Rapport de projet de C/SD

par

Nathan BARLOY, Valentin CRÔNE,
Johan TOMBRE

1 Avenue Paul Muller
54600 Villers-lès-Nancy
France

Projet de C/SD

Table des matières

Chapitre 1

État de l'art

1.1 Introduction

1.1.1 Qu'est ce qu'un système de recommandation ?

Un système de recommandation a pour but de fournir des recommandation de résultats pertinentes à un utilisateur, en fonction de différents paramètres comme ses préférences, son historique, le temps passé sur le contenu, etc.., afin de lui proposer directement un contenu ciblé sans efforts de sa part.

Un algorithme de recommandation peut de ce fait être utilisé pour faciliter les recherches de contenus sur un thème donné, ce qui est bénéfique pour l'utilisateur car il aura plus d'informations pertinentes à regarder, et permettre aux gérants de la plateforme de fidéliser la clientèle.

Prenons un exemple, un site de e-commerce sur lequel on trouve facile des articles en rapport avec l'article que l'on est en train de consulter rendra plus facile la recherche, et donc aidera le client a trouver rapidement le produit adéquat, ce qui réduira le temps nécessaire à l'achat, et évitera la perte du client. Un autre exemple est une plateforme de visionnage de vidéos en ligne, qui génère des bénéfices grâce a la publicité, un tel système proposera du contenu ciblé intéressant pour l'utilisateur, qui consultera un plus grand nombre de vidéos sur le site, ce qui permet à la fois de satisfaire l'utilisateur et améliorer la génération de revenus liés à la publicité. Ces effets sont liés au fait que l'utilisateur reçoit des suggestions pertinentes de la part du système de recommandation auxquelles il n'aurait pas forcément pu prêter attention autrement.

1.1.2 La place des systèmes de recommandation dans la recherche d'information

La recherche d'information sur internet est fondée sur un principe d'indexation des données. Afin de pouvoir répondre aux requêtes des utilisateurs dans un temps raisonnable, on indexe les données dans une base de données, et on les filtre en fonction de la requête des utilisateurs. Les requêtes sont entrées à partir de mots clés (requêtes ad hoc), et le moteur de recherche va alors retourner les résultats qui correspondent. Le nombre conséquent de résultats obligera le moteur de recherche a limiter la quantité d'informations retournée en limitant un certain nombre de résultats par page. Ce système fonctionne plutôt bien, mais nécessite que l'utilisateur traite lui même un grand nombre de résultats, ce qui peut se limiter a la ou les premières pages car il lui est impossible de tout traiter. On cherche donc à rendre les résultats de la première page plus pertinents. On fait alors immédiatement le lien avec les systèmes de recommandation, car au lieu de filtrer par mots clés, on va pouvoir rechercher des contenus à thème proche, date proche, et personnaliser les résultats en fonction des habitudes de l'utilisateur.

Ces exemples montrent un certain nombre de cas où l'utilisation d'un système de recommandation s'avère utile. Cependant il en existe de nombreux autres, car leur champ d'application est immense, c'est pourquoi il est intéressant de s'intéresser aux systèmes existants pour tenter de les reproduire et les améliorer.

1.1.3 Classer les différents systèmes de recommandation

Il existe différents types de systèmes de recommandation, en fonction du type de données à classer, de la puissance disponible, de activités utilisateurs, des besoins de l'entreprise, et de nombreux autres facteurs. Les facteurs principaux à retenir sont :

- La connaissance des préférences utilisateur.
- La similarité entre les utilisateurs, les uns par rapport aux autres. (Métrique)
- La connaissance d'informations sur des données à recommander
- La connaissance d'informations de regroupement sur des données à recommander

A partir de ces facteurs on produit différents types de recommandations. Les systèmes les plus utilisés sont le filtrage basé sur le contenu, et le filtrage collaboratif.

Le filtrage basé sur le contenu

Le filtrage basé sur le contenu va essayer d'associer un utilisateur à un ensemble de données proche de ses préférences. On va alors rechercher des données qui pourraient l'intéresser en tenant compte uniquement de lui-même, et de ce qu'il a consulté. Pour cela, il faut dresser des liens entres les différents éléments de la base de donnée, pour pouvoir déterminé si un élément est proche ou non de ce qu'a consulté l'utilisateur. L'avantage de ce filtrage est qu'il ne nécessite pas d'avoir d'autres utilisateurs pour fonctionner, puisqu'on ne compare pas les utilisateurs entre eux. Cela est donc très pratique quand la base de donnée des utilisateur est petite.

