I J C T A, 9(41), 2016, pp. 565-573 ISSN: 0974-5572
© International Science Press

Language Independent Code Clone Detection Approach Using JSON String Parsing

Gaurav Singh*, Sandeep Kaur** and Bhavneesh Sohal***

ABSTRACT

Cloning is an easy way of reusing the software. Some of the studies revealed that about 25- 30 per cent of code in long term software development project may have been cloned. Though there are some benefits of software cloning but still software cloning is considered as a harmful practice. If there is an error in one code that is to be cloned then that error can be transferred to all the modules where cloning has been done. Cloning also increases the maintenance cost. So, the clone detection becomes an important part for project success. In this paper, a new language independent approach is proposed for code clone detection which will work on JSON (Java Script Object Notation) string parsing. This approach will trace code clones for nearly all the languages.

Keywords: Clone Detection; JSON; Language Independent; Parsing; Plagiarism; Reuse

1. INTRODUCTION

"Software Cloning" is reusing source code of a program by copy and paste technique. The activity of cloning is known as code cloning and the duplicated code is known as the software clone. It encourages the software reuse and saves the programmer's time but it results in software management problems. Software cloning is considered as a harmful practice for industries. Cloning enhances the bug probability in a software program and increases the maintenance costs.

1.1. Types of Clones

1) Exact Clones: These types of clones are the exactly similar copies of the original code. The only difference is of whitespaces and the comments. The whole code is textually similar.

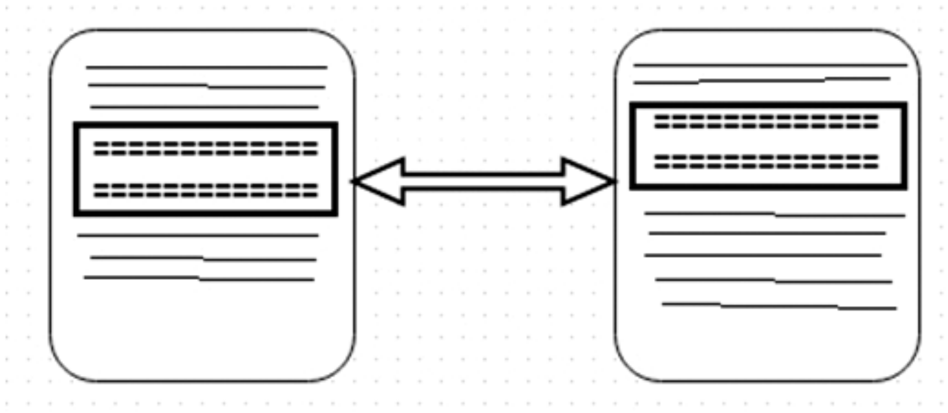

Figure 1: Example of code clones present in two files.

* Computer Science and Engineering Lovely Professional University Phagwara, India, *Email: gauravsingh0912@yahoo.in*

** Computer Science and Engineering Lovely Professional University Phagwara, India, *Email: sandeep.16827@lpu.co.in*

*** Computer Science and Engineering Lovely Professional University Phagwara, India, *Email: sohal.16042@lpu.co.in*

2) Syntactic Clones: These types are clones have syntax similarity with the original code. D. Rattan et al [1] states that the syntax of code is same but there may be slight variations in name of identifiers, literals, types, comments and layout. The reserved words and sentence structure is same as the original.
3) Near Miss Clones / Modified Clones: Project sections that have been duplicated with further alterations like proclamation insertions/cancellations with standing changes in identifiers, literals, sorts and formats as stated by D. Rattan [1]. These types of clones are hard to be detected.
4) Semantic Clones: Those code clones that have a semantic similarity with original code but the text may be different. By using meta analyses C.K Roy et al [2] states that Type 4 clone fragments perform same task but with different implementation. They may be developed by different programmers but may end up giving the same results as an output.

1.2. Disadvantages of Cloning

Software cloning is used by programmers widely but it has some adverse effects because of which it is considered as a harmful practice. Some of the harmful effects of cloning have been listed here below:

1) Bug propagation: There is a lot of probability of bug propagation in cloning. If there is some bug or error in piece of code and it is being cloned in a program then it spreads that bug everywhere in program.
2) Higher maintenance costs: Code clones eventually increase the after execution maintenance (preventive and adaptive) effort in software.
3) Resource depletion: More lines of code will lead to more resource depletion.
4) Negative impact on design: Code cloning disheartens the utilization of refactoring, inheritance and so on. It prompts terrible design rehearse concluded by D.Rattan[1].
5) Originality loss: The originality of the program gets lost after copying the content. Original content will remain less in program.

