Atal Bihari Vajpayee - Indian Institute Of
Information Technology and Management,
Gwalior

B.Tech Compute Science and Engineering

Summer Project

SOURCE CODE PLAGIARISM DETECTION SYSTEM

Divyansh Falodiya
2019BCS-020

Under the supervision of
Dr. Santosh Singh Rathore

Contents

1 Abstract

Plagiarism, in its essence, is the representation of other's ideas and thoughts as one's own. In academic context, there are differing definitions of Plagiarism which is rather subjective. Plagiarism is considered to be ethically wrong, be it of any kind. It is considered as a violation of academic integrity and may lead to different kind of penalties. There can be serious consequences to plagiarism :

- As a student, it can lead to failure in the course or even cancellation of the degree.
- As an academic professional, it can lead to some serious legal issues including copyright infringement and can take down the professional's reputation.

From an academic perspective, plagiarism of source code files is considered as a form of academic dishonesty. Students usually try to submit assignments by copying from their peers or from the internet. Detecting plagiarism in source code files manually is often time consuming as there can be a lot of files involved.

This project aims at providing a solution to the 'Plague of Plagiarism' in the academic world so as to detect plagiarism among source code files.

2 Introduction

2.1 Plagiarism

Plagiarism, as we know, is the act of imitating or copying someone else's work and presenting it as your own. [2] It is very common in the academic environment where students usually copy assignments from their peers, modify it to conceal the fact that it is plagiarised and submit it as their own work. This can make it difficult for the instructor to identify plagiarised assignments manually.

From the perspective of academic professionals, there is no commonly agreed definition of what constitutes a source code plagiarism. For instance, the problem statement may require the use of a common algorithm, so every submission will have the same algorithm. So, surely the assignments will be similar but nothing can be said about it being plagiarised.

2.2 Background

A lot of research has been put into the field of source code plagiarism and a ton of tools have been developed that work quite well. One of the earliest tools developed was Plague. It converts the code into structure files and compares them using Heckel algorithm and it only supports C language. Another such tool is MOSS which uses a string matching algorithm to divide the code into n-grams, hash them and create fingerprints of the hashes using Winnowing algorithm and compare these fingerprints. Not only Plague and MOSS, but there are quite a handful of other tools as well such as JPlag, YAP, Sherlock, and many more, some of which are open source as well.

2.3 Motivation

Plagiarising code assignments constitutes a form of academic deception. Students usually plagiarise due to either laziness or the lack of academic ability. It is important that students try and do these assignments by themselves. Not only in academic institutions, plagiarism of source code is also found on online coding platforms and sometimes, we may even find that an entire software is plagiarised.

Plagiarisation of any academic material is considered to be an ethical offense and may lead the person to some serious consequences. In legal terms, it is considered as literary theft. The ethics of plagiarism is the ethics of stealing [1].

2.4 Problem Statement

Detecting plagiarism in source code files, be it two files or a hundred files, is a tedious task to do manually for any instructor. With this in mind, the question that comes up is "Can plagiarism be detected among different source code files?" or "Is it possible to automate the process of plagiarism detection in source code". This project aims to answer the above mentioned questions.

2.5 Objective

One of the major objectives of Plag is to provide a reliable solution to the academic institutions to detect plagiarism in the submitted assignments with a hassle-free process. Plag has been developed in the form of a CLI tool which, in general, is considered to be faster and more efficient as compared to other interface tools.

3 Literature Review

Plagiarising assignments in academic institutions is getting way too normalized now-a-days and in this era of online education, young students tend to get attracted towards plagiarising solutions as it seems to provide them an easy way to get things done. Several methods have been proposed for efficient identification of plagiarism after intensive research.

Deimos is the result of such a research which is described in "Automatic Source Code Plagiarism Detection" [4]. It works in the following steps : (a) Tokenization, (b) Comparison of tokens using RKR-GST algorithm.

Another approach described in "An AST-based Code Plagiarism Detection System" [5] makes use of abstract syntax trees to identify the similarity of the syntactic structure of the code. This approach is better suitable for plagiarism detection in repositories or programs that may not require the same algorithm for solutions and so the flow of program may be different.

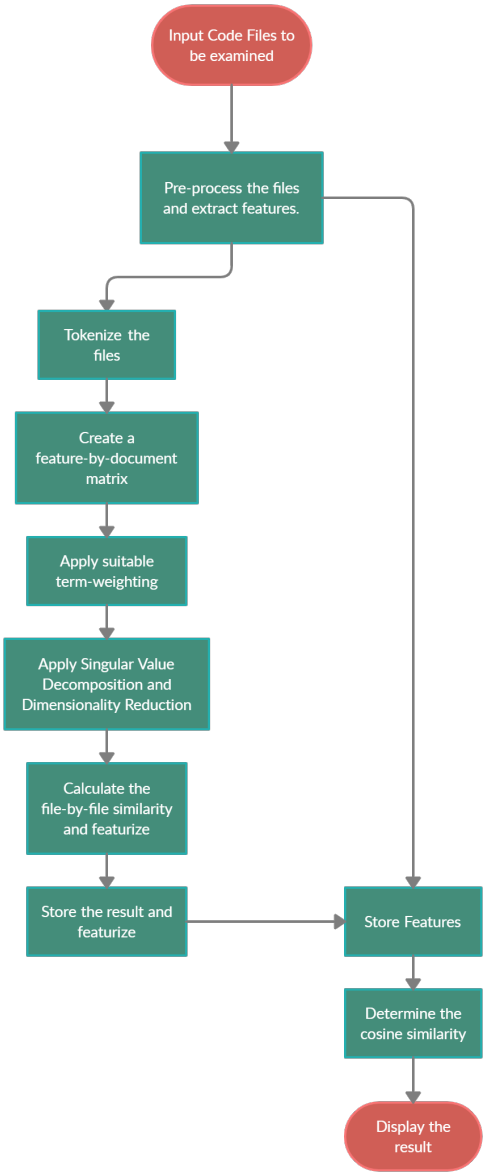
Yet another interesting approach is the one described in "Machine Learning for source-code plagiarism detection" [6] where the author approaches the problem of plagiarism detection in source code files using machine learning and deep learning and describes multiple methods to solve it include supervised and unsupervised learning algorithms.