Le filtrage collaboratif

Le filtrage collaboratif va mesurer une certaine distance entre un utilisateur donné, et tous les autres utilisateurs pour sélectionner les utilisateurs les plus proches uniquement. On utilisera alors le profil de ces utilisateurs proches pour regarder les données qui les ont intéressés pour les recommander à notre utilisateur initial. L'avantage de ce filtrage est qu'il ne nécessite pas de créer un algorithme qui détermine si deux données sont proches ou non. Cependant, si la base de donnée des utilisateurs n'est pas assez fournie, ce filtrage ne sera que très peu efficace.

Évidemment, les filtrages les plus efficaces sont ceux qui mélangent les méthodes décrites précédemment, de manière à en tirer les avantages.

1.2 Les données

Pour pouvoir lier des produits avec les utilisateurs, il est indispensable de mettre au point un système de notation afin de déterminer si un utilisateur apprécie ou non un produit. On distingue deux approches différentes :

— l'approche "réactive", où le système de recommandation demande à l'utilisateur des informations / avis sur certains produits afin d'affiner ces recommandations. Cette approche présente cependant l'inconvénient de solliciter l'utilisateur.
— l'approche "proactive", qui anticipe plus les goûts de l'utilisateur.

1.2.1 L'approche réactive

Cette approche consiste à déterminer les goûts de l'utilisateur à travers un processus conversationnel. Le système demande régulièrement des informations à l'utilisateur sur ses préférences sur ses goûts... puis va proposer des recommandations que l'utilisateur va accepter ou alors critiquer les résultats afin d'affiner le système. L'avantage de ce système de critique est qu'il est simple à appliquer et ne requiert pas de connaissance de l'utilisateur dans ce domaine. Toutefois, l'inconvénient majeur est qu'il demande directement à l'utilisateur de faire un effort pour exprimer son avis et donner un retour.

1.2.2 L'approche proactive

L'approche proactive ce base davantage sur la déduction des appréciations pour fournir ensuite une recommandation. Ainsi l'utilisateur n'a plus à guider le système avec des retours sur les recommandations proposées mais va observer les interactions de l'utilisateur pour déterminer les goûts de celui-ci. Ces observations peuvent être directes ou indirectes.

Observation directe

Ce sont toutes les informations que l'utilisateur donne de manière explicite en notant, postant un commentaire sur un produit par exemple ou encore les informations du profil de l'utilisateur comme son âge, son sexe, sa localité géographique... La notation des produits est un élément centrale dans un système de recommandations. Ces notes sont principalement numériques peuvent avoir différentes formes qui délivrent plus ou moins d'informations.

Il y tout d'abord la notation unaire : cette notation ne délivre qu'une seule information, si l'utilisateur a aimé un produit. On peut donner par l'exemple de réseau social Instagram qui donne la possibilité aux utilisateurs de "liker" une photo uniquement, il est impossible de mettre un avis négatif.

Il y a ensuite la notation binaire qui prend en compte l'avis négatif de l'utilisateur. L'information transmise est donc binaire : "J'aime" / "Je n'aime pas". C'est par exemple ce qu'utilise Facebook ou Youtube. L'avantage de cette notation est facilite le choix de l'utilisateur.

Enfin, la notation ordonnée est celle qu'on retrouve dans la majorité des site de e-commerce (Amazon, C Discount et autres). L'utilisateur donne une note à un produit suivant une échelle de notation comprise entre deux valeurs (entre 1 et 5 par exemple). Plus ce chiffre est élevé, plus l'utilisateur a aimé le

produit. Cette notation permet plus de nuances sur l'appréciation des produits et donc améliore les recommandations qui en découle. Cependant, le coût en calcul en est impact car l'information n'est plus binaire. De plus cette notation est sensible à la manière de noter de chaque utilisateur : une note de 3/5 sera considéré comme une mauvaise note par certains utilisateurs alors que pour d'autres, ce sera un note neutre / moyenne. Cela peut donc impacter la qualité des recommandations.

Partie sur l'analyse des commentaires pour en extraire des tags + utilisations des données profil utilisateur Une autre ressource importante dans les systèmes de recommandations est l'analyse des commentaires, descriptions, noms des produits... afin de produire des tags spécifiques à l'utilisateur. Diverses méthodes existent pour cela et nous aborderons plus en détail la méthode TF-IDF plus loin dans le rapport. Grâce aux tags produits par le système, il est possible de faire des rapprochements entre utilisateurs et/ou produits.

Observation indirecte

Ce sont l'ensemble des informations implicites données par l'utilisateur ou déduites par le système. Il peut s'agir par exemple d'actions réalisées sur une page web comme une recherche, un ajout dans un panier d'achat ou tout simplement de la fréquence de consultation d'une page. Il est ensuite possible à partir de ces informations d'estimer les préférences de l'utilisateur. Par exemple, si on utilisateur ajoute un produit dans son panier ou tout simplement le consulte fréquemment, on peut conclure que ce type de produit l'intéresse puis intégrer cette information à notre système.

