

Data Cleansing Technique Based on Neural Networks

7-5096

DATA ENTERED



Developed By:

Uzma Altaf 143-CS/MS/03

Saba Farooqi 144-CS/MS/03

Supervised By:

Dr. Sikandar Hayat Khiyal

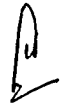


Department of Computer Science
International Islamic University Islamabad

2008

28/7/10

~~SECRET~~



Accession No. TH-5096

DATA ENTERED

RF
17/2/2012

T05096

MS

006.32

U Z D

1. Neutral network

D.F.

10

9-12-10

International Islamic University, Islamabad

Department of Computer Science

Dated: _____

Final Approval

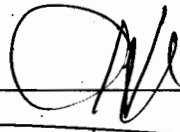
It is certified that we have read the thesis report submitted by Ms. Uzma Altaf (143-CS/MS/03) and Ms. Saba Farooqi (144-CS/MS/03) and it is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of MS Computer Science.

EXTERNAL EXAMINAR

Dr. Abdul Sattar

Ex. D.G.

Computer Bureau, Islamabad.




INTERNAL EXAMINAR

Dr. Muhammad Sher

Head, Department of Computer Science,

Faculty of Applied Sciences,

International Islamic University, Islamabad.



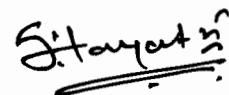
SUPERVISOR

Dr. M. Sikandar Hayat Khiyal

Chairperson,

Computer Science and Software Engg. Department,

Fatima Jinnah Women University, Rawalpindi.



A dissertation submitted to the
Department of Computer Science,
Faculty of Applied Sciences,
International Islamic University, Islamabad, Pakistan,
as a partial fulfillment of the requirements for the award of the degree of
MS in Computer Science

Dedicated To

The holiest Man Ever Born,
Prophet Muhammad (SAW)

To

Our Parents and Families

We are most thankful to our Parents and Families, whose affection and encouragement helped us a lot in the completion of our thesis work and whose prayers have always been a key to our success.

To

Our Honorable Teachers

Who have trusted us and helped us through tough times. Who are source of inspiration for our whole life.

DECLARATION

We, hereby declare that “Data Cleansing Technique Based on Neural Networks” software, neither as a whole nor as a part thereof has been copied out from any source. We have developed this software and the accompanied report entirely on the basis of our personal efforts made under the sincere guidance of our supervisor. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or any other university or institution of learning.


Uzma Aitaf

(143-CS/MS/03)

Saba Farooqi

(144-CS/MS/03)

ACKNOWLEDGEMENT

All praise for Allah (SWT) who has enabled us to complete our thesis of MS Computer Science successfully. Few words of appraisal can never match the support and help our parents have extended to us throughout our lives.

First and foremost, we would express our gratitude to our loving parents, without their support and help we would never had been able to reach this far in our lives.

We would also extend our gratitude to Dr. Sikandar Hayat Khiyal, our respectful teacher and supervisor, who cooperated, guided and facilitated us a lot during our thesis. He kept our morale high with his kind suggestions and guidance. Without his sincere and cooperative nature, we could never have been able to complete this task. We would also thank Mr. Shahid Rauf for his kind guidance.

It will not be out of place to express our profound admiration and gratitude for our teacher Mr. Munir Hussain Naveed, Associate Professor, Fatima Jinnah Women University, Rawalpindi, for his dedication, inspiring attitude, untiring help and kind behavior throughout the thesis work. He did a lot for our success.

Project In Brief

Project Title: Data Cleansing Technique based on Neural Networks

Organization: International Islamic University Islamabad

Under Taken By: Uzma Altaf

Reg #: 143-CS/MS/03

Saba Farooqi

Reg #: 144-CS/MS/03

Supervised By: Prof. Dr. Malik Sikandar Hayat Khiyal

Chairperson Computer Science and Software Engineering

Fatima Jinnah Women University, Rawalpindi.

Starting Date: 09-08-2006

End Date: 18-09-2008

Tools Used: Microsoft Visual C# 2005, Microsoft SQL Server 2005

System Used: Dell Latitude D 510, Intel Pentium M Processor 735A

ABSTRACT

Within data warehousing, data cleansing is especially applied when different databases are merged in a single databases. Records that refer to the same entity are represented erroneously. Because of this reason duplicate records appear when different databases are merged. The issue is to identify and eliminate these duplicates from database. This problem is known as the *merge/purge problem, semantic integration, instance identification, or object identity problem*. Name cleansing is an important factor for organizations that deal with names data as vendors, suppliers, employees and customers. It will simplify processes, reduce costs and enable important business objectives, as strategic sourcing initiatives. In this paper we examine different available data cleansing algorithms and provide a design which can be used for developing effective and efficient name cleansing solutions.

Table of Contents

1. Introduction	1
1.1 Unsupervised Learning	2
1.2 Supervised Learning.....	2
1.3 Evolutionary Algorithms.....	2
1.4 Artificial Neural Networks.....	3
1.4.1 Background	3
1.4.2 Structure	4
1.4.3 How They Learn?.....	5
1.4.4 Types of Neural Networks	5
1.4.4.1 Feed Forward Neural Network	6
1.4.4.2 Radial Basis Function Network	6
1.4.4.3 Kohonen Self-Organizing Network	6
1.4.4.4 Recurrent Network	6
1.4.4.5 Stochastic Neural Networks	7
1.4.4.6 Modular Neural Networks	8
1.4.4.7 Other Types of Networks	8
1.5 Associative Memory Matrix	9
1.5.1 Description	10
2. Literature Survey	11
2.1 Related Work	11
2.2 Problem Statement	18
2.3 Objectives	20
3. Proposed Solution	21
4. Methodology	25
4.1 Data Standardization	26
4.1.1 Tokenization	26
4.1.2 Associative Matrix Memory (AMM) for Removing Spelling Mistakes	29
4.1.2.1 Associative Matrix Memory	30
4.1.2.2 Knowledgebase	32

4.1.2.3.a. Associative Matrix Memory - Novel Method of Matching by Changing Positions of Letters	34
4.1.2.3.b. Associative Matrix Memory - Novel Method of Matching by Sorting Letters	35
4.1.2.4 Set of Possible Matches	37
4.2 Duplicate Elimination	37
4.2.1 Duplicate Elimination technique based on Associative Matrix Memory	37
4.2.2 Evolutionary Algorithms	40
4.2.2.1 Execution of Evolutionary Algorithm	41
4.2.2.2 Components of Evolutionary Algorithm	42
4.2.2.3 Pseudo Code	46
4.2.3 Artificial Neural Network.....	47
4.2.3.1 Execution of Neural Network.....	48
4.2.3.2 The Back-Propagation Algorithm	49
4.2.3.3 Pseudo Code	49
5. Analysis	52
5.1 System Specifications	52
5.1.1 Tokenization Method	52
5.1.2 Associative Matrix Memory (AMM) for Removing Spelling Mistakes	54
5.1.3 Duplicate Elimination technique based on Associative Matrix Memory	59
5.1.4 Duplicate Elimination Technique Based on Evolutionary Algorithm	61
5.1.5 Neural Network Algorithm for Duplicate Detection	64
6. Conclusion	66
References	67
Appendix A	

INTRODUCTION

1. Introduction

Within the data warehousing environment, data cleansing is specifically applied when many databases are merged in one single database. Records that refer to the same entity are represented in different formats in the different data sets or are represented erroneously. Therefore, duplication of records will appear in the merged database. The purpose is to identify and eliminate such duplicates in order to bring consistency in database. This problem is addressed as the *merge/purge problem*. Merge/Purge problem is also sometimes referred as record linkage, semantic integration, instance identification, or object identity problem.

