Thesis Title

John Doe

24 septembre 2015

ABSTRACT

Short summary of the contents of your thesis.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMS

DRY Don't Repeat Yourself

API Application Programming Interface

UML Unified Modeling Language

Première partie

CONTEXTE ET ENVIRONEMENT

1

Internet est un outils efficace pour diffuser savoirs et connaissance. Depuis quelques années, par exemple, les MOOCs se sont développés. Un MOOC, acronyme pour Massive Open Online Course, est, assez simplement, un cours en ligne.

Si chaque cours pourra être organisé différemment, une caractéristique contraignante demeure : le nombre d'élève, disproportionné par rapport au nombre d'enseignant. On peut atteindre plus de 100 000 élèves dans le cas de cours anglophone. Cela affecte particulièrement le suivi des élèves : il devient très compliqué, voir impossible, de suivre les élèves et de leur fournir une aide adapté le cas échéant.

Une solution peut alors être de chercher des outils informatique pouvant aider ou dans certain cas remplacer l'enseignant. Le format informatique du cours permettant par ailleurs une intégration aisée d'un tel outil dans le cours.

Plusieurs objectifs peuvent être envisagé. On pourrait essayer de sélectionner des exercices pertinent pour l'élève, d'évaluer son niveau, lui fournir des conseils automatique,.... La problématique qui va nous intéresser ici sera de détecter des élèves en difficulté, afin d'ensuite pouvoir intervenir et leur fournir une aide adaptée.

La définition d'en difficulté, cependant, peut être délicate. La question est de savoir ce quel est le but du cours pour les étudiants. Dans certains cas, se sera la réutilisation professionnel des connaissances acquises, ce qui sera quasiment impossible à mesurer. Dans d'autres, cela pourra être une utilisation quotidienne (langues,...), une meilleure culture générale,... Il faut donc trouver un moyen d'évaluer les élèves. Assez logiquement, les notes d'examens devraient être un indicateur correct ; de même, on pourra évaluer d'éventuels exercices intermédiaire. Il faut cependant garder à l'esprit que l'on devra choisir un indicateur, qui sera au moins partiellement arbitraire.

Par ailleurs, le but exact est de savoir quel élève a besoin d'une aide, et quand. Un élève en tête de classe, par exemple, pourrait dans certains cas profiter grandement d'une intervention sur un exercice avancé. L'intérêt de l'intervention n'est pas nécessairement lié au niveau de l'élève. La question est également de savoir comment l'on veut répartir les ressources de l'enseignant si elle sont limité : dans le cas d'un système automatique, dispenser des conseils ne pose pas de problème, mais si le l'enseignant doit intervenir lui-même, aider un

élève signifie qu'un autre ne profitera pas de cette attention. Idéalement, il faudrait mesurer l'efficacité d'une intervention. On pourrait par exemple tester un outils ou une méthode sur une partie d'une classe, et laisser les autres élèves sans aide. Cela demanderait des résoudre une série de problème. Ethique, tout d'abord, mais également de moyen ou les biais possible.

La solution pour définir en difficulté est au final d'imposer une définition. Le but sera d'avoir des éléments mesurable, afin de pouvoir exploiter cette définition, tout en étant crédible d'un point de vue pédagogique.

2

La PLM, pour Programmer Learning Machine, est un logiciel d'apprentissage de la programmation. Il est principalement utilisé par les élèves de première année de Telecom Nancy, et a été développé pour ce public. Le projet a été initié par Gérald Oster et Martin Quinson en 2008, et a bien évolué depuis. Le dernier changement majeur fut le passage d'une structure de client lourd vers une architecture client-serveur web. Cette dernière version est celle qui a été utilisé pour la rentrée 2015, et est le fruit du travail de Matthieu Nicolas.

La PLM se présente comme un environnement de développement simplifié, dans lequel l'élève peut résoudre les exercices qui lui sont proposé. Différentes leçons sont proposées, de la plus basique, welcome, qui reprend les bases de la programmation, à certains concept plus avancés, comme les tris. Chaque exercice est associé à son énoncé, son API (la liste des fonctions utilisables dans le "monde" correspondant), et son monde, donc. Par exemple, dans beaucoup d'exercice, il sera question de déplacer un pion (un buggle pour être exact) sur un quadrillage, et de lui faire effectuer différente tâche.

La PLM est un projet libre (sous license GPL GNU pour être précis), et parfaitement utilisable au dehors de Telecom Nancy. De ce fait, si la majorité des utilisateurs et des traces associées sont des étudiants de première année, nous disposons également de traces d'utilisateurs autres, sans plus d'informations sur ceux-ci.

