Cerisara Nathan et Lafont Loïc

Rapport du Projet Programmation

Description du projet

Description Générale

NextWatch est une application de prédiction de films basée sur une base de données de vieux films sur Netflix.

Ce programme est composé de deux parties :

- Un parser qui lit les fichiers textes de la base de données et qui va les convertir en un fichier binaire bien plus rapide à lire (1~2 minutes pour la lecture des fichiers textes contre quelques secondes pour les fichiers binaires).
- Une partie composée d'algorithmes de prédictions. Ceux-ci peuvent prendre plusieurs options en compte qui devront être rentrées dans la ligne de commande qui lance l'application.

Description des Algorithmes

On a implémenté plusieurs algorithmes dans ce projet, avec chacuns leurs points forts et leurs points faibles :

- Un *algorithme aléatoire*, qui est très rapide, mais qui n'est absolument pas pertinent, mais qui permet de découvrir une très large variété de films, que l'on ne verrait pas forcément ailleurs.

  Le principe de cet algorithme est le suivant : On choisit des films aléatoirement sur la liste complète de films.

- Un algorithme que l'on a nommé *Partial Clustering* qui est très lent mais qui donne de plutôt bons résultats.

  Le principe de cet algorithme est le suivant : On cherche une liste d'utilisateurs qui sont le plus proches de la liste de films donnée en entrée, on récupère l'union des films qu'ils ont regardés à laquelle on passe un filtre, et on tire aléatoirement un film dans la liste restante.

- Un algorithme appelé *Random Indexing* qui utilise des vecteurs d'embedding pour les films qui ont été calculées disponible sous deux versions (une version avec des vecteurs de dimension 10, et une version avec des vecteurs de dimension 100)

  Le principe de cet algorithme est le suivant : On précalcule à l'avance des vecteurs d'embedding pour chaque films; au début, pour chaque film, on initialise son vecteur aléatoirement, puis pour chaque utilisateur, si deux films ont été aimés par cet utilisateur (note >= 3), on rapproche leurs vecteurs (*v_film_1 = lin_norm(v_film_1 + coeff * v_film_2), v_film_2 = lin_norm(v_film_2 + coeff * v_film_1)*). (*algo2.c -> prepare_algo2*). Ensuite, dans le programme, on charge les poids de l'algorithme puis on renvoie juste les films les plus proches des films sélectionnés.

- Un *algorithme de bons films aléatoires*, qui est très rapide, pas très pertinent, mais qui permet de découvrir une bonne variété de bons films.

  Le principe de cet algorithme est le suivant : On choisit des films aléatoirement sur un sous-ensemble de bons films (note moyenne >= 4).

- Un algorithme qu'on a appelé *Movie from Movie* qui est très rapide, et qui peut donner des résultats très intéressants. On peut le voir comme une sorte de simplification de l'algorithme *Partial Clustering*.

  Le principe de cet algorithme est le suivant : On prend un film donné en entrée, on prend un utilisateur qui a aimé ce film, et on renvoie un autre film que cet utilisateur a aimé.

Pour la prédiction de films, lorsqu'un utilisateur a choisi un algorithme, **75%** des prédictions totales seront réalisées par l'algorithme sélectionné, et les **25%** restants seront fait par l'algorithme de bons films aléatoires, pour toujours permettre à l'utilisateur de découvrir de nouveaux films, et avoir un peu de diversité pour ne pas rester enfermer dans une bulle.

Les limites du projet

Certaines “options” ne sont pas implémentées dans notre algorithme, comme la sélection de bons ou mauvais clients car nous n’en avons pas vu l’utilité (et pas le temps).

Il aurait été possible d’encore compresser les données du parser mais cela aurait pris beaucoup de temps pour une faible optimisation nous avons donc préféré nous concentrer sur des éléments plus importants.

