ALVES Vincent

FOURNIER Anthony

TESTIER Marc-Antoine

A l’intention de

Rapport de projet de parcours

Sommaire

[Introduction 3](#_Toc390333407)

[Etat de l’art 4](#_Toc390333408)

# Introduction

Avec le développement d’internet et des nouvelles technologies, le phénomène de plagiat s’est beaucoup développé, en particulier dans le milieu universitaire. De nombreux enseignants cherchent alors des moyens efficaces pour lutter contre le plagiat. Il existe plusieurs solutions.

Dans ce projet, nous étudierons les méthodes utilisant des logiciels de détection de plagiat pour les codes sources. Les outils pour détecter la similarité entre deux codes sources utilisent plusieurs techniques différentes :

- Compter le nombre de variables, nombre d’opérateurs uniques, …

- Comparaison de chaîne de caractères.

- Comparaison de la structure du code source.

Nous développerons un programme permettant la comparaison de deux codes sources, et qui donne un pourcentage de similitude. La méthode qui sera utilisé par ce programme est la comparaison de structures de deux codes sources.

# Etat de l’art

Le plagiat dans les codes sources informatiques est assez fréquent, et celui-ci demande d’utiliser des outils différents que ceux utiliser pour la comparaison de texte dans les documents. Un aspect différent du plagiat de code source est qu’il n’existe pas « d’usine à dissertations » que l’on peut trouver dans le plagiat classique. Puisque les devoirs de programmation attendent des étudiants d’écrire des programmes avec des exigences précises, il est difficile de trouver des programmes qui répondent déjà à ces exigences. Puisque intégrer un code externe est souvent plus difficile que de l’écrire à partir de zéro, la plupart des étudiants qui plagient choisissent de le faire avec les code-sources de leurs camarades.

Les algorithmes de détection de plagiat dans les codes sources peuvent se faire sur différents niveaux :

* On peut comparer le code par segment de texte, mais cet algorithme peut être contré en changeant les noms des variables et des fonctions.
* On peut parser le code, afin de retirer les espaces, les commentaires et d’ignorer le nom des variables et fonctions. On peut ainsi comparer les structures de code. Mais cet algorithme peut être contré en rajoutant des boucles ou des conditions inutiles.
* On peut aussi placer la structure dans des arbres et comparer les sous arbres des codes sources.
* On peut comparer les types d’affectations et de déclarations des variables et des fonctions mais la complexité du programme en devient beaucoup plus grande.
* On peut compter le nombre de boucles, conditions et affectations par segment et en déterminer un score selon la similarité de ces nombres par segment entre deux codes. Cependant cela peut entraîner des faux positifs car deux segments de code peuvent avoir les mêmes nombre de boucles, conditions et affectations.
* Un autre algorithme consiste à créer les arbres des suffixes des mots des codes et de les comparer.

Il existe déjà différents logiciels répondant à ce problème gratuits ou non tel que : MOSS, JPlag, The Sherlock Plagiarism Detector et beaucoup d’autres.

# Conception et implémentation