Data cleansing, also known as data scrubbing is the process of detecting and removing and/or correcting a dirty data that exist in a database. By dirty data we mean data that is incorrect, out-of-date, redundant, incomplete, or formatted incorrectly. The goal of data cleansing is not just to clean the data in a database but also to bring consistency to different sets of data that have been merged from separate databases. Many sophisticated and complex softwares are available to clean a database's data using different algorithms, which made the task of data cleansing easier for human as before that data cleansing task was once done manually and therefore still subject to human error.

There are two areas of data cleansing, the unification process where common formats and abbreviations are enforced and the scrubbing process, where complex decision rules are applied to identify and scrub data errors. Since the types of errors and inconsistencies can be domain-specific, it is important and challenging to develop generic domain specific data cleansing solutions.

“To be of value as an asset to the organization, large, detailed databases require a considerable amount of maintenance so ensuring this is carried out as efficiently and effectively as possible, is of concern to every database manager”. [1]

“Whether internal (i.e. personnel databases, inventory, parts lists etc) or external (i.e. prospects, clients, suppliers, media contacts, service providers etc), a simple yet efficient means of regular reviews of the data is the ideal way to ensure the data remains in optimum condition”. [1]

Name/address cleansing is an important factor for projects that deals with name/address data of vendors, suppliers, customers, employees etc. the cleansing of such data will simplify processes in a sense that one can uniquely identify a record with assurity that it contains correct information regarding customer, vendor or employee, reduce costs in context of searching of particular records and enable important business objectives such as strategic sourcing initiatives and deriving useful information from the cleaned data. If business intelligence systems include

name and address related data in the target environment, we may have the concerns such as “Do the persons registered to an address live there, or Is the address entered by the person is valid? Do the companies mentioned exist?” Also, if we know the correct address, but we do not know the zip code, could we trace out that particular company or person?

The focus on names and address cleansing sometimes causes a misconception that it requires only names and addresses in the database be correct where as it is not the case. The goal of name/address cleansing is to identify and match information of customer reliably based on their name and address data and for this it is necessary that a correct entry of name/address have been made in warehouse for each customer. Our goal is to examine different available data cleansing algorithms and provide a design which can be used for developing effective and efficient name cleansing solutions.

1.1 Unsupervised Learning

Unsupervised learning is one of the techniques of machine learning where the algorithm is provided with examples from the input space only, and a model is trained to these observations. For example, a clustering algorithm is a form of unsupervised learning.

"Unsupervised learning is a method of machine learning where a model is fit to observations. It is distinguished from supervised learning by the fact that there is no *a priori* output. In unsupervised learning, a data set of input objects is gathered. Unsupervised learning then typically treats input objects as a set of random variables. A joint density model is then built for the data set." [2]

1.2 Supervised Learning

"*Supervised learning* is a machine learning technique whereby the algorithm is first presented with training data which consists of examples which include both the inputs and the desired outputs; thus enabling it to learn a function. The learner should then be able to generalize from the presented data to unseen examples." [3]

1.3 Evolutionary Algorithms

Charles Darwin was the first one to identify the process of natural selection in his remarkable work of “The Origin of Species”. Certain characteristics or features that have a high chance for the survival of individual are passed to its offspring during the process of reproduction. Individuals that have poor characteristics/features die off, making the species stronger in general. Inspired by this natural process of reproduction named as "survival of the fittest," evolutionary algorithms (EAs) was developed which tries to find a solution to a problem by producing offspring that holds better solution to a problem. [4]

There are two related, yet distinct, types of EAs. The first type is known as genetic algorithms (GAs) that deals with manipulation of a fixed-length bit string. The bit string represents a solution to the problem that need to be solved, however it depends on programmer to depict the meaning of the bit string. The second type of EAs is genetic programming which deals with generating expression trees. With genetic programming, actual programs are created and then executed. In general, the process used in evolutionary learning starts with random generation of a population of individuals where each individual is a possible solution to the problem. Each individual contains a genome, which is the content produced while EA is executed. In the case of GAs, the fixed-length bit string is the genome. In next step, each individual in the population is evaluated by using a function called a fitness function. Fitness function gives a particular fitness value to each individual. For example, if a genetic algorithm was designed to determine a single-variable algebraic function $F(n)$, the fitness function may use a training set of inputs and outputs to determine how close an individual is to $F(n)$. In this case, the fitness could be inversely proportional to the sum of the absolute differences between expected output and actual output for each input. How this fitness value of an individual is used that totally depends on the implementation of the evolutionary algorithm. Usually, an individual with higher fitness value has a greater chance to be selected for reproduction of next generation. There are different techniques that exist for the production of the next generation, but most commonly, two operators are used: Mutation and Breeding. Mutation is a single variate operator in which a new offspring is generated by randomly altering one or more genes in an individual's genome. Breeding uses a crossover operation to combine features of two parents' genomes to produce one or more children. Once a new generation is created, the old one is discarded. A cycle of evaluation and reproduction is repeated through several generations, as specified by the programmer till a desired output is obtained. The evolution process ends when it meets the termination criteria, and the result is the individual found to be best-so-far. [4]

1.4 Artificial Neural Networks

An artificial neural network (ANN), also referred to as "neural network" (NN), is a mathematical/ computational model which is based on biological neural networks. It consists of an interconnected artificial neurons and processes information similar to the human neural system. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

In more practical terms neural networks are non-linear statistical data modeling tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

1.4.1 Background

A traditional digital computer does many tasks quite fast, and it processes exactly what you tell it to do. Unfortunately, it can't help you when you yourself are not fully aware of the problem you that require solution. In worst scenario, standard algorithms don't deal properly with incomplete

data or the data that contains error; however it is the only kind which is available to deal with such problems. One answer to address such problems is the use of an artificial neural network (ANN) which is a computing system that has the ability to learn on its own.

The first ever artificial neural network invented was by psychologist Frank Rosenblatt, in 1958 named as Perceptron, which dealt with how the human brain processed visual data and learned to recognize objects. Since then other researchers started to use ANNs to study human cognition.

Ultimately, it was realized that ANNs could be a useful tools in their own right in addition to providing insights into the human brain functionality. Their pattern-matching and learning capability features allow them to address issues which were difficult or nearly impossible to be solved by other standard computational/ statistical techniques. By the end of 1980s, many institutes were using ANNs for a wide range of problems. [5]

1.4.2 Structure

The working of artificial neural network starts with creating connections between many different processing elements (neurons). These neurons may be physically constructed or simulated by a digital computer. A neuron can take many input signals, and then based on an internal weighting system; it produces a single output signal that is passed to another neuron as input.

The neurons in ANNs are tightly interconnected and are arranged in different layers. There are three types of layers; the input layer which receives the input; the output layer that produces the final output. In between the input and output layer exists the hidden layer which consists of one or more layers. Because of this structure it is impossible to predict or to know the exact flow of data. [5]

A simple neural network is presented as under in order to give a rough picture of its structure:

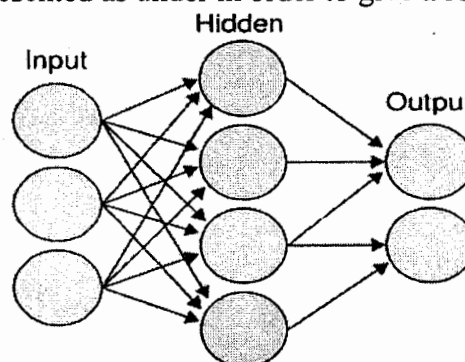


Figure 1.1: Rough Structure of Artificial Neural Network

1.4.3 How They Learn?

Artificial neural network starts with assigning weights randomly to all the neurons. This means that they don't "know" anything about the problem and that must be trained to solve the particular problem for which they are intended. In broader aspect, there are two methods for training an ANN, depending on the problem to be solved.

