop faible peut mener à des baisses de qualité. Cette technique est également sensible aux valeurs aberrantes et attributs non pertinents. Ils sont plus souvent utilisés pour faire les premières coupes dans l'espace [17].

Bayes Naïf

Ce classificateur est basé sur la probabilité conditionnelle et le théorème de Bayes. Chaque attribut et classe est une variable aléatoire et nous cherchons à déterminer la meilleure classe en fonction des attributs précédents ($P(C_k|A_1, ..., A_N) \propto P(C_k)P(A_1, ..., A_N|C_k)$). La seconde expression est plus simple à calculer d'autant plus si nous effectuons la simplification dite *naïve* qui consiste à considérer les attributs comme étant indépendants entre-eux. Nous obtenons donc : $P(A_1|C_k)...P(A_N|C_k)$ ce qui devient trivial. L'avantage de cette technique est sa robuste face au bruit et aux attributs non pertinents. Cependant, la simplification effectuée est forte et nous pouvons dès lors utiliser les *Bayesian Belief Networks* (BBN) pour atténuer ces effets, comme les données incomplètes ou les modèles sur-calibrés [18].

Réseaux neuronaux artificiels

Les réseaux neuronaux sont des graphes avec des nœuds inter-connectés par des liens pondérés. Le principe consiste à additionner les valeurs d'entrée majorées par le chemin qu'elles empruntent, transformer cette somme par une fonction et comparer le résultat à un seuil. Si celui-ci est atteint, un signal est émis en sortie, sinon, l'information ne transite pas [19]. Nous classons les couches de neurones en trois catégories : entrée, caché et sortie. Entrée répond à l'arrivée de données, le caché modifie l'entrée et la sortie génère le résultat. Généralement, les sorties ne sont pas utilisées en amont du flux de transmission. Ces réseaux ont l'avantage de pouvoir effectuer des tâches non-linéaires, de facilement passer à l'échelle et d'être parallélisés et enfin de tolérer des parties défaillantes. Seulement, il est difficile de concevoir la topologie du réseau surtout que celle-ci possède une grande influence sur l'erreur [20].

2.3 Filtrage sur contenu

2.3.1 Introduction

Les systèmes de recommandations de type *Content-based* essayent de trouver des objets partageant des similarités avec ceux que l'utilisateur a aimés auparavant. Nous commencerons dans ce chapitre par présenter les concepts de base associés à ces outils. Puis, nous tenterons de déterminer les principaux avantages et inconvénients de cette technique. Enfin, nous présenterons un bref aperçu de l'état de l'art sur la question.

2.3.2 Concepts et bases

Les systèmes de recommandations qui implémentent une approche de type *Content-based* commencent par analyser un ensemble de données liés aux anciennes préférences

d'un utilisateur. À partir de ces informations, ils construisent des modèles, ou profils, des intérêts du consommateur basés sur un classement qu'a effectué l'internaute, au travers de notes explicites (étoiles pour une recette) ou implicites (par le biais de la longueur de la recherche, si il se remet à chercher, la solution n'était probablement pas suffisante) [21]. Les profils sont des représentations structurées des intérêts de l'utilisateur, ils servent à recommander des nouveaux objets en comparant les caractéristiques de celui-ci à celles attendues. Bien sûr, tout l'enjeu réside dans la manière de créer ce profil afin qu'il corresponde au mieux aux attentes du consommateur [22].

Tout cela est bien joli mais comment pouvons-nous structurer au mieux notre programme et quelles sont ses principales composantes ? Généralement, on utilise trois grandes étapes pour effectuer des recommandations.

Le *Content Analyzer* s'occupe d'extraire, à partir de données brutes grâce au *preprocessing*[2.2.2], les informations pertinents aux structures de données (les concepts, mots-clefs, ...). Sa principale tâche consiste donc à traduire les entrées en un format qui puisse être utilisé par les prochaines étapes au travers des techniques d'extraction qui transforme l'espace originel dans lequel se décrivent les données en celui cible (par exemple : convertir une page web en dictionnaire inversé [3.3.4]). Il fournit les données nécessaires aux deux prochaines composantes [1].

Le *Profile Learner* collecte les données qui représentent les préférences de l'utilisateur et essaye d'interpréter celles-ci. Généralement, on effectue la synthèse grâce à des techniques de *Machine Learning* [23]. C'est ici que les influences des *ranking* interviennent avec les notions de pertinence sans action directe de la part de l'utilisateur.

Enfin, le *Filtering component* exploite les données qui lui sont fournies pour effectuer des propositions d'objets en essayant de faire correspondre le profil de l'utilisateur aux potentielles recommandations. Cette correspondance est généralement évaluée au travers des métriques de similarités[2.2.2]. Évidemment, les préférences de l'utilisateur évoluent avec le temps et cette partie doit donc présenter le plus grand potentiel évolutif possible afin d'être toujours le plus proche de l'optimum [24].

2.3.3 Avantages et incovénients

Cette approche des systèmes de recommandations propose certains avantages fort intéressants. On peut citer l'indépendance par rapport aux autres utilisateurs, ce qui signifie que le traitement peut être effectué de manière purement locale. La transparence ; en effet, on a *officiellement* aucun intérêt à espionner le reste de la population avant de fournir des résultats. La légèreté, peu d'informations doivent être récoltées avant d'être effectif ce qui permet de rajouter des nouveaux objets très facilement au système sans souffrir du manque de notes a contrario des *collaborative filtering* [2].

Néanmoins, les inconvénients de cette technique ne sont pas négligeables. En effet, l'analyse du contenu peut être très limitée et donc nécessiter une connaissance du domaine qui va être traité pour être plus efficace. Le manque d'activité de la part de l'utilisateur ou le peu d'interaction avec le système peuvent faire manquer des centres d'intérêts. La sur-spécialisation à la personne et le manque de généralisation peuvent mener à des blocages de résultats (*serendipity*). Enfin, on peut se demander ce qui va se passer lors de l'arrivée d'un nouvel utilisateur dans le système [1].

2.3.4 État de l'art

La recherche dans les systèmes de recommandations est le lieu de réunion de divers pans de l'informatique, notamment l'intelligence artificielle ou *Information Retrieval / Filtering*. En effet, ces systèmes sont à la recherche d'informations sur les objets et peuvent se manifester au travers d'une recherche textuelle explicite (Google). Ainsi que le côté d'intelligence artificielle parce qu'ils prennent en comptent le passé pour effectuer des choix et utilisent des techniques liées au *Machine Learning*. Beaucoup de travaux ont été effectuées sur le sujet mais nous n'effectuerons qu'un choix purement arbitraire parmi tous ceux-ci [25].

Représentation des objets

Les informations brutes peuvent contenir des imperfections par l'intermédiaire des mots utilisés par exemple. En effet, les mots peuvent avoir plusieurs paradigmes (polysémie) ou être des synonymes. Il faut donc effectuer une analyse sémantique des termes employés. Heureusement pour nous, ils existent diverses approches possibles. Notamment analyser de manière pure et brute une grande quantité d'informations diverses et variées comme le site web *Wikipedia* afin de déterminer les concepts et leurs champs sémantiques [26]. Une des techniques célèbres pour déterminer l'importance des mots est lié à la notion de :

Modèle d'espace vectoriel des mots-clefs Généralement, on emploie la technique du *TF-IDF* (*Term Frequency-Inverse Document Frequency*) dans la représentation spatiale du document appelée Vector Space Model dit *de Salton* [27]. Dans ce modèle, chaque document est représenté par un vecteur de dimension n où chaque composante représente un terme employé avec sa fréquence. Nous définissons l'ensemble des N documents comme étant D avec un vocabulaire T, le vecteur associé à un document par $d_j = \{w_{1j}, ..., w_{nj}\}$ où w_{kj} est le nombre d'occurrence du terme t_k dans le document d_j. Ce formalisme ayant été effectué, cette représentation n'est pas suffisante parce qu'il manque la mesure des poids et la notion de similarité. C'est pourquoi, on emploie souvent le *TF-IDF* qui se base sur quelques hypothèses : les termes rares ont beaucoup de poids (*IDF*) s'ils apparaissent rarement (*TF*) tout en ne prenant pas en compte la taille du document [28]. Voici une des fameuses fonctions *TF-IDF* parmi tant d'autres :

$$TF - IDF(t_k, d_j) = TF(t_k, d_j) \log \frac{N}{n_k} \quad \text{avec} \quad TF(t_k, d_j) = \frac{f_{k,j}}{max_z f_{z,j}}$$

Où n_k est le nombre de documents qui possèdent le terme t_k et où le maximum de la fréquence t_z dans le document d_j. Enfin pour que les vecteurs aient la même norme, on normalise :

$$w_{k,j} = \frac{TF - IDF(t_k, d_j)}{\sqrt{\sum_{s=1}^{T} TF - IDF(t_s, d_j)^2}}$$

Enfin, pour estimer la similarité, on emploie généralement celle du cosinus. Il suffit donc de tester le lien entre le profil et l'objet considéré [29].

$$sim(d_i, d_j) = \frac{\sum_k w_{ki} w_{kj}}{\sqrt{\sum_k w_{ki}^2} \sqrt{\sum_k w_{kj}^2}}$$

D'autres propositions ont été effectuées sur ce sujet, on parle de Letizia [30], Personal WebWatcher [31], Syskill & Webert [32], du livre *Recommender Systems Handbook* [1] et dans l'article *Content based recommendation systems* [25].