1.2.3 Conclusion sur les données

Nous avons vu différentes manières de collecter des informations relative à l'utilisateur dans le but d'alimenter notre algorithme de recommandation. Que ce soit par une approche réactive ou proactive, ces techniques peuvent soulever un débat sur la collecte des données. Il faut donc mener une réflexion pour trouver un bon compromis entre efficacité du système et collecte de données "abusive".

1.3 La matrice d'usage

Également appelé le modèle utilisateur, la matrice d'usage contient des données collectées sur les utilisateurs associés aux produits. On le représente généralement sous la forme d'une matrice avec les utilisateurs sur les lignes et les produits en colonne. Les scores attribués peuvent être une note, un nombre de consultation, un durée, un like... Cette matrice comporte généralement de nombreuses valeurs manquantes qu'il va falloir chercher à déterminer grâce à notre système de recommandation. Il est également intéressant de remarquer

que les goûts des utilisateurs évoluent au cours du temps. Il est donc important de réajuster les données de cette matrice régulièrement.

1.4 Réduction de matrice

Reprenons la matrice d'usage tout juste présentée. Elle est de dimension $m.n$ avec m le nombre d'utilisateurs et n le nombre de produits. Dans la pratique, cela représente habituellement des milliers / millions d'utilisateurs pour autant voire plus de produits. On obtient donc une matrice possédant des milliards de valeurs. Il serait donc très long d'effectuer des calculs dessus pour en déduire des recommandations. La solution est donc de réduire la dimension de la matrice.

Plusieurs approches sont possibles. On peut, par exemple, ne pas prendre en compte les utilisateurs dont on ne possède que très peu d'informations ou de la même manière retirer les produits peu populaires et donc ayant moins de chance d'intéresser les utilisateurs. Cependant, cette méthode a comme gros inconvénient de réduire le champ des recommandations proposées. En effet, comment proposer à un utilisateur un produit qui lui conviendrait parfaitement s'il est retiré de la liste car peu connu ? Cela restreint plus ou moins les produits recommandables à ceux qui sont populaires et donc déjà connus par la plupart.

Une autre approche consiste à ne conserver qu'une gamme de produits répondant à un critère spécifié. Par exemple, il est possible de supprimer les films de genre romantique si l'utilisateur ne manifeste aucun intérêt pour ce genre. Enfin, une dernière méthode consiste à ne prendre en compte que les utilisateurs jugés comme étant proche de l'utilisateur cible, c'est ce qu'on appelle le clustering (détaillé dans la section suivante).

L'avantage de cette méthode est donc de réduire le temps de calcul pour déterminer une recommandation. Cependant, la qualité de la recommandation s'en retrouve impactée. En effet cela réduit le nombre de personnes analysés par lalgorithme dans le cas dune réduction du nombre dutilisateur, et une réduction du nombre de produits réduira le champs des recommandations à un certains type.

Clustering

Lidée est décarter les items (resp. les utilisateurs) jugés non représentatif pour un utilisateur (resp. item) donné. On réduit ainsi la dimension de la matrice tout en minimisant la perte de données avant deffectuer les calculs de recommandation. Pour ce faire, on utilise des méthodes de partitionnementă: cela consiste à regrouper les éléments similaires. La méthode la plus connue est k-means

1.5 Calcul de similarités

Un calcul de similarité est utile pour déterminer la distance entre un utilisateur et un produit mais également entre deux utilisateurs (ou produits). Plusieurs outils existent pour nous permettre d'effectuer ce calcul. Nous verrons dans cette partie quelques unes de ces méthodes et discuterons des avantages et inconvénients qu'ils présentent.

1.5.1 La distance Euclidienne

La distance euclidienne est obtenue par la formule suivante :

$$L(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \tag{1.1}$$

Paragraphe à terminer...

1.5.2 Distance cosinus

On considère les vecteurs ligne de x et y. Le but est de calculer la distance, langle entre ces deux vecteurs. Dans notre cas, les vecteurs x et y seront des vecteurs associés aux profils de deux utilisateurs par exemple.

$$CosSim(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{m} x_i y_i}{\sqrt{\sum_{i=1}^{m} x_i^2} \sqrt{\sum_{i=1}^{m} y_i^2}} \tag{1.2}$$

L'inconvénient majeur de cette méthode est quelle considère tout film non noté comme un zéro, qui est une mauvaise note, ce qui va réduire la distance entre deux utilisateurs qui en réalité seraient jugés différents.

