Abstract: - A code clone is a portion of code that is similar to some another code portion residing in the source file. Code cloning is the result of copy-paste activities. There are number of techniques which have been proposed till now for the detection of clones, but still these occur in larger software systems. In this paper we describe a neural classifier with the help of which the accuracy of the detection of textual similarity of clones can be enhanced.

Keywords: - Code clone, Bug propagation, Taxonomy, Granularity, Acceptance rate, Rejection rate.

I. INTRODUCTION

In software engineering, the software cloning is the concept in which the copying and pasting of the code is widely done by making minor or no modifications in their text or functions. Due to which these codes becomes the almost mirror replicas of one another, which are said to be the code clones or cloning of the code. In software system code fragments mainly shows textual and functional similarities. They said to be textual similar if their program text matches and functional similar if behaviour among them matches. Although code cloning is a standard practice in modern programming, but this approach has increased the probability of bug propagation, increased the probability of introduction of new bugs, leads to bad design, and also adds maintenance cost. It has proved that the cloning has great adverse effects on the maintenance life cycle of system software. Therefore it is advised to remove the clones.

Clone Detection Techniques:

The clone detection techniques are mainly two phase processes, which consist of a transformation and a comparison phase. In the first phase the source code is transformed into internal format which allows the efficient use of comparison algorithm, by which the clones are detected. Due the central role, it is reasonable to classify the clone detection techniques according to their internal format [3], which are as follows:

- Token based clone detection technique
- Abstract Syntax Tree clone detection technique
- Program dependency Graph clone detection technique
- Metrics based clone detection technique[4]
- Hybrid clone detection technique[5]

II. TAXONOMY OF DETECTION TECHNIQUES

Every clone detection technique consists of various properties also known as dimensions, on the basis of which they are categorized [2]. For example: what it does, how it does etc. Various such properties are discussed as below.

Transformation of the source code: Also termed as normalization. Every approach applies certain type of transformation on the source code rather than using it as it is. Some approaches do some simple transformations as just removing white spaces or comments. While rather perform some complex transformation to get an alternative of the source code best suitable for the comparison algorithm.

Representation of the source code: Source representation is the alternative code obtained by transformation of the source code which is used in the comparison phase.

Comparison Granularity: By this property we mean that which type of clone granularity is used by the approach in the comparison phase. Different algorithms work on different type of code representations on different levels of granularity.

Comparison Algorithm: This property defines that which kind of comparison algorithms is used by a particular method. Different approaches or methods use different kinds of algorithms. For example: Biological Science purposes we use sequence matching algorithms. [3]

Computational Complexity: The technique should be scaled to detect the clones of larger software holding millions of lines of codes. The complexity of the technique depends on the transformation and comparison algorithm used.

Clone Similarity: This property defines that what kind of clone similarity the technique detects. Some techniques detect exact match, some detect near-miss clones, parameterized clones etc.
Clone Granularity: The clone granularity can be fixed or free. If the code clones has predefined syntactic boundaries then that are termed as fixed granularity clones. And the ones which does not have predefined boundaries means no limit on their structure and size, they are termed as free granularity clones.

Language independency: Now a day the software systems developed with various languages. A language independent tool can be applied to any system.

Output: This property implies that how the clones of the software system are returned as clone pairs or one clone classes or both.

III.RELATED WORK

In this section the work related to this research is discussed. The previous work has already used several types of mechanism, classifiers like support vector machine and linear discriminate analysis in the detection of the code clone.

Md. Sharif Uddin et.al [1] presents a standalone tool for clone detection known as Sim Cad. This tool used an algorithm which detects both exact and near-miss clones in the larger software systems. Most potential aspect of Sim Cad is that its clone detection function is made more portable by packing it into a library called SimLib.

Kodhai.E et.al [4] devise an algorithm which determine duplicated code from the program which is based on Halsted metrics. The main objective was to design and analyse a hybrid approach which is the blend of metrics based method with text-based techniques. In the textual comparison, line by line of code is compared rather than token or word.

Amandeep kaur et. al [5] uses a hybrid technique to detect type-1 and type-2 of clone and a tool is developed on the basis of this approach using a Java. The Textual comparison is performed to detect exact matched clone of Type-1 and Metric value computation for each interested method helps in detection of near miss clones of Type-2.

Florien D. et al [9] provides details on challenges and solutions for scalability and relevance of the results. Also present tool support for easy evaluation of clones in model based quality assurance.

Rowdy M. et. al [8] presents a clone detection technique to extract clones from object oriented source code using Differential File Comparison Algorithm to improve systems reusability and maintainability.