Apprentissages des profils

L'idée principale dans les méthodes de *Machine Learning* est de déduire des profils à partir d'informations grâce à une phase d'entraînement afin d'être ainsi capable de déterminer si un objet correspond aux besoins de l'utilisateur ou non. Cela revient à créer un classificateur.

Bayes Naïf Nous cherchons à déterminer la probabilité que le document d appartienne à la composante c ($P(c|d)$) et chaque composante est paramétrée par un sous-ensemble disjoint θ qui intervient dans la génération des documents. Seulement, cela peut s'avérer fort complexe et on peut utiliser la propriété de Bayes. Nous obtenons $P(c)P(d|c)$ car $P(d)$ est le même pour toutes les classes. On tombe sur $P(d_i|\theta) = \sum_{j=1}^{|C|} P(c_j|\theta)P(d_i|c_j;\theta)$. On effectue alors l'hypothèse d'indépendance des classes et on peut utiliser le modèle *multivariate Bernouilli* ou celui *multinomial* [33]. Dans le cas multinomial, on tombe sur :

$$P(d_i|c_j;\theta) = P(|d_i|)|d_i|!\prod_{t=1}^{|V|}\frac{P(w_t|cj;\theta)^{N_{it}}}{N_{it}!}$$

Avec le vocabulaire V, le nombre t de fois qu'apparaît le mot w et N_{it} le nombre de w_t dans d_i. Et avec l'estimation optimale de Bayes :

$$\theta_{w_t|c_j} = P(w_t|c_j;\theta_j) = \frac{1+\sum_{i=1}^{|D|} N_{it}P(c_j|d_i)}{|V|+\sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N_{is}P(c_j|d_i)}$$

Pertinence du retour et algorithm de Rocchio Les méthodes de *Relevance Feedback* tentent d'améliorer les recherches en se basant sur celles précédentes. C'est dans ce but que l'algorithme de Rocchio a été proposé. Il consiste à modifier la demande en pondérant différemment les éléments en fonction de leur pertinence. Voici la formule en question un peu modifiée [34] :

$$Q_{i+1} = \alpha Q_i + \beta\sum_{rel}\frac{D_i}{|D_i|} - \gamma\sum_{nonrel}\frac{D_i}{|D_i|}$$

L'idée est de réorienter le vecteur de recherche dans l'espace afin de découvrir de nouvelles régions en effectuant une combinaison linéaire sur les vecteurs des documents. Attention, ni la convergence, ni la performance n'est garantie par cette méthode.

2.4 Collaborative filtering

2.4.1 Introduction

En opposition aux méthodes qualifiées de *Content-based* qui utilisent les caractéristiques des objets précédemment évalués par l'utilisateur, l'approche *Collaborative filtering* se base sur l'idée que si une personne a émis les mêmes avis sur les mêmes

produits qu'un autre, alors il est probable que ces individus partagent les mêmes goûts et donc que les propositions puissent être partagées. On distingue deux grandes sous-catégories, les méthodes *Neighbourhood* et *Model-based* [35].

2.4.2 Avantages et inconvénients

Ces techniques possèdent certaines limitations parce qu'elles peuvent recommander des objets qui sont interdits dans certains pays. Elles ne se basent uniquement que sur les notes des produits et non leurs qualités. Elles ont dû mal à gérer les nouveaux produits non évalués. Elles sont sujettes à des tentatives de triche avec des personnes essayant de profiter du système. Elles sont également par essence liées à une connexion Internet. Mais cette approche permet d'obtenir des objets dans plusieurs catégories sans se retrouver coincé dans des systèmes sur-calibrés. Elle permet de gérer très facilement l'arrivée de nouveaux acteurs [36].

2.4.3 Approches locales

Dans ces approches, les notes d'un utilisateur pour un objet sont directement stockées dans le système et utilisées pour effectuer les prédictions. Elles se subdivisent encore en *User-based* comme dans GroupLens [37] et Ringo [38] et en *Item-based*, beaucoup plus vaste. Les avantages de ces approches sont leur aspect intuitif et justifiable, les performances et la stabilité.

Pour trouver les voisins, on peut employer la technique du *k-NN*[2.2.3] mais on peut se poser la question d'effectuer un choix de classificateur ou de régression. Bien sûr, la proximité entre les objets ou les personnes peut également être débattue. Cette décision, souvent importante, doit s'effectuer selon certains critères comme la précision et l'efficacité (s'il y a plus d'objets que d'utilisateurs), la stabilité (qui est le plus évolutif), le pouvoir justificatif (qui donnerait avantage aux objets) ou la sérendipité (qui donnerait avantage aux utilisateurs) [39].

Normalisation du classement Il est généralement préférable que l'on puisse comparer des choses comparables entre-elles. Notamment les notes attribuées, il existe deux grandes techniques, le *Mean centering*, normalisation pure et dure [37] et le *Z-score* qui prend en compte la variance [26].

Calcul de la similarité du poids Dans la même optique, on peut équilibrer plus ou moins les recommandations en fonction de la confiance qu'on a dans le système au travers de celle des notes ou des voisins. Bien choisir la méthode est important car elle impacte fortement la précision et la performance du système. Outre la méthode classique de la similarité cosinus[2.2.2], il existe la *Mean Squared Difference* ou la *Spearman Rank Correlation*[40].

Sélection voisine Généralement, on effectue deux étapes afin de déterminer le voisinage, on garde ceux qui ont l'air de convenir dans tout l'ensemble et dans ceux-ci, on évalue les prédictions. Le choix du nombre de voisins est très ardu, trop peu et les résultats seront trop sensibles aux valeurs aberrantes et un excès pourrait induire un affaiblissement des liens avec la cible. On prend généralement un nombre entre 20 et 50 [26, 41].

2.4.4 État de l'art

Énormément de techniques et d'approches ont été proposées pour résoudre certains des problèmes liés à la méthode *Collaborative Filtering*, nous ne ferons que nommer certaines de ces tactiques parmi une multitude de techniques.

Les modèles à factorisation de matrices ont gagné en popularité ces derniers temps grâce à leur précision et leur mise à l'échelle. Outre, la méthode dite SVD[2.2.2], il existe la SVD++ ou la *Probabilistic Matrix Factorization* [10]. On signale également les modèles au courant du temps [42].

D'autres procédés peuvent être découverts dans *Recommender systems handbook* [1], *Item-based collaborative filtering* [43] ou encore *Recommender systems survey* [44].

2.5 Hybrid

2.5.1 Introduction

Nous l'avons vu, les méthodes collaboratives et basées sur le contenu sont différentes, elles ne partagent ni les mêmes avantages ni les mêmes inconvénients. Il est donc logique qu'on ait essayé de les combiner en profitant de certaines caractéristiques de l'une et en comblant les lacunes par l'autre. Par exemple, les *collaborative filtering* souffre d'un problème lié aux nouveaux produits à cause de leur manque de notes, mais les *content-based* savent se débrouiller uniquement sur base de la description des biens. Elles ne souffrent donc pas de l'ajout de nouveaux produits, par contre elles souffrent de l'émergence de nouveaux utilisateurs, ... [45].

2.5.2 Techniques de recommandation

Il faut garder à l'esprit qu'on décrit généralement deux grandes familles de systèmes de recommandations, mais ce serait oublié les *Demographic*, *Utility-based* ou *Knowledge-based*. L'idée principale est de combiner des systèmes de recommandations différents afin de bénéficier d'un gain en performances et de limiter les désavantages de chacun des procédés [25].

2.5.3 Hybridation

Mais comment fait-on pour combiner des systèmes de recommandations me direz-vous ? Et bien comme d'habitude, plusieurs techniques sont disponibles :

Pondération L'idée est relativement triviale, nous avons plusieurs méthodes de recommandations, pourquoi ne pas simplement les combiner ! *P-Tango system* [46] propose de faire une combinaison linéaire et d'ajuster les paramètres en fonction du retour des utilisateurs par le biais des notes.