Because of all these negative effects, it is desirable to detect clones. The process of identifying similar clones is termed as clone detection.

1.3. Clone Detection Advantages

1) Bug Detection: The bug detection will be easier if copied clones are detected earlier using some clone detection tools.
2) Code Compression: The identified clones will be removed, so it will automatically save the space.
3) Originality Factor: The code will contain only genuine and original content written by the programmer himself rather than copied code.
4) Find Usage Patterns: If all the cloned fragments of specific source file can be detected, the functional usage patterns of that source file can be discovered.
5) Detection of Plagiarism: The plagiarism can be detected in the major projects and programs using clone detection tools.

1.4. Generic Clone Detection Process

Generic clone detection process refers to the sequence of tasks and operations which are performed by a clone detection technique. Though it is not necessary that every clone detection technique follows the same process and steps, it is a generalized process. A short description of the steps involved in the code clone detection process is given below.

a) Pre-processing: Before the actual clone detection process, code is preprocessed. The main motives of pre-processing are :

   • Removing uninterested code

   • Regulate source units

   • Define comparison units/granularity

b) Transformation: It involves the conversion of source code into an in-between illustration. Some main steps like tokenization, parsing, control and data flow analysis, removal of whitespaces, removal of comments, normalizing identifiers and structural transformations are performed in this step.

c) Match Detection: It involves comparison between transformed code segments using a suitable comparison algorithm.

d) Formatting: In this phase, the clone pair list obtained with respect to the transformed code is converted to a clone pair list with respect to the original code base. Normally, each location of the clone pair obtained from the previous phase is converted into line numbers on the original source files.

e) Filtering: The obtained clones are filtered manually or using some automatic method. In manual filtering, the false positive or spurious clones are separated by human experts; whereas in automatic heuristics, some parameters like length, frequency, diversity etc. are set according to which filtering takes place.

f) Aggregation: In order to reduce the amount of data, to perform subsequent analysis or to gather overview statistics, clones may be aggregated into clone classes.

1.5. Existing Clone Detection Tools

a) CC Finder X: Code clones (copied code parts) written in Java, VB, C#, C/C++, COBOL are identified by Aist CCFinderX [3].

b) Nicad: The NiCad Clone Detector is scalable tool which is used to detect near miss or modified clones. Nicad can also be integrated with various IDE's [4].

c) CloneDR: The CloneDR, assembled with the DMS / Software Reengineering Toolkit, finds exact as well as near miss clones in programming frameworks and can be utilized on a wide assortment of dialects [5].

d) Asta: Asta traces clones on the basis of structural abstraction. It finds the clones which are not identical. The method for an arrangement of clones may take distinctive parameters so as to speak to the majority of the clones in the set. These parameters may be structural as opposed to just lexical. Asta discovers these sorts of clones by searching for rehashed sub-trees in theory language structure tree of the project [6].

e) DuDe: DuDe is clone detection tool which is text-based and language-independent, gravitating around the concept of duplication chain. The tool is written in Java and runs on every major platform [7].

2. RELATED WORK

Ekwa Duala-Ekoko, Martin P. Robillard [8] in 2007 proposed a method of tracking clones during the evolution of software. Authors method depends upon the concept of abstract CRD (clone region descriptors). CRD are clone regions present inside the methods in a well-built way that does not depends on the exact text of that region of clone. Author also gives the definition and implementations of these CRD.

TungThanh Nguyen et al [9] in 2009 introduced a tool called ClemanX which is an incremental clone detection tool. Code fragments are represented in the form of sub-trees of Abstract Syntax Trees

(ASTs) and their similarity levels are calculated based upon their characteristic vectors of structural features. It performs the task of incrementally detecting similar code as an incremental distance based clustering problem.

Chanchal K. Roy [10] in 2009 developed a hybrid clone detection method and compares all the tools available for the clone detection. A mutation based framework is developed for comparison of tools which automatically measures precision of clone detection tools. The author discusses about his hybrid tool named as Nicad which is parser-based and language specific .But it is a light weight tool and detects the both exact and the near-miss clones with high accuracy and recall, and with rational performance.