A self-organizing ANN often known as a Kohonen Network is used for large amounts of data and its purpose is to discover patterns and relationships in that data. Researchers often use this type of network where they need to analyze experimental data.

A back-propagation ANN, conversely, is trained by human to perform specific tasks. During its training period, the teacher evaluates the network for whether the output is correct or not. If it's correct, the neural weights that produced the output are reinforced; if the output is incorrect, then those weights which are responsible are discarded. This type is mostly used for problem-solving applications.

an artificial neural network is typically slower than traditional algorithmic solution when they are implemented on single computer. however, since ANN's have a feature of parallel functioning which allows it to be built using multiple processors, giving it a great speed advantage at very little development cost. The parallel architecture also enables ANNs to process very large amounts of data very efficiently. When dealing with large, continuous streams of information, such as speech recognition, ANNs can operate considerably faster hence reduces the time cost.

Artificial neural networks have proved to be a useful and productive solution in many real-world applications that deal with complex and often incomplete data. For example ANNs have proved to give better results in visual pattern recognition and speech recognition. Further more, programs that are developed for text-to-speech have used ANNs. Many handwriting analysis programs such as those used in Personal Digital Assistants (PDAs) utilize ANNs.

Automated and robotic factories are one of the industries which is now being monitored by ANNs to control machinery, adjust temperature settings, and diagnose malfunctions and many more. These ANNs can work with or replace skilled labor, making it possible for few people to do more work in a more efficient and better manner. Thus resulting in the reduction of chances of human errs. [5]

1.4.4 Types of Neural Networks

A brief introduction regarding different types of neural networks in the form of basic definitions is given in the following section:

1.4.4.1 Feed Forward Neural Network

“A feed forward neural network is an artificial neural network where connections between the units do *not* form a directed cycle. This is different from recurrent neural networks”. [6]

“The feed forward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network as shown in figure”. [6]

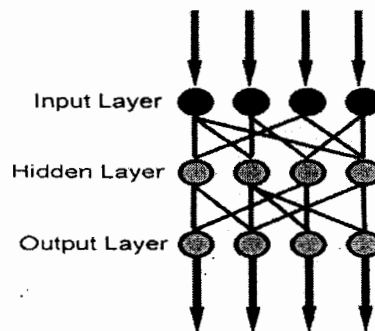


Figure 1.2 : In a feed forward network information at a later level, never back propagates to a previous level

1.4.4.2 Radial Basis Function Network

“A radial basis function network is an artificial neural network which uses radial basis functions as activation functions. They are used in function approximation, time series prediction, and control”. [7]

1.4.4.3 Kohonen Self-Organizing Network

“The self-organizing map (SOM) invented by Teuvo Kohonen uses a form of unsupervised learning. A set of artificial neurons learn to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space, and the SOM will attempt to preserve these”. [1]

1.4.4.4 Recurrent Network

“Contrary to feed forward networks, recurrent neural networks (RNs) are models with bi-directional data flow. While a feed forward network propagates data linearly from input to output, RNs also propagate data from later processing stages to earlier stages”. [1]

Simple Recurrent Network

“A simple recurrent network (SRN) is a variation on the Multi-Layer Perceptron, sometimes called an “Elman network” due to its invention by Jeff Elman. A three-layer network is used, with the addition of a set of “context units” in the input layer. There are connections from the

middle (hidden) layer to these context units fixed with a weight of one. At each time step, the input is propagated in a standard feed-forward fashion, and then a learning rule (usually back-propagation) is applied. The fixed back connections result in the context units always maintaining a copy of the previous values of the hidden units (since they propagate over the connections before the learning rule is applied). Thus the network can maintain a sort of state, allowing it to perform such tasks as sequence-prediction that is beyond the power of a standard Multi-Layer Perceptron”. [1]

“In a fully recurrent network, every neuron receives the input from every other neuron in the network. Such networks are not arranged in layers. Usually only a subset of the neurons receive external inputs in addition to the inputs from all the other neurons, and another disjunct subset of neurons report their output externally as well as sending it to all the neurons. These distinctive inputs and outputs perform the function of the input and output layers of a feed-forward or simple recurrent network, and also join all the other neurons in the recurrent processing”. [1]

Hopfield Network

“The Hopfield network is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield in 1982, this network guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable memory, resistant to connection alteration”. [1]

Echo State Network

“The echo state network (ESN) is a recurrent neural network with a sparsely connected random hidden layer. The weights of output neurons are the only part of the network that can change and be learned. ESN are good to (re)produce temporal patterns”. [1]

Long Short Term Memory Network

“The Long short term memory is an artificial neural net structure that unlike traditional RNNs doesn't have the problem of vanishing gradients. It can therefore use long delays and can handle signals that have a mix of low and high frequency components”. [1]

1.4.4.5 Stochastic Neural Networks

“A stochastic neural network differs from a typical neural network in the fact that it introduces random variations into the network. In a probabilistic view of neural networks, such random variations can be viewed as a form of statistical sampling, such as Monte Carlo sampling”. [1]

Boltzmann Machine

“The Boltzmann machine can be thought of as a noisy Hopfield network. Invented by Geoff Hinton and Terry Sejnowski in 1985, the Boltzmann machine is important because it is one of the first neural networks to demonstrate learning of latent variables (hidden units). Boltzmann

machine learning was at first slow to simulate, but the contrastive divergence algorithm of Geoff Hinton (circa 2000) allows models such as Boltzmann machines and products of experts to be trained much faster". [1]

1.4.4.6 Modular Neural Networks

"Biological studies showed that the human brain functions not as a single massive network, but as a collection of small networks. This realization gave birth to the concept of modular neural networks, in which several small networks cooperate or compete to solve problems". [1]

Committee of Machines

"A committee of machines (CoM) is a collection of different neural networks that together "vote" on a given example. This generally gives a much better result compared to other neural network models. In fact in many cases, starting with the same architecture and training but using different initial random weights gives vastly different networks. A CoM tends to stabilize the result. The CoM is similar to the general machine learning bagging method, except that the necessary variety of machines in the committee is obtained by training from different random starting weights rather than training on different randomly selected subsets of the training data". [1]

Associative Neural Network (ASNN)

"The ASNN is an extension of the committee of machines that goes beyond a simple/weighted average of different models. ASNN represents a combination of an ensemble of feed-forward neural networks and the k-nearest neighbor technique (kNN). It uses the correlation between ensemble responses as a measure of distance amid the analyzed cases for the kNN. This corrects the bias of the neural network ensemble. An associative neural network has a memory that can coincide with the training set. If new data becomes available, the network instantly improves its predictive ability and provides data approximation (self-learn the data) without a need to retrain the ensemble. Another important feature of ASNN is the possibility to interpret neural network results by analysis of correlations between data cases in the space of models". [1]

1.4.4.7 Other Types of Networks

Following are the special networks that do not fit in any of the previous categories:

Holographic Associative Memory

"Holographic associative memory represents a family of analog, correlation-based, associative, stimulus-response memories, where information is mapped onto the phase orientation of complex numbers operating". [1]

Instantaneously Trained Networks

“Instantaneously trained neural networks (ITNNs) were inspired by the phenomenon of short-term learning that seems to occur instantaneously. In these networks the weights of the hidden and the output layers are mapped directly from the training vector data. Ordinarily, they work on binary data, but versions for continuous data that require small additional processing are also available”. [1]