1.5.3 Corrélation de Pearson

Afin de pallier le problème cité précédemment, il est possible de normaliser au préalable les notes en leur soustrayant la moyenne des notes données par l'utilisateur. Cette méthode est également appelé la distance cosinus centrée. On a alors :

$$Corr(x, y) = \frac{\sum_{i=1}^{m} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{m} (y_i - \bar{y})^2}} \tag{1.3}$$

Le résultat obtenu se situe alors dans l'intervalle $[\![-1; 1]\!]$ avec -1 signifiant que les deux utilisateurs ont des goûts complétement opposés, 0 que les utilisateurs non rien en commum et enfin 1 que les deux utilisateurs sont totalement similaires.

Chapitre 2

Rapport

2.1 Amélioration de la base de données

Dans le but d'améliorer l'expérience de l'utilisateur sur l'interface graphique, nous avons décider de compléter la base de données mise à disposition afin d'y inclure les liens vers les bandes annonces des films ainsi que leurs affiches respectives. De plus, le fonctionnement de l'algorithme de recommandation collaboratif requière une base de données d'utilisateurs ayant notés plusieurs films.

Nous avons décidé de réaliser ces opérations à partir de scripts python qui sont plus simple et rapide à mettre en oeuvre, grâce notamment à ses nombreuses librairies disponibles en ligne. Il est important de remarquer que ces scripts ont été exécutés en amont afin de récupérer les éléments souhaités mais ne font en rien partie du code du programme.

2.1.1 Liens des bandes annonces

Comme notre interface graphique sera gérée grâce à la librairie WebKit-WebView, il sera possible d'intégrer à notre page un lecteur youtube avec la bande annonce. Il suffit donc de récupérer les liens URL des vidéos pour que l'ensemble fonctionne. Ces résultats seront alors stockés dans un fichier texte avant d'inclure les liens au fichier JSON.

Les grandes étapes dans la réalisation de ce script ont étés :

— Générer la requête à partir du lien youtube (`www.youtube.com/results?search_query=`)
— Récupérer l'intégralité du code HTML générer l'exécution de cette requête
— Scanner le contenu HTML afin de ne conserver que le lien du premier résultat (on suppose que l'algorithme de recommandation youtube est efficace)
— Isoler la partie utile du lien récupérer (il s'agit des 11 derniers caractères après `https://www.youtube.com/watch?v=`)
— Stocker ce résultat dans la base de données

2.1.2 Récupération des affiches des films

Avoir une affiche de chaque film de la base de données permet une fois de plus d'améliorer grandement l'interface graphique. Après plusieurs recherches, nous avons décidé de récupérer les images du site [TheMovieDB].

Pour ce faire nous avons, de la même manière que pour les liens youtube utilisé les url pour effectuer nos recherches. Puis nous avons scanné le contenu de la page html résultante pour détecter le résultat correspond à notre recherche (correspondance des dates notamment). Une fois trouvé, nous naviguons vers la page du résultat et scannons à nouveau la page à la recherche de l'image souhaité avant de la télécharger dans un dossier spécifique et d'inclure le chemin d'accès dans notre base de données.

2.1.3 Création d'une base de données utilisateurs

Indispensable au fonctionnement de la méthode de recommandation collaborative, une base de données d'uilisateurs ayant attribués des notes aux films était nécessaire. Plusieurs pistes ont été explorées :

— Création fictive d'utilisateurs et de notes (aléatoires)
— Création d'un formulaire à partager avec des amis pour constituer notre propre base de données
— Trouver ces informations en ligne

Nous avons au final retenu cette dernière solution après avoir trouvé une base de données de [MovieLens] qui regroupe près de 100000 notes d'utilisateurs sur 10000 films. Le seul inconvénient est que cette base de données ne prenne pas en compte les séries. Par conséquent, les résultats exposés ne seront des plus justes mais permettra d'avoir un aperçu de cette algorithme.

La base de données de [MovieLens] est au format csv, il était donc très simple de récupérer les données souhaitées. Le seul problème était les titres des films qui étaient en anglais et ne correspondaient pas toujours à nos titres. Il a donc fallu prendre quelques minutes pour manuellement faire correspondre leurs titres avec nos id de films. Ensuite il n'est plus que question d'extraire les bons éléments du fichier csv et le tour est joué !

2.2 Création d'une distance entre les films

Chaque film est caractérisé par la donnée de différents attributs. Dans le cas présent, les attributs que nous considérerons sont :

— L'année de parution
— Les acteurs
— Le réalisateur
— Le genre
— Le type (film ou série)

Pour chacun de ces attributs, on va calculer une distance, dont la valeur sera comprise entre 0 et 1 (0 pour identique, et 1 pour complètement différent). On fait ensuite simplement une moyenne pondérée des distances obtenues. Les poids

de cette moyenne sont modulables, ce qui nous permet de les modifier comme on le souhaite, pour avoir des proportions qui correspondent à ce que nous voulons.
Cette distance doit vérifier certaines propriétés de la distance mathématique :

— $d(a, b) = d(b, a)$ (la symétrie)
— $d(a, b) \geq 0$ (la positivité)
— $d(a, a) = 0$

Si la distance créée ne vérifie pas ces propriétés, alors elle n'est pas bonne.