DaqingHou et al [11] in 2009 worked with a motive to enlighten us about making further enhancements in cloning detecting tools rather than just their simple working. These further enhancements can be like comparing and contrast code clones, or to help edit (a group of) clones constantly and rapidly. The author tells about the main features that should be in clone managements system. The features like divergence of clones, editing the clone, capturing the clones and to visualize the clones.

Nam H. Pham, Hoan Anh Nguyen et al [12] in 2009 presented their code clone detection tool named as ModelCD . This tool was for Matlab models and can detect code full matched and approximate model clones very effectively. ModelCD worked on the graph based algorithm of clone detection. Authors had developed two different algorithms for exact and approximate clones. In their approach, simulink model was shown as a sparse, labeled directed graph. Clones in that model were considered as its weakly connected and non-overlapping sub graphs.

William S. Evans et al [13] in 2009 proposed a new algorithm for the detection of code clones. Their algorithm worked upon the abstract syntax tree. Authors had extended the previous work and made clone detection more efficient by detection clones even while the sub-trees have been changed. The name given for their tool was Asta. Asta took input of Abstract syntax tree in the form of XML string. Asta combined multiple ASTs and made a single XML. Further, Asta generated the matching patterns and found the code clones.

Liliane Barbour, Hau Yuan [14] in 2010 proposed a technique to increase the detection speed of incremental clones. Representative codes are selected from base code and string based approach is used to compare both. A client server architecture is introduced which detects clone information of a software system. In large organizations, all software codes are stored in a version control system, for example: concurrent version system (CVS), super version system (SVN).The work of version control system is to store all the updated source code, and track changes in the source code. Whenever a developer makes a change it checks the version control system instead of entire base code. Client accesses the clone list through serve, client monitors that which methods are being changed by programmer. Client sends the signature of method and path to server to get clone classes list for the current method.

Kodhai. E, Kanmani S [15] in 2010 proposed a new technique by combining metric-based approach and textual comparison of the source code. Proposed method was used for the detection of functional Clones in C source code.

Perumal. A , Kanmani S [16] in 2010 presented a technique to track same code snippets and for quantifying their similarity in the form of percentage. This technique can be used to trace clone clusters, set of code blocks all within the user supplied similarity values. This approach uses a measure like edit distance: two statements are assumed to be alike if they are within some threshold edit distance of each other.

Anna Corazza, Sergio Di [17] in 2010 proposed an approach to detect code clones, based on syntactic information combined with lexical elements. The main contribution is of Tree Kernels to find out similarity among (sub) trees.

A li S elam at and N w ah id [1 8] in 2010 proposed a technique which works on xml parsing for clone detection. Main idea is to combine detection based on structure information and detection based on instances.

Authors first converted the input documents into their xml code. After conversion, detection based on structural similarity is applied with the help of frequent sub graph miner technique.

Niko Schwarz [19] in 2012 proposed a technique to maintain the links between the repositories of software clones and outlines that how these links or paths can be created and maintained. In this paper, techniques have been discussed to primarily generate and manage such links. The working of the search engine is to assists the developer by adding its results into the source code. The IDE then saves the source of the code snippet and send a message to the depot that a clone was created. So in this way, the link is created between original and the copy and these types of clones are referred as hot clones.

Robert Tairas, Jeff Gray [20] in 2012 worked on unifying the code clone maintenance process by bridging the gap between clone detection and refactoring.

Saif Ur Rehman , Kamran Khan, Simon Fong [21] in 2012 proposed a clone detection technique which worked with most of the programming languages and reduced the big disadvantage of single language clone detection .This technique works on two dimensional array for faster clone detection. This technique improves programming time and reduces effort. While copying code we often forget to change the identifiers which produce new error in a code and it gets worse when the size of the code increases. The most challenging part is error in the code is reproduced in the copied code.

Antonio Cuomo et al [22] in 2014 presented their tool named CD-Form. This tool specifically detects Type 2 code clones for java language. The tool operates on java byte code. The byte code is transformed into CCS (Calculus of Communicating Systems) processes, which are successively checked for equivalence. After checking the equivalence, clone formatting is done and clone classes/pairs are calculated.