Spiking Neural Networks

“Spiking neural networks (SNNs) are models which explicitly take into account the timing of inputs. The network input and output are usually represented as series of spikes (delta function or more complex shapes). SNNs have an advantage of being able to continuously process information. They are often implemented as recurrent networks”. [1]

Dynamic Neural Networks

“Dynamic neural networks not only deal with nonlinear multivariate behavior, but also include (learning of) time-dependent behavior such as various transient phenomena and delay effects”. [1]

Cascading Neural Networks

“Cascade-Correlation is architecture and supervised learning algorithm developed by Scott Fahlman and Christian Lebiere. Instead of just adjusting the weights in a network of fixed topology, Cascade-Correlation begins with a minimal network, then automatically trains and adds new hidden units one by one, creating a multi-layer structure. Once a new hidden unit has been added to the network, its input-side weights are frozen. This unit then becomes a permanent feature-detector in the network, available for producing outputs or for creating other, more complex feature detectors”. [1]

Neuro-Fuzzy Networks

“A neuro-fuzzy network is a fuzzy inference system in the body of an artificial neural network. Depending on the FIS type, there are several layers that simulate the processes involved in a fuzzy inference like fuzzification, inference, aggregation and defuzzification. Embedding an FIS in a general structure of an ANN has the benefit of using available ANN training methods to find the parameters of a fuzzy system”. [1]

1.5 Associative Memory Matrix

“A *content-addressable memory* is a type of memory that allows for the recall of data based on the degree of similarity between the input pattern and the patterns stored in memory. It refers to a memory organization in which the memory is accessed by its content as opposed to an explicit address like in the traditional computer memory system. Therefore, this type of memory allows the recall of information based on partial knowledge of its contents.” [8]

“An *associative memory* is a content-addressable structure that maps a set of input patterns to a set of output patterns. There are two types of associative memory:

- *Autoassociative*: An autoassociative memory retrieves a previously stored pattern that most closely resembles the current pattern.
- *Heteroassociative*: In a heteroassociative memory, the retrieved pattern is, in general, different from the input pattern not only in content but possibly also in type and format.”[9]

1.5.1 Description

Suppose we are given a memory that contains a set of names of several people as shown in the figure below. If the given memory is content-addressable, then the string with, say spelling error "Belal Choudhry" as key is sufficient enough to retrieve the correct name "Bilal Choudhry" from the memory. In this sense, this type of memory is robust and fault-tolerant, as it exhibits some form of error-correction capability.

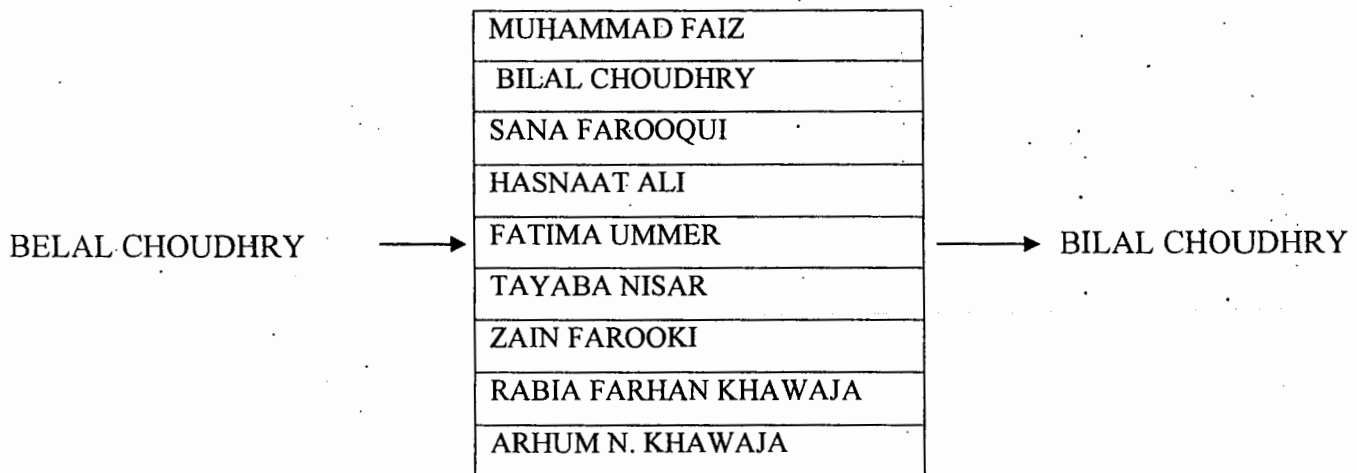


Figure 1.3: A content-addressable memory in action

LITERATURE SURVEY

2. Literature Survey

In this chapter we will discuss the work done by different authors in the area under review. A detailed study is made in the next section. Then we will relate the work done, with our study.

2.1 Related Work

The prevailing concept of data cleansing, also called record linkage, has gone through several steps of modification. In different eras novel definitions have been associated to data cleansing, which leads it to evolve as a huge term, giving it vast meaning and diversity of function.

The theory presented by Fellegi and Sunter [10] is an important milestone in the development of record linkage. It shapes a mathematical model of record linkage process, which serves as a bedrock of the modern concept of data cleansing. According to the definition expressed by Fellegi and Sunter, for two data sources A and B, two disjoint sets M and U are formed as a result of the cross product of A and B, the set $A \times B$. A record pair true match it is a member of set M. Otherwise, it is a member of set U. Mathematically it can be written as:

(There are two populations A and B. Their elements will be denoted by a and b respectively. We suppose that some elements are common to A and B. Consequently the set of ordered pairs)

$A \times B = \{(a,b): a \in A, b \in B\}$

is the union of two disjoint sets of matches

$M = \{(a,b): a=b, a \in A, b \in B\}$

and nonmatches

$U = \{(a,b): a \neq b, a \in A, b \in B\}$. [10]

The record linkage or data cleansing process proposed by Fellegi and Sunter classified each record pair as either a member of M or U. Their model is referred as Probabilistic Model since it is based on probability theory.

Following the same concept, as presented by Fellegi and Sunter [10], Jaro [11] implements the methodology with certain advances in it. Major enhancements encompass multiple blocking strategy and a novel matching algorithm (commonly called Jaro's Algorithm).

Moreover, EM (Expectation Maximization) Algorithm has been used for parameter estimation. Jaro [11] in his publication presents the theoretical background which is necessary to understand the statistical basis of record linkage process in general, the methodology being developed by Jaro to estimate parameters which are necessary for record linkage activity, the basic of matching algorithm used and the specific methodology used for matching the 1985 census of Tampa to the post enumeration survey (PES). However, additional work is required in a number of areas like,

instead of categorizing the record pairs as agreement or disagreement, the vector could be made to include cases where values are missing or it must be having a mechanism that can deal with values that apparently do not come under agreement/disagreement state.

Hernandez and Stolfo [12] addressed the record linkage problem as *merge/purge*, which is commonly used in business organizations to describe the record linkage problem. The authors proposed an equational theory for record linkage. By the term *equational theory*, they mean the specification of inference declarative rules that defines the logic of record equivalence. Hernandez and Stolfo [12] gave a solution in which sorting of the entire data set is used to bring the matching records close together in a bounded neighborhood in a linear list. In addition, data base is partitioned into ‘clusters’ in such a way that potentially matching records are assigned to same cluster. Nevertheless, the system is not truly automated due to the fact that a field expert is required to define equation theory (set of rules). Besides, the paper doesn’t provide an optimal solution for the clustering of records. Nonetheless, the proposed solution has been analyzed and adopted by number of succeeding research studies and softwares, which yield successful results.