2.2.1 La distance de Jacard

En mathématique, la distance entre 2 ensembles est souvent calculée comme étant le minimum des distances entre les points de ces ensembles :

$$d(E, F) = \min_{\substack{x \in E \\ y \in F}} d(x, y)$$

Cependant, dans notre cas, cette distance n'est pas très utile, puisqu'elle ne prend pas en compte les autres éléments des ensembles. Ainsi, une telle distance ne ferrais pas de différence entre 2 films où tous les acteurs sont les mêmes, et 2 films qui ont un seul acteur en commun. Il faut donc trouver autre chose : la distance de Jacard.

Cette distance permet de mesurer la similarité entre 2 ensembles finis : soient X et Y deux ensembles finis. Alors,

$$dist\_Jacard = 1 - \frac{|X \cap Y|}{|X \cup Y|}$$

On a bien que si $X = Y$, alors la distance de Jacard est nulle, et que si les deux ensembles sont complètement distincts, alors la distance vaut 1.
Cependant, on peut se dispenser de certains calculs pour trouver la distance de Jacard. En effet, on peut utiliser le fait que $|X \cup Y| = |X| + |Y| - |X \cap Y|$. Il n'y a alors plus qu'a calculer $|X \cap Y|$ pour pouvoir ensuite calculer la distance de Jacard.

Cette distance de Jacard pourra être utilisée pour calculer les distance entre les attributs des films qui sont des ensembles. Dans notre cas, cela représente les acteurs, les réalisateurs et les genres.
Après quelques tests, on se rend compte que même si l'algorithme marche la plupart du temps, il y a certains cas ou il plante à l'exécution et indique une division par zéro. Après quelques recherches, on comprend que cela est du au fait que parfois, la liste des acteurs, des réalisateurs ou des genres est vide. Ainsi, la distance de Jacard ne peut pas être calculée, puisqu'on divise par $|X \cup Y|$, qui vaut zéro si X et Y sont vides.

Ce bug nous permet aussi de nous rendre compte de l'absurdité de certains calculs : en effet, comment interpréter la distance entre en ensemble vide est un autre ensemble qui ne l'est pas forcement ? (par exemple, si on sait uniquement que le réalisateur du film 1 est Spielberg, et rien pour le film 2) Dans ce cas, donner une distance n'a aucun sens. Il faut donc aussi prendre en compte cela avant de faire les calculs.

Après quelques tests, on se rend compte que l'on obtient parfois une distance négative, ce qui est impossible pour une distance normalement : il y a donc un problème dans le code. Il se trouve en fait qu'il y avait un problème au niveau de certains types, ce qui faussait la fonction equals, qui sert a comparer 2 éléments entre eux. Après correction, on obtient quelque chose de cohérent. Notamment, la distance entre un film et lui-même est toujours 0.

Cette distance de Jacard permet de donner une distance entre 2 ensembles : on peut donc l'utiliser pour tous les attributs qui sont des listes, i.e. les acteurs, les réalisateurs, et les genres.

2.2.2 La distance entre les années

Pour créer une distance entre 2 années, on va bien évidemment considérer la distance entre ces 2 entiers : cela signifie prendre la valeur absolue de la différence ($|a1 - a2|$). Notre distance sera donc une fonction f qui sera à valeur dans $\mathbb{R}_+$. On va chercher des conditions sur cette fonction.

Tout d'abord, on veut obtenir une distance entre 0 et 1, donc doit aller de $\mathbb{R}_+$ dans $[0;1]$. De plus, on veut que l'ordre soit gardé : si $a > b$, on veut que $f(a) > f(b)$. Cela signifie que la fonction est croissante. Enfin, on doit avoir que $f(0) = 0$ et $\lim\limits_{x \to +\infty} f(x) = 1$.

En résumé, on cherche une fonction croissante de $\mathbb{R}_+$ dans $[0;1]$ qui atteint ses bornes.

une première fonction à laquelle on peut penser est une fonction que l'on rencontre souvent en physique :

$$f(x) = 1 - e^{-\frac{x}{\tau}}, \tau \in \mathbb{R}_+^*$$

On peut alors faire varier le paramètre τ pour moduler la fonction comme on veut. La valeur que nous avons choisi est $\tau = 10$, car on a alors que pour avoir une distance de 0.5, il que les deux dates soient séparées de 7 ans, ce qui correspond à peu près à ce qu'on veut.

