ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

Integrating Descriptive Modelling with Case Based Reasoning in Network Intrusion Detection

By

Dawit Hassen Abtew

Advisor

Dr. Million Meshesha

A Thesis Submitted To the College of Natural Sciences of Addis Ababa University
In Partial Fulfillment Of The Requirements For The Degree Of Masters Of Science
In

Information Science

June 2015

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

Integrating Descriptive Modelling with Case Based Reasoning in Network Intrusion Detection

By

Dawit Hassen Abtew

June 2015

Addis Ababa, Ethiopia

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

Integrating Descriptive Modelling with Case Based Reasoning in Network Intrusion Detection

By

Dawit Hassen Abtew

June 2015

Name and Signature of Members of the Examining Board

_____	_____
Chair person, Examining Board	Signature
_____	_____
Advisor	Signature
_____	_____
Examiner	Signature
_____	_____
Examiner	Signature

DECLARATION

This thesis is my original work and has not been submitted as a partial requirement for a degree in any University.

_____

DAWIT HASSEN ABTEW

June 2015

The thesis has been submitted for examination with our approval as University advisors.

Dr. Million Meshesha _____ _____

DEDICATION

I would like to dedicate my thesis to my grandmother EMAHOY ABEBECH ASFAW, my father HASSEN ABTEW and my beloved sister MEIRAF HASSEN. Their spirit will be with me forever.

ACKNOWLEDGMENT

The first and the most I would like to Thank Almighty God and His mother St. Marry (Yegishena Emebite).

I would like to thank my advisor Dr. Million Meshesha for his constructive comments, encouragement, suggestions, patience and full devotions from beginning to end of my thesis work.

Special thanks to Ato Getachew J., Dr. Solomon, Dr. Martha, Dr. Dereje, Dr. Tibebh, Dr. Rahil, Ato Ermias, Meseret, Adey and to all staff member of School of Information Science.

I thank Dr. Beniyam, Kemal, Dr. Dereb and to all staff member of Department of Linguistics for the opportunity that helps me to engage in this program.

I would like to thank to all staff members of Addis Ababa University ICT office for their support on my thesis work.

I don't know how to thank you Dr. Tadesse Fethi, his belief in education reached me here and his struggling and fighting for my education since I was a little boy. Tade "Congratulations"; your dream is now realized!! And I thank his wife Zewdneshi Teshome.

My deepest appreciation goes to my Family: Kelewa Fethi represents all Ethiopian mothers who devoted themselves to the best of their children. She gave me strength on every of my journey. Love you mom and long live to you. I thank Babi, Alex, Heaven, Dagi, Mesfen, Fetlework, Asrat, Woyneshet, Tefi, Mesfen, Beniyam, Gashe, Abaye, Marti, Metu, Mesa, Melat (Docho), Atu, Gash Gedli, Webet, Roman, Dalul for their love and support.

I thank my friends Nosmas, Tame (Chainaw), China, Amanu (absubset), Meaziti, Honey, Merry, Helu, David, Hani, Abush, Estifo, Mimi Cake, Zerwa, Akmed, Tesfu, Nina, Seyum, Tg, Emu, Beza, Bini, Danda, Abdi and my Classmates for your knowledge sharing, experiences and making my time with you so funny, I really enjoyed the time we had together.

Last, but not least, I would like to thank Mahiber Kidusna and Holy Trinity Cathedral Sunday School for their support and prayers.

TABLE OF CONTENTS

_Toc423683053ACKNOWLEDGMENT.....I
TABLE OF CONTENTS..... II
LIST OF TABELS.....VI
LIST OF FIGURES VII
LIST OF ABBREVATIONS AND ACHONYMS..... VIII
ABSTRACT.....X
CHAPTER ONE.....1
INTRODUCTION 1
1.1 Background.....1
1.2 Statement of the Problem.....4
1.3 Objectives of the study.....6
1.3.1 General Objective 6
1.3.2 Specific Objectives 6
1.4 Scope and limitation of the study.....7
1.5 Significance of the study.....7
1.6 Methodology 8
1.6.1 Research method.....8
1.6.2 Data selection.....9
1.6.3 Data preprocessing.....9
1.6.4 Data Mining 10
1.6.5 Evaluation and Interpretation.....12
1.6.6 Use of Knowledge.....12
CHAPTER TWO 15
LITERATURE REVIEW 15
2.1 Network Intrusion Detection (NID).....15
2.2 Categories of Network Intrusion Detection 16
2.2.1 Network based intrusion detection.....16
2.2.2 Host based intrusion detection system (HIDS) 16
2.3 Types of Network Intrusion Detection.....17

2.3.1 Signature Based Detection 17
2.3.2 Anomaly-based detection 17
2.4 Components of Intrusion Detection System 18
2.5 Types of Attack 19
2.5.1 Remote to Login (R2L) 19
2.5.2 Denial of Service Attack (DOS) 19
2.5.3 User to Root (U2R) Attacks 19
2.5.4 Probe 19
2.6 Intrusion Detection Using Data Mining Techniques 20
2.7 Data Mining 21
2.8 Data Mining Tasks 22
2.9 Creating Descriptive model 24
2.9.1 Clustering Technique 24
2.10 Knowledge Based System 29
2.11 Case Based Reasoning Approaches 30
2.11.1 What is Case 30
2.11.2 Why use Case Based Reasoning 32
2.12 CBR Cycle 33
2.13 Designing a CBR System 34
2.13.1 Case Representation 34
2.13.2 Indexing 35
2.14 Case retrieval 35
2.15 Case Adaptation 38
2.16 Evaluation 38
2.16.1 Standard Metrics to Evaluate Intrusion 38
2.16.2 Precision and Recall 39
2.17 Tools 39
2.17.1 Data Preparator 39
2.17.2 CBR-Based Framework jCOLIBRI 40
2.17.2.1 Basic Components jCOLIBRI 41

2.17.3 Weka 44
2.18 Related works 46
CHAPTER THREE 49
METHODS AND APPROCHES 49
3.1 Architecture of the system 49
CHAPTER FOUR 55
KNOWLEDGE ACQUISITION USING DATA MINING 55
4.1 Knowledge Source 55
4.1.1 Sampling data set 56
4.1.2 Data Preprocessing and Selection 56
4.2 Data Reduction 57
4.3 Experimentation 58
4.3.1 Experimental Setup 58
4.3.2 Creating Descriptive model 60
4.3.3 Experiment 1- K-means clustering 60
4.3.4 Experiment 2- Make Density Based Clusterer 61
4.3.5 Experiment 3- Expectation Maximization (EM) 61
4.4 Evaluation of K-means Algorithm 62
CHAPTER FIVE 63
INTEGRATION OF CLUSTERING WITH CASE BASED REASONING 63
5.1 Building the Case Base 63
5.2 Case Representation 64
5.3 Managing The Case Structure in COLIBRI Studio 64
5.4 Case Attributes 66
5.4.1 Managing Connectors 67
5.5 Manage Tasks and Methods 68
5.5.1 Managing Tasks 68
5.5.2 Managing Methods 69
5.6 Automatic Integration of Clustering Model with Case Based System 69
5.7 Retrieval 74

5.7.1 Similarity function 75
5.7.2 Case selection 75
5.8 Reuse 76
5.9 Revise 77
5.10 Retain 78
CHAPTER SIX 80
IMPLEMENTATION AND EXPERIMENTATION 80
6.1 Network Attack Diagnosis 80
6.1.1 Define query 81
6.1.2 Retrieve Cases 82
6.1.3 Case Adaption 83
6.1.4 Case Revise 84
6.1.5 Case Retain 85
6.2 Testing and Evaluation of CBNIDS 86
6.2.1 System Performance Testing 86
6.2.2 User Acceptance Testing 88
CHAPTER SEVEN 92
CONCLUSION AND RECOMMENDATION 92
6.1 Conclusion 92
7.2 Recommendation 93
References 95
Appendix I 99
Appendix II 100
Appendix III 102
Appendix IV 104
Appendix V 105
Appendix VI 108
Appendix VII 110

LIST OF TABELS

Table 2.1 Standard metrics for evaluations of Intrusions (attacks) 38
Table 4.1 Distribution of different types of attacks and normal instances 57
Table 4.2 Distribution of attacks and normal with percentage of sample instance after data reduction 57
Table 4.3 Distribution of attacks and normal with percentage of resample instance 58
Table 4.4 Default parameters and values for algorithms 60
Table 4.5 Performance of cluster 61
Table 4.6 Evaluation of test set 62
Table 5.1 Managing the Case Structure in COLIBRI Studio 64
Table 6.2 Confusion matrix for evaluation of CBNIDS compared to experts' judgment 85
Table 6.2 Performance evaluation based on Precision, Recall and F-measure 87
Table 6.3 Result of users' acceptance evaluation 89

LIST OF FIGURES

Figure 1.1 An overview of the steps that compose the KDD Process 9
Figure 2.1 Data mining in the knowledge discovery process 21
Figure 2.2 The k-means algorithm.....25
Figure 2.3 The k- medoids algorithm 26
Figure 2.4 Architecture of a Knowledge Based System.....29
Figure 2.5 A CBR System 31
Figure 2.6 A CBR System 32
Figure 2.7 CBR cycle.....34
Figure 2.8 Case retrieval process involved in CBR.....36
Figure 2.9 Schema of the layered architecture of COLIBRI 41
Figure 2.10 UML diagram with main elements of jCOLIBRI 43
Figure 3.1 Architecture of the proposed system.....50
Figure 5.1 Configurations of case structures and similarities.....65
Figure 5.2 The jCOLIBRI connector architecture 66
Figure 5.3 Standard CBR Application interface.....69
Figure 5.4 Load the cases and memory organization 70
Figure 5.5 Java code for case description.....70
Figure 5.6 java code for case solution 71
Figure 5.7 Plain Text Connector.....72
Figure 5.8 Similarity function.....74
Figure 5.9 Java code for case retrieval.....75
Figure 5.10 Java code for case reuses.....76
Figure 5.11 Java code for case revises.....76
Figure 5.12 Java code for case retain.....77
Figure 5.13 Postcyle of the application.....78
Figure 6.1 Normal, Probe, DOS, U2R and R2L cases.....79
Figure 6.2 Define Query step.....80
Figure 6.3 Retrieved cases 81
Figure 6.4 Case adaptation step 82
Figure 6.5 Revise case for retain.....83
Figure 6.6 Case Retain.....84

LIST OF ABBREVATIONS AND ACHONYMS

CBNIDS: Case Based Network Intrusion Detection System

CBR: Case Based Reasoning

CBROnto: Case Based Reasoning Ontology

CRN: Case Retrieval Nets

DARPA: Defense Advanced Research Project Agency

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

DENCLUE: DENsity-based CLUstEring

DOS: Denial of Service

EM: Expectation-Maximization

GUI: Graphical User Interface

HIDS: Host Based Intrusion Detection System

ICT: Information Communication Technology

IDPS: Intrusion Detection and Prevention Systems

IDS: Intrusion Detection System

IE: Inference Engine

IPS: Intrusion Prevention System

KBS: Knowledge Based System

KDD: Knowledge Discovery in data base

K-NN: K-Nearest Neighbor

LAN: Local Area Network

MIT: Massachusetts Institute of Technology

NIDS: Network Based Intrusion Detection System

OPTICS: Ordering Points to Identify the Clustering Structure

OSI: Open Systems Interconnection

PSMs: Problem Solving Methods

R2L: Remote to User

RBR: Rule Based Reasoning

TCP: Transmission Control Protocol

U2R: User to Root

WEKA: Waikato Environment for Knowledge Analysis

ABSTRACT

Nowadays, because of the advanced use of the Internet, number of applications including but not limited to e-commerce, online shopping, Internet banking are virtualized which required more secured network. As these operations increases, computer crimes and attacks compromising the security and the trust of a computer system and causing costly financial losses become more frequent and dangerous. While a number of effective techniques exist for the prevention and detection of attacks, intrusion detection system has been considered the most promising methods for defending complex and dynamic intrusion behaviors. Intrusion detection systems gather information from a computer or network of computers and attempt to detect intruders or system abuse. Data Mining is extracting the information from the huge set of data. In other word, data mining is mining the knowledge from data. Knowledge based system is part of Artificial Intelligence. Knowledge based system has come across a variety of approaches based on the knowledge representation method; case-based reasoning (CBR) is one of the popular approach used in knowledge-based system. CBR is a problem solving strategy that uses previous cases to solve new problems.

In this study, an integration of data mining with case based reasoning approach is proposed for network intrusion detection. The system is aiming at utilizing hidden knowledge extracted by employing descriptive algorithm of data mining, specifically K-means cluster from sampled KDDcup'99 intrusion dataset. Clustered case with centroid value is mapped to COLIBRI Studio IDE and then the integrator application creates using Eclipse IDE with JDK 6. The important part of a case based reasoning model includes case retrieval; the similarity measuring stage, reuse; which allows domain expert to transfer retrieval case solution to suit for the current case, revise; to test the solution and retain to store the confirmed solution to the case base for future use.

The performance of the system is evaluated by preparing test cases. The system achieves 89.5% accuracy with an average Precision and Recall of 89% and 90%. For user acceptance testing users are trained and evaluated the system in which 95.5% score is obtained. This shows the system has registered a promising result. The similar cases are retrieved based on k-nearest neighbor similarity measure. So, further study has to be done to improve the retrieval process by applying ontology based retrieval.

Keywords: - Intrusion detection, data mining, case based intrusion detection, case based reasoning

CHAPTER ONE

INTRODUCTION

1.1 Background

Computer Security is the protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability and confidentiality of information system resources such as hardware, software, firmware, information/data, and telecommunications [1].

These days most organizations are dependent on the Internet to communicate with people and systems to provide news, online shopping, email, credit card detail and personal information. Due to the rapid growth in technology and widespread use of the Internet, a lot of problems have been faced to secure the system's critical information within or across the networks because there are millions of people attempting to attack on systems to extract critical information [2] [3].

To defend against various cyber-attacks and computer viruses, lots of computer security techniques have been intensively studied in the area of cryptography, firewalls, anomaly and intrusion detection. Among them, network intrusion detection has been considered to be one of the most promising methods for defending complex and dynamic intrusion behaviors [8]. As firewalls and anti-viruses are not enough to provide full protection to the system, organizations have to implement the Intrusion Detection System (IDS) to protect their critical information against various types of attacks [3].

Usman et al. [3], defines intrusion as, an attempted act of using computer system resources without privileges, causing incidental damage. Intrusion Detection means any mechanisms which detect the intrusive behavior. IDS monitor network traffic and its suspicious behavior against security. According to Sumit et al. [4], nowadays IDS perform signature-based monitoring of the cyber infrastructure to identify any malicious activities and generate an alert when such activity is detected. IDS monitor network traffic and its suspicious behavior against security. If it detect any threat then alerts to the system or network administrator about intrusion. IDS are a set of

techniques and methods that are used to detect suspicious activities both at the network and host level [3].

Intrusion Detection System can be classified into three categories [5]. The first is Host-Based IDS that evaluates information found on a single or multiple host systems, including contents of operating systems, system and application files. The second is Network-Based IDS, which evaluates information captured from network communications, analyzing the stream of packets traveling across the network. Packets are captured through a set of sensors. The final one is Vulnerability-Assessment IDS, which detects vulnerabilities on internal networks and firewalls [6]. According to Fayyad et al. [6], there are two primary models to analyze events and detect attacks: Misuse detection and Anomaly Detection Models. Misuse detection is an IDS that detect intrusions by looking for activity that corresponds to known signatures of intrusions or vulnerabilities. On the other hand Anomaly Detection Model is an IDS that detect intrusions by searching « abnormal » network traffic. Data mining based methods are another paradigm for building intrusion detection system. The main advantage of these methods is that they leverage the generalization ability of data mining method and in order to detect new and unknown attacks. A data mining based IDS uses machine learning and data mining algorithms on a large set of system audit data to build detection modes [8]. Hyt et al. [8], noted that intrusion detection techniques using data mining have attracted more and more interests in recent years.

Data mining is the task of discovering interesting patterns from large amounts of data, where the data can be stored in databases, data warehouses, or other information repositories [7]. According to Fayyad et al. [6], data mining is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Data mining has two major tasks predictive modelling and descriptive modelling. As an important application area of data mining, it meliorates the great burden of analyzing huge volumes of audit data and realizing performance optimization of detection. Many data mining systems are great in deriving useful statistics and patterns from huge amounts of data, but they are not very smart in interpreting these results, which is crucial for turning them into interesting, understandable and actionable knowledge [9].

A knowledge Based System (KBS) is a computer system which represents knowledge about a specific problem domain and this knowledge can solve the problems from the problem domain [2] [11]. KBS is a sub-field of Artificial Intelligence (AI) that works on knowledge base for effective decision making by imitating the behavior of human expert within a well-defined, narrow domain of knowledge [35]. KBS has come across a variety of approaches based on the knowledge representation methods and the reasoning strategies applied during implementation. Rule based reasoning (RBR) and case based reasoning (CBR) are two popular approaches used in knowledge based systems [12]. According to Prentzas [12], the main advantages of RBR are compact representation of general knowledge, naturalness of representation, modularity and provision of explanations. However RBR, has some drawback such as knowledge acquisition bottleneck, brittleness of rules, inference efficiency problems, difficulty in maintenance of large rule bases, problem solving experience is not exploited and interpretation problems.

A Case-based representation store a large set of previous cases with their solutions in the *case base* (or case library) and uses them whenever a similar new case has to be dealt with [13]. Kolodner [14] also mention that, CBR is a problem solving strategy that uses previous cases to solve new problems. When a new input case has to be dealt with, the case-based system performs inference in four phases known as the CBR cycle: Retrieve, Reuse, Revise and Retain [12]. According to Prentzas [12], the main advantages of case-based representations are; ability to express specialized knowledge, naturalness of representation, modularity, easy knowledge acquisition, self-updatability, handling unexpected or missing inputs and inference efficiency.

Hervé et al. [5] noted that, knowledge based intrusion detection techniques apply the knowledge accumulated about specific attacks and system vulnerabilities. The intrusion detection system contains information about these vulnerabilities and looks for attempts to exploit them. When such an attempt is detected, an alarm is raised. Advantages of the knowledge based approaches are very low false alarm rates, and that the contextual analysis proposed by the intrusion detection system is detailed, making it easier for the security officer using this intrusion detection system to understand the problem and to take preventive or corrective action.

1.2 Statement of the Problem

Intrusion detection identifies a set of malicious actions that compromise the integrity, confidentiality, and availability of information resources [15]. According to Fares et al. [2], intrusion detection system (IDS) has become a standard component in security infrastructures as they allow network administrators to detect policy violation. This policy violation ranges from external attackers trying to gain unauthorized access to insiders abusing their access. Intrusion detection techniques using data mining have attracted more and more interests in recent years and IDS aim to identify attacks with a high detection rate and a low false alarm rate. Current IDS has a number of significant challenges nowadays data overload and false alarms. Data mining can help improve the performance of intrusion detection [16].

Different researchers propose different algorithms for predictive modeling, such as Bayesian approaches [52], decision trees [17] [53] [54] rule based models [51] and Neural Network [55]. As noted by Zhong [44], classification based data mining models for intrusion detection is often ineffective in dealing with dynamic changes in intrusion patterns and characteristics. Unsupervised learning methods have been given a closer look for network intrusion detection. To this end, Zhong et al. [44], proposed clustering based network intrusion detection using k-means, Mixture-Of-Spherical Gaussians, Self-Organizing Map, and Neural-Gas algorithms. Comparisons with classification based intrusion detection methods on a completely new network traffic dataset demonstrated the usefulness and feasibility of clustering-based methods in both detecting new attack types. Considering the dynamic nature of network traffic intrusions, unsupervised intrusion detection is more appropriate for anomaly detection than classification-based intrusion detection method. So, there is a gap between the results of data mining system can provide and taking action based on them [20].

Both Tigabu [17] and Sumit [4], recommended an integration of data mining with knowledge based system to improve the performance of network intrusion detection. The integrating data mining with knowledge base system is essential to deploy the knowledge extracted from data mining model. A knowledge-based system, knowledge about real world phenomena is represented in a way that enables the system to utilize the knowledge for its reasoning tasks [18]. According to Hervé et al. [5], Knowledge based intrusion detection techniques apply the

knowledge accumulated about specific attacks and system vulnerabilities. There are a number of researches that have been done on knowledge based IDS [4] [5], but integrating data mining and knowledge based system are a new research area. Abdulkerim [20], attempted to integrate data mining with knowledge based system, He conducted data mining experiment using JRip decision tree algorithm and design the knowledge based by using rule based knowledge representation method. According to Prentzas [12], the rule based approach to the development of knowledge based system has drawback such as; it is not possible to draw conclusions from rules when there are missing values in the input data. For a specific rule, a certain number of condition values must be known in order to evaluate the logical function connecting its conditions. In addition, rules do not perform well in cases of unexpected input values or combinations of them.

A rule-based system is not self-updatable, in the sense that there is no inherent mechanism to incorporate experience acquired from dealing with past problems. Such experience could contribute decisively in the inference process, as it could assist in handling special cases or exceptions not expressed by rules. The problem-solving capabilities of a rule-based system cannot be enhanced during real-time operation, unless the expert himself/herself intervenes in order to update the contents of the rule-base, which is not desirable. Finally, the maintenance of rule bases is getting a difficult process as the size of the rule base increases. The rule base may have problems such as, redundant rules, conflicting rules, rules with redundant or missing conditions, missing rules, etc [12] [19].

On the other hand, a CBR approach has the ability to self-updatability which is knowledge in the form of new cases faced during real-time operation can be incorporated into the case base extending the effectiveness of the system [12]. This self-updatability also facilitates the maintenance of the case base. A case based system can handle unexpected cases not recorded in the system or missing input values by assessing their similarity to stored cases and reusing relevant cases. The self-updatability of the system enhances handling of unexpected cases. Adapting preexisting cases to handle new problems is usually more efficient than having to solve a problem from scratch as in rule based systems [12] [19].

This study therefore aims to explore a way to integrate a knowledge acquired using descriptive data mining model with case based for network intrusion detection system.

To this end, this study explores and finds answers to the following research questions:-

- How to integrate data mining with case based knowledge representation method?
- How it is possible to enhance self-updatability of a knowledge base system from knowledge extracted using data mining clustering techniques?
- To what extent does the network intrusion detection system, identify unknown attacks and minimize false alarm rate?

1.3 Objectives of the study

1.3.1 General Objective

The general objectives of this study is to integrate descriptive model constructed using data mining with knowledge based system in order to develop a case based network intrusion detection system.

1.3.2 Specific Objectives

The specific objectives of this study are the following:-

- To identify suitable data mining techniques for constructing descriptive model and evaluate its performance.
- To integrate descriptive model with case-based reasoning approach for intrusion detection.
- To update the knowledge base based on the new knowledge obtained from the given query.
- To evaluate the performance of a case-based reasoning intrusion detection system.

1.4 Scope and limitation of the study

The aim of this study is integrating data mining with case based reasoning approach for network intrusion detection. The study allows to automatically acquire hidden knowledge using intrusion dataset collected from KDDcup'99. The KDDcup '99 dataset was created by processing the tcpdump portions of the 1998 DARPA intrusion detection system evaluation dataset, created by Lincoln Lab under contract to DARPA. For knowledge acquisition based on data mining technique, a descriptive modeling specifically k-means clustering technique is used in order to identify the natural categories of instances of network intrusion. The acquired knowledge using data mining technique is mapped to COLIBRI Studio for developing case based network intrusion detection.

In addition, the study did not include real life network data due to shortage of time to process such type of data in making them read for mining task and representing on COLIBRI Studio. Additionally, there are 21 types of network attacks form the KDDcup'99 dataset which are grouped into four classes namely Probe, DOS, R2L and U2R. The system is designed to detect based on the four classes rather than considering all 21 attack types.

1.5 Significance of the study

The study would benefit several areas of the corporate system. The aim of this study is integrating data mining with case based reasoning approaches and to design knowledge based network intrusion detection system. Advantages of the knowledge based approaches are very low false alarm rates and that the contextual analysis proposed by the intrusion detection system is detailed, making it easier for the security officer using this intrusion detection system to understand the problem and to take preventive or corrective action.

The proposed system provides self-updatability network intrusion detection. So, the system helps network administrators to take action in accordance of the detected network attack. So, the study would be able to enhance intelligence of the knowledge based intrusion detection system. The research would be able to provide them the idea on the integral of data mining with case based reasoning who use the technology. Finally, it contributes for other researchers to undertake further study in this area.

1.6 Methodology

Methodology is the systematic, theoretical analysis of the methods applied to a field of study [10]. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge [10]. According to Kumar [61], Methodology is a collection of procedures, techniques, tools, and documentation aids which will help the system developers in their efforts to implement a new information system The methodology is the general research strategy that outlines the way in which research is to be undertaken and, among other things, identifies the methods to be used in it [10].

1.6.1 Research method

Method described in the methodology, defines the means or approaches of data collection, algorithm selection, tool and technique selection [10]. Kumar [10] also mention that, research method understands of all those methods/techniques that are used for conduction of research. This study is an experimental research and in this study Knowledge Discovery Databases (KDD) method is selected. In order to make knowledge extraction as much as correct as possible (i.e. in order to keep the correctness of the knowledge as it is kept at the source) different techniques could be applied [22]. According to Mihaela et al. [21], among these techniques, data mining techniques and, more general, KDD techniques became the most used in the recent years. KDD is the process of extracting and refining useful knowledge from large data.