3. RESEARCH PROPOSAL AND METHODOLOGY

Code Clone Detection is of great concern for better maintenance and quality of a software system. Systems containing code clones are highly susceptible to bugs and defects and become hurdle for better evolution of software system. Hence, it is an open area of research from many years and results into various clone detection techniques and tools based on them. But as discussed in literature survey, certain limitations are associated with each clone detection technique and tool. A noble approach is needed which can detect code clones efficiently. So a new approach using JSON Parsing is proposed in this research work. JSON [23] is abbreviation of Java Script Object Notation. It is syntax for storing and exchanging data. By converting the code snippet into JSON format, we will try to detect code clones present in files. Main reason for using this approach is that it is language independent approach as JSON is a standard format, any language code can be converted to JSON string and by comparing those JSON strings, clones can be detected. JSON is very light weighted data-interchange format. So, we will be using this new and efficient approach for the purpose of code clone detection.

3.1. Dataset

This research work uses a new dataset of clone references. This dataset is a set of correct clones consisting of their locational information with their gapped lines. Bellon’s dataset is a commonly used clone dataset in existing clone detection research. Bellon’s dataset contains many clone references, thus the dataset is useful for comparing accuracies among clone detectors. However, Bellon’s dataset does not have locational information of gapped lines. Thus, Bellon’s benchmark does not evaluate some Type-3 clones correctly. In order to solve this problem, new dataset of clone references is being used instead of Bellon’s dataset.

3.2. Methodology

Proposed technique will be able to detect code clone of Type 1(Exact Clones), Type 2 (Syntactic Clones), Type 3 (Near Miss /Modified Clones). Based on the following steps, the clone detection takes place.

Table 1
Dataset

Project name	*Language*
weltab	C
cook	C
snns	C
postgresql	C
netbeans-javadoc	Java
eclipse-ant	Java
eclipse-jdtcore	Java
j2sdk1.4.0-javax-swing	Java

1. Select Valid Input Files of any type: Firstly the user will select the two valid files as input for the system. The code clone detection will be done in between these two files.
2. Conversion of Input code to JSON code: As our approach is language independent approach, so we will be converting input files code into the standard format that is JSON. For making all the types of input codes files like C,C++,Java, Html etc into one common standard format, we will convert these codes into JSON code. By converting both the file code into JSON code the whitespaces and comments will be removed that will bring code into structural similarity and the structural dissimilarities of the code will be removed. For converting the input code into JSON code we will be using a third party dynamic link library known as "NewtonSoft.JSON.Net20.dll"[24]. We will be importing this DLL file into our C# project for conversion of input code into the JSON code. This DLL file contains a library of functions and other information that can convert the input files into the JSON code. "JSONConvert.SerializeObject (inputcode,Format.Indented) " is the inbuilt function of "NewtonSoft.JSON.Net20.dll" file. This function will actually convert the input source file into the JSON code. We will read input code file line by line and pass the result as an input to above function for the conversion to JSON code. Format .Intended is the default format for JSON code.
3. String Matching for detection of Type 1: After we get the JSON code for the input files .We will be checking for the Type 1(exact clones) in the two files that are being converted into JSON. For the detection of Type 1, we will be comparing the two files with each other. One file's code is considered as one string and another file's code is considered as another string. Now these two strings are matched with each other for clone detection. "Equals()" function is used for this string matching like JSONString1.Equals (JSONString2) this will compare the values of the two strings and give the comparison results as the Boolean value.
4. JSON string's line by line matching for the detection of Type 2 and Type 3 clones: For the detection of Type 2, Type 3 clones, we will match the JSON code of both the files by line to line comparison. Every line of one code file is compared with each and every line of other code file. After that, clones are detected from both the files. For match detection in two codes, we will use Google's "Google Diff Match and Patch Library". The Diff Match and Patch libraries offer robust algorithms to perform the operations required for synchronizing plain text [25]. In our approach, we replace the plain text with JSON code in preceding method of Google.
   a. Diff [25]: Compare two blocks of plain text and efficiently return a list of differences. This implementation works on a character by character basis.
   b. Match [25]: Match searches for the pattern inside of a bigger content. This usage of match is fuzzy, which means it can discover a match regardless of the fact that the example contains

mistakes and doesn't precisely coordinate what is found in the content. This execution likewise acknowledges a normal area, close which the match ought to be found. The hopeful matches are scored in light of

- The quantity of spelling contrasts between the pattern and the content
- The separation distance between the applicant match and the normal location.
- The match separation parameter sets the relative significance of these two measurements.

c. Patch [25]: Two writings can be looked at against one another, producing a rundown of patches. These patches can then be connected against a third content. On the off chance that the third content has alters of its own, this adaptation of patch will apply its progressions on a best-effort premise, reporting which fixes succeeded and which failed..