La PLM est également prévu pour fournir des traces de l'activité des élèves : ces traces, locales, peuvent servir pour évaluer le travail effectué. Elles peuvent également être publiées publiquement si l'élève y consent. Elle sont alors anonymisées.

2.1 LORIA

2.2 TELECOM NANCY

2.3 LES TRACES : PLM-DATA

Les étudiants peuvent donc accepter de publier leurs traces. Celles-ci ne seront utilisées dans un but de recherche, et ne sont pas nominatives.

Différents événements sont enregistrés : ce qui nous intéresse particulièrement sont les exécutions de code, c'est à dire quand l'élève teste son programme. D'autres actions sont également enregistrés, comme le fait de changer d'exercice, ouvrir ou quitter la PLM, ou de rester inactif pendant un certains temps. En fait, un maximum d'information est conservé : le but est de créer un base de données permettant de future étude ; il est difficile de savoir à l'avance quelle information seront pertinentes. A chaque fois, diverses informations sont enregistrées : le résultat pour l'exercice, l'exercice en cours, le langage de programmation utilisé, l'heure, etc.

Les données sont stockées grâce à un gestionnaire de version : git. Les gestionnaire de versions permettent assez logiquement de conserver différentes versions d'un code, et sont en général utilisé lors d'un projet informatique, afin de permettre à plusieurs personnes de collaborer sur un même projet. Cela s'adapte bien au données que nous souhaitons conserver : principalement, différentes versions de réponses à des exercices, ainsi que quelques données contextuelles. Celles-ci sont conservées sous forme de messages JSON.

La structure actuelle est la suivante : à chaque élève correspond une branche du git. Chaque événement correspond à un commit : le message du commit contient le JSON, et le code de l'élève est stocké normalement, comme le code du commit.

Le code est stocké sur un dépôt gitHub, sous le nom PLM-data (https ://github.com/BuggleInc/PLM-data).

Quelques points à noter sur ces traces, cependant. Premièrement, n'importe qui peut utiliser la PLM et publier ses traces. Ce qui en soit une bonne chose, puisque que cela augmente nos données, mais peut également les fausser. En effet, si l'on souhaite étudier le comportement de débutant en informatique, les traces d'un enseignant souhaitant tester le logiciel, par exemple, ne sont pas pertinentes.

Ensuite, un élève peut avoir plusieurs branches. Ce problème devrait se réduire avec la nouvelle version, mais si un élève ne s'identifie pas, ou change de compte, il n'est pas possible de relier ses branches.

Enfin, si les branches sont non nominales, cela ne garantit pas nécessairement que les données sont anonymes. On ne peut pas relier le nom de la branche a un compte d'élève, mais rien n'empêche le code lui-même de contenir des indications.

3

Deuxième partie

PAROURIR ET MANIPULER LES TRACES

4

PLM-REAPER

A l'heure actuelle, PLM-data a environ 1 800 branches, qui peuvent aisément avoir plus de 1 000 commits. Si le site de gitHub permet de parcourir le dépôt, il n'est pas adapté à une telle masse de données. Martin Quinson avait développé un programme permettant de récupérer des statistiques sur élèves, mais ce prototype n'a pas été conçu pour être réutilisé.

Il a donc fallu re-créer un programme capable de parcourir le dépôt git, et d'en extraire les données que l'on souhaite. Cette bibliothèques, que l'on a nommé PLM-repear, est disponible sur le compte BuggleInc sur gitHub.

Les messages de commit sont en format Json, et contiennent les informations utiles du commit. Le commit contient également les fichiers de l'élèves : son code, le message d'erreur le cas échéant, et quelques autres fichiers, comme un compte des exercices réussi dans la leçon en cours.

Plm-reaper permet de sélectionner les commits et branches que l'on recherche, selon leurs dates, les exercices traités, le langage utilisé,... Cela permet non seulement de faire un premier tri des données, mais également d'accélérer les performances en ignorant les branches non pertinentes.

La mise en place de cette bibliothèque fit face à quelques difficultés.

Problème classique, l'API fournit par git a une documentation clairsemé sur certaines fonctionnalité un peu avancé. Rien d'insurmontable, mais une perte de temps néanmoins.

La taille des données a traité fut également problématiques. Une première implémentation naïve consista à récupérer d'un bloc les données souhaitées, données qui dépassèrent la capacité de mémoire alloué à la jvm. Un itérateur et un traitement des données au fur et à mesure fut rapidement mis en place.