According to Fayyad et al. [6], KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. KDD refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process. Data mining is the application of specific algorithms for extracting patterns from data. Steps in the KDD process, such as data preparation, data selection, data cleaning, incorporation of appropriate prior knowledge, and proper interpretation of the results of mining, is essential to ensure that useful knowledge is derived from the data [3]. According to Usama [3], data mining is a step in the KDD process that consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns (or models) over the data. KDD has been used by different researchers to discover knowledge from large collection of records. It has seven steps in which data mining is included as the fifth steps. The steps involved in knowledge discovery are depicted in the Figure 1.1

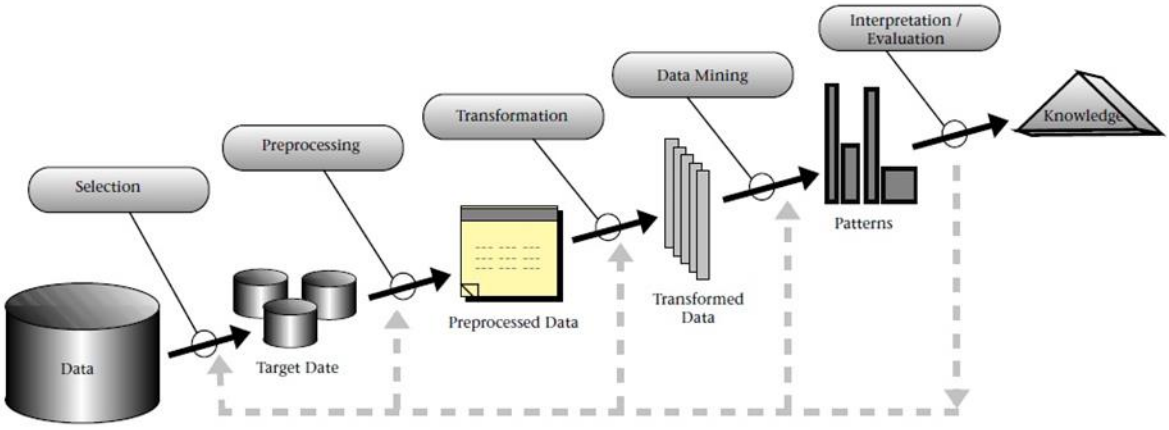


Figure 1.1 An overview of the steps that compose the KDD Process [32].

As illustrated in figure 1.1 the KDD process has seven steps. In an attempt to acquire knowledge using KDD, the following steps are included in this study; data selection, data preprocessing, data transformation, data mining, evaluation, and interpretation and use of knowledge.

1.6.2 Data selection

Data selection is where data relevant to the analysis task are retrieved from the data [33]. The KDD (Knowledge discovery in Database) Cup 1999 intrusion detection contest data (KDD cup 99 Intrusion detection data sets) has been used in this study. This data was prepared by the 1998 DARPA (Defense Advanced research Project Agency) Intrusion Detection Evaluation program by MIT Lincoln Labs (MIT Lincoln Laboratory). The KDDcup'99 data set from Lincoln lab available at ACM Knowledge Discovery web site [58]. Considering the time it takes to process the KDDcup'99 data set and coming up patterns, a sample size of 54, 111 instances are taken for this study.

1.6.3 Data preprocessing

The data set is preprocessed before the actual mining task is undertaken on the data set. According to Han et al. [7], data preprocessing is an important issue for data mining, as real-world data tend to be incomplete, noisy and inconsistent. Data preprocessing includes data cleaning, data integration, data transformation and data reduction.

As mentioned in above, the data set is prepared and is built based on the data captured in DARPA'98 IDS evaluation program. KDDcup'99 is about 4 gigabytes of compressed raw (binary) tcpdump data of 7 weeks of network traffic, which can be processed into about 5 million connection records, each with about 100 bytes. The KDDcup'99 data set is labeled with exact one specific attack type i.e., either normal or an attack. Deviations from 'normal behavior', everything that is not 'normal', are considered attacks. So, the KDDcup'99 data set labeled no need of data integration sub process of data preprocessing.

During the preprocessing task, instances are classified as normal or attack. Attacks were further classified into four sections represented into DOS (deny of service), Probe (information gathering), U2R (user to root) and U2L (remote to local).

DataPreparator tool is used for data preprocessing. DataPreparator is a free software tool designed to assist with common tasks of data preparation (or data preprocessing) in data analysis and data mining. DataPreparator can assist exploring and preparing data in various ways prior to data analysis or data mining. It includes operators for cleaning, discretization, numeration, scaling, attribute selection, missing values, outliers, statistics, visualization, balancing, sampling, row selection, and several other tasks. DataPreparator is written in Java and requires Java Runtime Environment (JRE) to be installed on your machine [49].

1.6.4 Data Mining

According to Han et al. [7], data mining has two major tasks: Predictive modelling and Descriptive modelling and there are three data mining techniques; Association, Classification and Cluster. In Association, the relationship of a particular item in a data transaction on other items in the same transaction is used to predict patterns. In Classification, the methods are intended for learning different functions that map each item of the selected data into one of a predefined set of classes. In Clustering techniques, consider data tuples as objects. They partition the objects into groups or clusters, so that objects within a cluster are similar to one another and dissimilar to objects in other clusters. Similarity is commonly defined in terms of how close the

objects are in space, based on a distance function [2] [7]. For this study descriptive modelling particularity clustering technique is applied.

According to Khubeb [59], clustering or cluster analysis as it is widely known is a focused type of data mining technique for large scale analysis of datasets. In this study, clustering technique is applied it is well known unsupervised data mining technique. It is particularly useful where there are many cases and no obvious natural grouping. Here, clustering data mining algorithm can be used to find whatever grouping may exist. A cluster is a collection of data objects that are similar in some sense to one another. A good clustering method produces high quality cluster to insure that the inter-cluster similarity is low and the intra-cluster similarity is high. In other words, members of a cluster are more like to each other than they are like members of different clusters.

For analyzing the data set three algorithms are selected. The first is k-means algorithm, which is one of the most well-known and commonly used partitioning methods. The algorithm attempts to determine k partitions that minimize the square-error function. The second is Expectation Maximization (EM) is selected for this study because it extends the k-means paradigm in a different way. Whereas the k-means algorithm assigns each object to a cluster, in EM each object is assigned to each cluster according to a weight representing its probability of membership. The final one is Density based methods, it is the partitioning method that cluster objects based on the distance between objects. Such methods can find only spherical-shaped clusters and encounter difficulty at discovering clusters of arbitrary shapes.

In order to get hidden knowledge from the data set and compare the performance of cluster, WEKA 3.7.9 is used. WEKA (Waikato Environment for Knowledge Analysis) is chosen since it is proven to be powerful for data mining and used by many researchers for mining task and the researcher is familiar with the tool. WEKA is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. WEKA is free software available under the GNU General Public License. The WEKA workbench contains a collection of visualization tools and algorithms for data analysis and descriptive modeling, together with

graphical user interfaces for easy access to this functionality. The software has Header section and Data section and Supported attributes numeric, nominal, string and date data type.

1.6.5 Evaluation and Interpretation

The descriptive model, which is cluster result about intrusions are evaluated by experts. In addition, the performance of cluster algorithms is compared and the one which performed best is selected for the knowledge acquisition step. The accuracy of the developed system is evaluated by using different techniques such as, Precision, Recall and F-measure. The KBS is evaluated based on system performance testing method and user acceptance.

1.6.6 Use of Knowledge

After data mining task has been done, the natural grouping of intrusions are extracted from sampled collection of intrusion data set and knowledge base is constructed as case base that contains case of network incident with their centroid value. So, the clustered instance is considers as case. A case can be said to be the record of a previous experience or problem [36]. The information recorded about this past experience will, by necessity, depend on the domain of the reasoner and the purpose to which the case will be put. According to Julie [36], the case base in the CBR system is the memory of all previous stored cases.

To design CBR system COLIBRI Studio IDE used in this study. The main design principle of COLIBRI is its layered structure. COLIBRI Studio is the implementation of the top layer of the COLIBRI platform. It provides the visual builder tools required to generate CBR systems without dealing directly with the source code [40]. COLIBRI Studio is integrated into the popular Eclipse IDE. This way it takes advantage of the facilities provided to manage projects and Java source code. It enables the generation of CBR projects where required libraries are automatically configured. These libraries conform a new release of the jCOLIBRI framework numbered as jCOLIBRI. COLIBRI Studio includes the following tools: a Case Designer that enables the definition of the structure of your cases; a tool to configure the storage of cases in

textual files; a tool to select the in-memory organization of the case; a tool to define the similarity between cases; and several tools to define the behavior of the system [40].

In addition, in order to develop an application which maps the knowledge acquired form the data mining clustering with case based reasoning Java Eclipse IDE 4.4.1 with JDK 6 is employed. Eclipse is a vast extendable set of tools for software development and Eclipse runs on multiple platforms including Windows, Linux, and Mac OS.

1.7 System development methodology

Prototyping approach is followed to develop the knowledge based system. Prototyping allows participating users and domain experts for evaluating system performance and efficiency.

1.8 Organization of the Study

This study consists of seven chapters. Chapter one discusses introduction about network intrusion detection, data mining, knowledge based system and case based reasoning. Statement of the problem, objective, significance, scope and methodology of the study is also included.

Chapter two defines some terms and gives overviews of concepts on security. A detailed discussion about data mining, network intrusion detection, types of network intrusion detection. This study is an integration of data mining and case based reasoning for network intrusion detection, so discussion about knowledge based system, case based reasoning approaches, how case based reasoning are represented to develop knowledge based system. In addition, related works in the area of network intrusion detection by using data mining techniques and case based reasoning approaches also included.

Chapter three discussed the design the architecture of the system designed, techniques, formulas and algorithms that are used for implementing the system. Chapter four presents the knowledge acquisition process. Focus on knowledge acquisition using data mining techniques. Detailed on knowledge discovery steps and the result of WEKA cluster algorithms are analyzed, interpreted and evaluated.

Chapter five discusses case base knowledge representation and also integration of data mining with case based reasoning approaches. For the integration of the system case representation, managing the case structure and connectors in COLIBRI Studio are detailed discussed. Chapter six discusses about the implementation of the system and the basic functionality of the system, performance evaluation and user acceptance testing.

Finally, chapter seven is about conclusion and recommendation. In this chapter based on the result obtained from this study the researcher's concluded and give recommendation for future work.

CHAPTER TWO

LITERATURE REVIEW

2.1 Network Intrusion Detection (NID)

Intrusion detection is the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices [25]. Incidents have many causes, such as malware (e.g., worms, spyware), attackers gaining unauthorized access to systems from the Internet, and authorized users of systems who misuse their privileges or attempt to gain additional privileges for which they are not authorized [25]. And also Intrusion detection is the act of detecting unwanted traffic on a network or a device.

An IDS can be a piece of installed software or a physical appliance that monitors network traffic in order to detect unwanted activity and events such as illegal and malicious traffic, traffic that violates security policy, and traffic that violates acceptable use policies[26]. Karen [25] also mentioned that, an intrusion detection system (IDS) is software that automates the intrusion detection process.

According to Shaik [27], Intrusion Detection provides the following services for the system and network administrator:-

- Monitor and analyze both user and system activities
- Analyze system configurations and vulnerabilities
- Access system and file integrity
- Ability to recognize patterns typical of attacks
- Analysis of abnormal activity pattern
- Track user policy violation

2.2 Categories of Network Intrusion Detection

2.2.1 Network based intrusion detection

A Network Based Intrusion Detection System (NIDS) is one common type of IDS that analyzes network traffic at all layers of the Open Systems Interconnection (OSI) model and makes decisions about the purpose of the traffic, analyzing for suspicious activity [26]. A network-based intrusion detection system monitors and analyzes the traffic on its network segment to detect intrusion attempts [27]. Network-based intrusion detection systems (NIDS) addressed two key issues with host-based intrusion detection systems. First, in order to analyze audit and system logs, the logs had to be forwarded to a dedicated processing system or a program had to be installed locally. Secondly, the analysis was limited to an individual system. This meant only those systems forwarding logs or having local software installed were protected [28]. There are three principal benefits of network-based intrusion detection systems (NIDS) approach [28]; first, misuse can be detected as it occurs instead of after. Second, misuse can be detected against any system on the network. Lastly, the system is passive; there is no requirement to install software on each system or forward logs. Network based intrusion detection system can monitor multiple systems. The network intrusion detection systems can detect some of the attacks that use the network [28].

2.2.2 Host based intrusion detection system (HIDS)

A host-based intrusion detection system (HIDS) is an intrusion detection system that monitors and analyzes the internals of a computing system as well as the network packets on its network interfaces [29]. The host-based IDS look for signs of intrusion on the local host system. These frequently use the host system's audit and logging mechanism as a source of information for analysis. It looks for unusual activity that is confined to the local host such as logins, improper file access, unapproved privilege escalation, or alterations on system privileges. HIDS comprises sensors that are located on servers or workstations to prevent attacks on a specific machine. A HIDS can see more than just network traffic and can make decisions based on local settings, settings specific to an operating system (OS), and log data [26]. Bilal [30] also mention that, a HIDS works with a software agent on a host. It is derived from mere log file analyzers. Modern host based Intrusion Detection Systems are designed as host based applications running in the

background of presumed critical, sensitive hosts, such as Mail Servers, Domain Name Service (DNS) Servers, web servers, database servers, etc. It identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability databases, Access control lists, etc.) and other host activities and state. Host-based systems tend to have lower false positive rates than do network-based systems. This happens because the ranges of commands executed on a specific host are much more focused than the types of traffic flowing across a network.

2.3 Types of Network Intrusion Detection

As noted by IATR [26], there are two types of network intrusion detection, such as signature-based detection and anomaly-based detection.

2.3.1 Signature Based Detection

IDS can use signature-based detection, relying on known traffic data to analyze potentially unwanted traffic [29]. It has advantages and disadvantages. One of the advantages of Signature-based approaches result in fewer false alarms because they can be very specific about what it is and they are looking for. Because the IDS is looking for something known, a lot of information regarding what the misuse is, the potential impact, and how to respond can be provided. This knowledge is extremely important in understanding what is occurring and effectively responding.

Signature based detection has also disadvantages, Signature-based approaches can only detect misuse for which a signature exists. For a signature to exist, the form of misuse must be known about beforehand so it can be researched and programmatically identified. This means any new form of misuse will not be detected by a signature based system until it is identified, analyzed, and then incorporated into the product. Depending on the circumstances, this could be hours, days, weeks, or even months.

2.3.2 Anomaly-based detection

An IDS that looks at network traffic and detects data that is incorrect, not valid, or generally abnormal is called anomaly-based detection [26]. Intrusions can be detected by observing

deviations from the expected behaviors of the system monitored. These "normal" behaviors can either correspond to some observations made in the past or to some forecasts made by various techniques. Everything that does not correspond to this "normal" pattern will be flagged as anomalous. Therefore, the core process of anomaly detection is not to learn what is anomalous but to learn what is normal or expected [27]. The key advantage of anomaly-based detection over the other two techniques is the ability to detect new attacks, attacks for which no signature or known protocol violation exists [28].

2.4 Components of Intrusion Detection System

According to IATR [26], Intrusion Detection Systems are generally made up of the following main types of components:-

- **Sensors—**these are deployed in a network or on a device to collect data. They take input from various sources, including network packets, log files, and system call traces. Input is collected, organized, and then forwarded to one or more analyzers.
- **Analyzers—**Analyzers in an IDS collect data forwarded by sensors and then determine if an intrusion has actually occurred. Output from the analyzers should include evidence supporting the intrusion report. The analyzers may also provide recommendations and guidance on mitigation steps.
- **User interface—**the user interface of the IDS provides the end user a view and way to interact with the system. Through the interface the user can control and configure the system. Many user interfaces can generate reports as well.
- **Honeypot—**in fully deployed IDS, some administrators may choose to install a "honeypot," essentially a system component set up as bait or decoy for intruders. Honeypots can be used as early warning systems of an attack, decoys from critical systems, and data collection sources for attack analyses. Many IDS vendors maintain honeypots for research purposes, and to develop new intrusion signatures. Note that a honeypot should only be deployed when the organization has the resources to maintain it. A honeypot left unmanaged may become a significant liability because attackers may use a compromised honeypot to attack other systems.

2.5 Types of Attack

Anand et al. [31] stated that, network attack can be categorized as Remote to Login (R2L), Denial of Service Attack (DOS), Probe and User to Root (U2R):-

2.5.1 Remote to Login (R2L)

A remote to user (R2L) attack is a class of attacks where an attacker sends packets to a machine over a network, then exploits machine's vulnerability to illegally gain local access as a user [31]. The goal of these attacks is to access some resources that your machine should not provide the attacker. Examples are Dictionary, FTP_write, Guest, Imap, Named, Phf, Sendmail, Xlock . [27].

2.5.2 Denial of Service Attack (DOS)

Denial of Service (DOS) is a class of attacks where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine [31]. Shaik [27] also mentioned that, DOS attacks are probably the nastiest, and most difficult to address. These are most horrible, because they are very easy to launch, difficult (sometimes impossible) to track, and it is not easy to refuse the requests of the attacker. Examples are Apache2, Back, Land, Mailbomb, SYN Flood, Ping of death, Process table, Smurf, Teardrop

2.5.3 User to Root (U2R) Attacks

User to root exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system. Most common exploits in this class of attacks are regular buffer overflows, which are caused by regular programming mistakes and environment assumptions Examples are Eject, Loadmodule, Ps, Xterm, Perl, Fdformat. [31].

2.5.4 Probe

Probing is a class of attacks where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits. Examples are Ipsweep, Mscan, Saint, Satan, Nmap. [31].

2.6 Intrusion Detection Using Data Mining Techniques

An intrusion detection system for a large complex network can typically generate thousands or millions of alarms per day. Beside on the detection an intrusion detection system has drawback. Current IDS has data overload and false negative drawback. Data mining can help to improve intrusion detection by addressing the above problem. A network intrusion detection system is expected to identify such attacks based on the pattern using data mining technique [2] [7] [32]. Several data mining and Machine Learning (ML) algorithm, including Neural Network, Fuzzy Logic, Support Vector Machine, and Data Mining and more have been extensively employed to detect intrusion activities both known and unknown from large quantity of complex and dynamic [17] [32] [51].

Data mining based intrusion detection system can be categorized into two major groups [5] [7] [8]: *misuse detection* and *anomaly detection*. In misuse detection, a model is trained with labeled data to recognize the patterns of "normal" and "intrusion" attempts. A classifier can then be derived to detect known intruions. Misuse detection has been shown to be very successful in detecting previously known attacks. However, since the misuse model is highly dependent on the labeled data used in the training stage, its capabilities of detecting new intrusion types is limited. Reseach in this area has included the application of classification algorithms, association rule mining, and cost sensitive modeling [4] [17] [15].

On the other hand, anomaly detection first establishes a model of normal system behaviors, and anomaly events are then distinguished based on this model. The implicit assumption is that any intrusive activity will be anomalous. Anomaly detection is able to detect newly emerging attacks, but it also has some drawbacks. It may fail to detect some known attacks if the behaviors of them are not significantly different from what is considered to be normal. Moreover, the false alarm rate tends to be high when the data of some normal system behaviors are not involved in the training phase. Anomaly detection research has included the application of classification algorithms, clustering algorithms, statistical approaches and outlier analysis [4] [17] [44] [53] [51].

2.7 Data Mining

Data Mining, also popularly known as Knowledge Discovery, refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data [32]. Data mining is a process that uses a variety of data analysis tools to discover patterns and relationships in data that may be used to make valid predictions [33]. Vivek [34] noted that, Data Mining is the process of extracting knowledge hidden from large volumes of raw data. The knowledge must be new, not obvious, and one must be able to use it. Figure (Fig. 2.1) shows data mining as a step in an iterative knowledge discovery process.

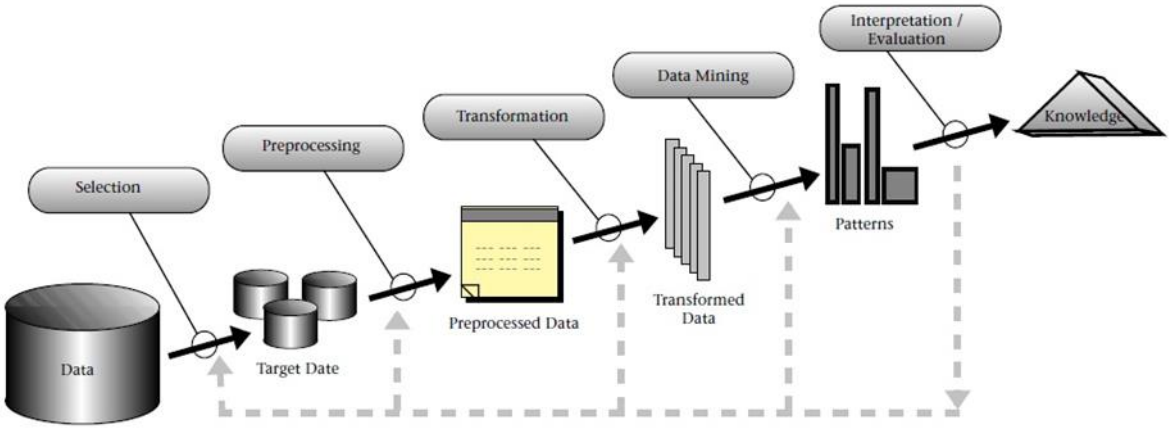


Figure 1.1 Data mining in the knowledge discovery process [32]

According to Osmar [32], the Knowledge Discovery process comprises of a few steps leading from raw data collections to some form of new knowledge. The iterative process consists of the following steps:

- **Data cleaning**: it is a phase in which noise data and irrelevant data are removed from the collection.
- **Data integration**: at this stage, multiple data sources, often heterogeneous, may be combined in a common source.
- **Data selection**: at this step, the data relevant to the analysis is decided on and retrieved from the data collection.
- **Data transformation**: also known as data consolidation, it is a phase in which the selected data is transformed into forms appropriate for the mining procedure.
- **Data mining**: it is the crucial step in which clever techniques are applied to extract patterns potentially useful.

- **Pattern evaluation**: in this step, strictly interesting patterns representing knowledge are identified based on given measures.
- **Knowledge representation**: is the final phase in which the discovered knowledge is visually represented to the user. This essential step uses visualization techniques to help users understand and interpret the data mining results.

Data mining is the task of discovering interesting patterns from large amounts of data, where the data can be stored in databases, data warehouses, or other information repositories [7]. Predictive modeling permits the value of one variable to be predicted from the known values of other variables. Classification, regression, prediction etc. are some examples of predictive modeling. The descriptive model identifies the patterns or relationships in data and explores the properties of the data examined. It describe all the data, it includes models for overall probability distribution of the data, partitioning of the p-dimensional space into groups and models describing the relationships between the variables. Clustering, association rule discovery, sequence discovery etc. are some of the examples [7].

2.8 Data Mining Tasks

According to Vivek [34], the various techniques of data mining are grouped in to three categories.

A. Association

Association and correlation is usually to find frequently used data items in the large data sets. It is the technique of finding patterns where one event is connected to another event. Association rule mining consists of first finding frequent item sets (set of items, such as A and B, satisfying a minimum support threshold, or percentage of the task-relevant tuples), from which strong association rules in the form of A to B are generated. These rules also satisfy a minimum confidence threshold (a prespecified probability of satisfying B under the condition that A is satisfied). Associations can be further analyzed to uncover correlation rules, which convey statistical correlations between item sets A and B. According to Jiawei [7], Association rule algorithms can be classified into three categories: (1) The Apriori algorithm is a seminal algorithm for mining frequent item sets for Boolean association rules. (2) Frequent pattern

growth (FP-growth) is a method of mining frequent itemsets without candidate generation. It constructs a highly compact data structure (an FP-tree) to compress the original transaction data. (3) Mining frequent itemsets using vertical data format is a method that transforms a given data set of transactions in the horizontal data format of TID-itemset into the vertical data format of item-TID set.

B. Clustering

Clustering can be said as identification of similar classes of objects. This is the technique of combining the transactions with similar behavior into one group, or the customers with same set of queries or transactions into one group. Osmar [32] mentioned that, clustering is the organization of data in classes. In clustering, class labels are unknown and it is up to the clustering algorithm to discover acceptable classes. Clustering is also called unsupervised classification, because the classification is not dictated by given class labels [7].

C. Classification

Classification and prediction are two main forms of data analysis that can be used to extract predictive models describing important data classes or to predict future data trends. Classification is a two-step process consisting of model construction and model usage. In the first step, a classifier is built describing a predetermined or labeled set of data classes. This is the learning step (or training phase), where a classification algorithm builds the classifier by analyzing or "learning form" a training set made up of database instances and their associated class labels. This step is called model construction [7].

Generally, classification is a process of building model that describe data class and used to predict the class of objects whose class label is unknown. It finds out the relationship between predictor value and the target value. The model is based on the analysis of a set of training data. The data for a classification is typically divided into two datasets: one for building the model; the other for testing the model [7] [32].