Changement Selon des critères bien déterminés, le programme préférera utiliser un système de recommandations à un autre. C'est le cas pour *DailyLearner* [47] sans pour autant règler les problèmes intrinsèques de ces méthodes.

Mixage Cela concerne davantage la représentation de la solution avec une combinaison des éléments des différents systèmes. Par exemple, les solutions locales ou *content-based* à la suite des recommandations collaboratives. C'est le cas pour *PTV-systems* [48].

Bien sûr, nous n'oublierons pas de citer les techniques de combinaisons de fonctionnalité [49], de cascade [50], d'augmentation [37, 51] et de meta-level [22].

2.6 Conclusion

Nous l'avons vu, les systèmes de recommandations sont un grand pan de l'informatique et beaucoup de littérature a été écrite sur ce sujet. Nous espérons toute fois avoir su faire percevoir les difficultés et les manières d'aborder les problèmes liés à ces techniques. Nous espérons également que vous vous passionnerez pour ce champ et irez chercher plus loin tant le nombre de techniques et de sciences impliquées est grand.

Bien sûr, nous n'avons pas aborder d'autres problématiques telles que la manière de tester de telles techniques, comment comparer les performances, comment les entraîner et les faire évoluer, ...

Bibliographie

[1] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*. New York, NY, USA : Springer-Verlag New York, Inc., 1st ed., 2010. pages 5, 9, 11, 13

[2] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, *Recommender systems : an introduction*. Cambridge University Press, 2010. pages 5, 9

[3] J. Han, M. Kamber, and J. Pei, *Data mining : concepts and techniques*. Elsevier, 2011. pages 6

[4] D. Pyle, *Data preparation for data mining*, vol. 1. Morgan Kaufmann, 1999. pages 6

[5] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936. pages 6

[6] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 917–922, 1977. pages 6

[7] P.-N. Tan, M. Steinbach, V. Kumar, *et al.*, *Introduction to data mining*, vol. 1. Pearson Addison Wesley Boston, 2006. pages 6

[8] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, 1995. pages 6

[9] R. Bellman and R. Corporation, *Dynamic Programming*. Rand Corporation research study, Princeton University Press, 1957. pages 7

[10] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009. pages 7, 13

[11] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. pages 7

[12] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970. pages 7

[13] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition : A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 1, pp. 4–37, 2000. pages 7

[14] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, January 1967. pages 7

[15] D. L. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *Systems, Man and Cybernetics, IEEE Transactions on*, no. 3, pp. 408–421, 1972. pages 7

[16] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986. pages 7

[17] O. Maimon and L. Rokach, *Data mining and knowledge discovery handbook*, vol. 2. Springer, 2005. pages 8

[18] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997. pages 8

[19] F. Rosenblatt, "The perceptron : a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958. pages 8

[20] M. Pazzani and D. Billsus, "Learning and revising user profiles : The identification of interesting web sites," *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997. pages 8

[21] D. W. Oard, J. Kim, *et al.*, "Implicit feedback for recommender systems," in *Proceedings of the AAAI workshop on recommender systems*, pp. 81–83, 1998. pages 9

[22] M. Balabanović and Y. Shoham, "Fab : content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997. pages 9, 14

[23] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning : An artificial intelligence approach*. Springer Science & Business Media, 2013. pages 9

[24] M. E. Prieto, V. H. Menéndez, A. A. Segura, and C. L. Vidal, "A recommender system architecture for instructional engineering," in *Emerging Technologies and Information Systems for the Knowledge Society*, pp. 314–321, Springer, 2008. pages 9

[25] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The adaptive web*, pp. 325–341, Springer, 2007. pages 10, 11, 13

[26] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 43–52, Morgan Kaufmann Publishers Inc., 1998. pages 10, 12

[27] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, pp. 613–620, Nov. 1975. pages :1975 :VSM :361219.361220

[28] G. Salton, "Automatic text processing : The transformation, analysis, and retrieval of," *Reading : Addison-Wesley*, 1989. pages 10

[29] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972. pages 10

[30] H. Lieberman *et al.*, "Letizia : An agent that assists web browsing," *IJCAI (1)*, vol. 1995, pp. 924–929, 1995. pages 11

[31] D. Mladenic and B. A. P. Webwatcher, "Machine learning used by personal webwatcher," 1999. pages 11

[32] M. J. Pazzani, J. Muramatsu, D. Billsus, *et al.*, "Syskill & webert : Identifying interesting web sites," in *AAAI/IAAI, Vol. 1*, pp. 54–61, 1996. pages 11

[33] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Citeseer, 1998. pages 11

[34] J. J. Rocchio, "Relevance feedback in information retrieval," 1971. pages 11

[35] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems : A survey of the state-of-the-art and possible extensions," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 6, pp. 734–749, 2005. pages 12

[36] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004. pages 12

[37] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens : applying collaborative filtering to usenet news," *Communications of the ACM*, vol. 40, no. 3, pp. 77–87, 1997. pages 12, 14

[38] U. Shardanand and P. Maes, "Social information filtering : algorithms for automating "word of mouth"," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217, ACM Press/Addison-Wesley Publishing Co., 1995. pages 12

[39] J. Wang, A. P. De Vries, and M. J. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 501–508, ACM, 2006. pages 12

[40] R. L. Iman and W.-J. Conover, "A distribution-free approach to inducing rank correlation among input variables," *Communications in Statistics-Simulation and Computation*, vol. 11, no. 3, pp. 311–334, 1982. pages 12

[41] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, "Combining collaborative filtering with personal agents for better recommendations," in *AAAI/IAAI*, pp. 439–446, 1999. pages 12

[42] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010. pages 13

[43] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, ACM, 2001. pages 13

[44] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013. pages 13

[45] R. Burke, "Hybrid web recommender systems," in *The adaptive web*, pp. 377–408, Springer, 2007. pages 13

[46] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *Proceedings of ACM SIGIR workshop on recommender systems*, vol. 60, Citeseer, 1999. pages 13

[47] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," *User modeling and user-adapted interaction*, vol. 10, no. 2-3, pp. 147–180, 2000. pages 13

[48] B. Smyth and P. Cotter, "A personalised tv listings service for the digital tv age," *Knowledge-Based Systems*, vol. 13, no. 2, pp. 53–59, 2000. pages 14

[49] C. Basu, H. Hirsh, W. Cohen, *et al.*, "Recommendation as classification : Using social and content-based information in recommendation," in *AAAI/IAAI*, pp. 714–720, 1998. pages 14

[50] R. Burke, "Hybrid recommender systems : Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002. pages 14

[51] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the fifth ACM conference on Digital libraries*, pp. 195–204, ACM, 2000. pages 14

Chapitre 3

Méthodologie

3.1 Introduction

Dans cette partie de ce travail, nous allons nous intéresser tout particulièrement aux techniques mises en œuvre afin de résoudre notre type de problème. Nous commencerons par expliquer la manière dont nous avons perçu le problème, nous aborderons ensuite les diverses manières qui ont été proposées pour résoudre cette question. Et enfin, nous présenterons la solution que nous avons développée, ses tenants et aboutissants ainsi que les divers choix que nous avons dûs effectuer.

Nous vous le rappelons, notre problématique consiste, sur la base d'une liste d'ingrédients définie par l'utilisateur, d'en proposer d'autres qui sont gustativement associés. Un exemple serait que sur la base des mots "poulet" et "curry", notre algorithme nous propose du "riz". Il faut pour effectuer cette action, être capable d'associer des ensembles de mots ou *ingrédients*, sur la base de concepts ou *recettes*. Mais avant de voir comment nous pouvons parvenir à de tels résultats, nous allons faire un rappel des principes sous-jacents.

3.2 Modèle vectoriel

3.2.1 Modèle

La manière d'associer des concepts ensemble ou de rechercher des informations sont intrinsèquement liées. En effet, la recherche d'information [2.2.3] a pour objectif de trouver de manière performante une information perdue dans un vaste ensemble, souvent appelé *corpus*. Ce corpus peut être très large ou contenir des informations de nature très variée et il peut donc être nécessaire d'avoir un retour de la part de l'utilisateur sur la pertinence de la réponse envoyée, afin de potentiellement mieux réorienter la recherche ou de simplement d'améliorer les résultats futurs.