Though the results are good enough but with the increasing complexities of time, classifiers also has been introduced in the same contrast:

SVM: SVM stands for support vector machine. This send and receives the data only in the form of binaries values. The classification of this classifier is 70 to 80%. The problem with this classifier is that it is quite difficult to understand and even cannot be easily implemented on each and every platform.

LDA: LDA is linear discriminate analysis. This method is used in statistics, pattern recognition, document modelling, object categorization and machine learning to find a linear combination of features which separates two or more classes of objects. LDA discards word order means it incorporates the ordering information of data.

IV. PROPOSED WORK

As it has been already discussed that a lot of tools, techniques and classifiers has been already tried in this scenario of textual parameter code cloning detection but there are chances of improvement of the accuracy pattern of classification. The problem of this research work is to enhance the accuracy of the detection using NEURAL network classifier. For this purpose, BPA (back propagation neural network) would be used which involves the training and the testing pattern of the data.

A. OBJECTIVES:

- To reduce False Acceptance Rate of the classification
- To increase False Rejection Rate of the classification
- To enhance the accuracy of implementation.

B. OUTLINE OF THE ALGORITHM:

The real uniqueness of the network exists in the values of the weights between its neurons; a method is needed to adjust the weights to solve a particular problem. For this type of network the most common learning algorithm used is called back propagation (BP). Back propagation is the abbreviation of backward propagation of errors. This is the most common method of training the artificial neural networks. For a particular output value the network learns from many inputs. Since the BP network learns from examples, so we must provide a learning set that consists of some input examples with known correct outputs. Then these input and output examples are used to tell the network that what kind of network is expected and the BP algorithm allows network to adapt.
V. METHODOLOGY

The methodology is described by the following steps:

1) First of all the base content will be uploaded.
2) Then its clusters will be formed on the basis of the base content uploaded.
3) The content which has to be checked will be uploaded.
4) Both the clusters will be saved to the database.
5) To check the clone process, Back Propagation algorithm will be called.
6) The BPA algorithm would take the first input as the training set and the second input as the target set.
7) Number of hidden layers will also be defined in terms of calling the BPA algorithm.
8) The BPA algorithm will match the contents of base and target on the basis of weight formation which is inbuilt in the architecture of the neural network.
9) If the weight of both the contents are found equivalent or under a certain range which would be decided by the neural network itself, the uploaded content would be termed as clone else it would be termed as no clone.

A) FLOW CHART:

![Flow Chart Image]

B) PARAMETERS OF THE JUDGEMENT:

1) FAR (FALSE ACCEPTANCE RATE): The false acceptance rate is a term which is 1 if the data is not a clone of the base content and still the system terms it as a clone.
2) FRR (FALSE REJECTION RATE): The false rejection rate is a term which is 1 when if the data is a clone and the system does not identify it as a clone.

VI. CONCLUSION

The Clone detection is an active research area. Large number of researches has been done in clone detection tools, techniques and classifiers. The presence of clone effects less the normal functionality of system, but it will make the further development more expensive by creating design problems as inheritance or abstractions may be missed. In this paper a comprehensive survey is presented on the clone similarities, various techniques and their taxonomy and about the effect of neural classifier on the detection of clone. We hope that this will provide help for the future researches.

VII. ACKNOWLEDGMENT

I express great thanks to my guide for his valuable guidance and advice. I would also like to thanks to all the faculty members and colleagues of my college for their continuous support and encouragement.

REFERENCES

1) Md.S.Uddin, Chanchal K. Roy, et al. SimCad: An Extensible and faster clone detection tool for large scale software systems Dept.comp.sci.unv.of Saskatchewan, Saskatoon, Canada
6) Semantic Design et al., Clone Detection Using Abstract Syntax Trees 1998 IEEE.
7) Mitali, Dr. S. Garg, Analytical review on the techniques opted for detection of cloning in spreadsheets, in LIARCSSE vol 1, issue 9, nov, 2013
9) Florien Deissenboeck, Benjamin Hummel, Model Clone Detection in Practice IWSC’10, May 08, 2010, Cape Town, South Africa. Copyright 2010 ACM
16) M. Worlein, T. Meinl, I. Fischer, and M. Philippsen. A quantitative comparison of the subgraph miners MoFag Span, FFSM.
18) M. Zhang and V. Ciesielski, Using Back propagation algorithm & Genetic Algo. To train & refine neural networks or Object detection, by dept. of comp. sci. royal melbourne institute of technology, Australia.