PROJET SD - FILMFINDER

Réalisé par : Levy Yoni et Tissier Romain
TELECOM Nancy 1ère année
Date : 26 Mai 2015

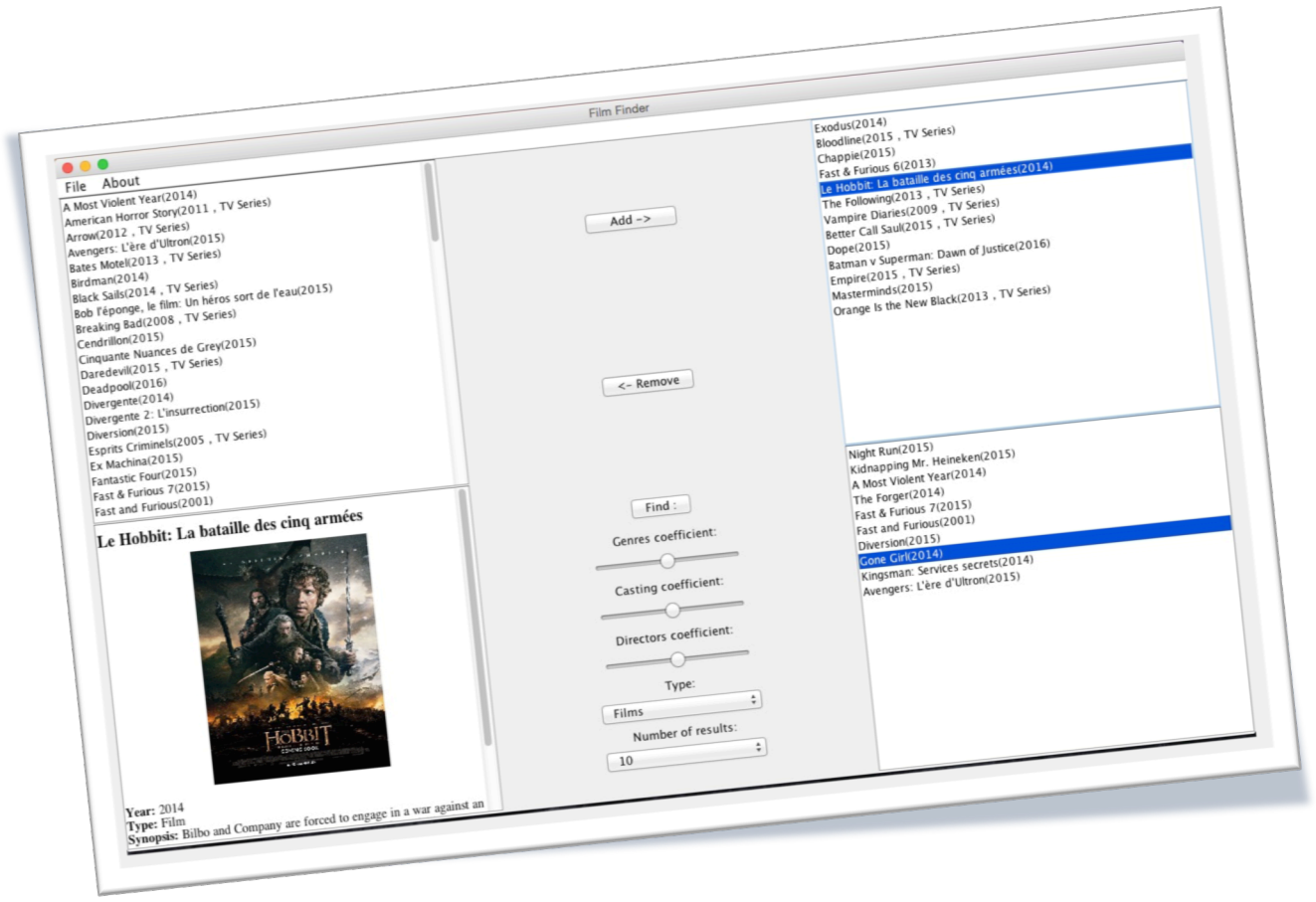


Table des matières

Introduction 3

Dépendances logicielles 3

Utilisation du programme 3

- Mode console 3
- Mode graphique 4

Organisation du code 5

- Diagramme de classe 5
- Utilisation des structures de données 6
- Algorithme de recherche 6
- Difficultés rencontrées 6
- Gestion du temps 7
- Amélioration de l'algorithme 7

Introduction

Ce projet de SD avait pour but de créer un programme que nous avons nommé FilmFinder et qui permet, grâce à une liste de films et de séries, de conseiller des films à son utilisateur en fonction de ceux qu'il a déjà vu.

Dépendances logicielles

Pour faire tourner ce programme la dernière version de la machine virtuelle java (1.8) est suffisante. Au niveau des performances graphiques et matérielles, nous n'avons pus tester le programme que sur des machines relativement récentes de type UNIX. Nous garantissons son fonctionnement pour Fedora 21 et Mac OS X 10.10.

