Final Year Project

**Software plagiarism detection system using machine learning approach**

**Lian Wang**

**2021**

• Chapter 1 Introduction

* + **–**Outline the problem you are working on, why it is interesting and what the challenges are.
  + my program started with
  + **–**List your aims and goals. An aim is something you intend to achieve (e.g., learn a new programming language and apply it in solving the problem), while a goal is something specific you expect to deliver (e.g., a working application with a particular set of features).
  + **–**Give an overview of how you carried out the project (e.g., an iterative approach).
  + **–**A brief overview of the rest of the chapters in the report (a guide to the reader of the overall

structure of the report).

This chapter is relatively short (2-4 pages) and must leave the reader very clear on what the project is about and what your goals are.

• Chapter 2 Context (or choose a chapter title that fits the material best)

This chapter should cover background information, related work, research done, and tools or software selected for use in the project.

* + **–**Provide necessary context and background information to describe how your project relates to what is already known or available.
  + Graphical user interface, text

    Description automatically generated
  + Text, letter

    Description automatically generated
  + **–**A description of the research carried out to learn out about the nature of the problem(s) being investigated and potential solutions. The form of the research will vary widely depending on the kind of project. For example, it might involve searching through research publications and online resources, or might involve an exploration of design possibilities for a user interface or program structure.
  + **–**The sources of information you are drawing on (papers, books, websites, etc.) should all be cited or referenced clearly. In addition, state how each source relates to your work and avoid

14

the temptation to pad out the chapter by including sources that you didn’t make use of during the project.

* + **–**If relevant, a survey of similar solutions, programs or applications to yours, and how yours is differentiated.
  + **–**Introduce the software, programming languages, library code, frameworks and other tools that you are using. Discuss choices and make clear which you made use of and why.

You should not include well known things (e.g., HTML or Java) or try to give tutorials on how to use a tool or code library (use references to books and websites for that information). Everything you include should be directly relevant to your work and the relationship made clear. This chapter is likely to be fairly substantial, perhaps 8-10 pages.

• Chapter 3 Requirements and Analysis

* + **–**Give the detailed problem statement. This elaborates on what you may have included in the introduction chapter, and represents the starting point from which requirements were derived.
  + **–**A structured list of requirements.
  + **–**Use cases (a use diagram and list of use case titles, with the full use cases appearing in the

appendix).

* + **–**Results of analysing the requirements to extract information. For example, data modelling to find the data to be stored (ER diagram), views/web pages needed and so on.

The level of detail of the requirements and use cases will depend on the nature of your project. If you are doing a Software Engineering based design and implementation project, then they will need to be done thoroughly. If there is a substantial body of requirements and use cases, then a summary should be given in the chapter, with the full set included in an appendix section.

If your project is not Software Engineering oriented, then you still need to describe the requirements you are working to and relevant analysis information. Use cases may not be needed or be relevant.

The analysis part of the chapter is what you did to map the requirements information into the first pass design. You can think of analysis as the first stage of design, and the purpose is to show how the requirements were used to inform the design.

The length of this chapter depends on the kind of project, but you are typically looking at 5-6 pages.

• Chapter 4 Design and Implementation

* + **–**Describe the design of what you have created.
  + **–**Start with the application architecture, giving its overall structure and the components that make up that structure.
  + **–**Give a description of the design of each of the the components that make up the architecture.
  + **–**Include the database or storage representation.

Diagram

Description automatically generated

Figure x. program schema

**–** Provide implementation details as necessary.  
As with other chapters, the structure and contents of this chapter will depend on the nature of

your project, so the list above is only a suggestion not a fixed requirement.

Find an ordering and form of words so that the design is clear, focusing on the interesting design decisions. For example, what were the alternatives, why select one particular solution? You have a limited number of pages so be selective about details. Also remember that someone (your examiners!) has to read this so don’t overwhelm them with intricate descriptions of everything that only you can follow – but do make sure the key details of the solution are in place. Use appropriate terminology and demonstrate that you have a good understanding of the Computer Science principles involved.

You can use diagrams and screen shots to help explain the design but don’t overuse them. Diagrams and screen shots should add information, not duplicate what is written in the text, and definitely avoid page after page of diagrams as this will disrupt the flow of your text. Where relevant, UML diagrams can certainly be used but, again, don’t flood the chapter with diagrams. Additional diagrams can always be included in an appendix section. It is not the case that a full set of UML diagrams must be provided for a software development project, and they shouldn’t be added in the belief that there must be UML diagrams to do a good project. Think about what you need to communicate and use UML diagrams if and when they fit the need.

It may be useful to include sections of code to highlight how a particular algorithm is implemented or how an interesting programming problem was solved. However, avoid lengthy sections of code, as this can also disrupt the flow of the text. Also make sure that your code fragments are readable, easy to follow and properly laid out. It may be better to use pseudo-code rather than actual code, especially when describing an algorithm. If you need to make use of longer sections of code, you can put the code in the appendix and reference it from the text.

An alternative way to organise the content of both this chapter and the preceding one, suitable for some projects, is to have a sequence of chapters or sections for each major iteration of the project. This allows the progression of the project to be shown, with each iteration building on the last, and the opportunity for interesting discussion about the decisions that needed to be made.