This project, however, is heavily inspired from the thesis "An Approach to Source-Code Plagiarism Detection and Investigation Using Latent Semantic Analysis" [3] which approaches the problem using information retrieval techniques, specifi-

cally Latent Semantic Analysis, so as to find the similarity among different programs.

4 Methodology

4.1 Flow Chart



4.2 Implementation

Based on the fact that this tool needs to be used by instructors for detecting plagiarism, understanding the needs of the instructor is an important part in developing it. It is developed as a python package that works directly from the command line. To install the package on a system, cd into the root of the package [1] where setup.py is located and run the following command:

```
>> python setup.py install
```

This will install the package on the system and add the "plag" command to the PATH variable. The command to access the tool is "plag".

Some examples of using the "plag" command are:

- Display the version of the tool.

  ```
  >> plag --version
  ```

- Display information about the tool.

  ```
  >> plag --help
  ```

- Sets the process comment argument (-pc) as True and sets the path to the primary file (-p) as "c:/Users/divyansh/data/file1.cpp".

  ```
  >> plag -pc -p "c:/Users/divyansh/data/file1.cpp"
  ```

There are many more ways to customize the processing other than those described here.

It takes two inputs from the user : (a) the primary file that needs to be compared, (b) the folder that contains the files that need to be compared with the primary file.

The process that occurs under the hood is as follows:

1. All the files input into the system are preprocessed accordingly.
   - Features are extracted from the source code such as the number of loops, number of variables, etc.
   - Comments are removed from the file so that they don't interfere with similarity calculation.

[1] https://github.com/DivyanshFalodiya/plagiarism-detector

- The extracted comments are tokenized if the user intends to.
- The source code files are tokenized and any irrelevant tokens are removed.

2. A term-by-document matrix (TD Matrix) for the tokenized files and the tokenized comments (if required) is created.
3. The TNC [2] weighting scheme is applied to the code's TD Matrix.
4. If comments are to be processed, the TF-IDF [3] weighting scheme is applied to the comment's TD Matrix.
5. Singular Value Decomposition is applied to the code's TD Matrix and the dimensions of the resultant matrix are reduced suitably. Result = U, σ, V^T (T represents the transpose of the matrix).
6. The similarity matrix (S) is calculated for the code as well as the comments by $(V\sigma)(V\sigma)^T$. This matrix is a document-by-document matrix and so, the value S_{ij} represents the similarity between the i^{th} and the j^{th} document.
7. The features extracted during preprocessing and the similarity value calculated for the code and comments make up another feature vector for the document. Now, the simiarity is calculated using Cosine Similarity.
8. A detailed report indicating the feature values and the similarity is presented to the user.

5 What does the future hold?

Plag is CLI based tool which means that it operates from the command line interface. One of its restraint is that it is limited to the Windows operating system only, i.e., it is not cross-platform. So, one of the foremost feature to be added is support for different operating systems.

At present, Plag is limited to detecting plagiarism in C++ files only which is a bit of trouble because there are many languages the assignments can be submitted in. So, support for a bunch of programming languages is to be added.

Another upgrade is user customization. As of now, users do not have much control over the process of detection which is important for customizing the process according to the needs.

[2] TNC Weight = Term Frequency × Normal Weight of the document
[3] TF-IDF Weight = Term Frequency × Inverse Document Frequency

6 Conclusion

There has been a lot of research in the field of source code plagiarism detection and different kinds of solutions have been proposed from diving deep into the syntactical structure of the code to find out the similarity to matching code strings. The approach used in Plag is somewhere in the middle. Though, it does not dive into the syntactical structure, it tries to observe the semantics of the code, thus producing average results.

References

[1] Ethics of Plagiarism. https://www.plagramme.com/ethics-of-plagiarism.

[2] Plagiarism. https://en.wikipedia.org/wiki/Plagiarism.

[3] Georgina Cosma and Mike Joy. An approach to source-code plagiarism detection and investigation using latent semantic analysis. *IEEE Transactions on Computers*, 61(3):379–394, 2012.

[4] Bhrama Deokate and Dinesh Hanchate. Software source code plagiarism detection: A survey. *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, 1:3159–0040, 02 2016.

[5] Cynthia Kustanto and Inggriani Liem. Automatic source code plagiarism detection. In *2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing*, pages 481–486, 2009.

[6] Jitendra Yasaswi, Suresh Purini, and C.V. Jawahar. Plagiarism detection in programming assignments using deep features. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 652–657, 2017.