Nous pouvons faire le parallèle avec la "vie réelle", en considérant le cas de la recherche d'un livre dans une bibliothèque. Imaginons, que vous soyez très intéressés par les populations, vous vous dirigerez de manière naturelle vers les rayonnages d'anthropologie, sachant pertinemment que votre recherche est bien vague. Mais si vous vous vouez de passion pour les indigènes d'Amazonie, alors vous aurez une bien plus grande précision dans votre démarche et chercherez le livre aux confluents de l'anthropologie et de l'Amazonie. Seulement, vous imaginez bien que ce système a des "failles" et que certains résultats, qui pourraient vous intéresser, se trouvent ailleurs, en mathématiques

par exemple [1]. Il semble également évident que certains mots ont un poids beaucoup plus important et fournissent davantage de renseignements que d'autres. Il faut donc mieux orienter la recherche vers ces mots remplis de sens tout en respectant la volonté générale de la demande.

Une approche purement ensembliste et binaire n'est pas suffisante. Il nous faut définir un nouveau modèle qui nous permettra de trouver des réponses pertinentes aux requêtes, et ce, quelque soit la demande. Il peut donc être nécessaire d'observer un haut niveau d'abstraction. Bien sûr, pour pouvoir traiter nos informations, nous devrons mettre en place un formalisme afin de définir avec exactitude le contenu de notre corpus. Se poser des questions sur la nature même de ces objets, afin de mieux guider nos choix de représentations et ainsi avoir une plus grande appréciation sur la pertinence.

Ainsi, dans le domaine de l'informatique, les modèles nous aident à apporter un cadre déterminé et logique. Ils visent également à supporter notre intuition lors de la description de notre problématique. Ils servent d'abstraction et permettent de mettre en place toutes les propriétés mathématiques que nous pourrons employer. En l'occurrence, notre modèle servira à définir la représentation que prendront les documents du corpus, les requêtes et la notion de pertinence entre une demande et un résultat éventuel.

Mais comment élabore-t-on un modèle ? Il n'y a pas de réponses absolues, en ce sens que rien n'est jamais figé. Des découvertes ou des raisons techniques poussent à les faire évoluer, ils se peaufinent peu à peu avec le temps et les retours des utilisateurs. Comme les problèmes sont souvent mal définis, il reste une part de choix dans les paramètres qui constituent notre modèle et ceux-ci obtiennent une valeur souvent de manière empirique pour aboutir à des résultats toujours meilleurs. Par la suite, nous présenterons un modèle, parmi d'autres, qui est relativement commode à étudier et largement utilisé à travers la communauté informatique.

3.2.2 Modèle vectoriel

Énormément de modèles sont issus de diverses observations de la nature et le modèle vectoriel n'échappe pas à cette règle. L'idée est relativement simple, et permet une représentation aisée graphiquement. Cela a l'avantage de nous permettre de trouver plus facilement des liens, et de pouvoir disposer des outils proposés par la géométrie afin de mieux étudier ces corrélations.

On associe généralement le modèle vectoriel, avec son application aux données de type texte, aux travaux fournis dans les années 70 par Gerard Salton avec notamment son article *A Vector Space Model for Automatic Indexing* (1975) [2] et ses ouvrages sur le sujet qui restent encore toujours des références actuellement : *Introduction to Modern Information Retrieval* (1983) [3] ou *Automatic Text Processing* (1989) [4].

Le principe de base de ce modèle est qu'on peut assimiler un document à une suite de termes appartenant au lexique d'une langue. En algèbre linéaire, chacun des mots de ce lexique correspondrait à un vecteur de base d'un espace vectoriel. Un exemple serait la phrase : "Plus blanc que blanc" qui pourrait se représenter mathématiquement par $1*plus+2*blanc+1*que$. Ainsi, par cette application, nous pouvons faire correspondre nos documents à des vecteurs dont la dimension est égale à la taille du lexique. Il faut également avoir en tête que, de fait, le vecteur est épars, en ce sens qu'un document ne contient qu'une petite partie d'un corpus de mots, beaucoup de composantes valent zéro.

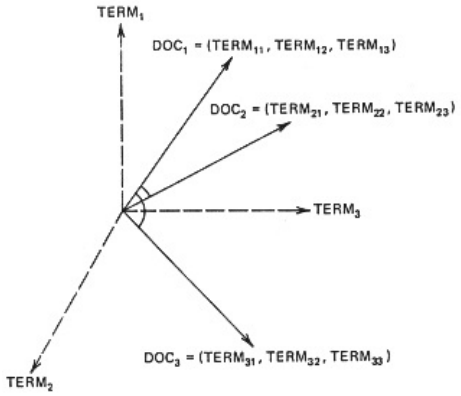

FIGURE 3.1 – Modèle vectoriel avec trois termes - tiré de Salton

3.2.3 Similarité cosinus

Cette représentation est relativement agréable mais ne nous permet pas encore de définir les documents les plus adéquats pour une requête donnée. Nous allons ici supposer que la similarité entre deux documents est simplement liée à la proximité de leur champ lexical et qu'une requête est un document par nature très court. Le problème revient à déterminer si le vecteur que représente notre requête nous envoie bien dans la bonne direction des résultats qui lui serait pertinente.

Nous pourrions estimer la pertinence d'un résultat comme sa distance par rapport au vecteur de demande, ainsi prendre une métrique qui lie les deux extrémités des vecteurs ou une notion qui va souvent de pair avec le concept même de distance, celui de la mesure de l'angle. Pour des raisons expliquées un peu plus loin, la distance L_2 (ou euclidienne) a moins de sens que la similarité cosinus et, on privilégie l'emploi de cette dernière. Nous pouvons caractériser la similarité entre deux documents comme étant l'angle formé entre le vecteur de requête et celui du document. Plus l'angle entre les deux est faible, plus les documents sont analogues. Il suffit alors simplement d'appliquer la formule brute du cosinus (si les deux vecteurs ne sont pas nuls) et nous obtenons le taux de similarité [5].

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

où A_i et B_i correspondent à la composante associée au terme i. Si ces poids sont strictement positifs, nous obtenons donc que plus la similarité est grande, plus ce terme tend vers 1 :

$$0 \leq similarity(A, B) \leq 1$$

3.2.4 Vous avez dit mesure ?

Il existe à priori plusieurs moyens de quantifier le taux de similarité entre deux documents dans ce modèle, toute métrique pouvant répondre à ce problème. Seulement, le choix d'une peut avoir de grandes conséquences sur les résultats obtenus. Par exemple, si nous employons la similarité cosinus, la taille des vecteurs n'a absolument aucune

importance puisque nous nous intéressons uniquement à l'angle formé entre ceux-ci. Alors que si nous avions employé une métrique de type L_p, nous n'aurions regardé les résultats que dans un "cercle". Dans les deux cas, il y a une perte d'information et celle-ci peut avoir plus ou moins d'impact sur les résultats, quitte à les biaiser totalement. En pratique, le nombre élevé de dimensions combiné avec celui de documents que l'on considérera pourrait occulter en partie ces effets locaux. Quoiqu'il en soit, il vaut mieux toujours réaliser des tests sur la pertinence [6].

De manière générale, choisir une "bonne" mesure se fait de manière expérimentale avec, potentiellement, une étude statistique sur la qualité de ces résultats. Nous verrons comment vérifier les réponses obtenues et leurs qualités plus tard[4.2]. Il faut également savoir qu'il existe plusieurs phénomènes dont il faut avoir conscience :

- Lorsque le nombre de dimensions devient élevé, des phénomènes qui semblent ne pas exister à faible dimensionnalité commence à avoir un impact de plus en plus important : on parle du *fléau de la dimensionnalité* ou *curse of dimensionality* [7].
- Comment peut-on interpréter les différences de résultats entre les diverses méthodes ? Toujours rester critique sur les observations effectuées !
- Notre corpus est-il de qualité ? Est-ce que les objets fortement décrits sont-ils plus souvent favorisés ou bien le contraire ?
- Nos mesures sont-elles réellement cohérentes ? Nos techniques employées également ? Ont-elles une signification particulière et adaptée en accord avec nos données et notre idée de résultat ?

Bien sûr, en recherche d'information, la mode est d'utiliser la similarité cosinus parce qu'elle semble mieux convenir aux résultats espérés puisqu'elle ne dépend pas de la norme des vecteurs et retourne un résultat entre 0 et 1. La majorité des composantes qui apparaissent dans un document sont nulles dans un autre. Alors que les distances peuvent obtenir des grands nombres qui sont difficilement comparables et sont sensibles à la norme. Les composantes sont également élevées à une puissance et ont donc un poids bien plus grand. Les résultats empiriques semblent indiquer que la similarité cosinus serait, en effet, en moyenne plus pertinent que les méthodes sur base d'une métrique [8]. A noter qu'il existe également des outils statistiques tels que les coefficients de corrélation [9].