Pinheiro and Don [13] presented a dynamic programming technique for linking the records in multiple heterogeneous databases by using loosely defined fields that allow free-style verbatim entries. They developed an interesting measure which was based on non-parametric randomization tests, that can be used for mining potentially useful relationships among variables. This measure uses distributional characteristics of historical events, hence accommodating variable length records in a natural way. However logistic regression model used for prediction discussed in this paper allows one to predict a discrete outcome. Handling of false true matches is not discussed in this paper. Moreover optimization problem that arises from text similarity measure equation has not been discussed in details.

Lee et al. [14] emphasize on pre-processing of the records before sorting them so that potentially matching records are brought to a close neighborhood subsequently. In this context, the techniques presented in this paper include scrubbing of data fields using external source files to remove typographical errors and use of abbreviations, tokenizing data fields and then sorting the tokens in the data fields. Yet, scalability and accuracy of the proposed algorithm is not tested with the real world data. Nonetheless, later researches have verified that pre-processing or standardization of data generates better results in terms of efficiency and accuracy.

McCallum et al. [14] introduces a technique for clustering which is efficient and effective when the problem is large. The key idea proposed is to perform clustering procedure in two stages, first a rough and quick stage clustering is performed in which it divides the data into overlapping subsets we call “canopies”, then a more rigorous final stage of clustering is performed in which expensive distance measurements are only made among points that occur in a common canopy. This differs from previous clustering methods in that it uses two different distance metrics for the

two stages, and forms overlapping regions. In this paper McCallum et al. focused on *reference matching*, a particular class of problems that arise when one has many different descriptions for each of many different objects, and wishes to know (1) which descriptions refer to the same object, and (2) what the best description of that object is?

T. Churches et al. [15] describe an approach to standardization of data, using combination of lexicon based tokenization and probabilistic hidden Markov models (HMMs). It is supposed that this model is a viable alternative to traditional rule based standardization methods. Furthermore, it is argued that this method is efficient in terms of time and accuracy. Yet, there is a margin to further expand this concept.

Jonathan et al. [16] examines the literature related to data cleansing process which were existing in that period and revised methods for error detection which go beyond integrity analysis. Jonathan et al. also discussed future work addressing different problems of data cleansing in this paper.

In a similar paper, Gu et al. [17] look into current practices and future directions of record linkage. The paper profoundly analyzes the literature available on record linkage and compares the softwares that have been proposed for data cleansing. Besides that, it presented the “standard” probabilistic record linkage model and the associated algorithm which is entirely based on the probability theory given by Fellegi and Sunter. [10]

Remarkable work has been presented by Winkler [18]. Record linkage, is required for creating and maintaining names/addresses that helps in the operations for evaluations of a Census. Winkler described enhancements to a record linkage methodology that uses string comparators to deal with strings which do not match character wise, an enhanced methodology to deal with differing, simultaneous agreements and disagreements between matching variables associated with pairs of records, and a new assignment algorithm for forcing 1-1 matching. Because of the interactions between the differing techniques, improving one method without accounting for how the method interacts with the others can actually reduce matching efficacy. However, considerably more research is needed before the techniques can be used by naive practitioners on a large variety of administrative lists. Furthermore the techniques can be applied by experienced practitioners to a narrow range of high-quality lists such as those for evaluating Census undercount that have known matching characteristics.

Much of the record linkage work in the past has been done manually or via elementary but ad hoc rules. In this paper Winkler [19] focused on computer matching techniques based on formal mathematical models subject to testing via statistical and other accepted methods. Winkler [19] provided background that relates Fellegi-Sunter model of record linkage to general automated software for establishment lists. The presentation focused on how various existing techniques

have been developed to deal with specific problems that occur in name/address parsing or inappropriate assumptions that are used to simplify computation associated with the Fellegi-Sunter model.

Data cleansing methods are used to find out duplicates within a file or with in the sets of files. Winkler [20] provides background on the Fellegi-Sunter model of record linkage. The Fellegi-Sunter model provides an optimal theoretical classification rule. Fellegi and Sunter [10] introduced methods that automatically estimate optimal parameters without training data that we used in many real world situations. Winkler [20] described the main model of record linkage, automatic methods of parameter estimation and error-rate estimation, methods of accounting for dependencies between fields, methods of finding better sets of pairs on which matching is done, and some straightforward extensions to situations when small amounts of labeled training data can be combined with unlabeled data. This paper provides an overview of different methods to find duplicates within and across files. Some of the most advanced methods use statistical ideas that correspond to and generalize methods from the computer science literature. One novelty is the emphasis on sophisticated methods of unsupervised learning for getting optimal matching parameters. Another is BigMatch technology for efficiently matching moderate size files having a 100 million or more records.

Enormous research efforts have been made by Cohen [21, 22, 23 and 24] regarding data cleansing solutions. In one of his papers, he proposes a logic called WHIRL, which reasons explicitly about similarity of local names as measured using the vector space model commonly adopted in statistical information retrieval. WHIRL is a query language to access the relations and manipulate the data residing in it. Beside its interactive nature, WHIRL is not flexible enough to include future enhancements. Furthermore, there is difficulty with using it in practice. Also, not every relational algebra query can be expressed in this language since there is no equivalent of negation or set difference, nor are SQL operations like grouping and sorting are supported by WHIRL. On the topic of data cleansing, the latest publication by Cohen [25] in conjunction with other authors comprises entity name matching and clustering. The review of this paper can be viewed in the coming portion of this article.

String data is everywhere, and its management is an important task in the past few years. Approximate queries are of importance, that are executed on string data especially for more complex queries that involves joins. This is because of the existence of typographical errors in data and multiple conventions used for recording attributes such as name and address. Since commercial databases do not support approximate string joins directly, therefore it is a challenge to implement this technique with user-defined functions (UDFs). In this paper, Gravano et al. [26] developed a technique to build an approximate string join capabilities on top of commercial databases by using facilities that are already available in them. This mechanism relies on matching short substrings of length q , called q grams, and considering both positions of

individual matches and the total number of such matches. However according to their experiments, a small value of q yields better results. When q is greater than three it gave worse results mostly. Moreover the concept of extended edit distance has been introduced in this paper but neither its implementation is given nor any reference to this concept has been made.

Li and Mehrotra [27] in their paper described an efficient approach to record linkage. Consider two lists of records, the record-linkage problem determine all pairs that are similar to each other, where the overall similarity between two records is defined based on domain-specific similarities over individual attributes making the record. For each attribute of records, it first maps records to a multidimensional Euclidean space that contains domain-specific similarity. Given the merging rule that defines when two records are similar, a set of attributes are chosen along which the merge will proceed. A multidimensional similarity join over the chosen attributes is then applied to determine similar pairs of records.

An outstanding framework comes from Galhardas et al. [28, 29 and 30], which then evolved as data cleansing tool called AJAX, whose purpose is to facilitate the specification and execution of data cleansing programs for both, a single source or for integrating multiple data sources. AJAX offers the fundamental services essential for the data cleansing task: data transformation, duplicate elimination and multi table matching. These services are implemented using a set of purposely designed macro-operators. Moreover, for specification of each of the macro-operators an SQL extension is also proposed. An exclusive feature of Galhardas et al. work is the ability of explicitly including the human interaction in process. The main novelty of the work is that the framework permits the following performance optimizations which are tailored for the data cleansing applications: mixed evaluation, neighborhood hash joins, decision push down and short circuited computation. By inclusion of classical edit distance functions, yet, the approach can come up with the properties that can be used for optimization purposes.