2.9 Creating Descriptive model

2.9.1 Clustering Technique

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. Clustering is a tool for data analysis, which solves classification problems. Its objective is to distribute cases (people, objects, events etc.) into groups, so that the degree of similarity can be strong between members of the same cluster and weak between members of different clusters [48]. In clustering, there is no preclassified data and no distinction between independent and dependent variables. Instead, clustering algorithms search for groups of records (the clusters composed of records similar to each other). The algorithms discover these similarities. A cluster of data objects can be treated collectively as one group and so may be considered as a form of data compression [7]. Clustering is also called data segmentation in some applications because clustering partitions large datasets into groups according to their similarity. Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases [7].

According to Jiawei [7], types of clustering methods are:-

1. **Partitioning Methods**

Given *D*, a data set of *n* objects, and *k*, the number of clusters to form, a partitioning algorithm organizes the objects into *k* partitions ($k \leq n$), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are "similar," whereas the objects of different clusters are "dissimilar" in terms of the dataset attributes. The most well-known and commonly used partitioning methods are *k-means* and *k-medoids*,

- **Centroid-Based Technique: The k-Means Method**

The *k*-means algorithm takes the input parameter, *k*, and partitions a set of *n* objects into *k* clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured in regard to the *mean* value of the objects in a cluster, which can be viewed as the cluster's *centroid* or *center of gravity*. The *k*-means algorithm proceeds as

follows. First, it randomly selects *k* of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges. Typically, the square-error criterion is used, defined as

$$E = \sum_{i=1}^{k} \sum_{p \varepsilon c_i} |p - m_i|^2 \qquad \text{Equation 1}$$

Where *E* is the sum of the square error for all objects in the data set; ***p*** is the point in space representing a given object; and m_i is the mean of cluster C_i (both ***p*** and m_i are multidimensional). The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster is given below in figure.

Input:

k: the number of clusters,

D: a data set containing n objects.

Output: A set of k clusters.

Method:

(1) arbitrarily choose k objects from D as the initial cluster centers;

(2) ***repeat***

(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;

(5) ***until*** *no change;*

Figure 2.2 the k-means algorithm

❖ **Representative Object-Based Technique: The k-Medoids Method**

The k-means algorithm is sensitive to outliers because an object with an extremely large value may substantially distort the distribution of data. This effect is particularly exacerbated due to the use of the square-error function. Instead of taking the mean value of the objects in a cluster as a reference point, pick actual objects to represent the clusters, using one representative object per cluster. Each remaining object is clustered with the representative object to which it is the most similar. The partitioning method is then performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. That is, an absolute-error criterion is used, defined as

$$E = \sum_{j=1}^{k} \sum_{p \varepsilon c_j} |p - o_j| \qquad \text{Equation 2}$$

Where E is the sum of the absolute error for all objects in the data set; p is the point in space representing a given object in cluster C_j; and o_j is the representative object of C_j. In general, the algorithm iterates until, eventually, each representative object is actually the medoid, or most centrally located object, of its cluster. This is the basis of the k-medoids method for grouping n objects into k clusters. The *k*- medoids algorithm, PAM, a *k*-medoids algorithm for partitioning based on medoid or central objects.

Input: *k: the number of clusters,*

D: a data set containing n objects.

Output: A set of k clusters.

Method:

(1) arbitrarily choose *k* objects in *D* as the initial representative objects or seeds;

(2) ***repeat***

(3) *assign each remaining object to the cluster with the nearest representative object;*

(4) *randomly select a non-representative object,* $\boldsymbol{o}_{random}$*;*

(5) compute the total cost, *S*, of swapping representative object, $\boldsymbol{o_j}$, with $\boldsymbol{o}_{random}$;

(6) if $S < 0$ then swap $\boldsymbol{o_j}$ with $\boldsymbol{o}_{random}$ to form the new set of *k* representative objects;

(7) ***until*** *no change;*

Figure 2.2 The *k*- medoids algorithm

The k-medoids method is more robust than k-means in the presence of noise and outliers, because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the k-means method. Both methods require the user to specify k, the number of clusters.

2. Hierarchical method

A hierarchical clustering method works by grouping data objects into a tree of clusters. Hierarchical clustering methods can be further classified as either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) fashion. The quality of a pure hierarchical clustering method suffers from its inability to perform adjustment once a merge or split decision has been executed. That is, if a particular merge or split decision later turns out to have been a poor choice, the method cannot backtrack and correct it. There are two types of hierarchical clustering methods:-

- **Agglomerative Hierarchical Clustering**

Agglomerative hierarchical clustering is this bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category. They differ only in their definition of intercluster similarity.

- **Divisive Hierarchical Clustering**

This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the diameter of each cluster is within a certain threshold.

3. Density based methods

To discover clusters with arbitrary shape, density-based clustering methods have been developed. These typically regard clusters as dense regions of objects in the data space that are separated by regions of low density (representing noise). DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is grows clusters according to a density-based connectivity analysis. OPTICS (Ordering Points to Identify the Clustering Structure) extends DBSCAN to produce a cluster ordering obtained from a wide range of parameter settings. DENCLUE (DENsity-based CLUstEring) clusters objects based on a set of density distribution functions.

4. Grid-based method

The grid-based clustering approach uses a multiresolution grid data structure. It quantizes the object space into a finite number of cells that form a grid structure on which all of the operations for clustering are performed. The main advantage of the approach is its fast processing time, which is typically independent of the number of data objects, yet dependent on only the number of cells in each dimension in the quantized space.

5. Model-based method

Model-based clustering methods attempt to optimize the fit between the given data and some mathematical model. Such methods are often based on the assumption that the data are generated by a mixture of underlying probability distributions. There are two types of Model-based clustering methods:-

- **Expectation-Maximization**

The EM (Expectation-Maximization) algorithm is a popular iterative refinement algorithm that can be used for finding the parameter estimates. It can be viewed as an extension of the k-means paradigm, which assigns an object to the cluster with which it is most similar, based on the cluster mean.

❖ **Conceptual Clustering**

Conceptual clustering is a form of clustering in machine learning that, given a set of unlabeled objects, produces a classification scheme over the objects. Conceptual clustering goes one step further by also finding characteristic descriptions for each group, where each group represents a concept or class. Hence, conceptual clustering is a two-step process: clustering is performed first, followed by characterization.

2.10 Knowledge Based System

A KBS is a computer based system, which uses and generates knowledge from data, information and knowledge. This system is capable of understanding the information under process and can take decision based on the residing information/knowledge in the system whereas the traditional computer systems do not know or understand the data/information they process [35]. Figure 2.2 shows the architecture of a knowledge based system.

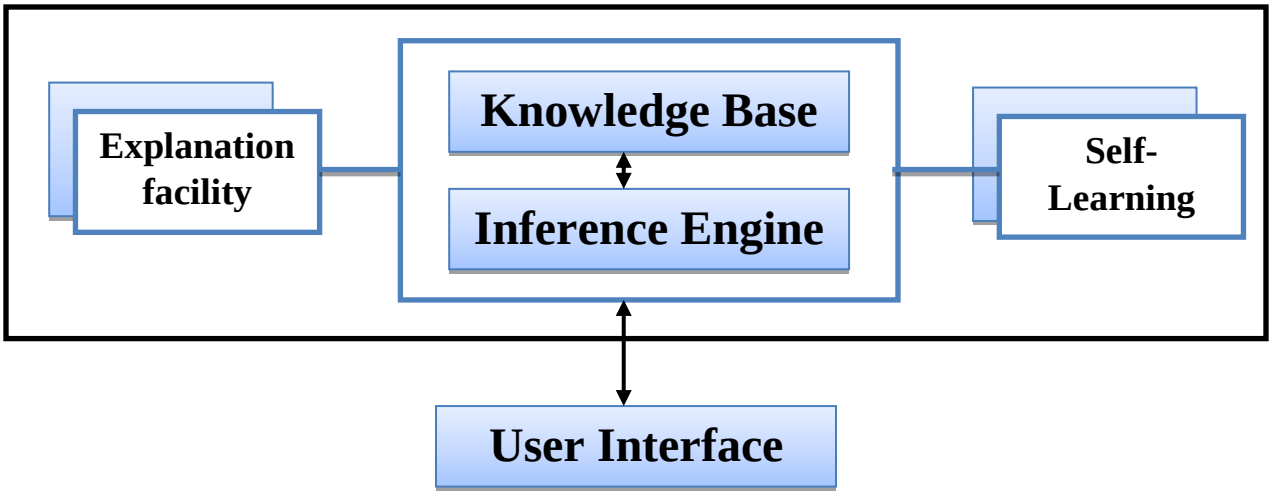


Figure 2.4 Architecture of a Knowledge Based System [35]

On the knowledge based system, the inference engine is usually set up to mimic the reasoning, or problem solving ability, that the human expert would use to arrive at a conclusion. The inference engine simulates the evaluation process of relating the information and rules in the knowledge base to the answers to a series of questions given by the operator [45].

A KBS tool is a set of software instructions and utilities taken to be a software package designed to assist the development of knowledge-based systems. Personal computers, typical programming languages like java and framework like .NET can also be used in KBS development. These programming languages are general purpose and also being used to develop other application than AI applications [35].

Knowledge-based systems applications are divided into two broad categories namely: (i) pure knowledge-based systems applications and (ii) applied knowledge-based systems application. Pure applications include research contributing in knowledge-based systems and AI development techniques such as knowledge acquisition, knowledge representation, models of automated knowledge-based systems development (knowledge engineering approaches, models and CASE tools for KBS), knowledge discovery and knowledge management types of tools [35]. The KBS consists of a Knowledge Base and a search program called Inference Engine (IE). The IE is a software program, which infers the knowledge available in the knowledge base. The knowledge base can be used as a repository of knowledge in various forms [35].

2.11 Case Based Reasoning Approaches

2.11.1 What is Case?

A case can be said to be the record of a previous experience or problem [36]. The information recorded about this past experience will, by necessity, depend on the domain of the reasoner and the purpose to which the case will be put.

According to Julie [36], the case base in the CBR system is the memory of all previous stored cases. There are three general areas that have to be considered when creating a case base.

- The structure and representation of the cases themselves
- The memory model used for organizing the entire case base
- The selection of indices which are used to identify each case

Case-based reasoning is a methodology for solving problems by utilizing previous experiences. It involves retaining a memory of previous problems and their solutions and, by referencing these, solves new problems [36]. According to Agnar [37], Case-based reasoning is a recent approach to knowledge-based problem solving and decision making: A new problem is solved

by remembering a previous similar situation and by reusing information and knowledge of that situation. Kolodner [38] mention that, case-based reasoning (CBR) is a problem solving strategy that uses previous cases to solve new problems. A case-based reasoning system can be viewed as a black box (see Fig. 2.4) that incorporates the reasoning mechanism and the following external facets:-

- The input specification or problem case
- The output that defines a suggested solution to the problem
- The memory of past cases, the case base, that are referenced by the reasoning mechanism Figure 2.5 depicts a typical CBR

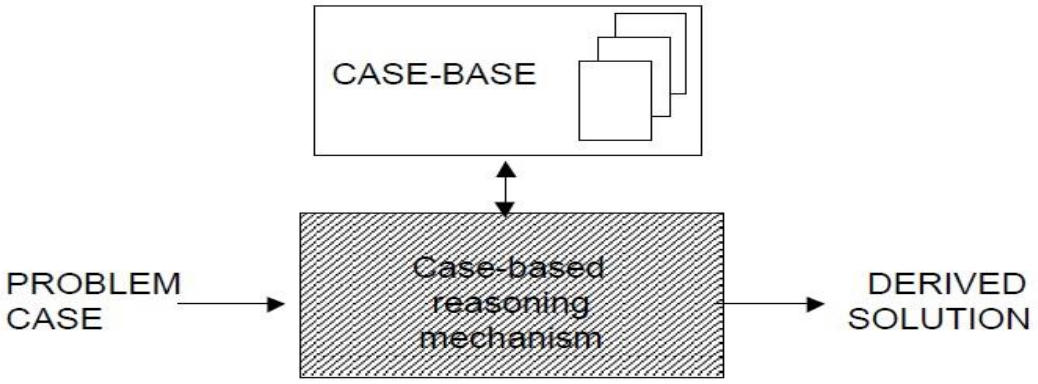


Figure 2.5 A CBR System [39]

Figure 2.6 incorporates the reasoning mechanism and the external aspects; the input specification, (or problem case), the output suggested solution, the memory of past cases that are referenced by the reasoning mechanism [39]. According to Julie [39], in most CBR systems, the case-based reasoning mechanism has an internal structure divided into two major parts; the case retriever and the case reasoner (see Figure 2.5). The case retriever's task is to find the appropriate cases in the case base while the case reasoner uses the retrieved cases to find a solution to the given problem description.

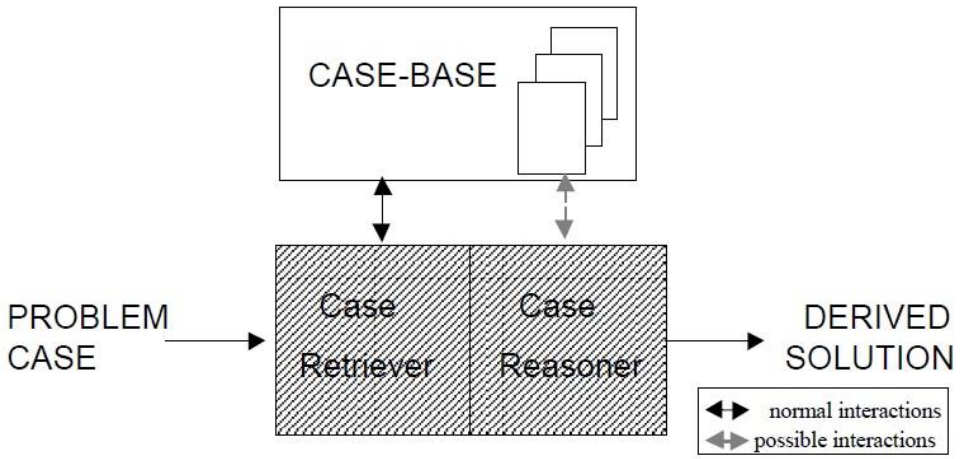


Figure 2.6 A CBR System [36]

2.11.2 Why use Case Based Reasoning?

Julie [39] mentions that case based reasoning approaches are used for the following reasons:-

- Reduction of the Knowledge Acquisition Task

  By eliminating the extraction of a model or a set of rules as is necessary in model/rule based systems, the knowledge acquisition tasks consists mainly of the collection of the relevant existing experiences/cases and their representation and storage.

- Avoid repeating mistakes made in the past

  In systems that record failures as well as successes, and perhaps the reason for those failures, the system can use the information about what caused failures in the past to predict any failures in the future.

- Graceful degradation of performance

  Some model based systems cannot even attempt to solve a problem on the boundaries of its knowledge or scope, or when there is missing or incomplete data. In contrast case-based systems can often have a reasonably successful attempt at solving these types of problem.

- Able to reason in domains that have not been fully understood, defined or modeled.

  While insufficient knowledge may exist about a domain to build a causal model of it or derive a set of heuristics for it, a case-based reasoner can function with only a set of cases from the domain. The underlying theory does not have to be quantified.

- Reason with incomplete or imprecise data and concepts

  Cases are retrieved, not just when identical to the current query case but also when they are within some measure of similarity, case based reasoning deal with since.

- Avoid repeating all the steps that need to be taken to arrive at a solution

  In problem domains that require significant processes to carry out the creation of a solution from scratch, the modifying of a previous solution can significantly reduce this processing. By reusing a previous solution, the steps taken to reach the retrieved solution can be reused themselves.

2.12 CBR Cycle

As shown in Figure 2.7, the problem solving cycle of CBR consists of four major processes [40] [41]: Retrieve, Reusing, Revise and Retain

- Retrieve: phase is an initial step which inquires about previous experiences that are similar to the new case. In this phase most similar cases will be retrieved from the general knowledge.
- Reuse: phase is the second step which is responsible in suggesting a solution for the new case from the available solutions of the cases that were retrieved from the general knowledge.
- Revise: the proposed solution from attempt to solve of new problem or adopt new proposed solution.
- Retain: when the new solution was confirmed then new proposed solution was retained.

The whole process of CBR cyclic in this order that new problem is solved by retrieving past experienced cases, by reusing the pervious case revising the solution, and retaining the new solution of experience in to existing knowledge system. As shown in the figure (2.7) that from collection of *pervious cases*, the new case is used RETRIEVES. The *retrieved* case is combined with the new case by reuse in to *solve case* and with the REVISE procedure the solution is also tested. In the RETAIN, useful experience is retained for future reuse, and existing case bases or updates with this case.

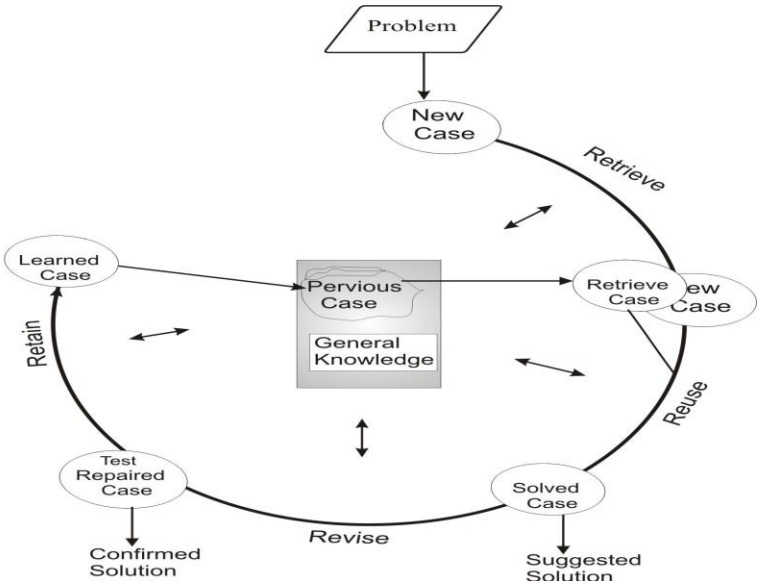


Figure 2.7 CBR cycle [40]

2.13 Designing a CBR System

There are step-by-step procedures involved in designing a CBR system.

2.13.1 Case Representation

Case is a representing an experience. It contains the information which is content of cases and situation where that information or experience can be used [2] [40]. Julie [39] mentioned that, Cases in a case base can represent many different types of knowledge and store it in many different representational formats. The objective of a system will greatly influence what is stored. A case based reasoning system may be aimed at the creation of a new design or plan, the diagnosis of a new problem, or the argument of a point of view with precedents. In general, a case consists of a:

- *Problem description*, which depicts the state of the world when the case occurred;
- *Problem solution* which states the derived solution to that problem; and/or
- *Outcome*, which describes the state of the world after the case occurred.

Julie [39] discussed that, in many practical CBR applications, cases are usually represented as two unstructured sets of attribute value pairs, i.e. the problem and solution features. Maja [42] stated that, cases can be represented as simple feature vectors, or they can be represented using any AI representational formalism such as frames, objects, predicates, semantic nets, or rules. Nasrullah [40] discussed that, case should represent two things: functionality of information and

the easiest way in which information is obtained in case. When we want to choose representation format of a case we should consider the following conditions [43]:-

- Language or shell in which we have to implement CBR system
- Indexing and search mechanism
- Type or structured associated with content

When, we want to choose format for case representation. Cases also have to be structured for efficient case retrieval. There are two types of structures. Structures types are Common Structure and Hierarchal Structure.

Memory model for chosen form of case representation depend on following factors [43]:

- Representation used in case base.
- Number of features that are used to match cases during search.
- Number and complexity of cases.

Case is a combination of two components. Components are description of a problem and its solution. Problem description consists of a set of attributes and values. That attributes predict a solution [40].

2.13.2 Indexing

Index is a computational data structure that can be held in memory and also can enable to search quickly [40]. Julie [39] stated that, Case indexing refers to assigning indices to cases for future retrieval and comparisons. This choice of indices is important to being able to retrieve the right case at the right time. This is because the indices of a case will determine in which context it will be retrieved in future. Nasrullah [40] mentioned that, Index should have the following features:

- Should show the purpose for which case will be used.
- It should be easy to recognize it in feature.
- It must address future use of case-base

2.14 Case retrieval

Case retrieval is the process of finding within the case base those cases that are the closest to the current case [39] [40].To carry out case retrieval there must be criteria that determine how a case is judged to be appropriate for retrieval and a mechanism to control how the case base is

searched. The selection criteria is necessary to decide which case is the best one to retrieve, that is, to determine how close the current and stored cases are [39]. Figure 2.8 depicts the process involved in case retrieval of CBR system.

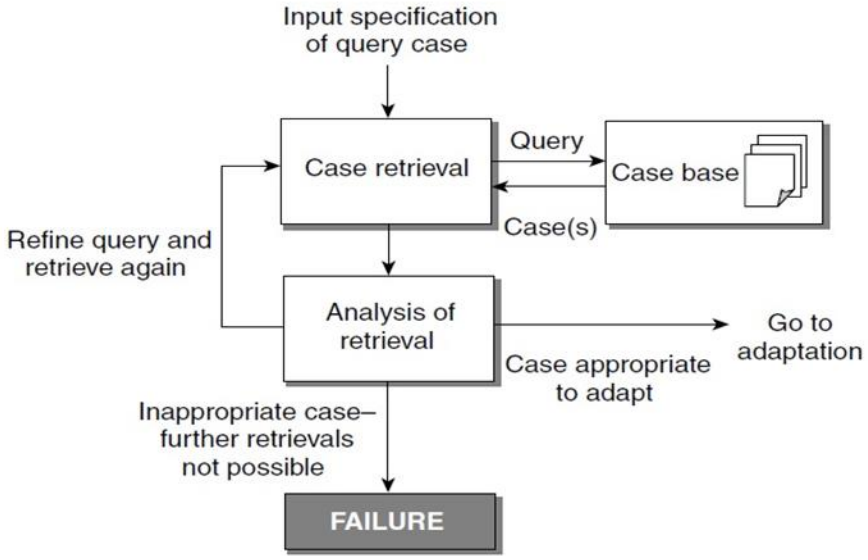


Figure 2.8 Case retrieval process involved in CBR [43]

According to Nasrullah [40] there are four retrieval techniques to identify similar cases:-

- **K-Nearest-neighbor Retrieval**

K-Nearest-neighbor retrieval technique is to measure similarity between source case and query case. If case is not matched with CBR library, then CBR system will return nearest match. In nearest neighbor retrieval, the case retrieved is chosen when the weighted sum of its features that match that query is greater than the other cases in the case base [39]. Nearest match can be computed using the following equation.

$$\textbf{Similarity}\ (T,S) = \sum_{i=1}^{n} f(Tj,Si)xWi \qquad \text{Equation 3}$$

Where;

T= target case

S= source case

n= number of attributes in each case

I= individual attribute from 1 to n

f= similarity function for attributes I in cases T and S

w= importance weighting of attribute I

Nearest-neighbor is not efficient technique. Because, when new case is introduced, indexing should be performed and it could affect the efficiency of the system.

- **Inductive Retrieval**

When inductive approaches are used to determine the case-base structure, which determines the relative importance of features for discriminating among similar cases, the resulting hierarchical structure of the case base provides a reduced search space for the case retriever. This may, in turn, reduce the query search time.

- **Knowledge-Guided Approaches**

Knowledge-guided approaches to retrieval use domain knowledge to determine the features of a case that are important for retrieving that case in the future. In some situations, different features of a case will have different levels of importance or contribution to the success levels associated with that case. As with inductive approaches to retrieval, knowledge-guided indexing may result in a hierarchical structure, which can be more effective for searching [43].

- **Validated Retrieval**

There have been numerous attempts at improving retrieval. One of these is validated retrieval, which consists of two phases. Phase one involves the retrieval of all cases that appear to be relevant to a problem, based on the main features of the present case. Phase two involves deriving more discriminating features from the initial group of retrieved cases to determine whether these cases are valid in the current situation. The advantage of validated retrieval is that inexpensive computational methods can be used to make the initial retrieval from the case base, while more expensive computational methods can be used in the second phase, where they are applied to only a subset of the case base [43].

2.15 Case Adaptation

Case adaptation is the process of translating the retrieved solution into the solution appropriate for the current problem [36]. It is a technique to alter retrieved case for reproducing new solution for new problem [40]. Case adaptation improve overall problem solving ability of CBR. According to Nasrullah [40], two techniques are used mostly in CBR systems: Structural and Derivational adaptation. In structural adaptation, formulas and rules are directly applied to stored solution in CBR library. When case is applies on these rules and formulas. Then, CBR system adapts this case and match with new problem. On the other hand derivational adaptation is a technique to reuse the rules and formulas to produce a new solution of a current problem. Solutions which are retrieved must be stored as additional case in the CBR library so it reproduces new solution to new case.

2.16 Evaluation

General performance of intrusion detection systems is measured in terms of numbers of selected features and the detection accuracies of the algorithms giving the best clustered results. As discussed by Farhan et al [63], there are different techniques used for performance measuring of the IDS such as Precision, Recall and F-measure. Good IDS require high detection rate, low false alarm rate and lower average misclassification cost.

2.16.1 Standard Metrics to Evaluate Intrusion

To evaluate the approach, the four standard metrics of true positive, true negative, false positive and false negative developed for network intrusions, have been used. Table 3.5 shows these standard metrics.