3.2.5 TF-IDF

Nous venons de discuter des questions de similarité mais nous n'avons pas abordé un problème plus profond qu'est celui des poids des termes et des valeurs des composantes. Une bonne pondération est important puisque celle-ci détermine la direction générale de ce vecteur. S'ils sont mal orientés et reflètent donc mal la sémantique du document, alors le concept de similarité cosinus s'écroule. Vous accepterez l'idée que certains mots dans un corpus ont beaucoup plus de poids que d'autres. Il existe des mots donnant une idée vague d'un concept et ceux pointus correspondant à des petites variations sémantiques dans un domaine. Généralement, un mot générique apparaîtra dans beaucoup plus de textes que des mots relatifs à une spécialité. On peut donc estimer que l'importance d'un mot est liée à l'importance de ce mot dans la littérature de manière générale et de sa fréquence d'apparition dans un document. Classiquement, pour représenter ce concept, on utilise le modèle du *TF-IDF*.

Le principe même du *TF-IDF* nous vient de M. Salton [10] et part du constat qu'il

est peu judicieux de considérer uniquement la fréquence d'un mot lors du calcul de son poids, il le définit donc comme étant une combinaison de deux aspects.

— La fréquence (**Term Frequency**) [11] est le nombre d'occurrences du terme t dans le document d parmi l'ensemble des mots existants : $tf(t,d) = \frac{n_{t,d}}{N_d}$. Afin que les documents puissent être comparés malgré leur différence de taille ou la répétition excessive d'un même mot-clef, il est nécessaire d'introduire un facteur de normalisation. Ici, il s'agit simplement du rapport entre le nombre d'occurrences et le nombre de termes qui existe, il existe beaucoup d'autres formules possibles. Signalons également la loi de Zipf qui établit que la fréquence d'apparition d'un terme est proportionnelle à l'inverse de son rang [12].

— L'inverse de la fréquence (**Inverse Document Frequency**) [13] qui consiste à diviser le nombre total de documents présents dans le corpus par le nombre de documents contenant ce terme t et d'en prendre le logarithme par exemple : $idf(t,D) = log\frac{|D|}{|d \in D : t \in d|}$. L'idée est de mesurer la rareté du terme, celui-ci aura alors un impact plus ou moins grand sur la recherche. Le logarithme évite que cette fréquence ne soit trop prépondérante.

Le modèle *TF-IDF* consiste donc à estimer le poids d'un terme t qui n'est plus seulement dépendant du document d étudié, mais bel et bien de l'ensemble du corpus D. Nous avons donc :

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

3.3 Prétraitement et recherche

3.3.1 Une question de langue

Une des difficultés rencontrées dans le domaine de la recherche d'information concerne le traitement de la langue. En effet, afin de mieux orienter les recherches, il peut être intéressant d'être capable de déterminer les fonctions des termes employés. Une analyse sémantique ou syntaxique peut améliorer la compréhension globale des demandes effectuées et donc mieux correspondre aux besoins des utilisateurs.

L'étude même de cet aspect de l'informatique est liée à un domaine fort large, le *traitement automatique du langage naturel* ou *Natural Language Processing*. Cette branche vise à étudier les possibilités de traduction automatique, de réalisation des résumés ou de détecter des émotions au travers de l'écrit [14]. Ici, nous n'emploierons qu'une infime partie des techniques employées afin de mieux extraire les mots-clefs d'un texte.

3.3.2 Les mots vides

Il existe, dans les langues, ce qu'on appelle communément des *mots vides*, qui possèdent très peu de sémantique par eux-mêmes. Par exemple : le, du, ce, avoir, être, ... Si nous pondérons les termes par la technique du *TF-IDF*, nous savons que ceux-ci auront un impact très faible sur les recherches, le coefficient *IDF* tendra, en effet, vers zéro. Pour des raisons de ressources mémoires ou de performances, il peut donc être envisageable de les supprimer directement lors de la formation du corpus. Une méthode efficace, pour générer une liste de ces mots, consiste à analyser un corpus qui a l'air analogue et de sélectionner les termes dont le coefficient *IDF* est pratiquement nul. Attention, supprimer ces mots n'est pas sans danger. En effet, il existe des expressions

qui ne sont composées que de celles-ci : "être ou ne pas être" ou "the who". On peut imaginer qu'un moteur de recherche soit suffisamment malin pour repérer des motifs qui se répètent et donc générer des exceptions par lui-même. Ou plus simplement, changer de technique en fonction du taux de mots vides présents dans la recherche [15].

3.3.3 Orthographe et conjugaison

On attend d'un moteur de recherche qu'il soit insensible au genre des mots, à leur conjugaison ou à leur forme. Une méthode efficace mais certes brutale pour répondre à ce problème est le *stemming* ou *désuffixation*, qui consiste à convertir tous les mots vers leur radical associé. Il faut bien sûr veiller au traitement de mots analogues mais dont le sens diffère fortement : (tu) bois et (un) bois. On peut également veiller à supprimer toutes les notions d'accentuation. Ces algorithmes sont fortement dépendants de la langue étudiée : agglutinante ou flexionnelle, indo-européenne ou autre. On pourra citer l'algorithme de Levins [16], de Porter [17] ou de Paice [18] pour l'Anglais ou Carry [19] ou Flemm [20] pour le français.

Lorsqu'on écrit avec un clavier, on commet souvent des erreurs de frappe. Il peut alors être intéressant de proposer une correction orthographique à l'utilisateur pour obtenir de meilleurs résultats. La méthode la plus employée pour répondre à cette problématique est d'utiliser un dictionnaire et de comparer les mots de la requête avec celui-ci. Pour cela, on peut employer la distance de Levenshtein [21], ou une variante où la proximité des touches du clavier ou la phonétique interviennent.

3.3.4 Index inversé

L'idée de l'index inversé provient de deux constatations :

— Premièrement, calculer les *TF-IDF* de l'ensemble d'un corpus peut prendre du temps et la matrice résultant doit être complètement recalculée à chaque fois qu'un nouveau document est ajouté à cet ensemble.

— Deuxièmement, les requêtes contiennent un tout petit ensemble de mots en comparaison à celui du corpus. Calculer la similarité cosinus avec tous les documents prendrait trop de temps et ne servirait à pas grand chose puisqu'il est probable que la majorité ne possède même pas un des termes de la demande.

Pour palier à ces problèmes, on emploie un index inversé afin d'offrir à la fois une rapidité d'indexation pour les nouveaux documents mais également une recherche performante. L'idée consiste à ne s'intéresser qu'aux mots et non aux documents, d'où l'origine du mot "inversé", tout en offrant une représentation du contenu exploitable. Sa structure est relativement simple, nous avons d'une part, une liste contenant tous les documents et le moyen d'y accéder ainsi que la taille de celui-ci et d'autre part, un index où à chaque mot du dictionnaire est associé la liste des paires des documents et le nombre d'occurrences où le terme apparaît. L'avantage d'une telle représentation est que le calcul du *IDF*, au logarithme près, consiste simplement à faire le quotient entre le nombre de documents et le nombre de fois où le terme apparaît, soit la taille de la liste associée au mot. Le *TF* est également très simple puisque nous possédons la taille de chaque document et que le nombre de fois où le terme apparaît dans le corpus est repris dans la liste associée au mot. Enfin, vous conviendrez que l'ajout de documents est aisé [22]. Attention, lorsque cette structure de données devient vraiment trop importante, des problèmes peuvent survenir [23].

3.4 Décomposition en valeurs singulières

Nous avons vu dans le chapitre précédent différents concepts clefs permettant d'effectuer des recherches de manière relativement efficace. Seulement, même si ces techniques ont beaucoup de qualité, elles sont peu adaptées à la résolution d'un problème qui lui est analogue. En effet, nous avons construit, avec notre modèle, une représentation spatiale des documents où leur position est définie par le contenu et il existe donc des sous-régions de cet espace ayant une certaine sémantique. Seulement, comment peut-on faire pour trouver des associations entre les mots ? Comment trouver un document contenant le mot "bateau" à l'aide du mot-clef "navire" ?

Nous verrons dans ce chapitre, une méthode afin de parvenir à la recherche de mots fortement corrélés dans une langue. Il existe d'autres procédures comme la *Principal Component Analysis* [24] ou la *Non-Negative Matrix Factorization* [25], mais celles-ci ne seront pas abordées. Nous ne présenterons que la décomposition en valeurs singulières avant d'attaquer l'analyse sémantique latente.

