IFT3913 - Travail Pratique 4

Daniel El-Masri (20096261) et Rym Bach (20078272)

18 Decembre 2020

Introduction :

Ce programme est fait pour prendre l'url d'un repositoire git et de calculer pour chaque version de *commit* le degrée de code commenté.

Pour executer le programme, il faut executer le script python "proto.py" dans la console.

Le script peut prendre 2 valeurs comme paramètre:

1. -url OU –repository : qui est l'url du repositoire git.
2. -max OU –maximum : un paramètre optionel pour indiquer un nombre maximal de commit voulu (pour les repositoires de grandes tailles, on peut indiquer une limite). Si aucun maximum est entré le programme calculera le degrée pour tout les commits dans l'historique.

Il est important de savoir que le plus le nombre de commit est grand, plus les données sont précises. Puisque on utilise des librairies externes, il serait bien de tester avec un minimum de 30 commit dans la branche *master* du repositoire.

Note Importante : Ce travail pratique à permis de rééler un problème dans notre solution du Travail Pratique 1 qui fais que lorsque le fichier a plus que 1000 fichiers, le CodeAnalyzer plante... Cela n'est cependant pas un problème dans le code de la solution du Travail Pratique 4 car c'est une erreur *Java.Lang.ArrayOutOfBoundsException*.

Le code plantera si toute la colonne n_classe contient la meme valeur (ex: [2,2,2,2,...,2]) MAIS cela est causer par la librairie utiliser et arrive rarement (cela arrive pour jfreechart en bas de 100 commits)

Le code plantera si le lien de l'url ne contient pas l'extension .git à la fin

un exemple d'execution est :
python.py proto.py -url https://github.com/jfree/jfreechart.git -max 100 python.py proto.py -url https://github.com/demasri/IFT3913-TP1.git

Description du Code:

Le code est bien séparé pour chaque étape a suivre, nous allons décrire le fonctionnement de chaque étape une par une:

1. Cloner le repositoire git : la méthode *clone_git_repository* prend deux valeurs en paramètres. Le premier est l'URL du repositoire et le deuxième est le nom du repositoire. Le code commence par chercher si le repositoire existe déjà, si celui-ci existe déjà alors le code va simplement aller chercher les plus récentes modifications. Si le repositoire n'existe pas, le code va aller cloner le repositoire grâce au lien qui lui est passé en paramètres. Une fois le repositoire cloner, le code va aller chercher la version la plus récente.

2. Obtenir la liste des hashes de commits : La méthode *get_hashes_list* ne prend aucun paramètres. Cette méthode exécute la commande *git rev-list master* pour aller chercher la liste des hashes de versions puis elle retourne un tableau contenant cette liste.

3. Calculer les données : La méthode *go_through_master_history* contient la majorité de la logique du programme pour la partie I. Tout d'abord elle regarde si un maximum à été entré en paramètres. Si oui, il y aura une boucle qui s'assurera que le code s'exécute seulement autant de fois que le maximum le demande. Ensuite elle passe à travers la liste des hashes, et pour chacun, elle met à jour le repositoire avec la version du hash et créer un tuple le contenant le hash de la version, le nombre de fichiers .java et la médiane de la métrique classe_BC. Finalement, elle retourne un tableau contenant ces tuples.

4. Compter le nombre de fichier JAVA : La méthode *count_java_files* fait partie de la logique du programme. Elle permet de passer à travers un repositoire dont le path lui est passé en paramètres puis elle parcourt tous les fichiers et sous-répertoire pour aller compter le nombre de fichiers aillant une extension .java.

5. Compiler l'analysateur de code : La méthode *compile_CodeAnalyzer* prend en parametre le path du repositoire et compile l'analysateur de code (le *CodeAnalyzer* crée dans le TP1).

6. Executer l'analysateur de code : La méthode *execute_CodeAnalyzer* prend en parametre le path du repositoire et exécute' l'analysateur de code (le *CodeAnalyzer* crée dans le TP1) sur le path du repositoire qui lui ai pass]'e en paramètre.

7. Calculer la mediane de classe_BC : La méthode *calculate_median_BC* permet de prendre un fichier CSV et de calculer la médiane de métrique classe_BC. Pour faire cela, elle va simplement aller chercher la colonne contenant la métrique classe_BC puis elle calcule la médiane.

8. Créer le fichier CSV résultant : La méthode *write_output_csv_file* prends en paramètre le tableau de tuple *data* et le nom du repositoire *repo_name* puis elle écris un fichier CSV avec un nom de fichier contenant le nom du repositoire. Cela est pour permettre d'executer le code sur plusieurs repositoire sans perdre les donnés crée anciennement.

9. Faire le test de normalité : La méthode *check_for_normality* prends en paramètre le nom du repositoire à tester puis commence par aller chercher les donnée nécéssaire pour faire les tests statistiques. Ensuite, en utilisant la librairie *scipy*, calcule la p-value. Une fois le résultat obtenue, la fonction retourne la valeur obtenue et la conclusion associé à cette valeur.

10. Faire le test de corrélation : La méthode *check_for_correlation* prends en paramètre le nom du repositoire à tester puis commence par aller chercher les donnée nécéssaire pour faire les tests statistiques. Ensuite, en utilisant la librairie *scipy*, calcule le coéfficient de corrélation de Pearson. Une fois le résultat obtenue, la fonction retourne la valeur obtenue et la conclusion associé à cette valeur.

11. Fonction aidante : Cette méthode *get_specific_data_from_column* est une méthode aidantes privé qui prends en paranètre le nom du fichier et l'index de la colonne pour que celle-ci permet simplement de aller chercher les données une colonne spécifique dans un fichier CSV. Puisque cette logique fut utilisé à plusieurs reprises il était plus logique de séparer cela dans une fonction à part qui est appelé plusieurs fois.

Conclusion : je tiens à vous remercier pour l'extension du délai pour la remise du devoir! Ce fût un vrai plaisir apprendre avec vous et Mr. Famelis. J'ai énormément apprécié et appris au courant de l'année!