Confusion metrics (standard metrics)		Predicted connection label	
		Normal	Intrusions (Attacks)
Actual Connection	Normal	True Negative (TN)	False Alarm (FP)
	Intrusions (attacks)	False Negative (FN)	Correctly detected Attacks(TP)

Table 2.1 Standard metrics for evaluations of Intrusions (attacks) [19]

The representation of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are defined as follows [19]:

- ❖ True Positive (TP): The number of malicious records that are correctly identified.
- ❖ True Negative (TN): The number of legitimate record that correctly classified.
- ❖ False Positive (FP): The number of records that are incorrectly identified as attacks however in fact they are legitimate activities.
- ❖ False Negative (FN): The number of records that are incorrectly classified as legitimate activities however in fact they are malicious.

2.16.2 Precision and Recall

Precision is the number of class members classified correctly over the total number of instances classified as class members. Technically can be expressed as the attack has been occurred and the IDS detect correctly (19).

$$Precisin = \frac{TP}{TP+FP}$$ Equation 4

Recall measures the number of correctly classified examples relative to the total number of positive examples. In other words the number of class members classified correctly over the total number of class members [19].

$$Precisin = \frac{TP}{TP+FN}$$ Equation 5

$$F-measure = \frac{2*PR}{P+R}$$ Equation 6

2.17 Tools

2.17.1 Data Preparator

DataPreparator is a free software tool designed to assist with common tasks of data preparation (or data preprocessing) in data analysis and data mining [40].

It provides:

- A variety of techniques for data cleaning, transformation, and exploration.
- Handling of large volumes of data (since data sets are not stored in the computer memory)

- Standalone tool, independent of any other tools
- User friendly graphical user interface

DataPreparator can exploring and preparing data in various ways prior to data analysis or data mining. It includes operators for cleaning, discretization, numeration, scaling, attribute selection, missing values, outliers, statistics, visualization, balancing, sampling, row selection, and several other tasks. DataPreparator is written in Java and requires Java Runtime Environment (JRE) to be installed on machine. DataPreparator was developed by Dr. Bozena (Bobbie) Stewart from Sydney, Australia [49].

2.17.2 CBR-Based Framework jCOLIBRI

jCOLIBRI is an object-oriented framework for developing CBR systems It offers an easier development process that is based on the reuse of past designs and implementations. jCOLIBRI formalizes the CBR knowledge using domain independent CBR ontology (CBROnto), which is mapped over the classes of the framework, a knowledge-level description of the CBR tasks and a library of reusable Problem Solving Methods (PSMs) [50]. jCOLIBRI is selected for this study based in several aspects: availability (open source framework), implementation (the Java implementation implies to a great extend usability, extensibility and user acceptance), GUI (the provided graphical tools facilitate the system design). Another justification for this study is connected with the fact that jCOLIBRI affords the opportunity to apply Case Retrieval Nets (CRN) model as Case Base structure [50].

The main design principle of COLIBRI is its layered structure. A schema of its structure is shown in Figure 1. jCOLIBRI is just a white-box tool that permits programmer users to have total control of the internal details of the software. This framework represents the bottom layer of platform and lacks of any kind of visual interface. On the other hand, the top layer includes the graphical development tools to aid users in the development of CBR systems. These tools are enclosed in the COLIBRI Studio IDE and generates applications that use the components provided by jCOLIBRI.

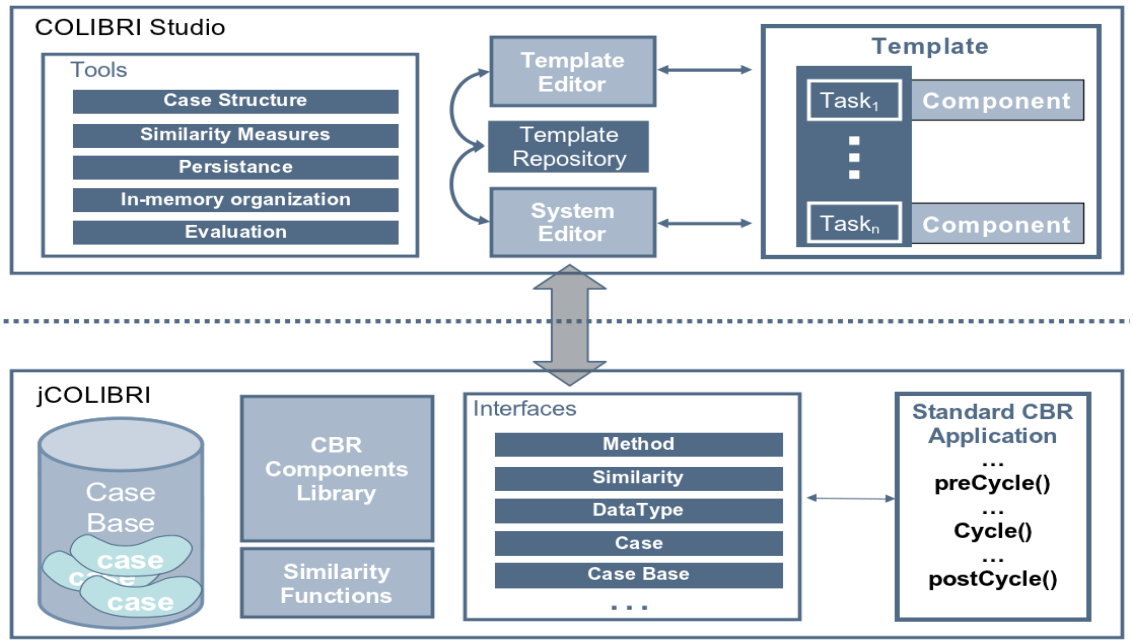


Figure 2.9 Schema of the layered architecture of COLIBRI [62]

2.17.2.1 Basic Components jCOLIBRI

Accourding to Juan [39], components of jCOLIBRI as shown in Figure 2.10 that is based on the following design principles:

- Organization into persistence, core and presentation layers. Cases –or experiences- can be stored using different media. Common media are data bases or plain text. Therefore, jCOLIBRI defines a family of components called connectors (Connector in Figure 2.10) to load the cases from different media into the in-memory organization: the case base (CBRCaseBase in Figure 2.10). There are several implementations of the in-memory case base using different data structures: linear lists, trees, hash maps, etc. This division into three layers enables an efficient management of cases, an issue that becomes more relevant as the size of the case base grows.
- Organization of the applications into precycle, cycle and postcyle. This organization is defined by the StandardCBRApplication interface shown in Figure 2.10. First, the precycle load the resource required and pre-processes the knowledge used by the application. This step is executed only once. Then the cycle is ready to be executed every

time a query is received. It computes the CBR processes. Finally, the postcycle releases the loaded resources to shut down the system.

- Case structure. The structure of cases is composed of four components: description, solution, result and justification of solution. Case components are represented using Java Beans. A Java Bean is any class that has a get() and a set() method for each public attribute. Using Java Beans, developers can design their cases as normal Java classes, choosing the most natural design. This way, each attribute of the Java Bean (integers, strings, booleans) becomes a simple attribute of the case. Moreover, you can follow a Composite pattern and include compound attributes that are other Java Bean classes that encapsulate some other attributes. These classes are referred to as case components because they must implement the CaseComponent interface shown in Figure 2.10. This interface only imposes the definition of an attribute as the identifier of the component. It is used by the connectors and methods as a kind of primary key to identify case components.

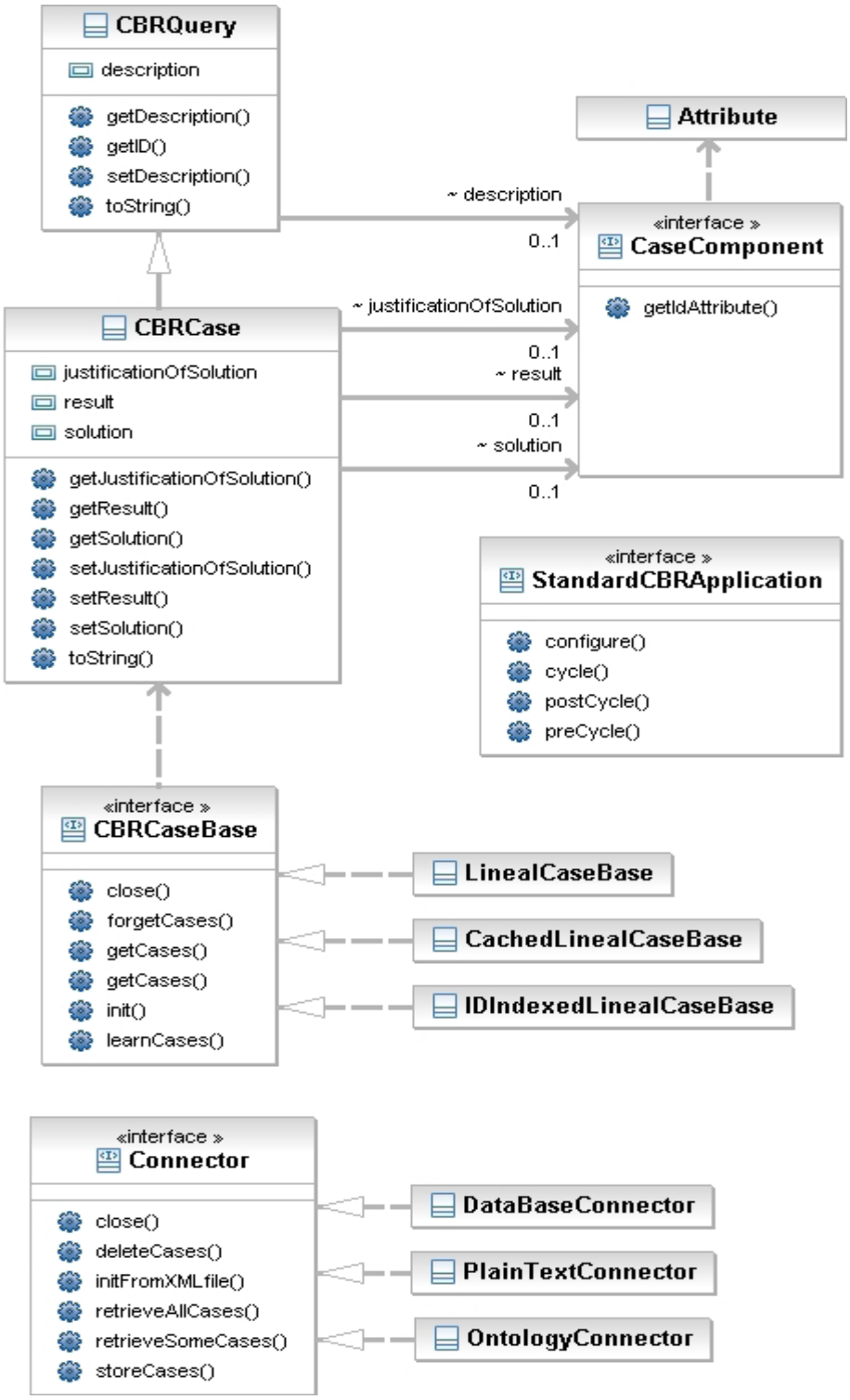


Figure 2.10 UML diagram with main elements of jCOLIBRI [39]

- Similarity measures to compare cases. The definition of the similarity between cases is based on the concept of similarity functions. There are global similarity functions that

compare components or compound attributes, and local similarity functions in order to compare simple attributes.

- Methods that implement and encapsulate the functionality of the CBR system. Methods implement different processes of the application (retrieval, selection, adaptation, etc.).

However, COLIBRI Studio provides a set of tools that hide the complexity of the Java code and are able to generate the source code for the system automatically. COLIBRI Studio is the implementation of the top layer of the COLIBRI platform. It provides the visual builder tools required to generate CBR systems without dealing directly with the source code. It is built on top of the jCOLIBRI framework and enables the composition of its CBR components. COLIBRI Studio is integrated into the popular Eclipse IDE. This way it takes advantage of the facilities provided to manage projects and Java source code. It enables the generation of CBR projects where required libraries are automatically configured. These libraries conform a new release of the jCOLIBRI framework numbered as jCOLIBRI [2] [39] [54].

COLIBRI Studio includes the following tools: a Case Designer that enables the definition of the structure of your cases; a tool to configure the storage of cases in textual files; a tool to select the in-memory organization of the case; a tool to define the similarity between cases; and several tools to define the behaviour of the system. Moreover COLIBRI Studio integrates several wizards that ease the design process [2] [39].

2.17.3 Weka

For this study the researcher used Weka for data mining. Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka is free software available under the GNU General Public License. The Weka workbench contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. The software has Header section and Data section and Supported attributes numeric, nominal, string and date data type.

The Main Features of weka 3.7.9:-

- 49 data preprocessing tools
- 76 classification/regression algorithms
- 8 clustering algorithms
- 3 algorithms for finding association rules
- 15 attribute/subset evaluators + 10 search algorithms for feature selection

2.18 Related works

Different researchers have been doing researches by using data mining technique for network intrusion detection. The researchers review some research works that related to data mining techniques, knowledge based system and case based reasoning.

Shi et al. [44] proposed clustering-based network intrusion detection system. They investigated unsupervised intrusion detection using multiple centroid-based clustering algorithms. They concluded that considering the dynamic nature of network traffic intrusions, unsupervised intrusion detection is more appropriate for anomaly detection than classification-based intrusion detection methods.

Sathya et al [47], they propose an intrusion detection algorithm based on data mining technique that is based on the analysis of usage data coming from multiple partners in order to reduce the number of false alarms. The performance is concentrated on the execution time of the algorithms by reducing the number of computations between sequences using dynamic cluster indexing approach and on the number of false alarm rate. The algorithms are analyzed for k nearest neighbor approach with CTI dataset for the effect of number of sequences in the clusters. For future work they recommend that, experimental evaluation and the comparison with existing method using real network audit data set and to exploit other data mining techniques such as association rule mining for network intrusion detection system and for other network security issues.

Solomon [45], has done research on design and develops a prototype self-learning knowledge-based system that can provide advice for physicians and patients to facilitate the diagnosis and treatment of diabetic patients. Knowledge is acquired using both structured and unstructured interviews from domain experts. He used decision tree that represents concepts and procedures. Rules are used to represent the domain knowledge and knowledge-based system and design the knowledge base SWI Prolog editor tool. For testing and evaluating the prototype system eighteen patients' histories are selected in order to test the accuracy of the prototype system. The overall total performance of the prototype system is 84.2%.

Alemu [46], proposed a case based reasoning (CBR) in solving complex side effects of HIV/AIDS cases for PLWHA. For knowledge acquired semi-structured interview and discussion are used. He used hierarchical tree structure for knowledge modeling. Case based reasoning

approach was used for knowledge representation and JCOLIBRI was used for developing CBR system. He applies prototype study for measure the performance of the study. User acceptance test and statistical analysis are used to evaluate the performance of the study. An average recall value of 72 %, with an average precision of 63 % has been achieved for overall performance of the prototype.

Tigabu [17], proposed predictive model for network intrusion detection, KDDcup'99 intrusion data set used for his experiment. Both J48 decision tree algorithm and the Naïve Bayes simple algorithm have been tested as a classification approach for building a predictive model for intrusion detection. The model created using 10-fold cross validation using the J48 decision tree algorithm with the default parameter values showed the best prediction accuracy. The model has a prediction accuracy of 96.11% on the training datasets. The model was validated with 3,397 testing dataset and performs a prediction accuracy of 93.2% for classifying new instances of datasets as normal, DOS, U2R, R2L and probe classes. For future works he recommended that an integration of knowledge bases system with data mining will help to improve the performance of network intrusion detection system.

Abdulkerim [20] study rule based integrating data mining with knowledge based system; He used for experiment KDDcup'99 intrusion data set. He conducted his data mining experiment by using J48 design tree algorithm and for data mining classifier, JRip is used for knowledge acquisition and an automatic knowledge acquisition mechanism also applied. And he design the knowledge based by using rule-based Knowledge representation method. On his proposed study a signatures network attack and automatic knowledge acquisition for constructing knowledge base performed at the same time with automatic integrator application. For the system performance testing RIDA-KBS are used and the system has 80 % of accuracy with Precision and Recall. And for user acceptance testing seven criteria of evaluation was performed and the system on average scored 80 %. Finally overall total performance the system has registered 80.5 % accuracy based on the system performance and user acceptance tests.

An attempt made by Alemu [46] and Tigabu [17] beside on the finding of their work they recommend integration of data mining with knowledge base system will help to improve the performance of network intrusion detection. Based on this assumption Abdulkerim [20] made an integration of data mining with knowledge based by using rule based knowledge representation

method. A rules based is one of the most popular knowledge representation method. The basic form of a rule is the if <conditions> then <conclusion>, where <conditions> represents the conditions of a rule, whereas <conclusion> represents its conclusion [12]. A rule based system represents general knowledge regarding a domain and solve problems from scratch. While case based systems use pre-stored situations to deal with similar new instances and cases for the specific knowledge. Therefore, as researcher knowledge an attempt need to works on integration of data mining with case base and to fill this gap, so this study is proposed.

CHAPTER THREE

METHODS AND APPROCHES

3.1 Architecture of the system

Figure 3.1 shows integration of data mining with case based reasoning so as to design a network intrusion detection system that can intelligently identify intrusions. As the users enter problem description (query) through user interface, the system searches the best matching cases form the case base. If relevant cases are found within the case base, then the system rank the relevant retrieved cases based on their global similarity. Next, the system proposes a solution. The proposed solution (solved solution) can be derived directly from a retrieved case. Therefore, the user of the system should have made an adaptation by altering the differences between the proposed case and the new case. In addition to adaptation, case inconsistencies are revised if the retrieved case is not the same as the new case. Finally, the revised solution is retained in the case base for future problem solving.

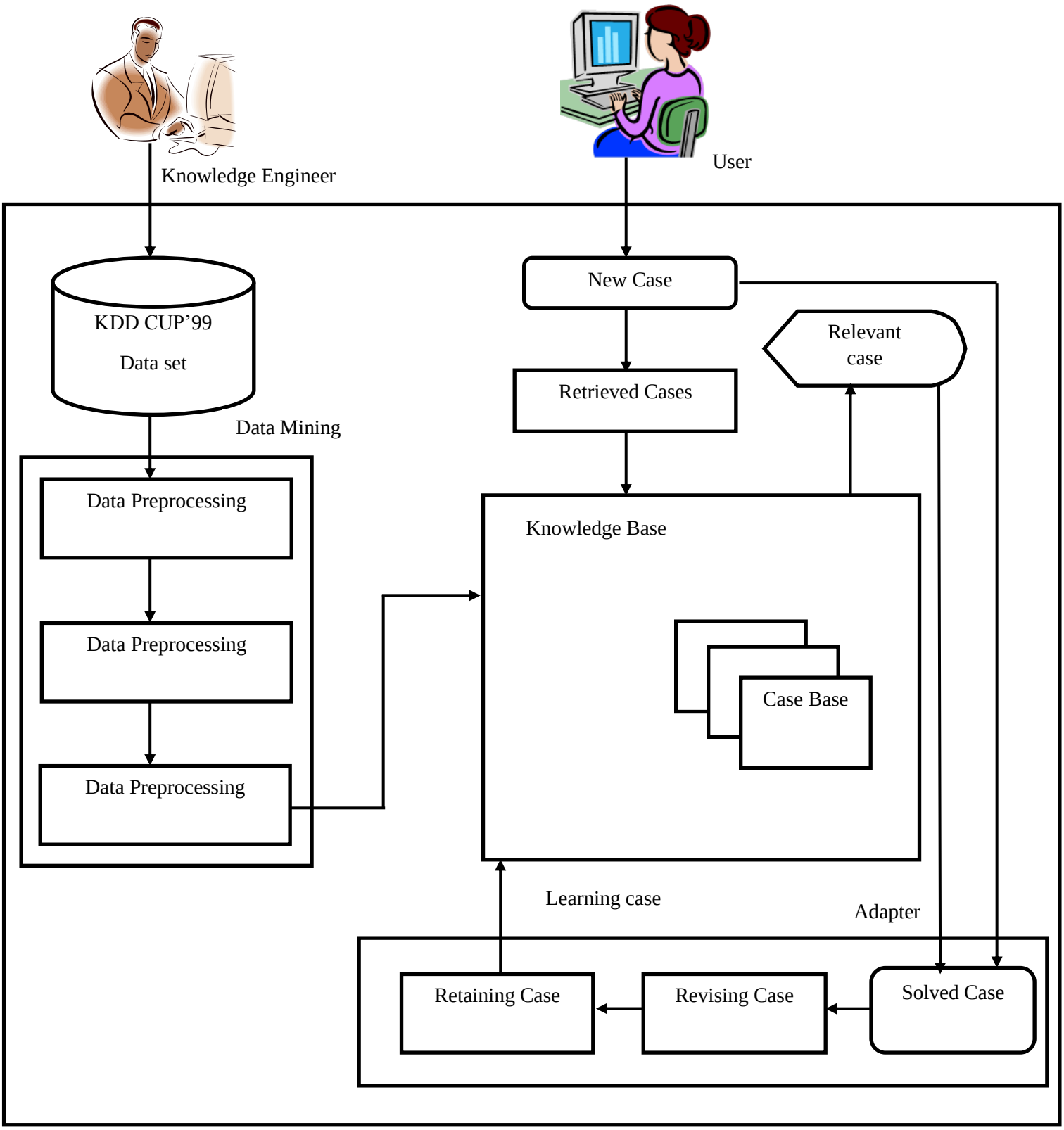


Figure 3.1 Architecture of the proposed system

The Framework shows that the data mining tasks used for generating knowledge from large collection of dataset. The detail of the framework is discussed below.

KDDcup'99 dataset: - is dataset which has been in use since 1999 for evaluation for anomaly detection method. The KDD (Knowledge discovery in Database) Cup 1999 intrusion detection contest KDD cup 99 Intrusion detection data sets. This data was prepared by the 1998 DARPA (Defense Advanced research Project Agency) Intrusion Detection Evaluation program by MIT Lincoln Labs (MIT Lincoln Laboratory).

Data preprocessing: - considering the time it takes to process data sets and coming up patterns, a sample size of 54, 111 instances are taken for this study. During the preprocessing task of instances are classified as normal and attack, into different distributions, while attacks were classified into four sections represented into DOS (deny of service), Probe (information gathering), U2R (user to root) and U2L (remote to local).

Clustering: - from the experimental result k-means algorithm registered better result than the other two algorithms. So, the k-means algorithm is used to mining hidden knowledge. The k-means algorithm takes the input parameter, k, and partitions a set of n objects into k clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. Cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity. The k-means algorithm proceeds as follows. First, it randomly selects k of the objects, each of which initially represents a cluster mean or center. For each of the remaining objects, an object is assigned to the cluster to which it is the most similar, based on the distance between the object and the cluster mean. It then computes the new mean for each cluster. This process iterates until the criterion function converges.

After k-means algorithm is applied the next step was design case based on the clustered instance based on centroid result. The system works the clustering process and designs the case base automatically. Case Base core tasks which are used in study application are precycle, main CBR cycle, post cycle, retrieve tasks, reuse tasks and retain tasks.

Design case structure: - Cases are saved in text file in applications. A case is composed of Description (describes the problem by means of several attributes), Solution (contains the

description of the solution of the case) and Result (represents the result of applying the case to a real situation). Description and Solution are sets of Attributes, and there are two types of attributes: simple and compound. Simple Attributes are described by Name, Type, Weight and Local Similarity Function. Compound Attributes collect other simple attributes, allowing complex case structure.

Configure connector: - the connector that is going to load the case base. The plan text connector is configured with the PalnTextconfig.xml file.

Case representation: - Case is a specific piece of knowledge representing as an experience. It contains the information which is content of case and situation where that information or experience can be used. In this study clustered instance considers as case. Case should represent two things: functionality of information and the easiest way in which information is obtained in case.

Case retrieve: - a new problem is solved by retrieving one or more previously experienced cases. Retrieve tasks will retrieve cases from the case base (Knowledge base). It consists of following subtasks [14]:

- Select working cases task from case base and stores them into current context.
- Compute similarity task using Similarity measurement determines how the new case is similar with old case stored in case base. The similarity measurement in Nearest Neighbor case base reasoning is calculated by local and global similarity functions. Global Similarity is linked with compound attributes and used to get similarity of collected attributes in unique similarity value. The global similarity measurement allows measuring the similarity between two cases, i.e. an old case from the case base with new case from the user. This measurement computes the similarity by using all local similarity of attributes which make up the case. It used to select the most relevant cases. Global similarity used in this study is average similarity. The local similarity is used to measure the similarity of attributes of the new case with their corresponding attributes value of old case from the case base. Local similarity functions are used to compare simple attribute values.
- Select best case, it shows the best matched of cases. It means that best matched case will be displayed.

Similarity Measurement

The local similarity is used to measure the similarity of attributes of the new case with their corresponding attributes value of old case from the case base. Local similarity functions are used to compare simple attribute values.

In this study, the following local similarity functions are used.

Equal: If you select equal local similarity for each attribute, then your input and value of case base must match. If value matches exactly then it will get result otherwise match failure.

Interval: When you select similarity interval and adjust interval value, then, jCOLIBRI match value as per the given interval. Exact value match is not compulsory in this similarity measure.

Global Similarity is linked with compound attributes and used to get similarity of collected attributes in unique similarity value. Global similarity used in this research is average similarity.

Average: It is a type of global similarity that considers the average of all attribute local similarity values. Figure 3.2 shows the global similarity algorithm.