3.4.1 Valeurs propres

Dans le terme décomposition en valeurs singulières, nous entendons *valeurs singulières*. Un petit rappel sur la nature de ces valeurs et leur lien avec les valeurs propres peut être nécessaire.

Le concept de vecteur propre est une notion algébrique s'appliquant à une application linéaire d'un espace dans lui-même (*endomorphisme*). On représente généralement les transformations linéaires de l'espace par des matrices qui sont des applications linéaires. En effet, celles-ci transforment les vecteurs en conservant les propriétés d'addition et de colinéarité. Cette propriété implique que, si on a un vecteur w, somme de deux vecteurs u et v, alors l'image de w sera également la somme des images des vecteurs u et v.

- On appelle vecteur propre, un vecteur qui conserve sa direction (à ne pas confondre avec le sens) après transformation. (1)
- Une valeur propre est associée à un vecteur et représente le facteur d'homothétie de cette transformation. (2)
- Un espace propre est le sous-espace vectoriel qui contient l'ensemble des vecteurs propres.

Mathématiquement : Soit E un espace vectoriel sur un groupe commutatif $\mathbb{K}$ et M un endomorphisme de E, alors :

- Le vecteur x de E non nul est dit vecteur propre de u si et seulement si il existe un élément λ de $\mathbb{K}$, tel que $M(x) = \lambda x$. (1)
- Le scalaire λ de $\mathbb{K}$ est dit valeur propre de u si et seulement si il existe un vecteur x non nul de E tel que $M(x) = \lambda x$. (2)

Attention, les valeurs propres et vecteurs propres ne sont pas toujours réels et peuvent avoir des multiplicités supérieures à 1.

3.4.2 Valeurs singulières

Cette présentation brève de ce vaste champ des mathématiques ayant été effectuée, nous allons présenter ce qui s'appelle les valeurs singulières de manière très synthétique :

Soit M, une matrice $m \times n$ dont les coefficients appartiennent à un corps $\mathbb{K}$: On appelle valeur singulière de M toute racine carrée d'une valeur propre de M^*M,

autrement dit tout réel positif σ tel qu'il existe un vecteur unitaire u dans $\mathbb{K}^m$ et un vecteur unitaire v dans $\mathbb{K}^n$ vérifiant :

$$Mv = \sigma u \quad \text{et} \quad M^*u = \sigma v$$

On dit que u est un vecteur singulier gauche et v, droit pour σ.

3.4.3 Décomposition

La décomposition en valeurs singulières est très ancienne, on l'associe généralement aux travaux de Beltrami et Jordan, qui ont été généralisés par Eckart et Young avec le théorème d'Autonne-Eckart-Young [26]. Ce théorème dit qu'on peut toujours décomposer une matrice en ceci :

$$M = U\Sigma V^*$$

avec :
— M, une matrice $m \times n$ avec au plus $min(m, n)$ valeurs singulières distinctes.
— U, une matrice orthogonale de $\mathbb{K}^m$ avec les vecteurs gauches de M.
— Σ, une matrice diagonale qui reprend les valeurs singulières de M. Habituellement, ces valeurs sont ordonnées dans l'ordre décroissant.
— V^*, une matrice orthogonale de $\mathbb{K}^n$ avec les vecteurs droits de M.

3.4.4 Réduction de la dimensionnalité

Nous l'avons dit, des problèmes peuvent apparaître avec un nombre de dimensions élevé. Il peut alors être intéressant de réduire cette dimensionnalité. On note Σ_k la matrice Σ où on n'a conservé que les k premières valeurs singulières.

$$\Sigma_k = \left(\begin{array}{cccc|ccc} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & & \vdots \\ 0 & \cdots & 0 & \sigma_k & 0 & \cdots & 0 \\ \hline 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & \vdots & \vdots & & \vdots \\ 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 \end{array}\right)$$

Grâce au théorème de Eckart-Young [27], $M_k = U\Sigma_k V^T$ est la meilleure approximation de rang k de M selon la norme de Frobenius ou de Hilbert-Schmidt [28].

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} = \sqrt{\text{trace}(A^*A)} = \sqrt{\sum_{i=1}^{\min\{m,n\}}\sigma_i^2}$$

On supprime généralement les termes ayant peu d'influences. En règle général, on garde toutes les valeurs représentant 90% de la trace de la matrice ou on ne garde qu'une petite centaine de valeurs singulières.

FIGURE 3.2 – Approximations successives d'une image, avec 1, 2, 4, 8, 16, 32, 64, 128 puis toutes les valeurs singulières (image originale à gauche) - tiré de Wikimedia

3.4.5 Interprétation géométrique

Certains ont une vue plutôt algébrique et d'autres géométrique. Il est peut être avantageux de visualiser ce que représente ces matrices U, Σ et V^*. Pour notre bien-être mental, nous allons nous cantonner à deux dimensions par la suite.

Si nous considérons une matrice M carrée et dont le déterminant est positif, U et V^* représentent une rotation et Σ une homothétie pour chacun des axes.

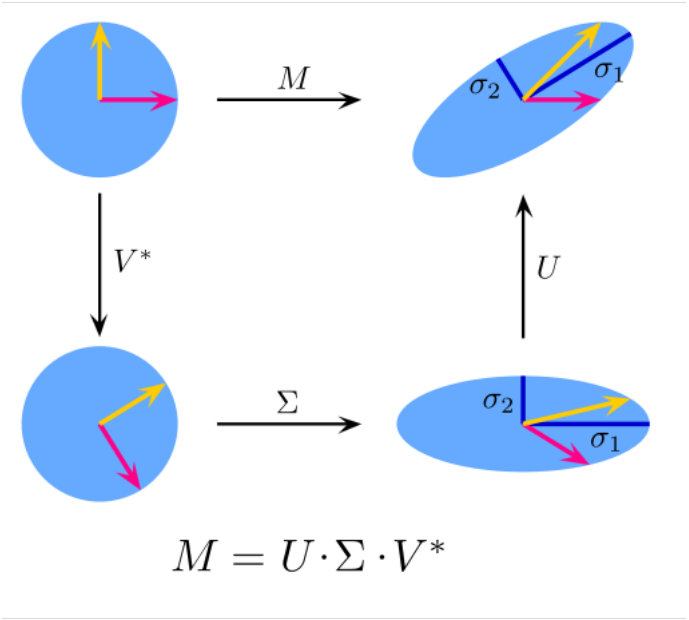

FIGURE 3.3 – Transformation du cercle unité par la matrice (1 1, 0 1) - tiré de Wikimedia par Georg-Johann

La décomposition consiste à effectuer une rotation de l'espace à l'aide de la matrice U afin de construire un nouvel espace orthogonal tel que la distance quadratique moyenne soit minimale par rapport à l'ensemble des points [29].

Et V, contient les nouvelles coordonnées dans la nouvelle base U. Conserver les premières valeurs singulières de Σ permet de garder les axes qui contiennent la variance la plus forte, et donc le plus d'informations. À noter que pour obtenir la matrice nécessaire à la technique du *PCA*, il suffirait de centrer la matrice au barycentre des données (ou centre de gravité pour les belges).

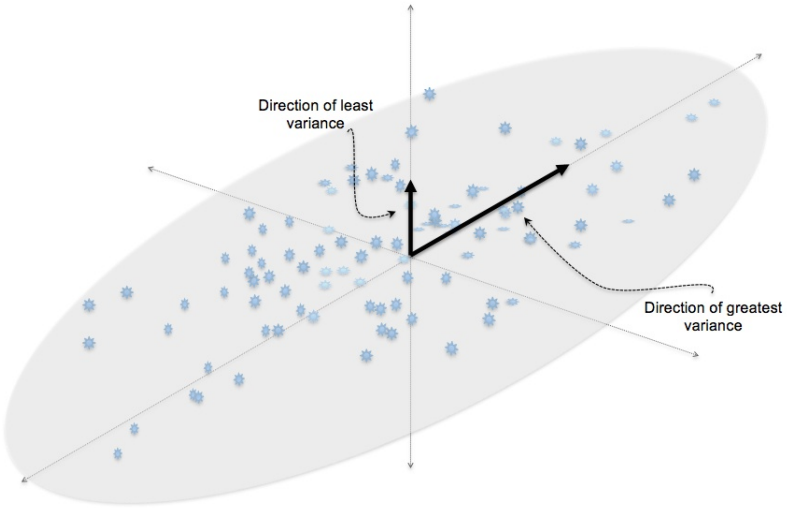

FIGURE 3.4 – Moindres carrés et orthogonalité - tiré de Mike Tamir

3.5 Analyse sémantique latente

La décomposition en valeurs singulières intervient dans plusieurs problèmes comme le calcul de matrice pseudo-inverse, la résolution de systèmes linéaires homogènes, les moindres-carrés, les matrices de rang inférieur, les modèles séparables ou encore dans le traitement du signal. Je vous invite à consulter la page Wikipédia pour obtenir encore plus d'informations sur ce sujet passionnant !