5. Size Comparison of files: After the detection of Type 1, Type 2, Type 3 clones by pattern matching technique, we will compute the file size of the two JSON codes. Then these files are compared on the basis of their file sizes to check whether they are size clones or not.
6. Interpreting Results: After the comparison of JSON codes for the detection of Type 1, Type 2, Type 3 clones, we will be interpreting the results by calculating the percentage of code that is cloned .If the clones are of Type 1(exact clones) then the percentage of comparison will be 100% and both Type 2 and Type 3 clones cannot be present so we will ignore them. If the clones are not of Type 1 then we will look for Type 2 and Type 3 clones present in JSON codes. After calculating the percentage of code clones in input files, we will show results graphically.

3.3. Architecture of Proposed Methodology

Figure 2 shows the working architecture of the proposed system. Our proposed system will work according to these phases. System will detect Type 1, Type 2, Type 3 code clones according to given architecture and

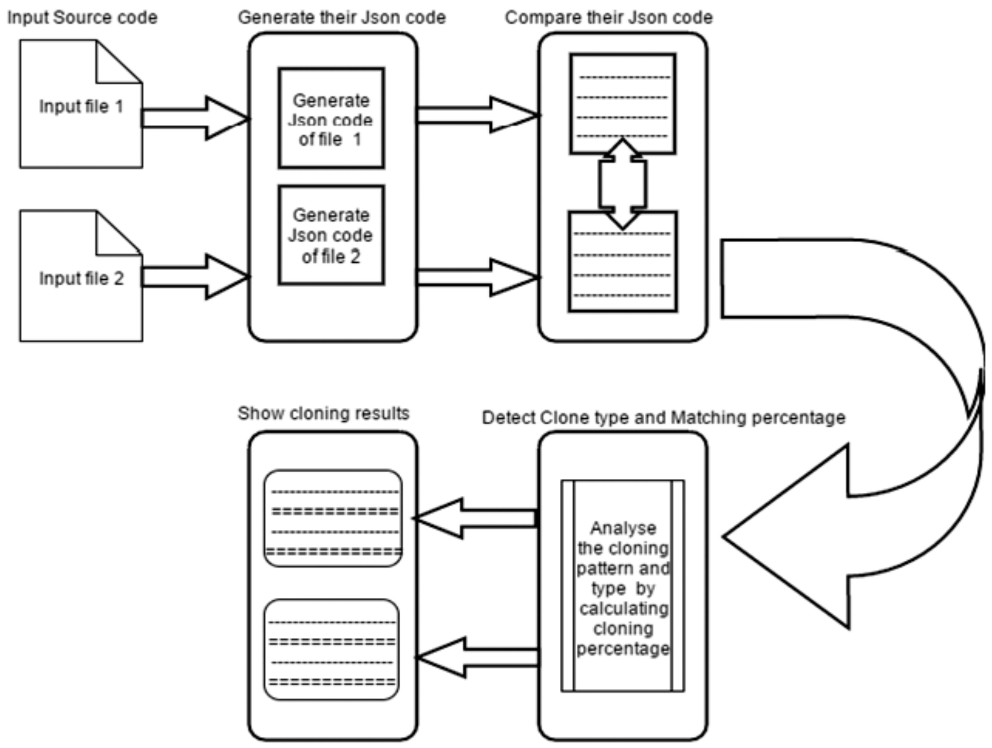
Input Source code
Generate their Json code
Compare their Json code
Input file 1
Input file 2
Generate Json code of file 1
Generate Json code of file 2
Show cloning results
Detect Clone type and Matching percentage
Analyse the cloning pattern and type by calculating cloning percentage

Figure 2: Architecture of Proposed Methodology

methodology. First of all, two input files will be taken for clone detection between them. Then the JSON code for those files will be generated. JSON code is generated to bring the input files into the same standard format for their comparison. First, the system will look for Type 1 clones and generate graphical results if Type 1 is present. If the clones are not of Type 1 then system will look for Type 2 and Type 3. After that system will generate graphical results by calculating the percentage of code cloning in files.