Step 1: *Find the local similarity of step for all attributes of the case which make up the case base*

Step 2: *Multiply the result of the local similarity of attributes with their corresponding attribute weight (importance value)*

Step 3: *Add the value of all attribute results of step 2*

Step 4: *Add all weights of attributes that represent the importance value of the attributes and multiply by the number of attributes*

Step 5: *Divide the result of step 3 by the result of step 4 and the result of this step is the global similarity that represents the degree of match of the old case with the new input case*

Figure 3.2 Global similarity function algorithms [23]

Case reuse: - is the information and knowledge in that case to solve the problem reusing the case in one way or another. Reusing task has the following subtasks:-

- Prepare Cases for Adaptation Task. It will select cases from case base and stores them into context. The researchers also specify the path of case structure in this method.
- Atomic Reuse Task, it should be resolved by reuse resolution method.
- Reuse Task, CBR case is reused in this task.

Case adaption: - It is a technique to alter retrieved case for reproducing new solution for new problem. jCOLIBRI offers two basic adaptation methods:- the first is Direct Attribute Copy Method. This method copies the value of an attribute in the query to an attribute of a case. The second is Numeric Direct Proportion Method. Performs a numerical direct proportion among attributes of the query and the case. For this study Numeric Direct Proportion Method are employed for case adaption.

CHAPTER FOUR

KNOWLEDGE ACQUISITION USING DATA MINING

In this study, the researcher focus on designing a case based reasoning approaches with knowledge based system for network intrusion detection. Data mining, particularly clustering algorithms are applied to get related cases of attacks. K-means clustering, expectation maximization (EM) clustering and Make Density Based Clusterer approaches are employed and their result is compared to generate cases for the development of knowledge based.

4.1 Knowledge Source

The dataset for this study is collected from KDDcup'99 dataset available at ACM Knowledge Discovery [58]. KDDcup'99 is a dataset collected from simulated network connection and made available for researchers to conduct study towards intrusion detection.

According to [58], the 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. A standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest uses a version of this dataset. Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic such as DOS (denial-of-service), U2R (unauthorized access to local superuser (root) privileges), R2L (unauthorized access from a remote machine) and Probe (surveillance and other probing). This was processed into about five million connection records. Similarly, the two weeks of test data yielded around two million connection records.

The KDDcup'99 data set have 41 features. Among the 41 features, 1-9 are used to represent the basic features of a packet, 10-22 employ the content features, 23-31 are used for traffic features with two seconds of time window and 32-41 for host based features. They are basically grouped into three categories: basic features of individual connection, content features within a connection, and traffic features which are computed using a two seconds time windows.

4.1.1 Sampling data set

The KDDcup'99 data set is very huge in size; even after removal of redundant instances the remaining dataset is so huge which requires time and memory space during the mining process. Hence, the sampling is found to be paramount to generation or extraction of knowledge from the dataset and considerable samples are taken. While sampling, all instances of R2L and U2R are taken since their size is so small compared to others. The number of instances included in the sample for normal, Probe and DOS is based on their proportion on the cleaned dataset.

4.1.2 Data Preprocessing and Selection

Today's real-world data are highly susceptible to noisy, missing, and inconsistent data due to their typically huge size (often several gigabytes or more) and their likely origin from multiple, heterogeneous sources. Low-quality data will lead to low-quality mining results [7]. One of the most important deficiencies in the KDD data set is the huge number of redundant records, which causes the learning algorithms to be biased towards the frequent records, and thus prevent them from learning unfrequented records which are usually more harmful to networks such as U2R and R2L attacks. In addition, the existence of these repeated records in the test set will cause the evaluation results to be biased by the methods which have better detection rates on the frequent records. Therefore, before the actual mining task is performed, these instances are removed at the data preprocessing stage.

The data set was arranged for data mining technique and also suitable for the selected CBR tools. As a result, the redundant instances in the record were removed during the preparation of dataset for data mining and case features for building the CBR case-base. The dataset sets contains four different categories of intrusions DOS, R2L, U2R, Probing and Normal. For this study 4,898,401 instances were collected from the website and among them 494,427 instances were found non-redundant. Table 4.1 shows the distribution of normal and attacks instances before after removing redundant instance.

	Total instances		
Attacks	**Before removing redundant**	**After removing**	**Percentage**
R2L	1,126	1,126	0.22
Probe	41,102	4,107	0.83
U2R	49	49	0.11
DOS	3,883,370	391,458	79.17
Normal	972,780	97,277	19.67
Total	4,898,427	494,427	

Table 4.1: Distribution of different types of attacks and normal instances

4.2 Data Reduction

Before data mining task is started, data reduction is one of the data mining tasks. Data reduction techniques can be applied to obtain a reduced representation of the data set that is much smaller in volume, yet closely maintains the integrity of the original data. That is, mining on the reduced dataset should be more efficient yet produce the same (or almost the same) analytical results [7]. The 494,427 instances are huge to processing clustering algorithm and it takes time for process it, so the researcher re-sampled the dataset into 54,111 instances. The size of U2R and U2L are smaller than as compared to others data set. As a result, the original number of instances of U2R and U2L are included in the sample dataset. Table 4 shows the distribution of the DOS, R2L, U2R, Probing and Normal of instances.

	Label	Count	%
1	Normal	24,503	45.28
2	DOS	24,326	44.95
3	Probe	4,107	7.58
4	R2L	1,126	2.08
5	U2R	49	0.11

Table 4.2: Distribution of attacks and normal with percentage of sample instance after data reduction

The percentage of total normal instances is 45.28% and the total percentage of attacks is 54.28%. The data set that used for experimentation is not balanced so, the researcher resamples a dataset by applying the Synthetic Minority Oversampling TEchnique (SMOTE). A data set is imbalanced if the classes are not approximately equally represented. One class (a minority class) includes much smaller number of examples than other classes [7]. According to Jiawei [7], class imbalance causes difficulties for learning and decrease the classifier performance.

	Label	Count	%
1	Normal	24,503	25.5
2	DOS	24,326	25.1
3	Probe	16,428	17.1
4	R2L	18,016	18.8
5	U2R	12,544	13.4

Table 4.3: Distribution of attacks and normal with percentage of resample instance

After preprocessing the dataset and resample the dataset by SMOTE, it is converted into comma delimited Excel file (CSV format), then to arff file suitable for mining using WEKA 3.7.9

4.3 Experimentation

After data preprocessing and preparation task have been finished then, the dataset converting to .csv suitable format for WEKA, the next task is the experimentation involving the selected clustering algorithms.

4.3.1 Experimental Setup

A total of three experiments aiming at building descriptive models are undertaken. The sampled dataset contains 54,111 instances containing DOS, R2L, U2R, Probing, and Normal. The dataset contains 42 attributes and all of them are used in experiments. The researcher made the clusters into five because the classes of the instance are known classes which are five. These classes are Normal, Probe, U2R, R2L and DOS. In addition, after undertaking a number of experiments, default value of parameters is taking into consideration for each clustering algorithm since it allows achieving better accuracy compared to modifying the default parameters' values. The parameters involved with their respective description and values are indicated in table 4.3

Parameters	Description	Default Value		
		EM	K-means	Make Density Based Clusterer
debug	If set to true, clusterer may output additional info to the console.	False	-	-
displayModelInOldFormat	Use old format for model output. The old format is better when there are many clusters. The new format is better when there are fewer clusters and many attributes.	False	-	-
maxIterations	maximum number of iterations	100	500	-
maximumNumberOfClusters	The maximum number of clusters to consider during cross-validation to select the best number of clusters	-1	-	-
minLogLikelihoodImprovementCV	The minimum improvement in cross-validated log likelihood required in order to consider increasing the number of clusters when cross-validiting to find the best number of clusters	1.0E-6	-	-
minLogLikelihoodImprovementIterating	The minimum improvement in log likelihood required to perform another iteration of the E and M steps	1.0E-6	-	-
minStdDev	Set minimum allowable standard deviation	1.0E-6	-	-
numClusters	Set number of clusters. -1 to select number of clusters automatically by cross validation.	-1	2	-
numExecutionSlots	The number of execution slots (threads) to use. Set equal to the number of available cpu/cores	1	1	-
numFolds	The number of folds to use when cross-validating to find the best number of clusters (default = 10)	10		-
seed	The random number seed to be used.	100	10	-
initializeUsingKMeansPlusPlusMethod	Initialize cluster centers using the probabilistic farthest first method of the k-means++ algorithm	-	False	-
displayStdDevs	Display std deviations of numeric attributes and counts of nominal attributes.	-	False	-
dontReplaceMissingValues	Replace missing values globally with	-	False	-

	mean/mode.			
fastDistanceCalc	Uses cut-off values for speeding up distance calculation, but suppresses also the calculation and output of the within cluster sum of squared errors/sum of distances.	-	False	-
initializeUsingKMeansPlusPlusMethod	Initialize cluster centers using the probabilistic farthest first method of the k-means++ algorithm	-	False	-
preserveInstancesOrder	Preserve order of instances.	-	False	-
distanceFunction	The distance function to use for instances comparison (default: weka.core.EuclideanDistance).	-	Euclidean Distance	-
clusterer	the clusterer to wrap	-	-	K-means
minStdDev	set minimum allowable standard deviation	-	-	1.0E-6

Table 4.4 Default parameters and values for algorithms

4.3.2 Creating Descriptive model

As this stage, three descriptive models involving K-means, Expectation maximization (EM) and Make Density Based clustering algorithm are constructed.

4.3.3 Experiment 1- K-means clustering

This experiment is undertaken using k-means clustering involves its default value of parameters. The principle goal of employing the K-means clustering scheme is to separate the collection of normal and attack data that behave similarly into several partitions which is known as K-th cluster centroids. In other words, K-means estimates a fixed number of K, the best cluster centroid representing data with similar behavior. In this study, the number of clusters has 5 based on the number of class, so predefined K=5, representing Cluster 1, Cluster 2, Cluster 3, Cluster 4, and Cluster 5. Hence, the distance has calculated using Euclidean distance. The algorithm correctly clustered is 48,396 instances and 5,715 instances have incorrectly clustered. 11.14 seconds is used to build the model.

4.3.4 Experiment 2- Make Density Based Clusterer

The second experiment is based on Make Density Based Clustere algorithm. Hence, the distance is calculated using Euclidean distance and the number of clusters has 5 based on the number of class. The correctly clustered instance 87.6% and from the total number of 54,111 instances 47,419 instances are correctly clustered and 6,692 instances are not correctly clustered. 13.8 seconds is used to build the model.

4.3.5 Experiment 3- Expectation Maximization (EM)

The third experiment is based on Expectation Maximization (EM) algorithm. Hence, the distance has calculated using Euclidean distance and the number of clusters has 5 based on the number of class. The correctly clustered instance 83.9% and from the total number of 54,111 instances 45,400 instances are correctly clustered and 8,711 instances are not correctly clustered. Table 4.3 shows the classes of Clusters.

As shows in table 4.3, the three cluster algorithm performed different result. Result of K-means algorithms has shown the better clustered accuracy than other. The highest incorrectly clustered is shown by Expectation Maximization (EM) algorithm. 516.38 seconds is used to build the model.

Clustere	Correctly clustered		Incorrectly clustered		Time taken to build model
	No.	Percentage	No.	Percentage	
K-means	48,396	89.43%	5,715	10.57%	11.14 seconds
Make Density Based Clusterer	47,419	87.6%	6,692	12.4%	13.8 seconds
EM	45,400	83.9%	8,711	16.1%	516.38 seconds

Table 4.5 Performance of cluster

Clustered instance generated from the clustering algorithms is used for to design and build case base. So as to develop effective case based system, acquiring relevant case is paramount. Hence form the three algorithms the researcher selected the cluster which best performed on clustering the data set. The K-means has best performance among the three clustering algorithm and it correctly clustered instance 89.43% and the time used to build the model is 11.14 seconds. So, the k-means clustering algorithm is used to build cases for the system.

The k-mean cluster similarity is measured in regard to the mean value of the objects in a cluster, which can be viewed as the cluster's centroid or center of gravity. The k-means algorithm takes the input parameter, *k*, and partitions a set of n objects into *k* clusters so that the resulting intracluster similarity is high but the intercluster similarity is low. In this study case based network intrusion detection are proposed and in order to develop the knowledge base cases are store into knowledge base then retrieve based on the similarity of the query. The k-means algorithm are undertaken to perform the clustered the dataset based on the centroid value. The case retrieval process retrieves case based on their similarity.

4.4 Evaluation of K-means Algorithm

To test k-means clustering algorithm, test dataset is prepared. The two weeks of test dataset yielded around of two million connections record and the researcher prepared 15,068 dataset by using datapreparator data preprocessing tool. Table 4.6 shows the evaluation result of test set.

=== Evaluation on test set ===

Clustered Instances

No.	Label	Count	%
1	Normal	5,833	38.7%
2	R2L	1,657	10.9%
3	Probe	3,227	21.4%
4	DOS	4,351	28%
5	U2R	699	1%

Table 4.6 Evaluation of test set

CHAPTER FIVE

INTEGRATION OF CLUSTERING WITH CASE BASED REASONING

In this study, an attempt is made to integrate clustering result with CBR system for Network Intrusion Detection. The development of a CBR application involves a number of steps, such as collection of cases and background knowledge, modeling a suitable case representation, defining an accurate similarity measure, implementing retrieval functionality, and implementing user interfaces [57].

The knowledge base can be used as a repository of knowledge in various forms. This container of case about network attacks which are generated by clustering, specifically k-means algorithm and automatically mapped into case based reasoning system.

5.1 Building the Case Base

During setting up the objectives of this study, one of the objectives is collecting attacks case in order to build a case base and represent the cases using the appropriate case representation method. Cases in a case base can represent many different types of knowledge that can be stored in many different representational formats. So, the researcher collects intrusion cases from KDDcup'99 data set using clustering algorithm. The acquired cases are used to build Case Based Network Intrusion Detection system that is used to assist by offering decision to protect the network from attacks. All the acquired cases are stored as plaintext file in a feature value representation format to represent cases.

The case base is presented as a plaintext comprising of *n* columns representing case attributes (A_1, A_2, A_3, …, A_n) and each *m* rows representing individual cases C ={C_1, C_2, C_3, …,C_m} each attribute has a sequence of possible *k* values associated to each column attribute A={V_1, V_2, V_3, …, V_k}. The reason for representing cases using feature-value representation is that this approach supports nearest neighbor retrieval algorithm and it represents case in an easy way [12] [14].

5.2 Case Representation

The case representation is made in a way that easily fit to COLIBRI Studio. Designing of such a case structure helps easily define the features available in the case and to measure the similarity between existing and new cases. Hence, the overall application of this research is to retrieve similar cases from the case base that can guide future reasoning of network intrusion detection. The collections of cases were structured to make efficient in retrieval process. This is done through case indexing process. Indexing refers to assigning indices to cases for retrieval and comparisons of query to the case in the case base [12] [14].

5.3 Managing The Case Structure in COLIBRI Studio

The acquired cases are saved in plaintext file format. All 41 attributes have significant impact on network intrusion detection.

Case Attributes	**Features**	**Data types**
Duration	Continuous	Integer
protocol_type	Discrete	String
Service	Discrete	String
Flag	Discrete	String
src_bytes	Continuous	Integer
dst_bytes	Continuous	Integer
Land	Discrete	Integer
wrong_fragment	Continuous	Integer
Urgent	Continuous	Integer
Hot	Continuous	Integer
num_failed_logins	Continuous	Integer
logged_in	Discrete	Integer
num_compromised	Continuous	Integer
root_shell	Continuous	Integer
su_attempted	Continuous	Integer

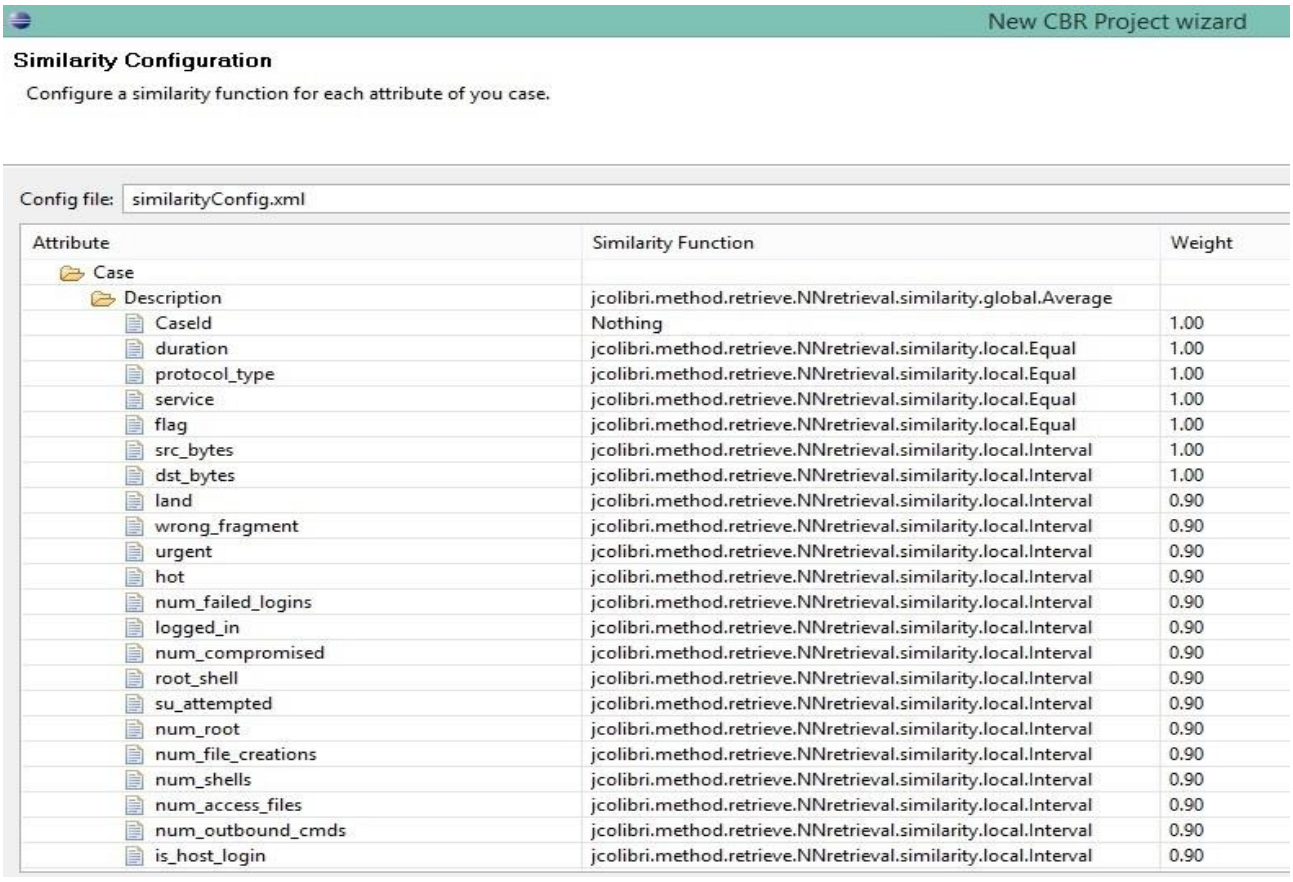
num_root	Continuous	Integer
num_file_creations	Continuous	Integer
num_shells	Continuous	Integer
num_access_files	Continuous	Integer
num_outbound_cmds	Continuous	Integer
is_host_login	Discrete	Integer
is_guest_login	Discrete	Integer
Count	Continuous	Integer
srv_count	Continuous	Integer
serror_rate	Continuous	Double
srv_serror_rate	Continuous	Double
rerror_rate	Continuous	Double
srv_rerror_rate	Continuous	Double
same_srv_rate	Continuous	Double
diff_srv_rate	Continuous	Double
srv_diff_host_rate	Continuous	Double
dst_host_count	Continuous	Integer
dst_host_srv_count	Continuous	Integer
dst_host_same_srv_rate	Continuous	Double
dst_host_diff_srv_rate	Continuous	Double
dst_host_same_src_port_rate	Continuous	Double
dst_host_srv_diff_host_rate	Continuous	Double
dst_host_serror_rate	Continuous	Double
dst_host_srv_serror_rate	Continuous	Double
dst_host_rerror_rate	Continuous	Double
dst_host_srv_rerror_rate	Continuous	Double

Table 5.1: Managing the Case Structure in COLIBRI Studio

5.4 Case Attributes

Defining case structure in COLIBRI Studio are done by using simple manage case structure window. It is very easy to define case structure with COLIBRI Studio. Because it is simple to add attributes in description of case structure and set properties of attributes of metadata of attributes. Metadata of attributes are weight of attributes, data type of attributes and similarity function. During configuration of case structures, COLIBRI Studio creates codes automatically and saved in xml file format. Most significant attributes are set by declaring higher weight as compared to other weights.

Descriptive modelling for designing case based reasoning in network intrusion detection prototype case base has 40 description attributes and 2 solution attributes. Solution attribute is used after finding best selected cases and show the type of attacks. Appendix VI shows the description of case attributes regarding name, data type, weight and local similarity.



Attribute	Similarity Function	Weight
Case		
Description	jcolibri.method.retrieve.NNretrieval.similarity.global.Average	
CaseId	Nothing	1.00
duration	jcolibri.method.retrieve.NNretrieval.similarity.local.Equal	1.00
protocol_type	jcolibri.method.retrieve.NNretrieval.similarity.local.Equal	1.00
service	jcolibri.method.retrieve.NNretrieval.similarity.local.Equal	1.00
flag	jcolibri.method.retrieve.NNretrieval.similarity.local.Equal	1.00
src_bytes	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	1.00
dst_bytes	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	1.00
land	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
wrong_fragment	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
urgent	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
hot	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_failed_logins	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
logged_in	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_compromised	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
root_shell	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
su_attempted	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_root	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_file_creations	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_shells	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_access_files	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
num_outbound_cmds	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90
is_host_login	jcolibri.method.retrieve.NNretrieval.similarity.local.Interval	0.90

Figure 5.1 configurations of case structures and similarities

5.4.1 Managing Connectors

Once case structures are configured in COLIBRI Studio, CBR system must access the stored cases in an efficient way. COLIBRI Studio splits the problem of case base management in two separate although related concerns: persistency mechanisms through connectors and in-memory organization.

A connector is an object which has the ability to access and retrieve cases from a specific case persistency when given the case structure, and give those cases to the CBR system in a standardized way. Because of this, COLIBRI Studio can deal with any case persistency as long as a connector is provided.

Connectors are objects that know how to access and retrieve cases from the storage media and return those cases to the CBR system in a uniform way. Therefore connectors provide an abstraction mechanism that allows users to load cases from different storage sources in a transparent way. As shown in figure 5.2, COLIBRI Studio includes connectors that work with plain text files, relational databases and Description Logics systems.

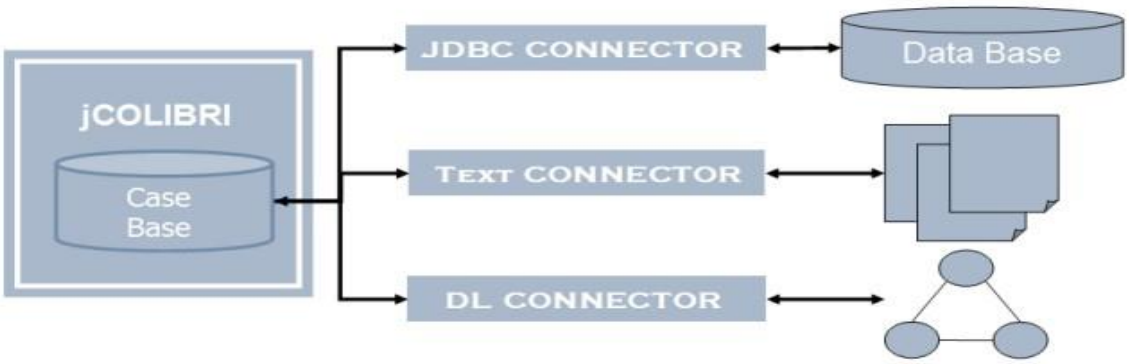


Figure 5.2 The jCOLIBRI connector architecture [24]

For the implementation of Case Based Intrusion Detection, the researcher used plaintext connector because dataset cases are stored in plaintext file format. Plaintext file case base connector is used for persistence of cases. In this connector, the researcher has to specify the path of case structure and also path of text file. All the attributes of a case should be mapped. This is connector's responsibility to retrieve data from case base and return it back to GUI. Like that of case structure, connector is also saved in xml format.

5.5 Manage Tasks and Methods

5.5.1 Managing Tasks

For the development of case based network intrusion detection, the researcher used core package tasks such as PreCycle, main CBR cycle and PostCycle. The first step, PreCycle task gets all the cases in case base. Therefore, it is necessary to define dataset path of connector in its subtask. There is only one subtask called obtain case task and it is used to retrieve data from case base before the execution of the main CBR cycle.

Main CBR cycle is the main task of CBR cycle and it also has sub tasks. The developer has to give path of case structure in it. It knows number of case attributes that are available. It is called obtain query task. In addition to obtain query task, there are other significant tasks under the main CBR cycle. These are retrieve tasks, reuse tasks, revise task and retain tasks.

Retrieve tasks used to retrieve case(s) from the stored case base. Retrieve tasks also decomposed in to different subtasks. The subtasks include select working cases task, compute similarity task and select the best case. Select working case task selects cases from case base and stores them into current context. Compute similarity task compute similarity of the stored cases with the case entered by the user using the query window. Select best case shows the best matched of case(s) after computing the similarity of stored cases against the new case. It means that the number of best matched case(s) is shown to the user depending on the method used and the threshold.