Une autre fonction utilisable possible est la fonction arctangente. On a alors

$$f(x) = \arctan(\alpha \cdot x) \times \frac{2}{\pi}, \alpha \in \mathbb{R}_+^*$$

Comme précédemment, on cherche une valeur du paramètre α qui corresponde à nos attentes. Dans ce cas, on prend $\alpha = 0.15$, ce qui donne une distance de 0.5 pour 7 années d'écart.

Dans tous les cas, c'est une bonne chose d'avoir des paramètres modifiables dans ces fonctions pour pouvoir mieux approcher ce que l'on veut.

2.2.3 Amélioration de la distance entre les genres

Pour la distance entre les genres, on utilisait jusqu'à présent la distance de Jacard. C'est une méthode qui marche, mais elle omet certains éléments dans son calcul, ce qui le rend moins précis. En effet, on ne prend pas en compte

les potentielles ressemblances entre genres, on regarde simplement si un même genre est présent dans les deux films. Ainsi, si l'on considère 3 films dont les genres sont respectivement crime, mystère et romance, la distance de Jacard entre ces 3 films sera nulle, et on ne pourra pas les classer. Cependant, on se rend bien compte que les genres crime et mystère sont proches, alors qu'ils ne le sont pas de romance. Il faut donc essayer de prendre en compte les distances intrinsèques entre deux genres.

Pour cela, la meilleure méthode aurait été d'utiliser le machine learning. Avec une telle méthode, l'humain n'a pas besoin d'intervenir, il suffit d'analyser la base de donnée des films et de repérer ceux qui sont proches d'après les utilisateurs pour estimer des distances entre les genres. L'avantage est que l'on peut alors rajouter un nouveau genre sans problème, et sans expliciter son lien avec les autres genres : l'algorithme fait cela tout seul. Cependant, pour qu'une telle méthode marche, il faut une base de données assez conséquente pour avoir des résultats corrects. De plus, c'est une méthode difficile à mettre en place, et qui requiert des connaissances que nous n'avons pas.

Il faut donc utiliser une autre méthode, plus fastidieuse et moins précise. On va pour cela créer nous-même une matrice contenant les distances entre les films, puis on fera une moyenne des distances entre les genres du premier film avec ceux du deuxième film. Pour cela, il faut d'abord lister les films existant, qu'on obtient grâce à un simple algorithme qui parcours tous les film. On obtient la liste suivante : Action, Thriller, Drama, Adventure, Family, Romance, Fantasy, Sci-Fi, Animation, Comedy, Horror, Crime, Mystery, History, Musical, Sport, Biography, War et Music ; soit un total de 19 genres. On crée ensuite une matrice de distances qu'on utilisera pour nos calculs :

