Documentation avancée

Dauphilme

4 février 2018

1 Introduction

Créé à partir d'une parcelle de la base de données de Netflix, Dauphilme est un projet de quelques semaines qui présentes de bonnes idées mais aussi des lacunes. C'est un projet ouvert, c'est-à-dire que le code source (en C) est disponible pour tous mais également ouvert vers l'internationnal car les variables et les finctions, ainsi que les commentaires sont en anglais. Nous avons néanmoins fait le choix de dévelloper notre interface utilisateur en français mais n'importe quel programmeur averti peut modifier la langue sans trop de difficulté.

2 Récupération des données

Cette partie du projet est la première mais également la plus importante car, si elle est mal faite, cela peut affecter la justesse du reste du projet ou même compromettre sa rapidité sur de grands fichiers (quelques centaines de Mo). On pourrait même dire que c'est un peu la pierre angulaire du projet.

Nous avons commencé ce projet en occultant la question de la rapidité et en exécutant nos programmes sur des fichiers de petite taille pour pouvoir vérifier les résultats de nos fonctions et débeuguer sans avoir une centaine d'affichages quand on veut voir une progression par exemple. Malheureusement, cette méthode nous a déservi car nous nous sommes rendu compte après coup qu'en triant à chaque fois les films de chaque utilisateur ainsi que les utilisateurs de chaque film, notre programme était beacoup trop lent pour faire face ne serait-ce qu'à une petite portion d'un combined data.

Nous avons aors décidé d'arrêter de trier, d'autant plus que nous ne l'avions presque pas utilisé dans les questions suivantes et cela nous a permi de diminuer drastiquement la complexité de notre algo. Nous executopns maintenant deux fichiers combined data en une dizaine de secondes mais, mystérieusement, le troisième fichier mets beaucoup plus longtemps à charger ... A ce jour, ce problème n'a pas été résolu mais -à bon entendeur- ne devrait pas être compliqué à contourner.

3 Voisins et similarité

Encore une question importante pour la suite du projet, la recherche de voisins à travers la similarité cosinus n'a pas été très difficile à coder de par notre compréhansion rapide de la quasi-nécessité de la création d'une structure Neig(pour Voisin). Néanmoins, cette partie a posé des problèmes de rapidité également quant-au calcul de la similarité où ,à chaque nouvel appel de la fonction BasicCosinus, on itérait en $O(n^2)$ sur les deux utilisateurs. A la place, nous avons opté pour un calcul au préalable de la somme des carrés des notes de u ainsi qu'un listing de ses notes dans un tableau afin de faire baisser cette complexité en O(n).

4 Recommandations

Voici la partie la plus intéressante mais aussi la plus longue du projet. Elle aura demandé beaucoup d'attention et de temps.

4.1 Simple Recommandation

En soi, la recommandation simple d'un film ne nous a pas posé de problème mais nous nous sommes étonné du fait qu'elle pouvait être faussée si peu de voisins avaient vu le film ,voire aucun. Nous avions caressé l'idée de ne considérer que les utilisateurs ayant vu le film en question dans la recherche des voisins pour une recommandation optimale mais, par manque de temps, nous avons abbandonné cette piste.

Nous avons également fait le choix de donner des recommandations réelles et non pas seulement entières pour une plus grande précision.

4.2 Suggestion de films

Nous entrons ici dans le vif du sujet et nous touchons au but de notre projet avec cette partie de recommandation de films. Encore une fois, nous avons privilégié à la création de tableau parallèles une structure ListAdvice pour être "plus propres". Malheureusement, nous aurions voulu réutiliser la structure List de la question 2 mais elle avait un attribut spécifique de type Neig*. Nous avons alors tenté de rendre cette structure plus générale pour pouvoir l'addapter à tout type de pointeur en changeant le Neig* par un void*, sans succès. Nous avons donc créé artificièlement un maillon beg qui faisait office de début.

4.3 Suggestion Dynamique

Grâce à cette partie, il est possible pour l'utilisateur de se faire suggérer des films en temps réel sur la base de ses notes (fictives) aux films les plus "populaires". Nous avons décidé d'incorporer le titres des films directement dans les Recommandations 2 et 4 pour ne pas avoir de redondance dans le code. Nous avons créé des macros qui permettent de changer le nombre de voisins ou de suggestions en un clic. On peut dire que c'est incohérent avec le reste des questions (où l'on obtient les paramètres directement de l'utilisateur) certes mais cette façon de faire nous a été bien utile et n'est pas difficile à supprimer si elle ne vous plaît pas.

5 Calcul du taux d'erreur

C'est en apparence une foinctionnalité facile à implémenter que le calcul du tau d'erreur du programme. Mais dans le fond, pour avoir un calcul rapide de ce taux sur plus de 0.01 pourcents des données, il faut avoir un calcul de voisins très rapide, ce qui n'est pas forcément notre cas. Nous avons pourtant essayé de limiter la complexité de cette question en cachant au hasard *les premiers* utilisateurs (c.à.d. les derniers utilisateurs à avoir noté le film) , ce qui nous permet d'avoir un calcul plutôt fiable sans devoir gérer si les maillons sont NULL ou autre.