Reuse tasks enable to reuse previously stored cases. It has three subtasks such as prepare cases for adaptation task, atomic reuse task and reuse task. Prepare cases for adaptation task select cases from case base and stores them into context. Here also specifying the path of case structure in this method is needed. Atomic reuse task should be resolved by reuse resolution method.

Revise task is the evaluation stage about the selected solution in reuse phase. After selecting the most similar cases from the retrieved results, the solution for the problem should be confirmed and validated before the solution is stored for future use.

Retain tasks also used to CBR case retention on a persistence layer. It has also its own subtasks like select cases to store task and store cases task. Select cases to store task give authentication to the user for storing case. The store cases task enables to store case(s) into the case base.

The last task in managing tasks in jCOLIBRI is PostCycle. PostCycle task have only one sub task called close connectors task which is usually executed after the main CBR cycle. Its main task is to close a connection between case base and GUI.

5.5.2 Managing Methods

The method library stores classes that actually resolve the task. These classes can resolve the CBR cycle using in programming or using GUI. All tasks that are mentioned above should have their own methods to be assigned in order to achieve the tasks goal. The following are lists of methods which are used to solve tasks for this Case Based Network Intrusion Detection application.

5.6 Automatic Integration of Clustering Model with Case Based System

In this study, an attempt is made to design an automatic integration descriptive model (constructed by data mining) with case based reasoning. To achieve this COLIBRI Studio, Eclipse IDE Version: 3.7.0 with JDK 6.0 has been used.

Before the first process of integration is to apply the researcher create cbr java packages and then create class for CBRApplication and link to k-means clustered result to automatically integrated to CBR. After CBRApplication and clustered result are applied, the next step could be experiment on COLIBRI Studio. A CBR application in jCOLIBRI must implement or extend the jcolibri cbrapplications standard CBRApplication interface.

COLIBRI Studio is a technological evolution of COLIBRI and it is an object-oriented framework in Java which is designed for building CBR systems. It is a java-based and uses JavaBeans technology for case representation and automatically generate of user interface. The Java code generated for jCOLIBRI for network intrusion detection is show in figure 5.3 below the general structure of a case based system and this study also develops based on JavaBeans technology.

```
import jcolibri.cbrcore.CBRCaseBase;
import jcolibri.cbrcore.CBRQuery;
import jcolibri.exception.ExecutionException;
        //Defines the method of an standard CBR application.
public interface StandardCBRApplication
{
//Configures the application: case base, connectors …
 public void configure() throws ExecutionException;
//Runs the precyle where typically cases are read and organized into a case base.
 public CBRCaseBase preCycle() throws ExecutionException;
        // Executes a CBR cycle with the given query.
 public void cycle(CBRQuery query) throws ExecutionException;
        //Runs the code to shut down the application. Typically it closes the
connector.
 public void postCycle() throws ExecutionException;
}
```

Figure 5.3 Standard CBR Application interface

The second layer of Case Base management is the data structure used to organize the cases once loaded into memory. The CBR methods to access the cases in-memory organization of the cases is jcolibri.cbrcore.CBRCaseBase method should be imported. In-memory organization has three different types of structure: jcolibri.casebase.LinealCaseBase: Basic Lineal Case Base that stores cases into a List. jcolibri.casebase.CachedLinealCaseBase: Cached case base that only persists cases when closing the application. jcolibri.casebase.IDIndexedLinealCaseBase: Extension of LinealCaseBase that also keeps an index of cases using their IDs. For this study jcolibri.casebase.LinealCaseBase memory organization are used. Figure 5.4 shows Load the cases and memory organization.

```
public void configure() throws ExecutionException {

    //try {

    _connector = new PlainTextConnector();

    _connector.initFromXMLfile(jcolibri.util.FileIO.findFile("C:\\Users\\GishenDebreKerbi\\
        Documents\\NetBeansProjects\\CBRDetection\\src\\cbr\\plainTextConnectorConfig.xml"));

    _caseBase = new LinealCaseBase();

        simConfig = new NNConfig();
```

Figure 5.4 Load the cases and memory organization

The whole program for developing by using COLIBRI Studio code for constructing case based is attached in Appendix. After the main function of CBR java code generated the next step could be, the case structure being designed defines a case component named *CaseDescription* for the description of the case and another component named *CaseSolution* for the solution. The id attributes (*DescriptionID* and *SolutionID*) are required by the components of jCOLIBRI to manage the cases and serve as primary keys to index and store the case base. Figure 5.5 shows java code for case description and figure 5.6 shows java code for case solution.

```
import jcolibri.cbrcore.Attribute;
public class CaseDescription implements jcolibri.cbrcore.CaseComponent {
public Integer getDuration() {
    return duration; }
  public void setDuration(Integer duration1) {
    this.duration = duration1;
  }}
```

Figure 5.5 Java code for case description

```
import jcolibri.cbrcore.Attribute;
public class CaseSolution implements jcolibri.cbrcore.CaseComponent {
Integer SolutionId;
String result;
@Override
public String toString() {
return "(" + SolutionId + ";" + result + ")";
}
}
```

Figure 5.6 java code for case solution

Once the case structure is defined the next step is to configure how cases are loaded from the persistance media. Configure a connector that loads cases from coma-separated text files. Again the Config File field contains the name of the xml configuration file that will store the settings and connectors are configured through XML configuration files. Figure 5.7 shows the Palin text connector of the system.

```
<TextFileConfiguration>
FilePath>C:\Users\GishenDebreKerbi\Documents\EclipjCOLIBRIProject\DataSet\DataSet_
For_Test.txt</FilePath>
   <Delimiters>,</Delimiters>
   <DescriptionClassName>cbr.CaseDawit</DescriptionClassName>
   <DescriptionMappings>
      <Map>duration</Map>
      <Map>protocol_type</Map>
      <Map>service</Map>
      <Map>flag</Map>
      <Map>src_bytes</Map>
      <Map>dst_bytes</Map>
      <Map>hot</Map>
      <Map>dst_host_count</Map>
      <Map>dst_host_srv_count</Map>
   </DescriptionMappings>
   <SolutionClassName>cbr.CaseSolution</SolutionClassName>
   <SolutionMappings>
      <Map>SolutionId</Map>
      <Map>result</Map>
   </SolutionMappings>
</TextFileConfiguration>
```

Figure 5.7 Plain Text Connector

The following step consists on defining the similarity configuration to compare cases with the query. Compound attributes are configured with a global similarity function, whereas simple attributes use local similarity functions. These functions are automatically obtained from the COLIBRI library.

The mapping files define how to map a Java Bean with a table of the data base. Firstly, the researcher defines which table is used to store the bean. Then researcher configures which

column of the table contains each attribute of the bean. Cases were composed by a description represented in the CaseDescription class, and a solution represented in the CaseSolution class. To map both components to the same table. Some columns will store the attributes of the description and other will store the attributes of the solution.

5.7 Retrieval

At this point, the researcher completed the configure() and precycle() methods of CBRApplication application. Those methods allows load the cases from the persistence into memory. The precycle() method loads the cases and stores them into the _caseBase object of main application. This section and the following ones show how to fill the cycle() method to perform the 4R's tasks (retrieval, reuse, revise, and retain).

The retrieval step obtains the most similar cases given a query. The main method of jCOLIBRI for computing the retrieval is the jcolibri.method.retrieve.NNretrieval.NNScoringMethod class. This method performs a Nearest Neighbor numeric scoring comparing attributes. It uses global similarity functions to compare compound attributes (CaseComponents) and local similarity functions to compare simple attributes.

The configuration of those similarity functions is stored in the jcolibri.method.retrieve.NNretrieval.NNConfig object. The values configured in this object are:

- Global similarity function for the description.
- Global similarity functions for each compound attribute (CaseComponents excepting the description).
- Local similarity functions for each simple attribute.
- Weight for each attribute.

The cases to compare with the query, the query and the similarity configuration. The method returns a collection of jcolibri.method.retrieve.RetrievalResult objects. These objects contain the retrieved case and a double that represents the similarity of that case to the query.

Most similar cases must be selected once they have been scored according to their similarity with the query. Usually, only the top k most similar cases are selected. This retrieval process that combines Nearest Neighbor scoring and top k selection is commonly called k-NN retrieval.

5.7.1 Similarity function

There are many similarity functions in the jcolibri.method.retrieve.NNretrieval.similarity package. For this study interval, equal and average similarity function is employed to measure the similarity function. Figure 5.8 shows similarity function.

```
import jcolibri.method.retrieve.NNretrieval.similarity.local.Interval;
import jcolibri.method.retrieve.NNretrieval.similarity.local.Equal;
import jcolibri.method.retrieve.NNretrieval.similarity.global.Average;
import jcolibri.method.retrieve.NNretrieval.NNConfig;
import jcolibri.method.retrieve.NNretrieval.NNScoringMethod;
```

Figure 5.8 similarity function

5.7.2 Case selection

Most similar cases must be selected once they have been scored according to their similarity with the query. Usually, only the top k most similar cases are selected. The case based network intrusion detection system retrieve the most similar five cases. This retrieval process that combines Nearest Neighbor scoring and top *k* selection is commonly called k-NN retrieval. Figure 5.9 shows java code for case selection.

```
import jcolibri.method.retrieve.selection.SelectCases;
import jcolibri.method.retrieve.RetrievalResult;
public void cycle(CBRQuery query) throws ExecutionException {
    // Execute Filter
    Collection<CBRCase> filtered =
FilterBasedRetrievalMethod.filterCases(_caseBase.getCases(), query, filterConfig);
    // Execute NN
    Collection<RetrievalResult> retrievedCases =
NNScoringMethod.evaluateSimilarity(filtered, query, simConfig);
    // Select cases
    Collection<CBRCase> selectedCases = SelectCases.selectTopK(retrievedCases, 5);
    for (Iterator<CBRCase> it = selectedCases.iterator(); it.hasNext();) {
      jcolibri.cbrcore.CBRCase c = it.next();
      // Show result
      resultDialog.showCases(retrievedCases, selectedCases);
      resultDialog.setVisible(true);
      System.out.println(c);
    }
```

Figure 5.9 Java code for case retrieval

5.8 Reuse

The reuse step adapts the solution of the retrieved cases to the requirements of the query. This step is very domain dependent and vary depending on the application. jCOLIBRI offers two basic adaptation methods: jcolibri.method.reuse.DirectAttributeCopyMethod. This method copies the value of an attribute in the query to an attribute of a case. On the other hand, jcolibri.method.reuse.NumericDirectProportionMethod. Performs a numerical direct proportion among attributes of the query and the case. In this study the NumericDirectProportionMethod method is used to adapt the solution of the attacks according with the description of the query is used. Figure 5.10 shows java code for case reuse.

```
// Show adaptation dialog
        autoAdaptDialog.setVisible(true);
// Adapt depending on user selection
        if (autoAdaptDialog.adapt_Dsthost_Servcount()) {
            // Compute a direct proportion between the "protocol_type" and "result" attributes.
            NumericDirectProportionMethod.directProportion(new Attribute("protocol_type",
CaseDescription.class),
                    new Attribute("result", CaseSolution.class),
                    query, selectedCases);
        }
```

Figure 5.10 Java code for case reuse

5.9 Revise

In the revise step the proposed solution is tested for success, e.g. by being applied to the real world environment or evaluated by a domain expert, and repaired if failed. jCOLIBRI only includes a method to define new Ids to the cases as they will be stored into the data base during the following step and they cannot use the original Id of the retrieved cases. This method is jcolibri.method.revise.DefineNewIdsMethod. Figure 5.11 shows java code for case revise.

```
public void showCases(Collection<CBRCase> cases) {
        this.cases = new ArrayList<CBRCase>(cases);
        currentCase = 0;
        showCase();
// Revise
        revisionDialog.showCases(selectedCases);
        revisionDialog.setVisible(true);
    }
```

Figure 5.11 Java code for case revises

5.10 Retain

In the retain step useful new cases are stored in the case base for future reuse. This way the CBR system has learned a new experience. jCOLIBRI includes the jcolibri.method.retain.StoreCasesMethod to include new cases into the case base. The new added cases will be stored in the persistence media depending on the chosen implementation of CBRCaseBase. The LinealCaseBase class will store the cases directly into the persistence layer, but the CachedLinealCaseBase will keep the new cases into memory and save them only when closing the CBR application (this is, when the postCycle() method is invoked). The following code shows the retain window in the application and calls the StoreCasesMethod method. The retain window allows the user to select which cases must be added to the case base and to define a new Id for each one. Figure 5.12 shows java code for retain case

```
public void showCases(Collection<CBRCase> eval, int casebasesize) {
        cases = new ArrayList<CBRCase>(eval);
        casesToRetain = new ArrayList<CBRCase>();
        currentCase = 0;
        if (numcases < casebasesize) {
           numcases = casebasesize + 1;    }
        idEditor.setText("Class" + (++numcases));
        showCase();  }
void setId() {
        CBRCase _case = cases.get(currentCase);
        cases.remove(_case);
        CaseDescription desc = (CaseDescription) _case.getDescription();
        desc.setCaseId(idEditor.getText());
        CaseSolution sol = (CaseSolution) _case.getSolution();
        sol.setSolutionId(idEditor.getText());
        casesToRetain.add(_case);    currentCase = 0;
        idEditor.setText("Result" + (++numcases));
        saveCheck.setSelected(false);
        enableSaveCase();
        showCase();     }
```

Figure 5.12 Java code for case retain

The postcycle() method of the CBR applications in jCOLIBRI is in charge of shutting down the CBR system or invoke maintenance tasks. This method usually calls the close() method of the connector to store (if required) the learned cases into the persistence media.

In the Case based network intrusion detection application call the close() method of the connector and also the shutdown() method of the case base manager.

```
public void postCycle() throws ExecutionException {
    this._caseBase.close();
}
```

Figure 5.13 postcyle of the application

The whole program for developing the data mining and case based reasoning system integration using java Eclipse IDE and COLIBRI Studio IDE code for construction is attached in Appendix VI, VII.

CHAPTER SIX

IMPLEMENTATION AND EXPERIMENTATION

So far, natural grouping of intrusion are extracted from sampled collection of intrusion data set and knowledge base is constructed as case base that contains case of network incidents. Case Based Network Intrusion Detection System (CBNIDS), is then designed that automatically detect a network incident as Normal, Probe, DOS, U2R and R2L.

The detection process is undertaken by interacting with the user through presenting a serious of questions for the user. The system asks the user by displaying questions containing attributes with their values. The user is expected to reply for the questions. Once CBNIDS detects network incident as Probe, DOS, U2R and R2L attacks, it display the result.

6.1 Network Attack Diagnosis

The diagnosis of network incident is aimed at identifying an attack as R2L, Probe, DOS, U2R or normal incidents. It is based on case based reasoning knowledge based system. Figure 6.1 depicts a set of sample instances of Normal, Probe, DOS, U2R and R2L.

[Description: [0 , tcp , http , SF , 209 , 301 , 0 , 0 , 0 , 209 , 1.0 , 0 , 0.0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 9 , 0.0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 8 , 9 , 0.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.22 , 60 , 255]][Solution: [normal. , case5 , 12.159955]]

[Description: [0 , udp , domain_u , SF , 41 , 91 , 0 , 0 , 0 , 41 , 1.0 , 0 , 0.0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 5 , 0.0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 5 , 5 , 0.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.0 , 6 , 227]][Solution: [DOS. , case1 , 12.159955]]

[Description: [0 , tcp , http , SF , 247 , 353 , 0 , 0 , 0 , 247 , 1.0 , 0 , 0.0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 4 , 0.0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 4 , 4 , 0.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.0 , 75 , 255]][Solution: [Probe. , case4 , 12.159955]]

[Description: [0 , tcp , http , SF , 235 , 1261 , 0 , 0 , 0 , 235 , 1.0 , 0 , 0.0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 14 , 0.0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 9 , 14 , 0.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.29 , 80 , 112]][Solution: [U2R. , case2 , 12.159955]]

[Description: [0 , tcp , smtp , SF , 1462 , 499 , 0 , 0 , 0 , 1462 , 0.94 , 0 , 0.0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0.0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0.0 , 0.0 , 0.0 , 0.0 , 1.0 , 0.0 , 0.0 , 34 , 245]][Solution: [R2L. , case3 , 12.159955]]

Figure 6.1 Normal, Probe, DOS, U2R and R2L cases

6.1.1 Define query

In this step the user defines the query to the system. To define which are the values of the different attributes of a network attacks: duration, protocol_type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent, hot, num_failed_logins, is_host_login, is_guest_login, count and so on.

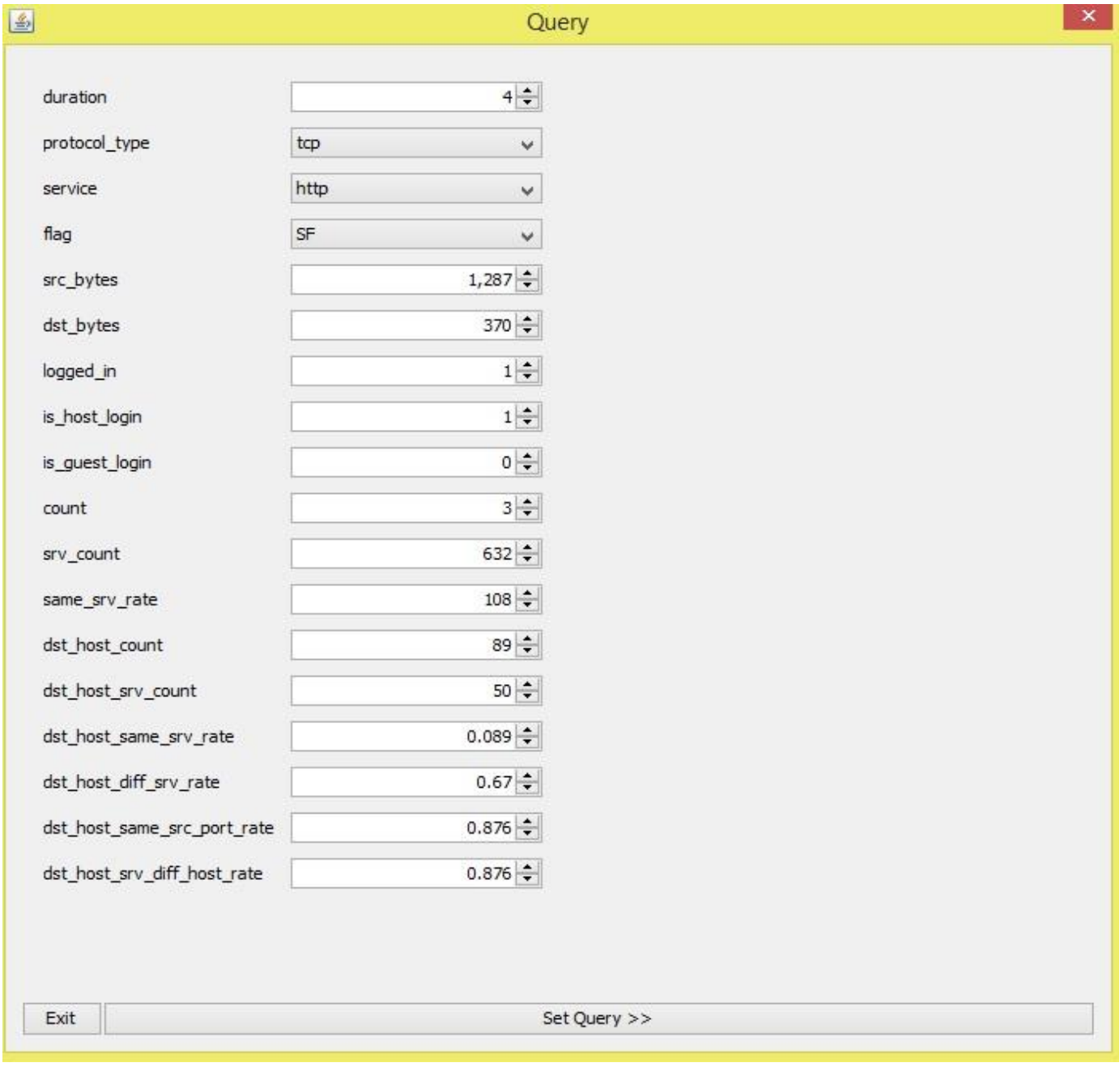


Figure 6.2 Define Query step

6.1.2 Retrieve Cases

This step shows the retrieved case by the k-NN method. It shows each case and its similarity to the query. The system will retrieve five best match cases.

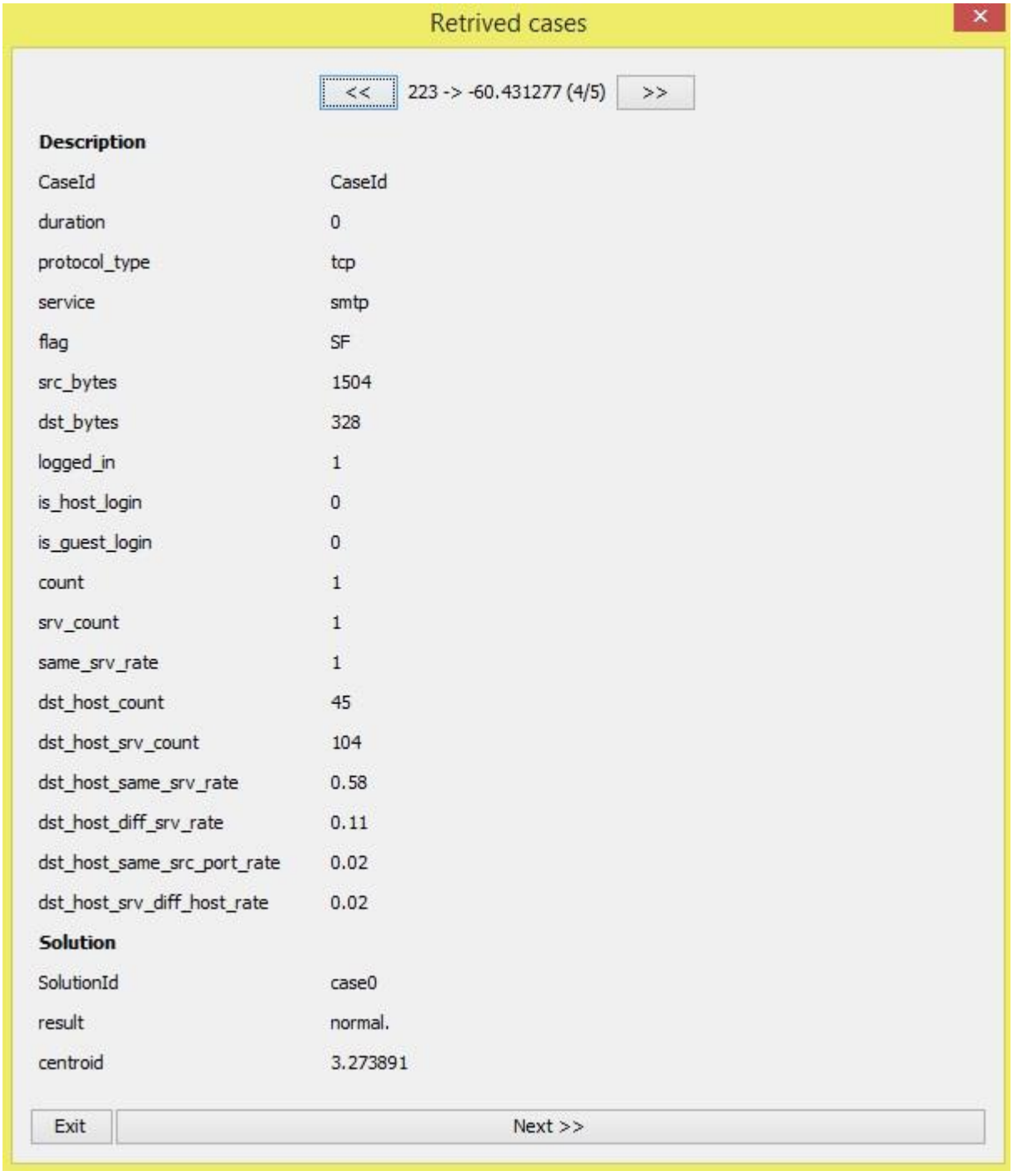


Figure 6.3 Retrieved cases

6.1.3 Case Adaption

In this step the system adapts the retrieved cases to the requirements of the user depending on the values defined in the query. This system will use simple direct proportions to perform the adaptation. After adapting the solution of the cases, their description can be substituted by the description of the query. At this point, the system will manage a list of working cases that are different from the cases in the case base. These working cases represent possible solutions to the problem described in the query.