$$\begin{pmatrix}
0.0 & 0.3 & 0.5 & 0.1 & 0.6 & 0.9 & 0.2 & 0.2 & 0.7 & 0.7 & 0.4 & 0.6 & 0.6 & 0.7 & 0.9 & 0.5 & 0.8 & 0.2 & 0.9 \\
0.3 & 0.0 & 0.3 & 0.4 & 0.7 & 0.9 & 0.5 & 0.4 & 0.8 & 0.8 & 0.1 & 0.2 & 0.3 & 0.6 & 0.9 & 0.9 & 0.8 & 0.5 & 0.9 \\
0.5 & 0.3 & 0.0 & 0.6 & 0.7 & 0.4 & 0.5 & 0.6 & 0.8 & 0.8 & 0.3 & 0.3 & 0.3 & 0.2 & 0.4 & 0.6 & 0.5 & 0.6 & 0.8 \\
0.1 & 0.4 & 0.6 & 0.0 & 0.2 & 0.4 & 0.1 & 0.3 & 0.4 & 0.3 & 0.7 & 0.8 & 0.4 & 0.7 & 0.8 & 0.5 & 0.8 & 0.3 & 0.9 \\
0.6 & 0.7 & 0.7 & 0.2 & 0.0 & 0.7 & 0.5 & 0.6 & 0.2 & 0.1 & 0.8 & 0.7 & 0.5 & 0.8 & 0.3 & 0.4 & 0.9 & 0.7 & 0.6 \\
0.9 & 0.9 & 0.4 & 0.4 & 0.7 & 0.0 & 0.7 & 0.8 & 0.7 & 0.5 & 0.9 & 0.8 & 0.6 & 0.4 & 0.1 & 0.6 & 0.6 & 0.5 & 0.3 \\
0.2 & 0.5 & 0.5 & 0.1 & 0.5 & 0.7 & 0.0 & 0.3 & 0.5 & 0.4 & 0.5 & 0.7 & 0.4 & 0.8 & 0.7 & 0.9 & 0.9 & 0.2 & 0.8 \\
0.2 & 0.4 & 0.6 & 0.3 & 0.6 & 0.8 & 0.3 & 0.0 & 0.7 & 0.4 & 0.5 & 0.4 & 0.2 & 0.8 & 0.7 & 0.9 & 0.9 & 0.6 & 0.8 \\
0.7 & 0.8 & 0.8 & 0.4 & 0.2 & 0.7 & 0.5 & 0.7 & 0.0 & 0.1 & 0.7 & 0.7 & 0.4 & 0.8 & 0.3 & 0.8 & 0.9 & 0.7 & 0.4 \\
0.7 & 0.8 & 0.8 & 0.3 & 0.1 & 0.5 & 0.4 & 0.4 & 0.1 & 0.0 & 0.8 & 0.8 & 0.4 & 0.9 & 0.3 & 0.5 & 0.7 & 0.9 & 0.6 \\
0.4 & 0.1 & 0.3 & 0.7 & 0.8 & 0.9 & 0.5 & 0.5 & 0.7 & 0.8 & 0.0 & 0.2 & 0.4 & 0.6 & 0.9 & 0.9 & 0.8 & 0.3 & 0.8 \\
0.6 & 0.2 & 0.3 & 0.8 & 0.7 & 0.8 & 0.7 & 0.4 & 0.7 & 0.8 & 0.2 & 0.0 & 0.1 & 0.3 & 0.8 & 0.9 & 0.7 & 0.3 & 0.9 \\
0.6 & 0.3 & 0.3 & 0.4 & 0.5 & 0.6 & 0.4 & 0.2 & 0.4 & 0.4 & 0.4 & 0.1 & 0.0 & 0.4 & 0.6 & 0.6 & 0.7 & 0.5 & 0.8 \\
0.7 & 0.6 & 0.2 & 0.7 & 0.8 & 0.4 & 0.8 & 0.8 & 0.8 & 0.9 & 0.6 & 0.3 & 0.4 & 0.0 & 0.7 & 0.6 & 0.1 & 0.2 & 0.8 \\
0.9 & 0.9 & 0.4 & 0.8 & 0.3 & 0.1 & 0.7 & 0.7 & 0.3 & 0.3 & 0.9 & 0.8 & 0.6 & 0.7 & 0.0 & 0.6 & 0.8 & 0.6 & 0.1 \\
0.5 & 0.9 & 0.6 & 0.5 & 0.4 & 0.6 & 0.9 & 0.9 & 0.8 & 0.5 & 0.9 & 0.9 & 0.6 & 0.6 & 0.6 & 0.0 & 0.7 & 0.9 & 0.9 \\
0.8 & 0.8 & 0.5 & 0.8 & 0.9 & 0.6 & 0.9 & 0.9 & 0.9 & 0.7 & 0.8 & 0.7 & 0.7 & 0.1 & 0.8 & 0.7 & 0.0 & 0.7 & 0.9 \\
0.2 & 0.5 & 0.6 & 0.3 & 0.7 & 0.5 & 0.2 & 0.6 & 0.7 & 0.9 & 0.3 & 0.3 & 0.5 & 0.2 & 0.6 & 0.9 & 0.7 & 0.0 & 0.8 \\
0.9 & 0.9 & 0.8 & 0.9 & 0.6 & 0.3 & 0.8 & 0.8 & 0.4 & 0.6 & 0.8 & 0.9 & 0.8 & 0.8 & 0.1 & 0.9 & 0.9 & 0.8 & 0.0
\end{pmatrix}$$

C'est assez indigeste et laborieux à créer, mais une fois cela fait, l'utilisation est très simple. Chaque ligne et chaque colonne correspond à un genre, dans l'ordre de la liste, et à la ligne i colonne j, on a la distance entre le genre i et le

genre j. Les propriétés de la distance nous donne alors aussi des propriétés sur la matrice : puisqu'une distance est symétrique ($d(a, b) = d(b, a)$), la matrice l'est aussi. De plus la distance entre un genre et lui même est nul, donc les éléments diagonaux sont tous nuls.

Pour créer notre distance entre 2 ensembles de genres, on va simplement parcourir les 2 ensembles, et faire la moyenne de toutes les distances que l'on trouve grâce à notre matrice. Ainsi, si le film 1 a n genres et le film 2 en a m, on va faire la moyenne de $m \times n$ distances entre genres.

Il y a cependant un inconvénient à cet algorithme : il donne toujours des distances assez basses, même s'il permet de mieux ordonner. En effet on peut constater que la distance entre 2 ensembles identiques ne vaut pas toujours 0, ce qui peux poser certains problèmes.