Mais cette décomposition intervient également pour un sujet qui va nous intéresser tout particulièrement, celui de l'analyse sémantique latente ou *Latent Semantic Indexing/Analysis*. Qui consiste à identifier les liens qui unissent les termes aux concepts dans un ensemble non-structuré de textes. Le principe fondateur réside dans le fait que les mots employés dans un même contexte ont généralement un sens analogue. Tout aurait commencé grâce aux travaux de Jean-Paul Benzécri [30] qui ont été améliorés par la société *Bell Core* [31]. Dans cette partie, nous ne présenterons pas la technique dite du *Probabilistic Latent Semantic Analysis* qui propose une approche purement statistique [32].

3.5.1 Espace des concepts

La matrice utilisée pour réaliser la décomposition en valeurs singulières est simplement une matrice qui contient l'occurrence du terme t dans le document d. On peut également utiliser le poids du *TF-IDF* pour chacun des termes. C'est généralement une matrice fort creuse, la majorité des termes étant égaux à zéro.

Lorsqu'on applique la décomposition en valeurs singulières, on crée une nouvelle base à partir des anciennes dimensions (les mots de notre corpus). Cette nouvelle base est obtenue par combinaison linéaire et on appelle communément la matrice U, la matrice des *concepts*. La matrice Σ reprend les taux de variations de ces concepts, les k plus grandes valeurs permettant ainsi de "capturer" les dimensions qui possèdent le plus d'information et d'éliminer le bruit. V aura peu d'importance pour la suite.

3.5.2 Similarité des termes et des documents

Il est possible de définir une matrice $U\Sigma$ (et $V\Sigma$) comme le produit matriciel de U (resp. V) et Σ, nous les noterons par la suite T et D. Ce choix est cohérent puisqu'il est facilement démontrable que M^*M peut se simplifier avec une décomposition spectrale parce que une matrice orthogonale et sa transposée donne l'identité. Intuitivement, cela consiste à exagérer les concepts ayant beaucoup de poids afin de les rendre encore plus visibles.

Enfin il est intéressant de regarder le produit de T^TT (et de D^TD), qui représentent les liens entre les différentes termes (et documents) deux à deux [33].

3.5.3 Requêtes dans l'espace

Le calcul de la décomposition en valeurs singulières est une opération coûteuse, il peut alors être nécessaire de ne pas travailler avec l'ensemble de la base de données mais avec un sous-ensemble. L'autre partie étant alors convertie dans l'espace des concepts. Cette opération est analogue à la traduction d'une requête dans cet espace.

$$\begin{aligned} & Q = U\Sigma V_q^T \\ \Leftrightarrow \quad & V_q^T = \Sigma^{-1}U^TQ \\ \Leftrightarrow \quad & V_q\Sigma = Q^TU\Sigma^{-1}\Sigma \end{aligned}$$

Avec Q, le vecteur de requête et V_q, la matrice orthogonale pour la requête. La projection de Q sur l'espace réduit à k dimensions s'obtient en calculant $V_q\Sigma_k$, ce qui revient à calculer les k premières colonnes de Q^TU.

Bibliographie

[1] C. Lévi-Strauss, "Les structures élémentaires de la parenté," 1949. pages 19

[2] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, pp. 613–620, Nov. 1975. pages 19

[3] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986. pages 19

[4] G. Salton, "Automatic text processing : The transformation, analysis, and retrieval of," *Reading : Addison-Wesley*, 1989. pages 19

[5] P.-N. Tan, M. Steinbach, V. Kumar, *et al.*, *Introduction to data mining*, vol. 1. Pearson Addison Wesley Boston, 2006. pages 20

[6] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. MIT press, 2001. pages 21

[7] R. Bellman, "Dynamic programming princeton university press," *Princeton, NJ*, 1957. pages 21

[8] G. Qian, S. Sural, Y. Gu, and S. Pramanik, "Similarity between euclidean and cosine angle distance for nearest neighbor queries," in *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 1232–1237, ACM, 2004. pages 21

[9] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pp. 58–64, 2000. pages 21

[10] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988. pages 21

[11] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM J. Res. Dev.*, vol. 1, pp. 309–317, Oct. 1957. pages :1957 :SAM :1661832.1661836

[12] G. K. Zipf, "Selected studies of the principle of relative frequency in language.," 1932. pages 22

[13] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972. pages 22

[14] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*, vol. 999. MIT Press, 1999. pages 22

[15] S. Tong, U. Lerner, A. Singhal, P. Haahr, and S. Baker, "Locating meaningful stopwords or stop-phrases in keyword-based retrieval systems," May 17 2011. US Patent 7,945,579. pages 23

[16] J. B. Lovins, *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory Cambridge, 1968. pages 23

[17] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980. pages :10.1108/eb046814

[18] C. D. Paice, "Method for evaluation of stemming algorithms based on error counting," *J. Am. Soc. Inf. Sci.*, vol. 47, pp. 632–649, Aug. 1996. pages :1996 :MES :230789.230796

[19] M. Paternostre, P. Francq, J. LAMORAL, D. Wartel, and M. Saerens, "Carry, un algorithme de désuffixation pour le français," *Rapport technique du projet Galilei*, 2002. pages 23

[20] F. Namer, "Flemm : un analyseur flexionnel du français à base de règles," *Traitement automatique des langues*, vol. 41, no. 2, pp. 523–547, 2000. pages 23

[21] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966. pages 23

[22] J. Zobel and A. Moffat, "Inverted files for text search engines," *ACM computing surveys (CSUR)*, vol. 38, no. 2, p. 6, 2006. pages 23

[23] S. Brin and L. Page, "Reprint of : The anatomy of a large-scale hypertextual web search engine," *Computer networks*, vol. 56, no. 18, pp. 3825–3833, 2012. pages 23

[24] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901. pages 24

[25] P. Paatero and U. Tapper, "Positive matrix factorization : A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994. pages 24

[26] C. Eckart and G. Young, "A principal axis transformation for non-hermitian matrices," *Bulletin of the American Mathematical Society*, vol. 45, no. 2, pp. 118–121, 1939. pages 25

[27] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936. pages 25

[28] R. Horn and C. Johnson, "Norms for vectors and matrices," *Matrix Analysis*, pp. 257–342, 1990. pages 25

[29] C. L. Lawson and R. J. Hanson, *Solving least squares problems*, vol. 161. SIAM, 1974. pages 26

[30] J.-P. Benzécri and L. Bellier, *L'analyse des données*, vol. 1. Dunod Paris, 1976. pages 27

[31] S. Deerwester, S. Dumais, G. Furnas, R. Harshman, T. Landauer, K. Lochbaum, and L. Streeter, "Computer information retrieval using latent semantic structure," June 13 1989. US Patent 4,839,853. pages 27

[32] T. Hofmann, "Probabilistic latent semantic indexing," in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 50–57, ACM, 1999. pages 27

[33] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998. pages 28

Chapitre 4

Résultats et interprétation

4.1 Introduction

Dans cette partie, nous allons nous intéresser aux résultats obtenus avec la technique que nous avons employée ainsi qu'aux méthodes d'expérimentation afin de s'assurer de la validité du système.

Il existe trois grandes familles de tests :

— Hors ligne : Souvent la méthode la plus simple et la moins onéreuse, elle consiste, sur la base d'un ensemble de données, à créer un protocole qui permet d'estimer la pertinence des résultats ainsi que ses performances.

— Études sur une population : Cette technique consiste à utiliser le système de recommandations sur une (petite) population d'utilisateurs et de leur demander leur avis au travers de questionnaires sur leurs ressentis.

— En ligne : On peut mener des expériences à large échelle sur un public qui a conscience d'être analysé (Youtube).

Dans ce travail, nous n'aborderons qu'un seul de ces aspects, celui de l'expérimentation sur les données mêmes. Les autres aspects ayant davantage attrait à la sociologie.