4. CONCLUSION

In this paper, a completely new approach of clone detection is proposed. Our technique uses JSON parsing for making this approach as language independent. In addition to this, Google's "Diff-Match-Patch" library is used for finding the code clones. JSON is very light weighted data interchange format for storing data. Proposed technique will be able to detect the code clones of Type-1(exact clones), Type-2(syntactic clones) and Type-3(near miss /modified clones) for all the languages.

REFERENCES

[1] Rattan D, Bhatia R, Singh M. Software clone detection: A systematic review. Information and Software Technology. 2013;55(7):1165-99.

[2] Roy CK, Cordy JR, Koschke R. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. Science of Computer Programming. 2009;74(7):470-95.

[3] The archive of CCFinder Official Site [internet] available at: http://www.ccfinder.net/ccfinderxos.html

[4] NiCad4 Clone Detector [internet] available at: http://www.txl.ca/nicaddownload.html

[5] Clone Doctor: Software Clone Detection and Reporting[internet] available at: http://www.semdesigns.com/products/clone/

[6] Clone Detection via Structural Abstraction[internet] available at: https://archive.org/details/Microsoft_Research_Video_151664

[7] Description of Dude[internet] available at: http://wettel.github.io/dude.html

[8] Duala-Ekoko E. and Martin P R. Tracking Code Clones in Evolving Software. Proceedings - International Conference on Software Engineering (2007): 158–167.

[9] Nguyen, Tung Thanh et al. Scalable and Incremental Clone Detection for Evolving Software.IEEE International Conference on Software Maintenance, ICSM (2009): 491–494.

[10] Roy Chanchal K. Detection and Analysis of near-Miss Software Clones. IEEE International Conference on Software Maintenance, ICSM (2009): 447–450.

[11] Hou Daqing, Ferosh Jacob, and Patricia Jablonski. Proactively Managing Copy-and-Paste Induced Code Clones. IEEE International Conference on Software Maintenance, ICSM (2009): 391–392.

[12] Pham, Nam H et al. Complete and Accurate Clone Detection in Graph - Based Models. ICSE (2009): 276–286.IEEE.

[13] Evans WS, Fraser CW, Ma F. Clone detection via structural abstraction. Software Quality Journal. 2009;17(4):309-30.

[14] Barbour Liliane, Hao Yuan, and Ying Zou. A Technique for Just-in-Time Clone Detection in Large Scale Systems.2010 IEEE 18th International Conference on Program Comprehension (2010): 76–79.

[15] Kodhai E, Kanmani S, Kamatchi A, Radhika R, Saranya BV. Detection of Type-1 and Type-2 Code Clones Using Textual Analysis and Metrics. 2010:241-3.

[16] Perumal, A., S. Kanmani, and E. Kodhai. Extracting the Similarity in Detected Software Clones Using Metrics.2010 International Conference on Computer and Communication Technology, ICCCT-2010 (2010): 575–579.

[17] Corazza Anna et al. A Tree Kernel Based Approach for Clone Detection.2010 IEEE International Conference on Software Maintenance (2010): 1–5.

[18] Ali Selamat and Norfaradilla Wahid (2010). Code Clone Detection Using String Based Tree Matching Technique, Semantic Web, Gang Wu (Ed.), ISBN: 978-953-7619-54-1,InTech.

[19] Schwarz, Niko. Hot Clones: Combining Search-Driven Development, Clone Management, and Code Provenance.Proceedings - International Conference on Software Engineering (2012): 1628–1629.

[20] Tairas, Robert, and Jeff Gray. Increasing Clone Maintenance Support by Unifying Clone Detection and Refactoring Activities.Information and Software Technology 54.12 (2012): 1297–1307.

[21] Rehman Saif Ur et al. An Efficient New Multi-Language Clone Detection Approach from Large Source Code. Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics (2012): 937–940.

[22] Cuomo A, Santone A, Villano U. CD-Form: A clone detector based on formal methods. Science of Computer Programming. 2014;95:390-405.

[23] Introducing JSON[internet] available at: http://www.json.org/

[24] Downloading Json Files [internet] available at: http://www.newtonsoft.com/JSON

[25] google-diff-match-patch[internet] available at: https://code.google.com/p/google-diff-match-patch/