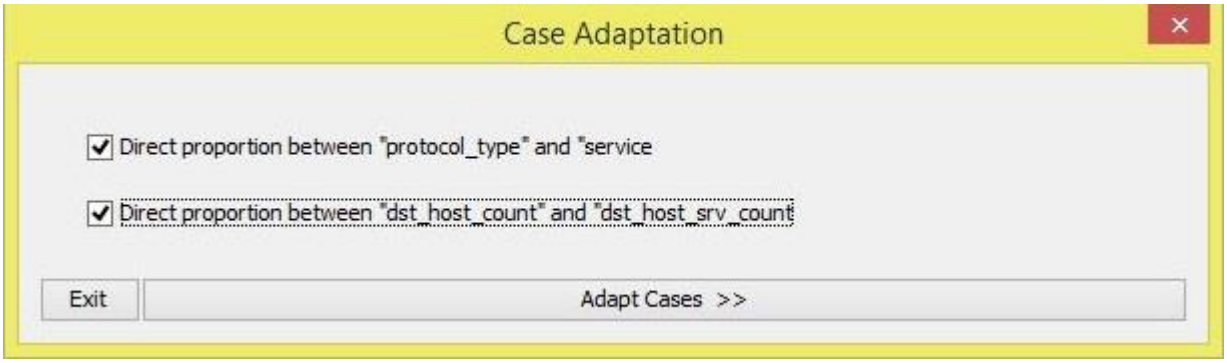


Figure 6.4 Case adaptation steps

6.1.4 Case Revise

Once cases have been adapted, the user (or a domain expert) would adjust the values of the working cases in a manual way. If the retrieved cases are not similar enough to the requirements of the query the user (or domain expert) has to define manually the solution of the case. Figure 6.5 shows case revise task of the system.

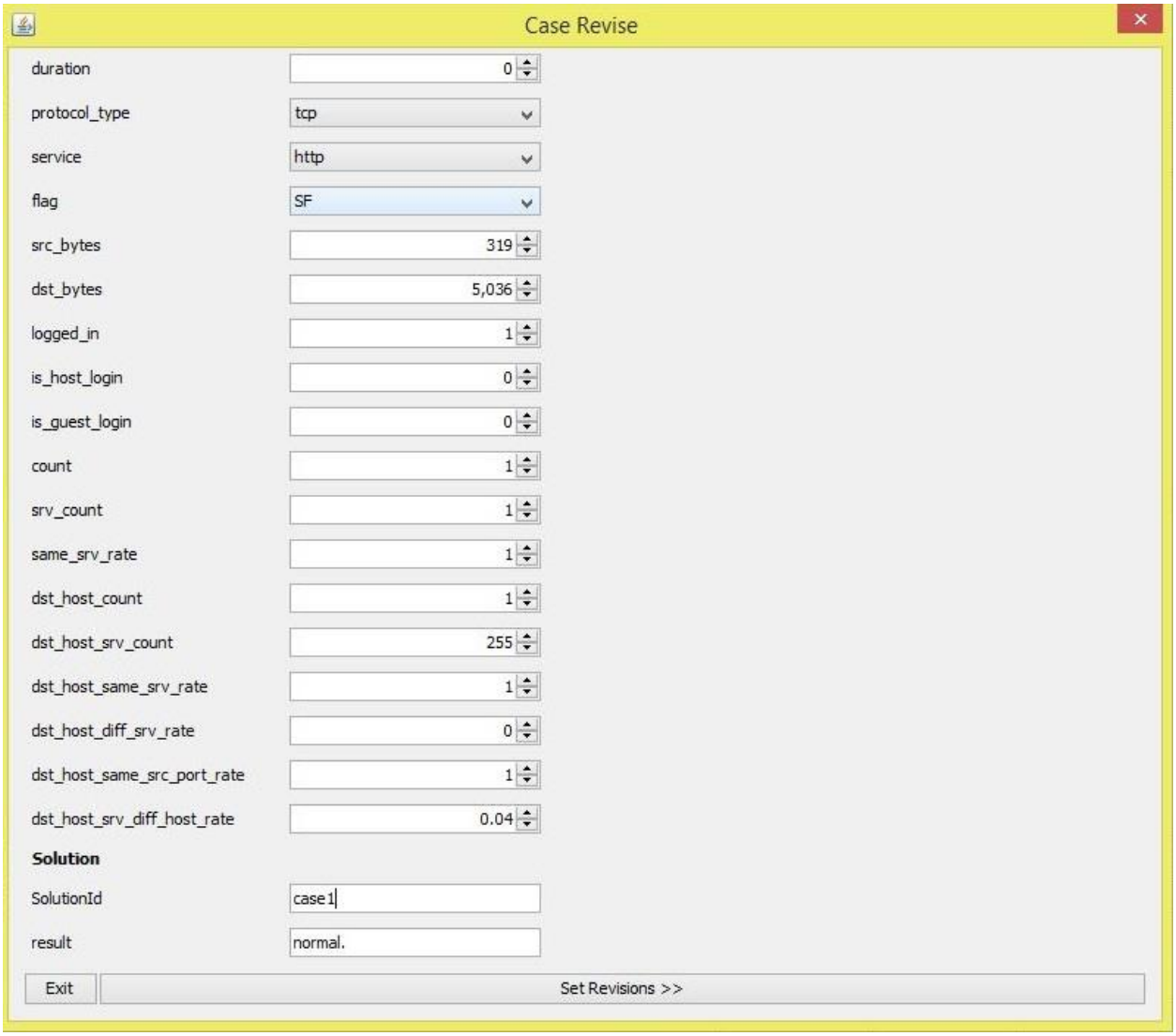


Figure 6.5 Revise case for retain

6.1.5 Case Retain

Finally, the user (or domain expert) would save the new network attack case into the case base for being used in future queries. If it is done, a new Id must be assigned to the attack. Figure 6.6 shows the case retain task.

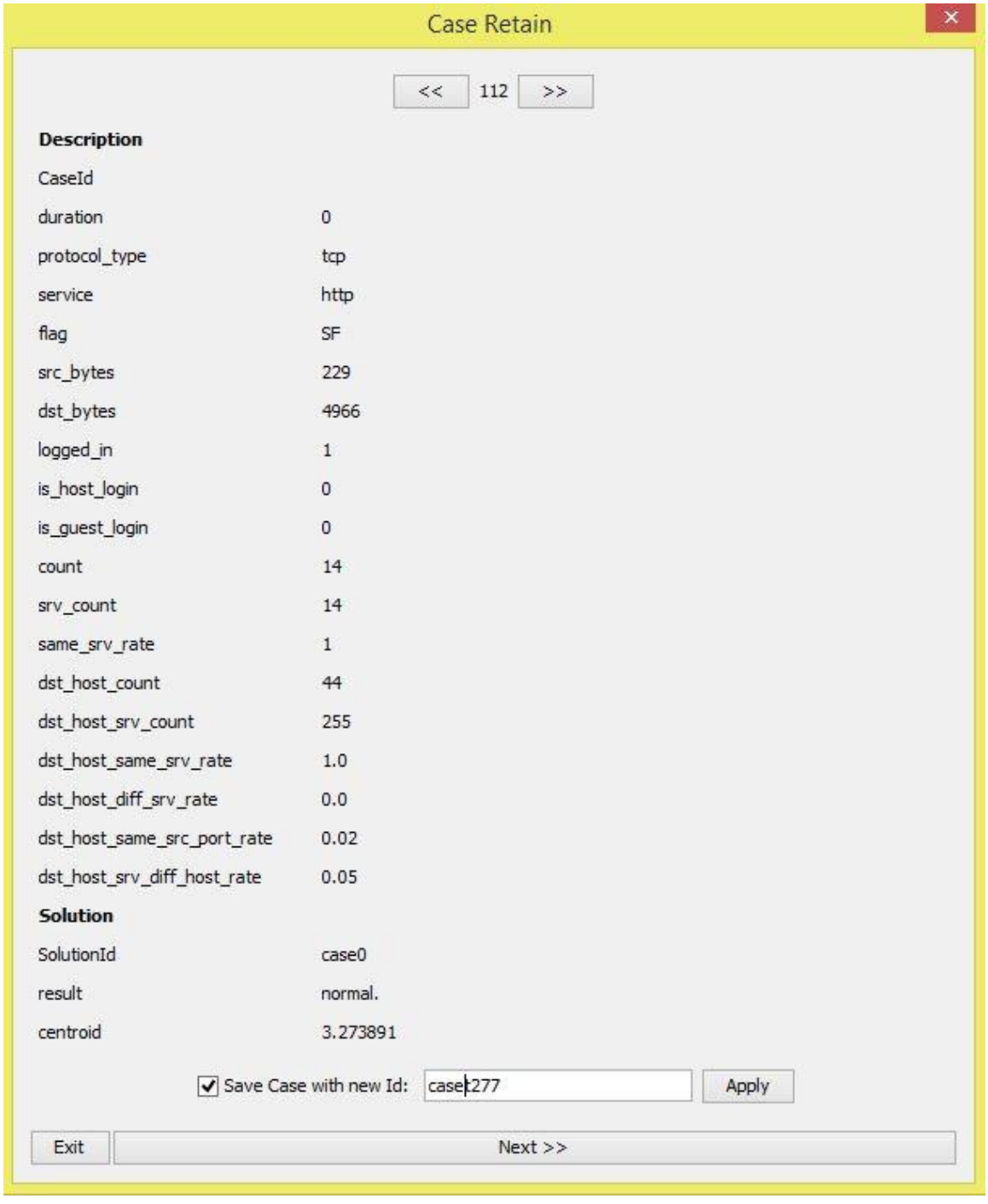


Figure 6.6 Case retain

6.2 Testing and Evaluation of CBNIDS

In order to assure CBNIDS meets the requirement it is developed for, it has to be tested. For evaluating the performance of the system, test cases are prepared and given to the system. The outputs of the system are compared against judgment of domain area experts'.

6.2.1 System Performance Testing

Testing was made to know how the system performs well and to evaluate whether the system selects the appropriate cases from the case-base. So, each test case was presented to the system individually, and the set of selected case was evaluated against the expected selection to check the performance of the similarity measures. The two weeks of test dataset yielded around two million connection records and by considering the number of attributes and the time, the researcher prepared only 30 test cases using datapreparator data preprocessing tool. Attributes of instances with their respective values describe the behavior of certain network incident.

Statistical analysis is used for evaluating the performance of the system. In order to perform statistical analysis of the results, the Precision, Recall and F-measure measurements are applied. Confusion matrix is used for comparing the performance of CBNIDS with test cases.

	CBNIDS						
Test Cases	**Attacks class**	**Normal**	**Probe**	**DOS**	**R2L**	**U2R**	**Total**
	Normal	7	0	0	2	0	9
	Probe	0	5	0	0	0	5
	DOS	0	1	6	0	0	7
	R2L	1	0	0	3	0	4
	U2R	0	0	0	0	5	5
	Total	8	6	6	5	5	30

Table 6.2 Confusion matrix for evaluation of CBNIDS compared to test case

The confusion matrix on table 6.2 shows matrix of test cases evaluation by CBNIDS and test cases. The rows illustrate test case and the columns illustrate result of CBNIDS.

The entry under column Normal indicates that the system identified seven instances as normal. The entries under column Probe testified that out of five instances, all of the instances are correctly identified as Probe attack. The entries for DOS and R2L column show that the system has correctly identified six instances as DOS and three instances R2L attack types respectively. The entries in the confusion matrix under U2R column depict that four instance are correctly identified. R2L attack type diagnosis the system achieved the lowest result as compared to the others.

It is clear that each type of attacks has their own way of attacking and causing damages to the victim computer. Identification of each class of attacks to their correct class is important to network administrators. But as shown in the confusion matrix, 2 Normal instances are incorrectly identified as R2L attacks, 1 DOS is identified as Probe and 1 R2L is identified as Normal. The researcher believes that the system's identification of these instances as an attack as strength though it is not to their correct class.

The system has correctly detected 27 test instances out of 30 to their correct class. This means the system has 86.66% detection accuracy. And four instances out of 30 are incorrectly classified which is 13.34%. But, this measure alone is not enough to measure performance of the system since it only tells us the overall performance. Hence Precision and Recall are employed to evaluate the system performance apart from detection accuracy. Recall is the proportion of real positive cases that are correctly predicted positive. Precision denotes the proportion of predicted positive cases which are correctly real positives [61].

As clearly illustrated in table 6.2, the system's performance is evaluated in term of Precision, Recall and F-measure which enables us to view in detail how accurate is the system is identifying network attacks and normal incidents.

	Precision	Recall	F-Measure	Attack Class
	0.87	0.77	0.80	Normal
	0.83	1	0.90	Probe
	1	0.87	0.93	DOS
	0.6	0.75	0.66	R2L
	1	1	1	U2R
Average	0.86	0.88	0.86	

Table 6.2 Performance evaluation based on Precision, Recall and F-measure

The system has registered its highest 100% Recall for Probe and U2R attacks and 77% for Normal, 87% for DOS and 75% for R2L attacks. The system registered its highest 100% Precision for DOS and U2R respectively. The system has 87% Precision for Normal, 83% for Probe and 60% for R2L. As compared to Precision, the system has performed well in its Recall for identifying network incidents accurately. To sum up, CBNIDS has an accuracy of 87.3%.

6.2.2 User Acceptance Testing

The aim of undertaking user acceptance testing is that to make sure how the CBNIDS is performing on usable by users.

Five domain experts were selected to test the system and all experts are from Addis Ababa University ICT director office. The experts, after providing training how CBNIDS is works, given test cases to use and evaluate the system.

The evaluators assessed CBNIDS by using the following standards

- Is the system efficient in time?
- Relevancy of the retrieved cases for new case
- Simplicity to use and interact with the system.
- Rate the significance of the system in the domain area
- Relevance of the attributes in representing network attacks
- The performance of case revision and solution adaptation

Evaluation of the system using user acceptance testing is making sure how the user or domain experts' view the system on the base of the abovementioned evaluation standards. Different researchers have used different types of user acceptance testing evaluation criteria. But for this study, the evaluation criteria suggested by Kolodner [14,]Abdilkerim [20], Tigabu [17] and Solomon [45] customized and used to ease the evaluation process, the weights scale suggested by Solomon [45], has been used such that Poor=1, Fair=2, Good=3, Very good=4 and Excellent =5

Table 6.3 depicts that summary of domain experts' evaluation of the system. The value indicates numbers of evaluator who evaluated the system as poor, fair, good, very good and excellent with respect to the evaluation criteria.

Thus, 100% of the evaluators replied excellent for efficient response of the system. On the other hand 60% and 80% of respondents good and very good. Thus, the relevancy of the retrieved case for new problem, with regard to the second question 60%, 80% and 100% of evaluator suggest good, very good and excellent.

The third question is about the simplicity to use and interactivity of the system. Two of the evaluator gives 80% and other 100%. The fourth evaluation criteria is about the significance of the system in the domain area. All of the evaluators (100%) agreed that, the system is significant on the domain area. Whereas 100%, evaluator scored the relevance of the attributes in the representation network attacks as excellent and 80% gives very good and 60% give good.

No	Criteria of evaluation	poor	Fair	Good	Very Good	Excellent	Average
1.	Is the system efficient in time	0	0	2	2	1	3.2
2.	Relevancy of the retrieved cases for new case	0	0	2	2	1	3.8
3.	Simplicity to use and interact with the system.	0	0	0	2	3	4.6
4.	Rate the significance of the system in the domain area	0	0	0	0	5	5
5.	Relevance of the attributes in representing network attacks	0	0	0	3	2	4.5
6.	The performance of case revision and solution adaptation	0	1	2	1	1	2.8
		Total Average					4.78

Table 6.3 Result of users' acceptance evaluation

The evaluation criteria are the performance of case revision and solution adaptation. Among the evaluator, 100% replied excellent, 80% replied very good, 60% replied good and 40% replied fair. Finally, according to the evaluation filled by domain experts', Case base network intrusion detection system has registered 4.78 out of 5 (95.5%), which is taken to very good achievement.

In general, the significance of CBR in network intrusion detection has achieved an encouraging result. In addition, the significance of the system in the domain area, rated by the respondents with a highest rating value. Domain experts assign a lower value compared to others is the performance of case revision and solution adaptation. Since the prototype learns over time, the case revision and solution adaptation will grow quickly. One interesting feedback provided by the domain experts is adding explanation facilities are in the system is very important. According to the domain experts, explanation facility is important in the adaptation of the retrieved solution.

If users of the system get a more explanation about the retrieved solution and the problems itself, the can easily decide whether to use or not use the retrieved solution. According to the experts, the user should get more explanation about the attacks such as adverse and instruction of the network attacks.

CHAPTER SEVEN

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

In this study, an attempt is made to explore the possibilities of integrating data mining with case based reasoning approaches for developing network intrusion detection system. For the data mining sample intrusion dataset is taken form KDDcup'99. The dataset is preprocessed and made suitable for data mining. During the preprocessing task of instances are classified as normal and attack, into different distributions, while attacks were classified into four sections represented into DOS, Probe, U2R and U2L. For acquiring knowledge from the dataset, an automatic knowledge acquisition using data mining specifically clustering for descriptive modelling is proposed in this study. Clustering provides natural groups of instance with intrusion dataset. Hence, clustering algorithms are computed such as, k-means clustering, expectation maximization (EM) clustering and Make Density Based Clusterer approaches are employed. Experimental results show that k-means clustering outperform others with an accuracy of 89.43%.

The case base was developed using COLIBRI Studio IDE, which is the most compatible and reliable CBR tool to deal with case based system. The induction of network attack signatures and automatic knowledge acquisition for constructing case base are performed simultaneously by automatic integrator application. A case is generated and the clustered instances with centroid values are added into the case base automatically.

Following the successful integration of data mining with case based system, case based network intrusion detection is built. The system has the capability of retrieve, reuse, revise and retain cases. The retrieval task starts with a new case description and ends when best matching similarity case of attacks or normal cases has been found in the case base. As users enter case description (query) through interface, the system searches the best matching cases from the case base and return possible solution in ranked order to the problems described in the query. If exact matching occurred in between the query and cases in the case base, users can directly derive the solution or if the similarity or matching is partial, then adaptation, revision and reuse tasks are done to make the proposed solution best for the problem at hand. At last, those best modified solution to the problem stored to the case base for future use.

System performance testing shows that the case based network intrusion detection system registered 87% accuracy with very good Precision and Recall 86% and 88% respectively. User acceptance testing is performed with an average score of 95.5% based on user acceptance evaluation. The system has showed 87% and 95.5% overall accuracy according to the system performance and user acceptance testing respectively.

In this study promising result is achieved in integrating clustered intrusion using data mining with case based reasoning for network intrusion detection system. The application has the ability of accommodating changes and automatically updating the knowledge base. Some challenge have been happened, which is the number of attribute that has been used in 45 which is prepares manually and time taken process during on the system integration.

7.2 Recommendation

Hence, the researcher believes further researches have to be done for improvement of data mining with case based reasoning and the following are recommended for future works:-

- Building hybrid knowledge based system which is capable of employing rule based reasoning and case based reasoning with integrated data mining techniques.
- The case based network intrusion detection system detects the major attack types, such as Probe, DOS, U2R and R2L. But there are 21 types of network attacks form the KDDcup'99 dataset. For future work, it is recommended to design case based network intrusion detection system which directly detects attacks separately.
- This study was carried out using clustering technique of simple K-means. So further investigation needs to be done using other clustering algorithms such as Hierarchical and Support Cobweb.
- As the domain experts suggested that the system need explanation facility is important in the adaptation of the retrieved solution. For future work the system that provides advice and information based on the adaptation of case.

- Nowadays the behaviors of attacks are frequently changing. To understand such attacks, it is needs to further investigate ontology based retrieval for enhance the performance of case based reasoning for designing network intrusion detection system.

References

[1] S.K Miller, "An Introduction to Computer Security," IEEE Computer, no.34, 2001.

[2] Ahmed H. Fares and Mohamed I. Sharawy, "Intrusion Detection: Supervised Machine Learning," Computing Science and Engineering, vol. 5, no. 4, pp. 305-313, December 2011.

[3] Usman A., Sajjad H., Salman N., and Obaid U., "A Survey of Intrusion Detection and Prevention Techniques," *International Conference on Information Communication and Management*, vol. 16, pp. 23-36, 2011.

[4] Sumit M., Mary M., Anupam J., and Tim F, "A Knowledge-Based Approach To Intrusion Detection Modeling," in *Computer Science and Electrical Engineering*, Maryland, 2006, pp. 19-30.

[5] Hervé D, "An Introduction to Intrusion-Detection Systems," in *Proceeding of connect'2000,* Zurich, 2002, pp. 1-18, IBM Research, IBM Reseach Laboratory.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth., "Knowledge discovery and data mining: Towards a unifying framework," in *Knowledge Discovery and Data Mining*, California, 1996, pp. 82-88.

[7] Jiawei H. and Micheline K., *Data Mining: Concepts and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2006.

[8] Huy A. and Deokjai C., "Application of Data Mining to Network Intrusion Detection: Classifier Selection Model," in *APNOMS*, Berlin, 2008, pp. 399–408.

[9] Carsten P, "Integrating and Updating Domain Knowledge with Data Mining," *Leipzig Graduate School of Management*, vol. III, pp. 50-62, June 2000.

[10] Kumar Ranjit, *Research Methodology-A Step-by-Step Guide for Beginners*, 2nd ed. Singapore: SAGE, 2005.

[11] SEO and PPC Mangement Solution. (2015, February) SEO & PPC Mangemnet Solution. [Online]. Http://www.theecommercesolution.com/useful_links/KBS_Data_Mining.php

[12] Jim Prentzas and Ioannis Hatzilygeroudis, "Categorizing Approaches Combining Rule-Based and Case-Based Reasoning," vol. 24, pp. 97-122, May 2007.

[13] L. Lewis, "Managing Computer Networks: A Case-Based Reasoning Approach," in *Artech House Publishers*, Boston, 1995, pp. 25-37.

[14] Janet Kolodner, *Case-Based Reasoning*, 2nd ed. California, San Mateo: organ Kaufmann, 1992.

[15] Paul Dokas and Levent Ertoz, "Data Mining for Network Intrusion Detection," *IEEE Computer*, vol. 2nd, no. 12, pp. 18-31, 2001.

[16] Armin Stahl and Thomas R., "Rapid Prototyping of CBR Applications with the open source tool myCBR," in *German Research Center for Artificial Intelligence*, Berlin, pp. 615-629, 2008.

[17] Tigabu Dagne, "Constructing Predictive Model for Network Intrusion Detection," School of Information Science, Addis Ababa University, Addis Ababa, MSc Thesis 2012.

[18] Agnar Aamodt, "Case-Based Reasoning - An Introduction," in Case Based Approchces , Dragvoll, vol. 6, no.3, pp. 39-49, August 1991.

[19] Ferri C., Flach P., and Henrandez-Orallo J, "Learning Decision Trees Using Area under the ROC Curve," in *the 19th International Conference on Machine Learning*, Chicago, pp. 139-146, 2002.

[20] Abdilkerim M, "Towards Integrating Data Mining with Knowledge Based System: The Case of Network Intrusion Detection," School of Information Science, Addis Ababa University, Addis Ababa, MSc Thesis 2013.

[21] Prof. Mihaela, "On the Use of Data-Mining Techniques in Knowledge-Based Systems," in *Economy Informatics*, Romania, pp. 22-24, 2006.

[22] Tom M. Mitchell, *Machine Learning*, 1st ed. Chicago: McGraw-Hill Science, 1997.

[23] Watson I, "Case-based reasoning is a methodology not a technology," in *Knowledge-Based Systems*, vol. 12, no. 5, pp. 303–308, 1999.

[24] Erik Stiklestad, "Reusing External Library Components in the Creek CBR System". Norwegian University of Science and Technology Department of Computer and Information Science, MSc Thesis 2007.

[25] Karen Scarfone and Peter Mell, *Guide to Intrusion Detection and Prevention Systems (IDPS)*, 1st ed. U.S.: National Institute of Standards and Technology, 2007.

[26] *Intrusion Detection Systems*, 6th ed. New York: Information Assurance Tools Report IATR, 2009.

[27] Shaik Akbar, K.Nageswara Rao, and J.A.Chandulal, "Intrusion Detection System Methodologies Based on Data Analysis," *International Journal of Computer Application*, vol. 5, no. 2, pp. 44-59, 2010.

[28] Chris Petersen, "An introduction to network and host based network intrusion detection," in *LogRhythm*, 2006, pp. 1-12.

[29] D.P.Gaikwad, Pooja Pabshettiwar, Priyanka Musale, Pooja Paranjape, and Ashwini S. Pawar, "A Proposal for Implementation of Signature Based Intrusion Detection System Using Multithreading Technique," *AISSMS*, vol. 27, pp. 56-70, January 2012.

[30] Bilal Maqbool Beigh and Prof.M.A.Peer, "Intrusion Detection and Prevention System: Classification and Quick Review," *ISSN*, vol. 2, no. 7, pp. 50-64, July 2012.

[31] Amrita Anand and Brajesh Patel, "An Overview on Intrusion Detection System and Types of Attacks It Can Detect Considering Different Protocols," *Advenced Research in Computer Science and Software Engineering*, vol. 2, no. 8, pp. 94-98, September 2012.

[32] Osmar R. Zaïane, "Introduction to Data Mining," *Principles of Knowledge Discovery in Databases*, vol. 6, no. 2, pp. 78-89, October 1999.

[33] Two Crows Corporation, "Introduction to Data Mining and Knowledge Discovery," vol. 10, no. 13, pp. 1-36, October 1999.

[34] Vivek Bhambri, "Application of Data Mining in Banking Sector," *IJCST*, vol. 2, no. 2, pp. 23-37, April 2011.

[35] Priti Srinivas and Rajendra Akerkar, "Advanced Knowledge Based Systems:Model," *Knowledge-Based Systems for Development*, vol. 1, no. 1, pp. 45-60, July 2010.

[36] Julie Mandy, Tharam D., and Simon S, "A Tutorial on Case-Based Reasoning Soft Computing in Case Based Reasoning," *Knowledge-based systems*, vol. 17, no. 4, pp. 1-28, August 2001.