2.2.4 Recommandation de contenu

Une fois qu'on a une méthode qui nous calcule une distance entre 2 films, on peut facilement créer une méthode qui prend en argument un film, et qui nous en recommande d'autres. Il suffit pour cela de calculer la distance entre ce film et chacun des autres, puis de trier par ordre décroissant. Les premiers films de la liste sont alors ceux à recommander.

2.3 Système de recommandation collaboratif

Comme vu dans l'Etat de l'art, la distance entre deux utilisateurs est définit à partir de toutes les données recueillies sur eux, ce qui comprend :

— les données personnelles
— les données de navigation
— les notes attribués par l'utilisateur
— et plus encore...

Toutefois, comme notre base de données ne contient que les notes des utilisateurs, nous avons décidé de ne prendre en compte que ces notes dans notre algorithme.

2.3.1 Création d'une distance entre 2 utilisateurs

La première étape consiste à déterminer, à partir d'une liste de notes pour chaque utilisateur, une distance entre ces derniers. La solution retenue a été d'utliser la corrélation de Pearson (détaillé dans l'Etat de l'art). Définissons U_1 comme étant l'utilisateur cible et U_2 l'utilisateur que l'on compare

On détermine dans un premier temps les notes moyennes pour U_1 et U_2 avant de calculer la corrélation de Pearson en ne prennant en compte que les films vus par U_1 et U_2.

Il en résulte une note comprise entre $[\![-1; 1]\!]$. Cependant, pour les suite des calculs, nous avons besoin d'une note comprise dans l'intervalle $[\![0; 1]\!]$. On ajuste

donc le coefficient de Pearson à l'aide de la formule :

$$distance(U_1, U_2) = \frac{1 + coef_{Pearson}}{2}$$

Maintenant qu'il est possible de calculer la distance entre deux utilisateurs, il suffit de ré-itérer cette opération pour tous les autres utilisateurs. On obtient alors liste de n-1 couples $(identifiant_{Ui}, distance(U_1, U_i))$ avec : $0 \leqslant i < n; i \neq 1$

2.3.2 Estimation d'une note

L'objectif est maintenant de pouvoir estimer la note que U_1 donnera à un film donné en se basant sur celles des autres utilisateurs qui ont déjà noté ce film. Pour cela, on applique la formule suivante.

Soit x un film donné et U, l'ensemble des utilisateurs ayant notés ce film. On a :

$$Note\_estime(x) = \frac{\sum_{i \in U} sim_i.note_i}{|sim_i|}$$

On effectue donc ce calcul pour tous les films non notés par U_1 et obtenons ainsi une liste non triée de tous les films non vus par U_1. Il ne reste alors plus qu'à les trier.

2.3.3 Tri des films

Pour trier les films en fonction de la note estimé, on applique un algorithme récursif inspiré des algorithmes MergeShort et QuickShort. Le principe est le suivant :

- On prend le premier élément de la liste qui prendra comme pivot et qu'on stocke dans un nouvelle liste appelé "centre"
- Pour tous les autres éléments de la liste :
  - si l'élément est supérieur au pivot, on le stocke dans la liste appelé "droit"
  - si l'élément est inférieur au pivot, on le stocke dans la liste appelé "gauche"
  - si l'élément est égal au pivot, on le stocke dans la liste "centre"
- On réitère cet le même algorithme jusqu'à ce qu'on a plus que 1 ou 0 éléments dans les liste "gauche" et "droite" (étape récursive)
- Enfin on concatène l'ensemble dans l'ordre : "gauche" + "centre" + "droit"

La liste obtenue est maintenant triée dans l'ordre croissant des notes estimés.

2.4 Structures de données

Les structures de données du projet peuvent être séparées en 2 types :

- Statiques
- Dynamiques

Les structures statiques sont celles qui sont directement inscrites dans le code du programme C, écrites en dur dans le code, ce sont des structures de bas niveau. Ce sont les structures que l'on va définir avec le mot clé struct en C, et qui sont destinées à contenir des données connues par avance, un tableau, une autre structure, des variables, etc...

Les structures dynamiques sont celle qui utilisent une autre structure plus flexible et qui permet l'agencement de données pour créer d'autres structures de données. Ce sont les structures basées sur le JSON, qui vont utiliser libcjson, et qui permettent d'ajouter et supprimer dynamiquement des éléments via des fonctions. Pour ces stuctures nous avons défini un fichier de référence en Markdown sur le dépot git, afin que le programme utilise un format uniforme de données, mais une structure JSON permet en réalité de représenter n'importe quelle structure de données.

2.4.1 Structures de données statiques

2.4.2 Structures de données dynamiques

2.5 Fonctionnement de l'application produite

2.5.1 Fonctionnement du serveur

2.5.2 Fonctionnement du client CLI

2.5.3 Fonctionnement du client GUI

2.6 Protocole de communication entre clients