Utilisation du programme

Ce programme a été conçut de manière modulable grâce à l'utilisation de la programmation orientée objet en Java. Ainsi, il propose deux interfaces distinctes : le mode console et le mode graphique.

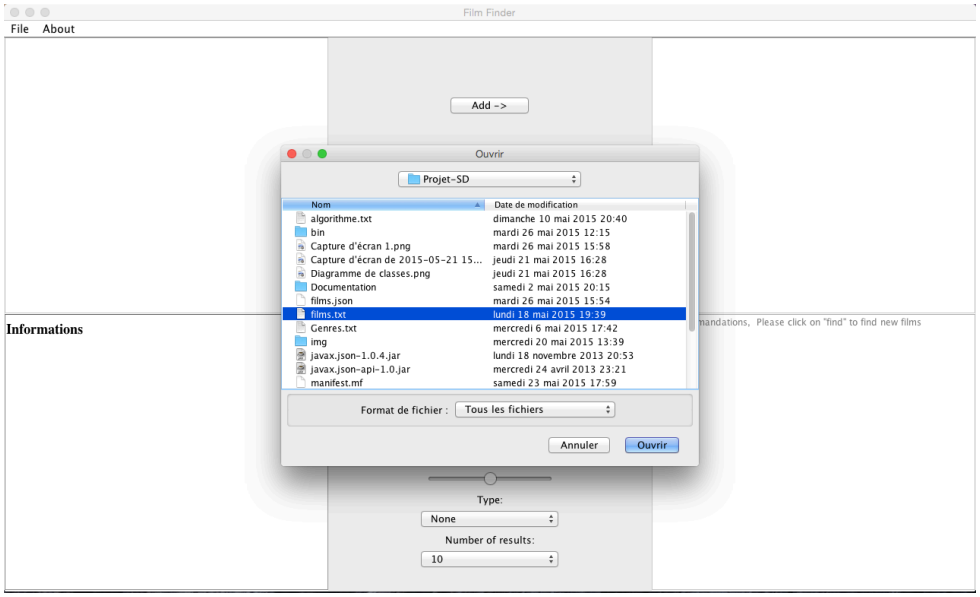
Mode console

Le mode console du programme interprète la commande, l'analyse, renvoi une erreur si besoin et traite la requête. Cependant, le mode console comporte de nombreuses options d'utilisation plus ou moins complexes qui doivent être rentrées en arguments. Voici la liste des différentes options :

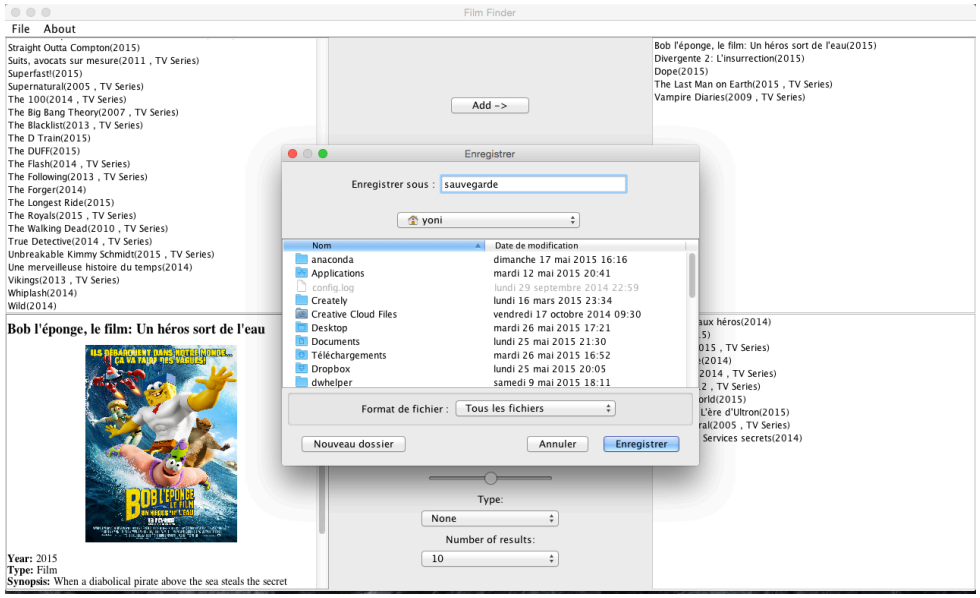
- Indiquer le chemin du fichier qui contient la liste des films (au format JSON ou TXT).
- Indiquer les films que l'on a vus.
- Affecter un poids pour définir l'importance du casting, des réalisateurs et des genres lors de l'exécution de l'algorithme. Par défaut, leurs valeurs sont 1.
- Activer un filtre sur la durée des éléments de sorties. (A l'usage, cette option n'est pas réellement pertinente et a été supprimée dans l'interface graphique.).
- Choisir le type de média voulu en sortie de l'algorithme. Par défaut la valeur est NONE.
- Indiquer le nombre de résultats souhaité en sortie. Par défaut, le nombre de résultat est 10.
- Exécuter l'interface graphique
- Afficher l'aide
- Activer la sortie au format JSON.