[37] Yongjian Fu, "Data Mining: Tasks, Techniques and Appliactions," in *Introduction to Data Mining and its appliaction.* Berlin, Germany:Springer Berlin Heidelberg, 2006, ch.7, pp.195-216.

[38] Janet Kolodner, *Case-Based Reasoning*, 1st ed. San Mateo, California: Morgan Kaufmann, 1993.

[39] Juan Anand, "COLIBRI Studio Tutorial," Group for Artificial Intelligence Applications Universidad Complutense De Madrid, August 2012.

[40] Nasrullah Iqbal, "Evaluation of jCOLIBRI," Malardalen University, Sweden, MSc Thesis 2006.

[41] Ramon López De Mántaras, David Mcsherry, Derek Bridge, and David Leake, "Retrieval, reuse, revision, and retention in case based reasoning," *The Knowledge Engineering Review*, vol. 1, no. 2, pp. 1-23, January 2005.

[42] Pantic Maja, "Introduction to Machine Learning & Case-Based Reasoning," in *Machine Learning*, London, 2001, pp. 56-70.

[43] Sankar K. Pal and Simon C. K. Shiu, "Foundations of Soft Case-Based Reasoning," vol. 14, no. 3, pp. 71-86, September 2004.

[44] Shi Zhong, Taghi Khoshgoftaar, and Naeem Seliya, "Clustering-Based Network Intrusion Detection," in *Computer Science And Engineering*, Florida, 2004.

[45] Solomon Gebremariam, *A Self-Learning Knowledge Based System For Diagnosis And Treatment Of Diabetes*, School of Information Science, Addis Ababa University, Addis Ababa, MSc Thesis 2013.

[46] Alemu Jorgi, *A Case-Based Approaches for Designing Knowledge Based System for AIDS Resource Center (ARC): The Case of Warmline Clinician Consultation Service*. School of Information Science, Addis Ababa University, Addis Ababa, MSc Thesis 2010.

[47] S.Sathya Bama, M.S.Irfan Ahmed, and A.Saravanan, "etwork Intrusion Detection using Clustering: A Data Mining Approach," *Computer Applications*, vol. 30, no. 4, pp. 1-23, September 2011.

[48] Ehsan Hajizadeh, Hamed Davari Ardakani, and Jamal Shahrabi, "Application of data mining techniques in stock markets," *Economics and International Finance*, vol. 19, no. 18, pp. 1-18, March 2010.

[49] (2015, January) datapreparator. [Online]. www.datapreparator.com

[50] Stoyanov S, Govedarova N, and Popchev I., "CBR-Based Search in BULCHINO Catalogue," University of Plovdiv, Bulgaria, 2005.

[51] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2005.

[52] G.H John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Uncertainty in Artificial Intelligence*, California, 1995, pp. 50-62.

[53] J. Quinlan, "C4.5: Programs for Machine Learning," in *Morgan Kaufmann Series in Machine Learning*, San Mateo, 1992, pp. 1-19.

[54] Kohavi R, "Scaling up the accuracy of naïve-bayes classifier: A decision-tree hybrid," in *Knowledge Discovery and Data Mining*, Menlo Park, 1996, pp. 202-207.

[55] P Werbos, *New Tools for Prediction and Analysis in the Behavioral Sciences*., 1994.

[56] A. Atanassov and L. Antonov, "Comparative Analysis Of Case Based Reasoning Software Frameworks jCOLIBRI and myCBR," *Chemical Technology and Metallurgy*, vol. 47, no. 1,

pp. 83-90, Feburary 2012.

[57] A. & Roth-Berghofer Stahl, "Rapid Prototyping of CBR Applications with the open source tool myCBR," in *German Research Center for Artificial Intelligence (DFKI)*, Berlin, 2005, pp. 97-105.

[58] (December, 2014) Association of Computing Machinery. [Online]. http://www.sigkdd.org/kddcup/index.php?section=1999&method=data

[59] Mohammad Khubeb Siddiqui and Shams Naahid, "Analysis of KDD CUP 99 Dataset using Clustering based Data Mining," *Database Theory and Application*, vol. 6, no. 5, pp. 23-34, June 2013.

Appendix I

Class of attack	**Attack type**
R2L	ftp_write
	Warezmaster

	guess_passwd
	Imap
	Multihop
	Spy
	warezclient
DOS	back
	neptune
	pod
	smurf
	teardrop
	land
Probe	satan
	portsweep
	nmap
	ipsweep
U2R	buffer_overflow
	loadmodule
	rootkit

Appendix II

Attributes relation and data declaration

@relation 'data set for CBNID'

@attribute duration numeric

@attribute protocol_type {tcp,udp,icmp}

@attribute service {http, smtp, finger, ntp_u, ecr_i, auth, domain_u, ftp, telnet, eco_i, private, ftp_data, pop_3, mtp, domain, ctf, name, systat, time, nnsp, http_443, echo, ssh, whois, remote_job, link, gopher, hostnames, csnet_ns, uucp_path, netbios_ns, vmnet, Z39_50, iso_tsap, pop_2, other, printer, efs, courier, uucp, discard, bgp, klogin, netbios_dgm, netbios_ssn, ldap, shell, daytime, supdup, nntp, imap4, login, kshell, rje, sql_net, exec, urh_i, urp_i, netstat, IRC, sunrpc, X11}
@attribute flag {SF, S1, RSTO, REJ, RSTOS0, S0, SH, RSTR, S2, S3}
@attribute src_bytes numeric
@attribute dst_bytes numeric
@attribute land numeric
@attribute wrong_fragment numeric
@attribute urgent numeric
@attribute hot numeric
@attribute num_failed_logins numeric
@attribute logged_in numeric
@attribute num_compromised numeric
@attribute root_shell numeric
@attribute su_attempted numeric
@attribute num_root numeric
@attribute num_file_creations numeric
@attribute num_shells numeric
@attribute num_access_files numeric
@attribute num_outbound_cmds numeric
@attribute is_host_login numeric
@attribute is_guest_login numeric
@attribute count numeric
@attribute srv_count numeric
@attribute serror_rate numeric
@attribute srv_serror_rate numeric
@attribute rerror_rate numeric
@attribute srv_rerror_rate numeric
@attribute same_srv_rate numeric
@attribute diff_srv_rate numeric
@attribute srv_diff_host_rate numeric
@attribute dst_host_count numeric
@attribute dst_host_srv_count numeric
@attribute dst_host_same_srv_rate numeric
@attribute dst_host_diff_srv_rate numeric

@attribute dst_host_same_src_port_rate numeric

@attribute dst_host_srv_diff_host_rate numeric

@attribute dst_host_serror_rate numeric

@attribute dst_host_srv_serror_rate numeric

@attribute dst_host_rerror_rate numeric

@attribute dst_host_srv_rerror_rate numeric

@attribute result {normal.,DOS,R2L,Probe,U2R}

@data

0,tcp,http,SF,230,591,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,21,21,0,0,0,0,1,0,0,78,213,1,0,0.01,0.03,0,0,0,0,normal.

0,tcp,http,SF,203,1626,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,9,17,0,0,0,0,1,0,0.12,28,255,1,0,0.04,0.05,0,0,0,0,normal.

0,tcp,http,SF,307,1366,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,25,25,0,0,0,0,1,0,0,25,255,1,0,0.04,0.04,0,0,0,0, Normal.

0,tcp,http,SF,330,2164,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,18,18,0,0,0,0,1,0,0,72,255,1,0,0.01,0.04,0,0,0,0, Normal

0,tcp,http,SF,324,3609,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,19,19,0,0,0,0,1,0,0,73,255,1,0,0.01,0.04,0,0,0,0, Normal

0,tcp,http,SF,227,296,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,7,8,0,0,0,0,1,0,0.25,99,255,1,0,0.02,0.04,0,0,0,0, Normal

0,tcp,http,SF,225,2000,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,18,19,0,0,0,0,1,0,0.11,110,255,1,0,0.02,0.04,0,0,0,0, Normal

0,tcp,http,SF,312,2195,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,8,0,0,0,0,1,0,0,255,255,1,0,0,0,0,0,0,Normal

0,tcp,http,SF,294,3369,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,11,11,0,0,0,0,1,0,0,11,255,1,0,0.09,0.01,0,0,0,0, Normal

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,112,8,1,1,0,0,0.07,0.06,0,255,8,0.03,0.07,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,134,8,1,1,0,0,0.06,0.06,0,255,8,0.03,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,120,4,1,1,0,0,0.03,0.07,0,255,4,0.02,0.07,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,120,16,1,1,0,0,0.13,0.07,0,255,16,0.06,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,7,1,1,0,0,0.05,0.07,0,255,7,0.03,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,137,11,1,1,0,0,0.08,0.07,0,255,11,0.04,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,138,12,1,1,0,0,0.09,0.07,0,255,12,0.05,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,150,20,1,1,0,0,0.13,0.06,0,255,20,0.08,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,129,14,1,1,0,0,0.11,0.06,0,255,14,0.05,0.07,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,132,17,1,1,0,0,0.13,0.06,0,255,17,0.07,0.06,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,8,1,1,0,0,0.07,0.07,0,255,8,0.03,0.07,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,105,3,1,1,0,0,0.03,0.08,0,255,3,0.01,0.07,0,0,1,1,0,0,DOS

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,106,4,1,1,0,0,0.04,0.08,0,255,4,0.02,0.07,0,0,1,1,0,0,DOS

Appendix III

Attribute description with their data type

Feature name	Description	Type

duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	continuous
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	discrete
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	

count	number of connections to the same host as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-host connections.	
serror_rate	% of connections that have ``SYN'' errors	continuous
rerror_rate	% of connections that have ``REJ'' errors	continuous
same_srv_rate	% of connections to the same service	continuous
diff_srv_rate	% of connections to different services	continuous
srv_count	number of connections to the same service as the current connection in the past two seconds	continuous
	Note: The following features refer to these same-service connections.	
srv_serror_rate	% of connections that have ``SYN'' errors	continuous
srv_rerror_rate	% of connections that have ``REJ'' errors	continuous
srv_diff_host_rate	% of connections to different hosts	continuous

Appendix IV

Dear Evaluator,

This evaluation form is prepared aiming at measuring to what exted does the CBNIDS is useable and acceptable by end users in the asrea of network administration. Therefore, you are kindly requested to evaluate the system by labeling (X) symbol on the space provided for the corresponding attributes values for each criteria of evaluation.

I would like to appreciate your collaboration in providing the informaiton.

Note:- the vaues for all attributes in the table arer realted as: Excellent=5, Very good=4, Good=3, Fair=2, and Poor=1.

No	**Criteria of evaluation**	**poor**	**Fair**	**Good**	**Very Good**	**Excellent**	**Average**
1.	Is the system efficient in time						
2.	Relevancy of the retrieved cases for new case						
3.	Simplicity to use and interact with the system.						
4.	Rate the significance of the system in the domain area						
5.	Relevance of the attributes in representing network attacks						
6.	The performance of case revision and solution adaptation						
		Total					

Appendix V

Description of case attributes

Case Attributes	Features	Data types	weight	Local Similarity
Duration	Continuous	Integer	1	Interval
protocol_type	Discrete	String	1	Equal
Service	Discrete	String	1	Equal
Flag	Discrete	String	1	Equal
src_bytes	Continuous	Integer	1	Interval
dst_bytes	Continuous	Integer	0.9	Interval
Land	Discrete	Integer	0.9	Interval
wrong_fragment	Continuous	Integer	0.9	Interval
Urgent	Continuous	Integer	0.9	Interval
Hot	Continuous	Integer	0.9	Interval
num_failed_logins	Continuous	Integer	0.9	Interval
logged_in	Discrete	Integer	0.9	Interval
num_compromised	Continuous	Integer	0.9	Interval
root_shell	Continuous	Integer	0.9	Interval
su_attempted	Continuous	Integer	0.9	Interval
num_root	Continuous	Integer	0.9	Interval
num_file_creations	Continuous	Integer	0.9	Interval
num_shells	Continuous	Integer	0.9	Interval
num_access_files	Continuous	Integer	0.9	Interval
num_outbound_cmds	Continuous	Integer	0.9	Interval
is_host_login	Discrete	Integer	1	Interval
is_guest_login	Discrete	Integer	1	Interval
Count	Continuous	Integer	1	Interval
srv_count	Continuous	Integer	1	Interval
serror_rate	Continuous	Double	1	Interval

srv_serror_rate	Continuous	Double	1	Interval
rerror_rate	Continuous	Double	1	Interval
srv_rerror_rate	Continuous	Double	1	Interval
same_srv_rate	Continuous	Double	1	Interval
diff_srv_rate	Continuous	Double	1	Interval
srv_diff_host_rate	Continuous	Double	1	Interval
dst_host_count	Continuous	Integer	1	Interval
dst_host_srv_count	Continuous	Integer	1	Interval
dst_host_same_srv_rate	Continuous	Double	1	Interval
dst_host_diff_srv_rate	Continuous	Double	1	Interval
dst_host_same_src_port_rate	Continuous	Double	1	Interval
dst_host_srv_diff_host_rate	Continuous	Double	1	Interval
dst_host_serror_rate	Continuous	Double	1	Interval
dst_host_srv_serror_rate	Continuous	Double	1	Interval
dst_host_rerror_rate	Continuous	Double	1	Interval
dst_host_srv_rerror_rate	Continuous	Double	1	Interval

Appendix VI

Java code for k-means clusterign algorithms

```
import java.io.*;
import java.util.Arrays;
import weka.clusterers.SimpleKMeans;
import weka.core.*;
import weka.clusterers.*;
import weka.clusterers.ClusterEvaluation;
import weka.filters.*;
import weka.filters.unsupervised.attribute.Remove;
import weka.clusterers.FilteredClusterer;
public class Cluster {
   private int m_InstanceNum;
   private int m_NumClusters;
   private int[] m_BestAssignments;
   public static void main(String[] args) throws Exception {
      // load data
      Instances data = new Instances(new BufferedReader(new FileReader("C:\\Users\\Gishen Debre Kerbi\\Documents\\NetBeansProjects\\Cluster\\src\\cluster\\s_c_Dataset June 132015_Final.arff")));
      data.setClassIndex(data.numAttributes() - 1);
      Remove filter = new Remove();
      filter.setAttributeIndices("" + (data.classIndex() + 1));
      filter.setInputFormat(data);
      Instances dataClusterer = Filter.useFilter(data, filter);
      // train clusterer
      SimpleKMeans clusterer = new SimpleKMeans();
            clusterer.setNumClusters(5);
      clusterer.setSeed(494021);
      String[] options = new String[2];
      int nClusters = 5;
      int seed = 494021;
      options[0] = String.valueOf(nClusters);
      options[1] = String.valueOf(seed);
      clusterer.setOptions(options);
```

```
        ClusterEvaluation eval = new ClusterEvaluation();
        eval.setClusterer(clusterer);
        eval.evaluateClusterer(data);
        System.out.println("# of clusters: " + eval.getNumClusters());
        clusterer.buildClusterer(dataClusterer);
        eval.setClusterer(clusterer);
        eval.evaluateClusterer(data);
        System.out.println(eval.clusterResultsToString());
        Instances centroids = clusterer.getClusterCentroids();
        int[] clusterSize = clusterer.getClusterSizes();
        String filename = "Intrusion";
        filename += "_ClusterDataSetJune122015";
        filename += ".txt";
        File file = new File(filename);
        try (Writer writer = new BufferedWriter(new FileWriter(file))) {
           for (int i = 0; i <= 494021; i++) {
              for (int j = 0; j < clusterSize[i]; j++) {
                 String centroidsStr = centroids.instance(i).toString();
                 String[] to = centroidsStr.split(",");
                 writer.write(Integer.toString(i));
                 writer.write(",");
                 writer.write(data.instance(j) + "," + "case" + i + "," + (to[0]));
                  writer.write("\n");
                           }
            }
            System.out.println("\nInformation: ");
        }
     }
}
```

Appendix VII

Java code for CBNID application

```
package cbr;
import jcolibri.cbraplications.StandardCBRApplication;
import jcolibri.cbrcore.CBRCase;
import jcolibri.cbrcore.CBRCaseBase;
import jcolibri.cbrcore.CBRQuery;
import jcolibri.cbrcore.Connector;
import jcolibri.exception.ExecutionException;
import jcolibri.exception.InitializingException;
import jcolibri.method.retrieve.FilterBasedRetrieval.FilterConfig;
import jcolibri.method.retrieve.NNretrieval.NNConfig;
import jcolibri.connector.PlainTextConnector;
import jcolibri.method.retrieve.NNretrieval.similarity.global.Average;
import jcolibri.method.retrieve.NNretrieval.similarity.local.Interval;
import jcolibri.method.retrieve.NNretrieval.similarity.local.Equal;
import jcolibri.method.retrieve.NNretrieval.NNScoringMethod;
import jcolibri.method.retrieve.selection.SelectCases;
import jcolibri.method.retrieve.RetrievalResult;
import jcolibri.casebase.LinearCaseBase;
import jcolibri.method.retrieve.FilterBasedRetrieval.FilterBasedRetrievalMethod;
//import jcolibri.method.retrieve.FilterBasedRetrieval.FilterConfig;
import java.util.Iterator;
import java.util.Collection;
import javax.annotation.Generated;
import representation.CaseDescription;
//import representation.CaseSolution;
//import jcolibri.method.retrieve.NNretrieval.NNConfig;
import jcolibri.cbrcore.Attribute;

public class CBRApplication implements StandardCBRApplication {
	private static CBRApplication _instance = null;

	public static CBRApplication getInstance() {
		if (_instance == null) {
			_instance = new CBRApplication();
		}
		return _instance;
	}

	CBRCaseBase _caseBase;
	Connector _connector;
	NNConfig simConfig;
	FilterConfig filterConfig;
	Collection<Attribute> hiddenAtts;
	ResultDisplay resultDialog;
```

```
	public void configure() throws ExecutionException {
		try {
			_connector = new PlainTextConnector();
			_connector
					.initFromXMLfile(jcolibri.util.FileIO
							.findFile("C:\\Users\\Gishen Debre
Kerbi\\Documents\\Eclips jCOLIBRI
Project\\CBRNetworkIntrusionDetection\\config\\plainTextConnectorConfig.xml"));
			_caseBase = new LinearCaseBase();
			simConfig = new NNConfig();
			simConfig.setDescriptionSimFunction(new Average());
			simConfig.addMapping(
					new Attribute("CaseId", CaseDescription.class),
					new Interval(1));
	simConfig.addMapping(new Attribute("duration", CaseDescription.class), new Equal());
	simConfig.addMapping(new Attribute("protocol_type",CaseDescription.class), new Equal());
	simConfig.addMapping( new Attribute("service", CaseDescription.class), new Equal());
	simConfig.addMapping(new Attribute("flag", CaseDescription.class), new Equal());
	simConfig.addMapping(new Attribute("src_bytes", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("dst_bytes", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("land", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("wrong_fragment", CaseDescription.class), new
Interval(1));
	simConfig.addMapping( new Attribute("urgent", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("hot", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("num_failed_logins", CaseDescription.class), new
Interval(1));
	simConfig.addMapping(new Attribute("logged_in", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("num_compromised", CaseDescription.class), new
Interval(1));
	simConfig.addMapping(new Attribute("root_shell", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("su_attempted", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("num_root", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("num_file_creations",CaseDescription.class), new
Interval(1));
	simConfig.addMapping(new Attribute("num_shells", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("num_access_files", CaseDescription.class), new
Interval(1));
	simConfig.addMapping(new Attribute("num_outbound_cmds", CaseDescription.class), new
Interval(1));
	simConfig.addMapping(new Attribute("is_host_login", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("is_guest_login", CaseDescription.class), new
Interval(1));
	simConfig.addMapping(new Attribute("count", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("srv_count", CaseDescription.class), new Interval(1));
	simConfig.addMapping(new Attribute("serror_rate", CaseDescription.class), new Interval(1));
```

```
        simConfig.addMapping(new Attribute("srv_serror_rate", CaseDescription.class), new
Interval(1));
        simConfig.addMapping(new Attribute("rerror_rate", CaseDescription.class), new Interval(1));
        simConfig.addMapping(new Attribute("srv_rerror_rate", CaseDescription.class), new
Interval(1));
        simConfig.addMapping(new Attribute("same_srv_rate", CaseDescription.class), new Interval(1));
        simConfig.addMapping(new Attribute("diff_srv_rate", CaseDescription.class), new Interval(1));
        simConfig.addMapping(new Attribute("srv_diff_host_rate", CaseDescription.class), new
Interval(1));
        simConfig.addMapping(new Attribute("dst_host_count", CaseDescription.class), new
Interval(1));
        simConfig.addMapping(new Attribute("dst_host_srv_count", CaseDescription.class), new
Interval(1));
        simConfig.addMapping(new Attribute("dst_host_same_srv_rate", CaseDescription.class), new
Interval(1));
        simConfig.addMapping(new Attribute("dst_host_diff_srv_rate", CaseDescription.class), new
Interval(1));
simConfig.addMapping(new Attribute("dst_host_same_src_port_rate", CaseDescription.class), new
Interval(1));
simConfig.addMapping(new Attribute("dst_host_srv_diff_host_rate",CaseDescription.class), new
Interval(1));
simConfig.addMapping(new Attribute("dst_host_serror_rate",CaseDescription.class), new Interval(1));
simConfig.addMapping(new Attribute("dst_host_srv_serror_rate", CaseDescription.class), new
Interval(1));
simConfig.addMapping(new Attribute("dst_host_rerror_rate", CaseDescription.class), new Interval(1));
simConfig.addMapping(new Attribute("dst_host_srv_rerror_rate",CaseDescription.class), new
Interval(1));
        configureConnector();
        configureCaseBase();
                } catch (Exception e) {
                        throw new ExecutionException(e);
                }
        }

        /** Configures the connector */
        private void configureConnector() throws InitializingException {
                        }

        /** Configures the case base */
        private void configureCaseBase() throws InitializingException {
        _caseBase = new jcolibri.casebase.LinearCaseBase();
        }
        public CBRCaseBase preCycle() throws ExecutionException {
                _caseBase.init(_connector);
                // Print the cases
                java.util.Collection<CBRCase> cases = _caseBase.getCases();
                for (Iterator<CBRCase> it = cases.iterator(); it.hasNext();) {
```

```
                    jcolibri.cbrcore.CBRCase c = it.next();
                    System.out.println(c);
                }
                return _caseBase;
        }
        public void cycle(CBRQuery query) throws ExecutionException {
                // Execute Filter
Collection<CBRCase> filtered = FilterBasedRetrievalMethod.filterCases( _caseBase.getCases(), query,
filterConfig);
                // Execute NN
Collection<RetrievalResult> eval = NNScoringMethod.evaluateSimilarity( filtered, query, simConfig);
                // Select cases
Collection<CBRCase> selectedCases = SelectCases.selectTopK(eval, 5);

                System.out.println("The selected Cases are...");
                for (Iterator<CBRCase> it = selectedCases.iterator(); it.hasNext();)
                 {
                 jcolibri.cbrcore.CBRCase c = it.next();
                 System.out.println(c);
                 }
        }


        public void postCycle() throws ExecutionException {
                _connector.close();
        }

//Main method
                public static void main(String[] args) {
                CBRApplication cbrApp = new CBRApplication();
                // cbrApp.UserSeectionGUL;
                try {
                        cbrApp.configure();
                        cbrApp.preCycle();
                        CBRQuery query = new CBRQuery();

                        CaseDescription cd = new CaseDescription();
                        // CaseDescription cd = new CaseDescription();
                        // cd.setCaseId(1);
                        cd.setDuration(0);
                        cd.setProtocol_type("tcp");
                        cd.setService("ftp_data");
                        cd.setFlag("SF");
                        cd.setSrc_bytes(345);
                        cd.setDst_bytes(5014);
                        cd.setLand(1);
                        cd.setWrong_fragment(1);
```

```
                cd.setUrgent(45);
                cd.setHot(2345);
                cd.setNum_failed_logins(654);
                cd.setLogged_in(98);
                cd.setNum_compromised(159);
                cd.setRoot_shell(636);
                cd.setSu_attempted(845);
                cd.setNum_shells(0);
                cd.setNum_access_files(0);
                cd.setNum_root(0);
                cd.setNum_outbound_cmds(0);
                cd.setIs_host_login(0);
                cd.setIs_guest_login(0);
                cd.setCount(0);
                cd.setSrv_count(0);
                cd.setSerror_rate(1.0);
                cd.setSrv_serror_rate(1.0);
                cd.setRerror_rate(1.0);
                cd.setSrv_rerror_rate(1.0);
                cd.setSame_srv_rate(1.0);
                cd.setDiff_srv_rate(1.0);
                cd.setSrv_diff_host_rate(1.0);
                cd.setDst_host_count(1);
                cd.setDst_host_srv_count(1);
                cd.setDst_host_same_srv_rate(1.0);
                cd.setDst_host_diff_srv_rate(1.0);
                cd.setDst_host_same_src_port_rate(1.0);
                cd.setDst_host_srv_diff_host_rate(1.0);
                cd.setDst_host_serror_rate(1.0);
                cd.setDst_host_srv_serror_rate(1.0);
                cd.setDst_host_rerror_rate(1.0);
                cd.setDst_host_srv_rerror_rate(1.0);
                query.setDescription(cd);
                cbrApp.cycle(query);
                cbrApp.postCycle();
            } catch (ExecutionException e) {
                System.out.println(e.getMessage());
            }
        }
}
```