Gérer les différentes versions de la PLM et des traces correspondantes est également une contrainte. En particulier, un bug dans la version déployé le premier jour de la rentrée produisit des messages de commit érronés, qui faisait planter le reaper. Ce genre de problème est généralement aisé à contourner, mais ce cumule dans le code à chaque versions créant des traces bancales.

Troisième partie

ETAT DE L'ART

5

MODELING HOW STUDENTS LEARN TO PROGRAM

Dans un papier de 2012, Chris Piech propose d'étudier non le comportement des élèves sur une série d'exercice, mais à l'intérieur d'un exercice. L'idée est que le chemin emprunté par l'élève pour résoudre l'exercice serait un indicateur intéressant, entre autre pour révéler son niveau et ses chances de réussir les exercices suivants.

En utilisant l'IDE éclipse modifié afin de récupérer le code des élèves, il récupère une base de données proche de PLM-data. Il est alors possible de regrouper le code des élèves en différent bloc logiques, puis d'étudier les chemins des élèves au travers de ces solutions intermédiaires. Cela se fait en utilisant des modèles de markov caché, et pose un sous problème : mesurer des distances entre deux codes.

Ces idées ne s'adapte cependant pas tout à fait à notre but. Tout d'abord, il nécessite des exercices un minimum complexes, afin d'avoir plusieurs solutions possible, et des étapes intermédiaire pertinentes. Il est difficile avant d'avoir essayé de savoir quel exercice apporterait des éléments intéressants ou non, mais une grande partie des exercices de la première leçon sont simple, et donc probablement non pertinents pour cette méthode.

Par ailleurs, cette méthode demande d'étudier en particulier quelques exercices, ce qui n'est pas idéal pour un premier système, surtout si les-dits exercice se trouve assez loin dans la leçon. Cela peut par contre être un ajout pertinent, pour mieux déterminer par exemple si l'élève à une bonne compression logique des exercices plus complexes.

Dernier point, moins primordial mais qui pourrait diminuer l'efficacité de la méthode, les distances utilisées pour regrouper les codes des élèves. La plus efficace est celle qui se base sur les appel à l'API de l'exercice, ce qui, vu les exercices de la PLM, serait souvent inutiles. Il propose deux autres distances, mais qui sont moins précises.

6

ERROR QUOTIENT WATWIN ALGORITHM

Le "quotient d'erreur" est une valeur proposée par Matthew C. Jadud, dans "Methods and Tools for Exploring Novice Compilation Behaviour". L'idée est de comparer deux codes successif d'un élève, et de lui attribuer une "note", selon qu'il ait réussi à rendre son code sans erreur, ou au moins à changer d'erreur. L'idée étant qu'un élève réussissant à changer ou supprimer des erreurs progresse.

Christopher Watson propose une amélioration de ce quotient dans "Predicting Performance in an Introductory Programming Course by Logging and Analyzing Student Programming Behavior". Il affine l'analyse de l'erreur (est-ce le même message exactement, ou le même type d'erreur à un autre endroits), et intègre également un bonus/malus en fonction du temps entre deux compilation.

7

COARSE-GRAINED DETECTION OF STUDENT FRUSTRATION IN AN INTRODUCTORY PROGRAMMING COURSE

Dans ce papier, l'idée est d'essayer de relier la frustration des élèves à leur comportement de code. Pour cela, il ont observé quelques dizaines d'élèves lors de de sessions d'exercice, en notant les comportements dénonçant la frustration ou le renoncement des élèves. Ensuite, via une régression linéaire, il est possible de relier ces comportements à des comportements "de code". Par exemple, l'étude semble montrer un lien, assez logique, entre la frustration des élèves et le nombre de compilation successive avec la même erreur.

Cependant, reproduire l'expérience demande la participation d'un grand nombre de personnes afin d'observer les élèves, pour des résultats incertains. Quant à reprendre directement les résultats de l'étude, cela pose deux problèmes.Tout d'abord, si l'étude permet de penser que l'idée à du potentiel, elle n'arrive pas à fournir un résultat assez pertinent pour être réutilisé ; et le réutiliser dans un contexte différent change probablement trop de paramètre. La durée des sessions, longueur des exercices, type d'exercice étant différents.

8

LEARNING PROGRAM EMBEDDINGS TO PROPOAGATE FEEDBACK ON STUDENT CODE

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.