Approach adopted by Dey et al. [31] estimates the closeness between entities using a distance measure and develops a decision model for matching these entities. However, in situations involving very large databases, the performance in terms of response time can be an issue. As a matter of fact, the concept very much relies on the right choice made by user. Otherwise, performance can suffer to a great extent. Owing to this reason, it is important to identify the right set of attributes that should be examined to derive the distance across entity instances in different databases.

Bilenko and Mooney [32] put forward a domain independent method for improvement of duplicate detection accuracy using machine learning techniques. First, trainable distance metrics are learned for each field, adapting to the specific notion of similarity that is appropriate for the field's domain. Second, a classifier is employed that uses several diverse metrics for each field as distance features and classifies pairs of records as duplicates or non-duplicates. It also proposes

an extended model of learnable string distance, which shows improvement over an existing approach. Further, the researchers intend to extend the approach to token-based distance metrics using Jaccard similarity or vector-space cosine distance.

Verykios [33] present a Bayesian decision model that minimizes the cost of making a decision. This model is similar to the one proposed by Fellegi and Sunter [10] as it uses the same criterion for discriminating between matches and non matches. A discriminating feature is that it minimizes the 'cost of making a decision' rather than the 'probability of error' in decision model. Practicality and accuracy of this method, however, is not clear. There is a room for work on this concept since not much of the research has been made in this line.

Monge and Elkan [34] describe and evaluate three field matching algorithms. A dynamic programming algorithm named "The Smith-Waterman Algorithm", which was first developed to find optimal alignments between related DNA or protein sequences, has been reformed in this paper. On account of its performance, it has been extended in future work.

Advancement in the field of Artificial Intelligence, machine learning in particular has opened the door for it to penetrate into other domains of computer science. Numerous intelligent algorithms have been adapted for data cleansing to acquire better performance and results. Specifically, learning algorithms have been put into practice to enhance the 'decision models'. The outcomes are quite encouraging as Machine learning based data cleansing models outperforms probabilistic models. In fact, ID3 Algorithm, Clustering and Genetic algorithm [35] have already been applied on data cleansing systems.

Winkler [36] used a moderate size of vocabulary of words chosen from all of the training data and all of the test data. All of the data patterns observed in the training and in the unlabeled test data were used. Various authors (Weiss et al. 1999, Lewis and Ringuette 1994) have observed that classification rules based on a subset of the vocabulary can work moderately well.

Winkler [36] in his paper said that in the applications developed by Nigam et al., a suitable training data is required to create structure for the EM algorithm. Nigam et al. combine labeled training data with unlabeled additional data. They show that classification decision rules can be improved if moderate amounts of training data are combined with the proper amounts of additional unlabelled data. Similarly, if a small amount of training data is combined with a large amount unlabelled data, then decision rules will likely be poor. Although the original work of Nigam et al. appeared very promising, recent work by Yang and Liu [37] has shown that new variants of other methods in machine learning work consistently better than Bayesian networks with a variety of representative test decks. Yang and Liu [37] indicated that the comparisons in Nigam et al. work were not entirely suitable as they did not use complete sets of test data and used only the largest classes in the test decks. Yang and Liu showed that methods such as

Support Vector Machines (SVM), k-nearest neighbor (kNN), and Linear Least Squares Fit (LLSF) all gave better results according to a variety of statistical measures on several test decks. Whereas Bayesian networks and Neural Nets performance was poor than SVM, kNN, and LLSF.

One of notable researches has been carried out by Elfeky et. al. [38], who has invented an interactive record linkage toolbox named TAILOR. The paper proposes a hybrid record linkage model that combines the advantages and features of both the induction and clustering record linkage models. To a great extent, however, the performance of the software depends upon the user and its criteria. Due to the reason that it is an interactive system, there is always a need to have a field expert to make right choice of comparison function according to the specifications of the domain.

As mentioned earlier, Cohen [39] presents techniques for entity name matching and clustering that are scalable and adaptive in a sense that accuracy can be improved by adopting training strategies. Their work is largely based on the theory proposed by Fellegi and Sunter [10]. Greedy Agglomerative Clustering (GAC) has been employed in this research. Implementation and results generated by 'Canopy Algorithm', presented by A.McCallum et al. [15], is the focal point of this paper. However, one of notable omissions is that it lacks any feature that directly measures TF-IDF similarity.

Zhu and Ungar [40] presents a flexible approach to string edit distance that can be automatically tuned to different data sets and can use synonym dictionaries. Dynamic programming is used to calculate the edit distance between a pair of strings based on a set of string edit rules including a new edit rule that allows words and phrases to be deleted or substituted. A genetic algorithm is used to learn cost corresponding to each edit rule based on a small set of labeled training data. However, experimental results show that large distance between the lengths of the strings lead to the high cost.

Although, considerable amount of study and research has been made for embedding machine learning techniques in data cleansing methods, nevertheless, neural networks is one of the regions that still seeks attention of the researchers on the subject of data cleansing. Despite the fact that the following paper is not directly linked with data cleansing, yet, it is a source of inspiration for our research effort.

Hodge and Austin [41] propose a simple, supple, and competent hybrid spell checking method based upon phonetic matching, supervised learning, and associative matching in the AURA neural system. Hamming Distance and n-gram algorithms have been integrated into the system that have high recall for typing errors and a phonetic spell-checking algorithm in a single novel architecture. The area of our concern; AURA is a collection of binary neural networks that may be implemented in a modular fashion. AURA utilizes 'Correlation Matrix Memories (CMMs)' to

map inputs to outputs through a supervised learning rule, similar to a hash function. However, there is ability to reuse the Hamming Distance CMM coupled with the superior quality of the shifting triple.

One of the major challenges of artificial intelligence is to make machines as intelligent as human brain. An enormous work is seen in the field of evolutionary learning by David Fogel [42]. David Fogel et al. in one of his paper proposed an algorithm through which computer learned to play chess based on evolutionary learning. This paper gives an idea that evolutionary programming can be utilized for problems for which there is no known solution and that solutions can be achieved by self learning process.

A detailed material on CMM is produced by Sam Clegg [43]. He describes CMM as it is a form binary neural network, also known as weight-less neural networks. Such systems are often given priority over other types of neural networks because their behavior is very well defined and their implementation is very easy on standard computer hardware. A CMM is essentially an associative memory that has the benefit of limited fault tolerance and error correction.

2.2 Problem Statement

The world is full of data. After aggregation and organization, data becomes information. In today's complex and interconnected world, information mostly exists in the form which can be stored and transmitted through electronic media, nearly at once. The challenge is to accurately understand, combine, and apply information to generate and use knowledge. John Naisbitt's words have never been truer: *"We are drowning in information but starved for knowledge"*.

Now the question arises that how to extract knowledge from this mess of data relationships. Clearly, mathematical solution provides better answer to the question. Essentially every source of data, i.e. data from daily activity in financial markets, data collected during medical research or patient examinations, or data regarding business purchase transactions that occur every day, is influenced in many ways by other data from the adjacent environment. In short, the world is a noisy source of data. Knowledge, then, is based on study that accommodates uncertainty. As Nietzsche said, *"There are no facts, only interpretations"*.

Interpretation requires, getting data, cleansing data, analyzing data, and finally presenting data in a way that decisions can be taken on the basis of the knowledge obtained from the data. By exploration and extraction one can acquire information about data relationships that are hidden within the data itself. And for this purpose, we require tools and technologies to aid us. While the human brain is the most powerful pattern recognition engine we have, it's not very good at serially processing and sorting huge quantities of discrete data items. So we need to build models of the world based on data from the world - we need empirical models. In turn, models must

rapidly and accurately find the patterns buried in data that reflect knowledge which is useful in the world - empirical models must learn from the data.