4.2 Protocole de tests

4.2.1 Mais qu'est-ce qu'une expérimentation ?

" Méthode scientifique exigeant l'emploi systématique de l'expérience afin de vérifier les hypothèses avancées et d'acquérir des connaissances positives dans les sciences expérimentales. " [1]

Dans cette définition, nous décelons plusieurs éléments clefs. Nous avons tout d'abord la notion d'hypothèse, qui doit être précise mais concise et nous devons fournir une expérience capable de tester cette proposition. Nous avons également la notion de répétabilité ainsi que celle de conclusion. Il existe d'autres concepts qui lui sont liés, le fait d'avoir des variables que l'on peut *mesurer*. Sous une hypothèse, il est important que les variables non testées restent constantes afin de ne pas compromettre l'expérience. Exemple : Comparer l'algorithme A et B mais sur des ensembles différents. Finalement, nous devons faire très attention aux conclusions que nous pouvons en tirer. Généraliser des résultats ne se fait jamais à la légère, il faut vérifier qu'ils soient cohérents ou sur différentes données vérifiant les mêmes hypothèses.

4.2.2 Expérimentation hors ligne

Dans les expérimentations hors ligne, nous ne travaillons que sur des données, celles-ci peuvent être brutes et utilisées par nos algorithmes ou bien déjà travaillées avec des profils préexistants. Nous allons essayer de simuler le comportement de l'utilisateur ou les résultats qui peuvent lui être proposés.

Le grand avantage de cette technique est quelle ne requière pas d'interactions avec les utilisateurs et permet donc de tester beaucoup d'algorithmes à moindres coûts. Malheureusement, on ne pourra jamais mesurer l'interaction qu'à ce système sur les utilisateurs et ne permettra que de répondre à des questions purement quantifiables. Cette technique d'expérimentation n'est pratiquement utilisée que pour comparer les résultats des algorithmes. Il est donc préférable que l'ensemble des données de test corresponde au problème réel.

4.2.3 Outils statistiques

Afin de départager les différentes techniques mises en œuvre, on cherche à déterminer des outils qui nous permettent de quantifier la précision des recommandations proposées. Il faut garder en tête que la qualité n'est pas signe de pertinence.

Les deux outils les plus souvent utilisés pour mesurer les prédictions sont la racine de l'erreur moyenne *Root-Mean-Square Deviation/Error* et l'erreur absolue moyenne *Mean Absolute Deviation/Error*. Ici $\hat{r}_{ui}$ est la prédiction pour l'ensemble de test T avec les pairs utilisateur-objet (u, i) et la solution supposée r_{ui}.

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} (\hat{r}_{ui} - r_{ui})^2} \qquad MAE = \sqrt{\frac{1}{|T|} \sum_{(u,i) \in T} |\hat{r}_{ui} - r_{ui}|}$$

Les deux outils possèdent leurs qualités et inconvénients, la différence fondamentale est la dépendance dans la grandeur de l'erreur. Cette sémantique pouvant avoir plus ou moins d'impact en fonction du problème étudié. On emploie également des tests croisés (χ^2) afin de déterminer la prévision d'utilisation lorsqu'on n'emploie pas un critère mesurable telle que des "étoiles". Ces outils sont préférés lorsque le système de recommandation travaille davantage sur les propositions. On ne s'intéresse donc pas à l'appréciation mais à la capacité de l'objet à être ajouté à la liste de l'utilisateur. L'idée est qu'on possède une liste d'objets choisis par les utilisateurs et on va occulter certains de ces éléments afin de voir si le système est capable de retrouver quels étaient ces objets. Enfin, il suffit de comparer le taux d'erreur réelle et les faux positifs.

Pour cela, il existe différents coefficients qualifiés de "corrélations" :

$$\text{Matthews Correlation Coefficient} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$\text{Pearson product-moment correlation coefficient} = \rho_{X,Y} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - E[X]^2}\sqrt{E[Y^2] - E[Y]^2}}$$

Les autres aspects des systèmes de recommandations sont nettement plus difficiles à tester et nécessite un retour humain. Il existe également des problèmes pratiquement impossibles à résoudre, on peut citer : le problème du démarrage à froid, la confiance

dans le système et les actions "normales" des utilisateurs, la nouveauté et la sérendipité, le risque, l'utilité, l'intimité du résultat, ...

4.3 Nos tests

Dans notre cas, nous avons simplement tiré des recettes aléatoirement dans notre base de données (5% de l'ensemble, soit ± 500 recettes). Pour chacune d'entre elles, nous avons considéré chacun des ingrédients et pris le résultat de notre système de recommandations pour chacun d'eux. Nous avons alors regardé quels étaient ceux présents dans la recette et pondéré le résultat par la position occupée dans la liste des recommandations (son poids est inversément proportionnel à sa position dans la liste des recommandations). Nous avons également testé les relations du second ordre, c'est-à-dire, pour chacune des recommandations qui n'a pas de correspondance dans la recette, nous avons regardé quels sont les ingrédients fortement corrélés ($> 90\%$) et comparé avec la vraie recette pour traiter les cas des synonymes (blanc de poulet, poitrine de poulet, poulet, ...).

4.4 Résultats

Pour le système de recommandations même, nous avons comparé l'usage de différentes pondérations du *TFIDF* sur la matrice des occurrences (en ligne), ainsi que le système de normalisation des lignes dans la matrice $U\Sigma$ (en colonne) et finalement l'importance du nombre de valeurs singulières à garder.

	Row Centering	Row Scaling	Z Score	Z Score Mean
Best Fully Weighted	12.6	13.8	19.8	19.9
Log Smooth	8.2	8.2	8.3	8.5
Nothing	12.3	12.5	19.7	19.8

Ces résultats du premier ordre ont été obtenus pour le meilleur pourcentage de valeurs singulières à conserver, soit 90%. On remarquera que les pourcentages sont relativement faibles même si corrects. Il faut avoir en tête que notre base de données connaît beaucoup de synonymes ($\pm 17\%$). Notons également que l'utilisation d'une pondération de type *TFIDF* a peu de sens dans notre cas. En effet, les ingrédients apparaissent généralement qu'une seule fois dans une recette et donc seul le coefficient qui agit sur l'ensemble des données a de l'importance ici.

Enfin, présentons les résultats pour les résultats du second ordre :

	Row Centering	Row Scaling	Z Score	Z Score Mean
Best Fully Weighted	51.3	53.9	57.6	58.9
Log Smooth	48.9	50.1	49.9	50.5
Nothing	51.3	53.1	57.7	57.2

Les résultats sont nettement meilleurs même s'ils sont loin d'être mirobolants. Peut-être utiliser une autre technique de la décomposition en valeurs singulières pourrait être intéressante ($SVD{+}{+}$). L'utilisation de la méthode du *Best Fully Weighted* et du *Z*

Score Mean semble conduire aux meilleurs résultats. Bien sûr, tous ces résultats sont à considérer avec des pincettes et nécessiteraient une étude plus approfondie pour voir s'ils correspondent aux réels besoins des utilisateurs. Finallement, nous signalerons qu'il aurait été préférable d'utiliser des outils plus adaptés à cette tâche telle que Scikit-learn [2].

Bibliographie

[1] A. française, "Dictionnaire de l'académie française (9e édition)," 1990. pages 31

[2] "scikit-learn : Machine learning in python." http ://scikit-learn.org/stable/. Accessed : 2016-02-29. pages 34

Chapitre 5

Conclusions et perspectives

5.1 Conclusions et perspectives

Vous l'aurez remarqué, les systèmes de recommandations représentent un vaste pan de l'informatique. Nous avons abordé dans ce travail que deux aspects : celui des recommandations dites basées sur le milieu (*Knowledge-based*) et l'analyse sémantique latente. Nous aurions tant aimé aborder d'autres sujets mais nous sommes limités tant par l'espace que par le temps.

Dans ce projet, nous avons vu qu'il était possible d'implémenter un système de recommandations, avec des concepts certes complexes ou plutôt tardifs dans l'histoire des Sciences, en peu de lignes de codes. Cependant, il ne s'agit que d'une simple ébauche, d'un premier aperçu de toute la complexité de cette problématique. Une première approximation est relativement triviale à implémenter mais l'amélioration des résultats ne l'est certainement pas. Nous comprenons mieux que des entreprises telles que Google ou Amazon prennent des années à se développer pour arriver à de tels résultats et devenir des géants tous puissants de l'économie de marché.

Nous espérons que la lecture de ce sujet vous aura passionné autant que nous. Nous vous invitons à vous renseigner d'avantage dans ces théories qui sont porteuses de beaucoup de découvertes et qui ont attrait à différents aspects de la Science en général. La recherche dans ce domaine vous tend les bras et ne demande que votre aide. Des concours sont organisés de manière plus casuelle afin que les gens puissent effleurer du bout du doigt ce champ de l'informatique et de véritables challenges sont créés par des entreprises afin d'améliorer leur propre système avec des lots intéressants à la clef.