Mode graphique

Le mode graphique permet une interface plus interactive avec le logiciel. Voici quelques captures d'écran :



Ouverture du fichier contenant la liste de films



Sauvegarde des résultats au format JSON

L'interface graphique est composé de deux menus : un menu « File » et un menu « About ». Le menu « File » permet de charger une base de donnée de film (l'argument --file fonctionne

toujours et chargera automatiquement la liste des films sur l'interface), il permet aussi d'enregistrer le résultat de la recommandation au format JSON, enfin il permet de quitter le logiciel. L'interface est divisée en 5 grandes parties :

- la première est une liste déroulante correspondant aux films non vu,
- la seconde contient les boutons permettant d'ajouter ou d'enlever un film vu,
- la troisième est la liste des films vu,
- la quatrième retourne les informations correspondants au dernier film sélectionné (dans une des trois listes),
- la cinquième affiche la configuration de l'algorithme et permet de l'exécuter,
- la dernière est la liste des films recommandés par l'algorithme.

Organisation du code

Diagramme de classe

Diagramme de classes UML 1

Le programme s'articule autour de trois classes principales :

- la classe Media, qui modélise un media (film ou série) et ses informations,
- la classe MediaAlgorithm qui contient l'implémentation de l'algorithme,
- la classe Database qui modélise la base de donnée de film

Ces trois classes sont utilisés par l'interface graphique, c'est d'ailleurs pour cela que ArrayListModel fait le lien entre Media et Database : l'interface graphique gère mieux les objets dérivés de AbstractListModel. Au niveau de l'implémentation, Media hérite de Comparable afin que l'on puisse trier les listes par ordre alphabétiques dans la classe ArrayListModel. Enfin View hérite de Jframe et s'exécute donc comme une fenêtre graphique.

Utilisation des structures de données

Trois types de structures de données ont été choisis. Tout d'abord les ArrayList qui offrent une gestion de tableau dynamique. Elles sont donc utilisées par le programme pour gérer des listes de films. Nous avons retenu son format pour sa rapidité d'accès. En effet, nous pensions dans un premier temps utiliser les LinkedList, qui sont des listes chainée classique. Cependant ArrayList gère en interne ses données par des tableaux, ainsi l'accès est plus rapide mais l'ajout est plus lent.

L'ajout est un élément important du programme, mais les accès aux données sont plus important : en effet dans l'interface graphique, il est plus correcte d'avoir une latence au démarrage plutôt qu'en cours d'utilisation pour afficher un film. La seconde structure choisie est tout simplement le tableau statique. En effet, nous l'utilisons pour la classe Media : la base de donnée n'est pas modifiée en cours d'utilisation, ainsi il est plus intéressant d'utiliser ce type de structure pour améliorer la vitesse d'accès aux informations.

Enfin les nous utilisons les HashMap qui nous ont permis d'associer facilement un entier à une clé unique (ici le type Media), afin de réaliser le tableau des différents coefficients.

Algorithme de recherche

L'algorithme de recherche qui a été implémenté dans ce projet se base sur les caractéristiques des films que l'utilisateur a marqués comme étant vus. En premier lieu, l'algorithme va compter le nombre d'occurrence de chaque genre, de chaque acteur et de chaque directeur dans la liste. Par la suite, en fonction des différents paramètres rentrés par l'utilisateur (les coefficients sur les genres, les acteurs et les directeurs) l'algorithme va, pour chacun des films non vus, incrémenter un compteur pour chaque caractéristique calculée précédemment. Enfin, l'algorithme va garder seulement les 10 vidéos (valeur par défaut) qui possèdent le compteur le plus élevé et qui correspondent au type rentré par l'utilisateur (films, séries ou les deux).

Difficultés rencontrées

Le projet n'a pas posé de difficultés particulières. Le point le plus complexe était la recherche d'un algorithme efficace pour le programme.

Gestion du temps

Nous nous sommes répartis le travail de la façon suivante :

- Création de la méthode qui permet de récupérer les différents médias depuis le fichier texte (Romain a passé 2h dessus)
- Implantation de l'algorithme de recommandation (Yoni a passé 3h dessus)
- Création de la fonction utilisant la ligne de commande (Nous avons passé 2h dessus)
- Améliorations de l'algorithme (Yoni a passé 2h dessus)
- Création de l'interface graphique (Romain a passé 4h dessus)
- Implémentation de l'option exportant en JSON (Romain a passé 2h dessus)
- Amélioration de l'utilisation en ligne de commande (Nous avons passé 2h dessus)
- Améliorations de l'interface graphique (Yoni a passé 2h dessus)

Amélioration de l'algorithme

Nous avons essayé d'intégrer un nouvel algorithme qui se base sur les films vus par différents utilisateurs et les garde en mémoire dans une base de donnée au format texte. Si le lien existe déjà, on incrémente le compteur, sinon on l'ajoute. Ainsi lors de la recherche, nous n'avons qu'à regarder quel lien à le plus grand incrément pour un film donné. Cette méthode peut être très efficace mais peut aussi comporter des erreurs.