Current data cleansing techniques perform better when the fields to be matched are well standardized and there is sufficient number of attributes for matching. Winkler identifies a key research question concerning the selection of matching/comparison technique. In principle, weighting attribute matching by frequency-based weights should be more informative than simple agree/disagree matching [44]. However, for less frequent attribute values and for noisy data sources, this is not actually the case. A method for deciding between these approaches is needed.

Most of the work has been done in this field by implementing different artificial intelligence techniques and many of the record linkage softwares are now available which allows user to competently parse, standardize, match and merge data, thus appearing to be able to address data quality problems.

Adaptive learning in which comparator function is learnt has also been proposed recently [39]. Predictive models (such as bagging methods and SVMs) of machine learning have been proposed for learning the match/non-match decision function.

Another aspect in which one is interested while performing data cleansing task, is to avoid the need to sort large datasets for blocking. This can be done by using recent developments in high-dimensional similarity joins [27]. These techniques use smart data structures to store records so that good candidates for matching are stored together based on the agreed distance or probabilistic measure.

Names and addresses are stored in many sources and usually have high cardinality. For example, finding customer matches is very important for customer relationship management. A number of commercial tools, e.g., IDCENTRIC (FirstLogic), PUREINTEGRATE (Oracle), QUICKADDRESS (QASSystems), REUNION (PitneyBowes), and TRILLIUM (TrilliumSoftware), focus on cleaning name/address kind of data. These tools claim to offer methods as extracting and transforming name and address information into individual standard elements, validating street names, cities, and zip codes. They integrate a huge library of pre-specified rules that deals with the problems usually found in processing of this type of data. For example, TRILLIUM's extraction (parser) and matcher module contains over 200,000 business rules. The tools also provide facilities to extend the rule library with user-defined rules for specific needs. Sample tools for duplicate identification and elimination consists of DATACLEANSER (EDD), MERGE/PURGELIBRARY (Sagent/QMSoftware), MATCHIT (HelpITSystems), and MASTERMERGE (PitneyBowes). Usually, they require the data sources that are already cleaned for matching process. Many approaches for matching attribute values are

supported. Tools such as DATACLEANSER and MERGE/PURGE LIBRARY also allow user-specified matching rules to be incorporated.

In spite of having a large variety of data cleansing tools for schema matching and data scrubbing, accuracy and efficiency are still the issues that need attention. Various intelligent algorithms have been adapted for data cleansing to obtain better performance. Machine learning techniques such as genetic algorithms and fuzzy logic are two of the most influential tools available for detecting and describing fine relationships in huge amounts of apparently unrelated data.

A significant amount of study and research has been made for embedding machine learning techniques in data cleansing methods, nevertheless, neural networks is one of the regions that still seeks attention of the researchers on the subject of data cleansing.

2.3 Objectives

Keeping in view the performance provided by the tools / techniques empowered with intelligent algorithms and presence of limited literature on Machine learning regarding data cleansing, this research proposes a data cleaning methodology based on Supervised learning, Un supervised Learning and Associative Memory Matrix. It aims to design a technique that offers better accuracy and focuses on offering higher efficiency. A distinctive feature of this research is the cleansing of Pakistani names in particular and South Asian names broadly.

PROPOSED SOLUTION

3. Proposed Solution

Keeping in view the performance provided by the tools / techniques empowered with intelligent algorithms and presence of limited literature on neural network regarding data cleansing, this research proposes a data cleaning methodology based on neural network. It aims to design a technique that offers better accuracy and focuses on offering higher efficiency.

A distinctive feature of this research is the cleansing of Pakistani names in particular and South Asian names broadly. Cleansing of South Asian names is difficult, since there is no defined format. For example consider the following table 3.1:

Table 3.1: Sample Data

Name
IQBAL JAVED RTD
J. DAVID STEPHEN
SIRTAJ NABI KHAN AFRIDI
MR KRISHIAN ACHERE SRINATH
SFY YASIR HAYAT KHAN

Names shown in *Table 3.1* contain noise and invalid words within a complete name. As we see there are different formats of names, which is why name cleansing is quite difficult.

Following the Elfeky et al. [38], data cleansing process is categorized into following layers as shown in figure 3.1:

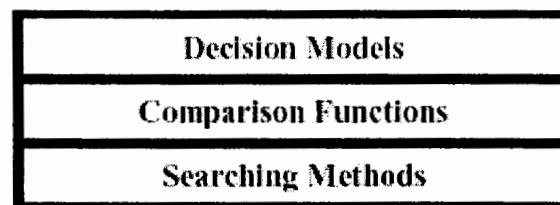


Figure 3.1: Layered Design

Searching Methods	<ul style="list-style-type: none"> ❖ Blocking <ul style="list-style-type: none"> ➤ Sorting ➤ Hashing ❖ Sorted Neighborhood
Comparison Models	<ul style="list-style-type: none"> ❖ Hamming Distance ❖ Edit Distance ❖ Jaro's Algorithm ❖ N-gram ❖ Soundex Code
Decision Models	<ul style="list-style-type: none"> ❖ Probabilistic Model <ul style="list-style-type: none"> ➤ EM-Based ➤ Cost-Based ➤ Error-Based ❖ Induction Model ❖ Clustering Model ❖ <i>Neural Network Based Model</i> ❖ <i>Evolutionary Learning Based Model</i> ❖ <i>Correlation Memory Matrix Based Model</i>

Figure 3.2: List of Possible Solutions

As obvious from the figure 3.2, each step can be carried out in a variety of ways, by adopting different strategies, which are going to be analyzed. The most appropriate among them is implemented finally. Analysis done so far reveals that combination of different techniques yield better results.

This research proposes a Correlation Matrix Memories (CMMs) based, Evolutionary learning based and Neural Network based model for data cleaning and is expected to be a significant enhancement in the domain of data cleaning.

CMM is a form of binary neural network, also known as weight-less neural networks. Such systems are given preference over other types of neural networks because their behavior is very well defined and their implementation is quite easy over desktop computers. CMM is actually an associative memory that is capable of limited fault tolerance and error correction. It is a simple

binary matrix. It works on binary vectors and store associations in a matrix. Other advantage of CMM is that memory requirement is smaller as compared to conventional memories. In our research we used CMM as associative memory matrix.

Evolutionary learning has developed a lot in last few years. Evolutionary computation is an iterative process, as it progresses with growth or development in a population. The common idea behind this algorithm is the same, that is, survival of the fittest. A population of individuals is given; randomly create a set of candidate solutions that comes on the criteria of fitness measures. From these candidate solutions, select the best candidates that can be used to create a new generation. The two operations used for this purpose are: Recombination and Mutation. Recombination is applied on two or more candidates which results in two or more new candidates where as Mutation is applied on a single candidate and a single new candidate is achieved as a result. Hence a new generation is created. This process is thus iterative. In our research we have used mutation operation to create a new generation. The outline of evolutionary algorithm is given in the following figure 3.3 as:

