**International Journal of Software Innovation**
Volume 9 • Issue 1 • January-March 2021

# Using Dynamic Time Warping to Detect Clones in Software Systems

Mostefai Abdelkader, Dr. Tahar Moulay University of Saida, Algeria

## ABSTRACT

Software clone detection is a widely researched area over the last two decades. Code clones are fragments of code judged similar by some metric of similarity. This paper proposes an approach for code clone detection using dynamic time warping technique (i.e., DTW). DTW is a well-known algorithm for aligning and measuring similarity of time series and it has been found effective in many domains where similarity plays an important role such as speech and gesture recognition. The proposed approach finds clones in three steps. First software modules are extracted. Then, the extracted modules are turned to time series. Finally, the time series are compared using the DTW algorithm to find clones. The results of the experiment conducted on a well-known Benchmark show that the approach can detect clones effectively in software systems.

## KEYWORDS

Clone Detection, Dynamic Time Warping, Effective, Module, Software, Time Series

## INTRODUCTION

Software clone detection is a widely researched area over the last two decades (Rattan et al. 2013; Saini et al. 2018; Wang et al. 2018; Farmahinifarahani et al. 2019). Code clones are generally the result of a copy-paste practice that consists of copying a code fragment from some place in a source code and then inserting it as -is or after making some modifications in another place in the source code. This approach is widely practiced by programmers to reuse and adapt existing code. Previous studies indicate that 9% to 17% of the source code of software systems are clones (Rieger et al. 2004). The studies show also that in some software 50% of the code is cloned (Zibran, 2011).

while cloning increases productivity of programmers, it leads to software code with many duplicated code fragments (i.e., code clones) which harm the quality of the produced software and increase the maintenance costs (Kapser & Godfrey, 2006). The maintenance activity is a costly activity in software engineering life cycle and the presence of clones will make it very costly. For example, if bug is found and fixed in some fragment, it is necessary to visit all its clones to fix the bug.

Therefore, it is important to detect clones in order to effectively manage them. A typical management process encapsulates activities that aim at detecting, avoiding and removing clones (Giesecke, 2007).

DOI: 10.4018/IJSI.2021010103

Despite that the literature proposed many approaches to detect clones in software systems, they were found ineffective and the clone detection problem still a challenging task (Bellon et al. 2007; Uddin et al.2011; Murakami et al. 2013; Wang et al. 2018; Farmahinifarahani et al. 2019).

Code clones are fragments of code judged similar by some metric of similarity (Baxter et al. 1998). While research communities proposed a variety of similarity metrics in different field, only a few of them are evaluated in detecting clones (Rattan et al. 2013).

DTW (Berndt & Clifford,1994) is a well-known algorithm for aligning and measuring similarity of data sequences. DTW has been found effective in many domains such as speech and gesture recognition (Berndt &Clifford,1994; Keogh &Pazzani, 2000; Bellman & Kalaba,1959; Salvador&Chan, 2004).

Thus, it is important to evaluate the performance of the DTW algorithm for clone detection in software systems.

This paper proposes an approach for code clone detection using dynamic time warping. The proposed approach is a process of three steps. The first activity extracts and pre-process all modules (i.e., methods, functions...) existing in the software system, the second activity turns modules to times series (i.e., data sequences) and the third one finds clone pairs using the DTW algorithm.

An experiment is done to assess the performance of the DTW algorithm in detecting clones type I, type II and type III. The results of the experiment are encouraging and show that the proposed approach has the potential to detect clones effectively.

The rest of the paper is organised as follows: section II presents general terms and definition. Section III presents the related work, section IV presents the concept of time series, section V introduces the DTW algorithm, section VI details the proposed approach, section VII presents the experimentation and the results, section VIII describes threats to validity and finally section IX conclude and outline the future work.

## RELATED WORKS

Many methods, techniques and tools have been proposed for software clone detection.

Lingxiao et al (2005) proposed an efficient algorithm for detecting similar subtrees from the tree representing the source code. First, The algorithm maps subtrees to numerical vectors. Then a clustering algorithm partitions these vectors into a set of clusters where each cluster contains similar subtrees. The experimentation done using the DECKARD tool showed that the approach is scalable and accurate.

Baxter & Yahin (1998) proposed an approach for detecting exact and near miss clones using abstract syntax tree. Three algorithms are used to find clones. the first algorithm, called the Basic algorithm, finds sub-tree clones. The second one, named the sequence detection algorithm, finds variable-size sequences of sub-tree clones. The third algorithm detects complex near-miss clones by generalizing the combinations of other clones.

Cordy & Chanchal (2011) developed the NICAD (Automated Detection of Near-Miss Intentional Clones) approach. The approach works as follow. First an activity called agile parsing is executed to remove noise, normalize and split program statements into parts in order to detect changes by line wise differences. Second, island grammars and agile parsing are used to extract candidate clones. third the Longest Common Subsequence (LCS) algorithm with a number of strategies are used to detect clone pairs and to accelerate the clone detection process.

Ducasse *et* al (1999) proposed an approach for detecting clones without parsing which make it language independent. The proposed process transforms source code to lines, pre-process lines by removing comments and whitespaces and detect clones by the string-based Dynamic Pattern Matching (DPM) algorithm. The process is accelerated using an optimization technique that uses has function for string.

Jens Krinke (Jens Krinke, 2001) presented the DUPLIX technique to detect similar code in programs. the proposed technique detects similar subgraphs in the program dependency graphs