Au niveau de l’interface graphique (qui soit dit en passant est très belle), nous aurions pu, avec plus de temps, ajouter la sélection des options directement depuis l’interface via des cases à cocher ou décocher lorsque l’on veut ou ne veut pas utiliser l’option. Il manque aussi des filtres de sélection qui n’ont pas été implémentés par manque de temps.
Au niveau de l’interface graphique, nous n’avons pas non plus eu le temps d’implémenter quelques fonctionnalités de conforts telles que le scroll au niveau d’un container scrollable avec la molette de la souris lorsque la souris est au-dessus de ce container. Il y a aussi un petit problème d’affichage au niveau du scroll de la liste de films,

Bien que nous n’ayons à lancer le parser qu’une seule fois, celui-ci est assez long à s'exécuter: il prend environ 1~2 minutes à s’exécuter selon sur quel machine on l’exécute.

La compilation prend aussi un peu de temps, c’est normal.

Il y a aussi quelques bugs dans le programme au niveau de la libération de la mémoire que nous n’avons pas réussi à débuger à temps.

Comment installer / utiliser le programme

Installation

Prérequis

Vérifiez que vous avez au moins 1,5 Go de libre sur votre espace de stockage.
Si ce n'est pas le cas, il faut faire de la place. C'est nécessaire pour que le parser puisse stocker les données binaires qu'il a calculées.

Vous aurez besoin d'avoir un compilateur C pour compiler le projet. Plus d'infos sur https://gcc.gnu.org/install/index.html et https://gcc.gnu.org/install/binaries.html.

Vous aurez besoin d'avoir la librairie SDL2 installée sur votre machine. Si ce n'est pas le cas, vous pouvez vous documenter ici : https://wiki.libsdl.org/SDL2/Installation.

Téléchargement

Cloner le dossier "NextWatch". (`git clone https://git.unistra.fr/llafont/my-awesone-project.git projet_CERISARA_LAFONT`)

Télécharger la base de donnée (https://academictorrents.com/details/9b13183dc4d60676b773c9e2cd6de5e5542cee9a) et la placer dans le sous-dossier data/ de telle sorte à avoir cette hierarchie :

```
data/
    movie_title.txt
    probe.txt
    qualifying.txt
    README
    training_set/
        mv_0000001.txt
        mv_0000002.txt
        mv_0000003.txt
            ...
        mv_0017770.txt
```

Compilation

Ouvrez un terminal et allez dans **à la racine** du dossier que vous venez de cloner.

Compilez le programme avec la commande : `make clean && make all`

Préparation des données

Lancer le parser avec la commande : `./bin/main -p`

Attendre que le parsing se termine, cela prend en général 1 à 2 minutes.

Vous pouvez maintenant utiliser l'application.

Utilisation de l'application

L'application se lance avec la commande : `./bin/main`

Il y a plusieurs paramètres pour utiliser l'application :

- `-h` : affiche l'aide.
- `-p` : lance le parser au lieu des algos.
- `-m` *`mode_utilisé`* : Si ce paramètre n'est pas utilisé, le programme ne va lancer l'algorithme qu'une seule fois. Si vous utilisez ce paramètre, vous pouvez utiliser deux modes supplémentaires
    - `"console"`, mode console : Permet de ne charger qu'une seule fois les données et de pouvoir lancer plusieurs commandes pour utiliser les algorithmes.
    - `"window"`, mode fenêtré : Ouvre une fenêtre qui permet de sélectionner les films à la main, de changer d'algorithmes à la volée, et de voir les résultats des algorithmes.
- `-t` : affiche dans la console le temps d'exécution des algorithmes.
- `-n` *`numéro`* : Précise le nombre de films que l'algorithme de prédictions va donner. Par défaut est égal à 10.
- `-a` *`nom_algorithme`* : Sélectionne l'algorithme choisi (pourra être changé dans l'interface graphique). Le nom de l'algorithme doit être dans la liste suivante : [`"random"`, `"partial_clustering"`, `"random_indexing_v10"`, `"random_indexing_v100"`, `"random_good_movies"`, `"movie_from_movie"`].
- `-r` *`[0, 1, 2, …, 17770 | chemin_fichier.txt]`* :
    - Prend en argument soit une liste d'id de films,
    - soit un fichier qui contient une liste de films (un film par ligne, pas d'autres caractères).

  Ces films seront la base de certains algorithmes de prédictions, et les algorithmes essaieront de donner des films qui sont en rapport avec les films donnés avec le paramètre `-p`.

Vous pouvez donc par exemple lancer l'application comme ceci : `./bin/main -m window`.