This is a core chapter in your report and will usually be quite substantial, 10 pages or more. • Chapter 5 Testing (and/or Results Evaluation)

* + **–**Describe your testing strategy (unit, functional, acceptance testing and how they are carried out). How were test cases selected?
  + **–**Examples of specific tests and how they were carried out (e.g., using mock objects to break dependencies). Focus on the interesting cases.
  + **–**A summary of the test results and what coverage was achieved. Detailed test reports should appear in the appendix, if they add useful information or you want to demonstrate the kinds of tests and coverage achieved.

If your project requires substantial evaluation of data and results, evaluation of algorithms, or other forms of testing that are not code-based, then adapt this chapter to suit.

16

This chapter will typically be 2-4 pages in length but could be more depending on the depth of testing done. If you need to do a detailed evaluation for a more mathematical or theory-based project, then this chapter could well be more substantial.

* + Chapter 6 Conclusions and Evaluation
    - **–**A summary of what the project has achieved. Make sure that you address each goal set out in the Introduction chapter, to show that you have achieved what you claimed you would. Don’t leave any loose ends.
    - **–**A critical evaluation of the results of the project (e.g., how well were the goals met, is the application fit for purpose, has good design and implementation practice been followed, was the right implementation technology chosen and so on).
    - **–**Future work. How could the project be developed if you had another 6 months. Take care to differentiate between what you have done to satisfy your stated project goals, and work that could be done to meet extended goals.
    - **–**Wrap-up and final thoughts on your project.  
      This chapter is typically 2-4 pages long but could be longer if the project work requires more

extensive evaluation.

* + List of references. Give publication details for all the items referred to by references you have made in main text of the report.
  + Bibliography. This lists all the sources of information that you made use of during the project but are *not* referenced in the text. The items in the list must be relevant to your project, so don’t just list everything you may have looked at or read.

**Background:**

**In the process of researching about the solution for software plagiarism, legacy software cannot fulfil the need of people. There are two type of software on the market: one analysis the semantics of the source code and compare the code to the resources. Another kind is to record the features of the file, for example the birthmark, and analysis those features. Both are not satisfied. The first one often lack accuracy that need human to examinate the result. The latter requires a high demand of right to access to those features that violate the privacy or refused by the users.**

**Machine learning is new trend on the market. Machine learning is a great tool in analysis the possibilities. The software plagiarism detection is a task about possibility.**

**Literature Review:**

**Moss:**

**Winnowing: Local Algorithms for Document Fingerprinting**

**Question:**

**What is used to be compared with in Moss?**

**Compare within the submissions**

**Compare with the web resource**

**What is the range/ how to choose the web resource used e.g. github?**

**What is hash function?**

**What is its function in the process of similarity check?**

**PIY:**

[Efficient clustering-based source code plagiarism detection using PIY](https://link-springer-com.libproxy.ucl.ac.uk/content/pdf/10.1007/s10115-014-0742-2.pdf)

* **Improved accuracy based on MOSS** 
  + **Tokenization**
  + **Token broken into k-grams**
  + **List of k-gram frequency**
  + **Treat frequency as vectors**
  + **Use vectors to compare similarity**
    - **Manhattan distance**
    - **Cosine distance**
      * **Correspond to the cosine of the angel btw the two vectors representing p(1st doc) and q(2nd doc) in n-dimensional space.**
      * **Divide the dot product of their magnitudes**

**Project goal:**

**By implement machine learning into the software plagiarism detection system, provide a solution that has higher accuracy of detections.**

**Requirements:**

* Must implement a text analysis function to analyse the semantic of the source code
* Must implement an unsupervised machine learning mode that generate an 70% prediction accuracy over training data and 60% over testing data

**Project plan:**

* **Find a database for software plagiarism**
  + [Can’t find a code database](https://ucl-new-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=UCL_LG_DSaz%2F33271858&context=L&vid=UCL_VU2&lang=en_US&search_scope=LSCOP_UCL_LG_DS&adaptor=Local%20Search%20Engine&tab=local&query=any,contains,code&offset=0) **in Moodle library**
  + **Does Turnitin has a** [API](https://help.turnitin.com/ithenticate/ithenticate-developer/api/api-guide.htm)**?**
  + **Using students’** [assignment/developers’ work](http://ijmlc.org/papers/50-A243.pdf)
  + **Use a simple program create by myself**
    - **Check ethics on moodle about FYP of github data.**
      * **What can we**
      * **How can use it**
    - **By changing the name of variables, create a plagiarism code for test use**
* **Build a machine learning model**
  + **Collect data**
    - **Where are the dataset in the papers coming from?**
      * [1](https://onlinelibrary-wiley-com.libproxy.ucl.ac.uk/doi/full/10.1002/cpe.5000#cpe5000-bib-0021)
* **Cleanse data**
* **Find a way to represent source code**
* **Choose a model**
* **evaluation**
* **Adjust the parameters**
* **prediction**

**Tasks break down:**

* **Read source file as input**
* **Extract features**
  + **Abstract syntax tree**
  + **Intermediate code generation**
* **Transfer features into matrix**
* **Use logistic regression model to classify source code**

**Weekly report:**

**References:**