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  END
```

Figure 3.3: A general Scheme of Evolutionary Algorithm

Neural Networks are mathematical constructs that follow the processes people use pattern recognition, learning tasks, and solving problems. Neural Networks are typically characterized in terms of the number and types of connections between individual processing elements, called neurons, and the learning rules that are used when data is presented to the network. Every neuron has a transfer function, usually non-linear, that generates a single output value from all of the input values that are applied to the neuron. Every connection has a weight that is applied to the input value associated with the connection.

A particular organization of neurons and connections is mostly called neural network architecture. The influence of neural networks comes from their talent to learn from experience.

A neural network learns how to identify patterns by adjusting its weights in response to data input. The learning that occurs in a neural network can be supervised or unsupervised. With supervised learning, every training sample has an associated known output value. The difference between the known output value and the neural network output value is used during training to adjust the connection weights in the network. With unsupervised learning, the neural network identifies clusters in the input data that are close to each other based on some mathematical definition of distance. In either case, after a neural network has been trained, it can be deployed within an application and used to make decisions or perform actions when new data is presented. Specifically, 'Perceptron', 'Associative Neural Network' and 'Back Propagation' seem to address the issues regarding data cleansing and provide an appropriate solution for them.

METHODOLOGY AND IMPLEMENTATION

4. Methodology

To identify the expected errors that occur in writing or while typing names, an in depth study and analysis of South Asian names has been performed. The analysis spots large number of expected errors. Besides, it finds probability of error with respect to its length, number of error occurrences and position of error within a word.

Over 5,000 names are present in Urdu dictionary, issued by National Language Authority, alone. Names that are in practice but not mentioned in the dictionary are apart from that. The script of writing South Asian names in local languages is different from English script. Hence, variations may exist in spellings of a name while translating it to English language. Due to this reason, novel ideas, along with implementation, have been presented to produce better results from those generated by conventional methods for removing anomalies in names.

To sum up a complete set of errors is rather unfeasible owing to the fact that each name may give rise to a diverse range of errors. To satisfy maximum number of scenarios and meet the utmost requirements, following model has been formulated:

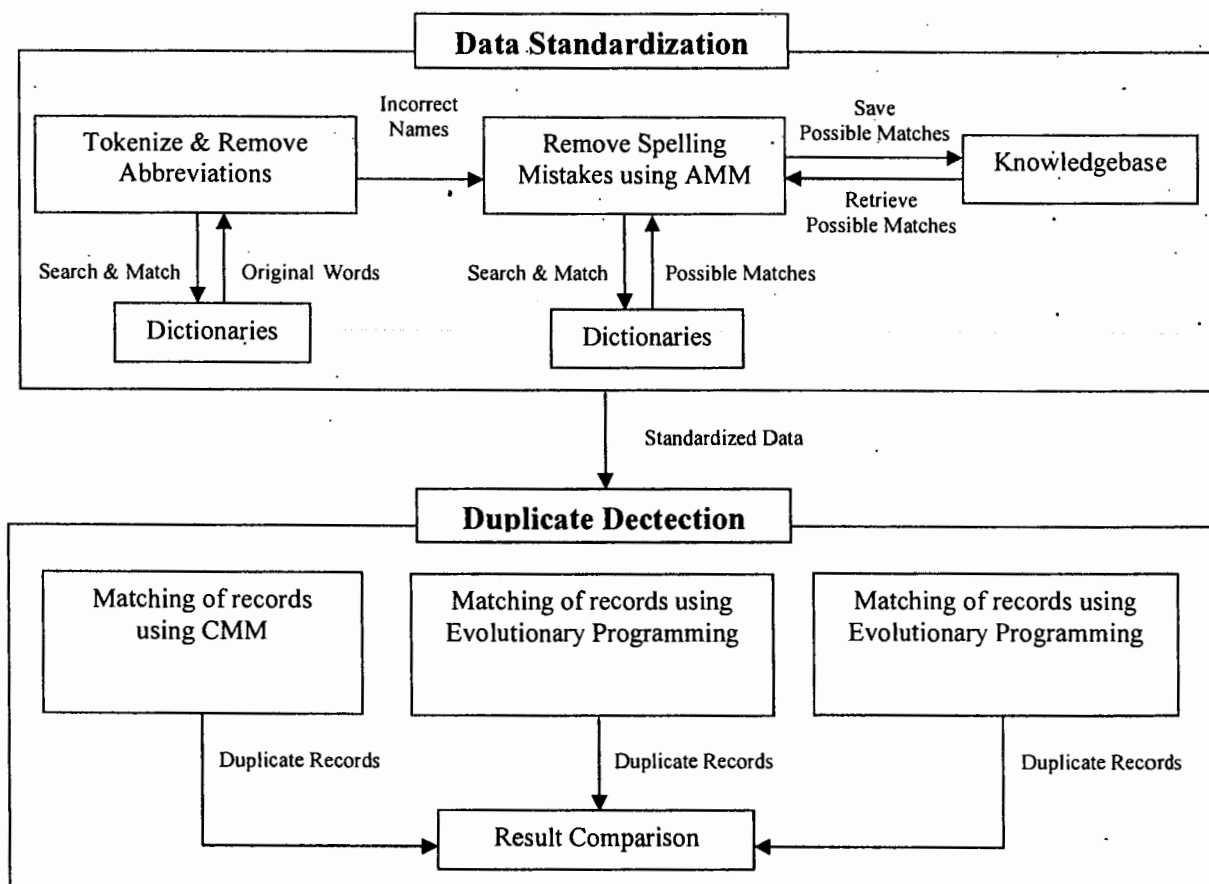


Figure 4.1: Architecture of Data Cleansing Model

The model, presented in *Figure 4.1*, consists of two key modules, which are as follows:

1. Data Standardization
2. Duplicate Detection

Each module is explained in the forth coming sections.

4.1 Data Standardization

Data standardization includes the following sub-modules:

4.1.1 Tokenization

For the purpose of tokenization, three dictionaries are maintained, which are as under:

- a) Dictionary of *Names* (*dNames*)
- b) Dictionary of *Abbreviations* (*of.Names*) (*dAbbs*)
- c) Dictionary of *Postfixes-Prefixes including their abbreviations* (*dPP*)

Dictionary of Names *dNames* includes valid names used for identification of individuals. While, dictionary of Abbreviations *dAbbs* keeps record of the short forms that are commonly used in place of the original names. For example, in South Asian Muslim names, letter 'M' is often used to represent the name 'Muhammad'. Likewise, 'CH' is a representation of the name 'Chaudhary'. Original names related to the short forms are also maintained in the dictionary.

Dictionary of Postfixes/ Prefixes *dPP* comprises words & abbreviations that are not valid names but rather used before or after a real name to show ones designation, marital status etc. 'Captain', 'Retired' & 'Mr' are few examples of prefix/ postfix. Separate dictionaries for prefixes and postfixes are not formed due to the fact that words falling in these dictionaries may overlap.

The designed algorithm parses/ splits a full name into tokens. Spaces and special characters are utilized as delimiters for parsing. Also, special characters and digits are weeded out of name. Each token is compared with all three of the dictionaries.

Table 4.1: Tokens of the input (full) name and specification of their status.

Tokens	Status
Capt	1
Retd	1
CH	2
Rehmat	2
Ali	2
Chairman	1
Golf	0
Club	1

1: Prefix/ Postfix
2: Valid Sub-Name
0: Unidentified

In case token is found in the dictionary of prefixes-postfixes, it is labeled as 1, representing *prefix-postfix* (see Table 4.1). If the token is found in dictionary of Names, it is denoted by 2, representing *valid sub-name*. The token is marked same even if it is found in dictionary of Abbreviations. However, abbreviations are replaced with the original word at the same stage. Token marked 0, symbolizing it as *unidentified*, indicates that it is found in none of the three dictionaries. All dictionaries are exclusive, therefore, no token can belong to more than one dictionary at the same time. Following example demonstrates the scheme of tokenization:

Name before Tokenization:

Capt Retd CH Rehmat Ali Chairman Golf Club

Tokens of Input Name:

Table 4.2: Scheme of Tokenization.

Prefix	Name1	Name2	Name3	Name4	Name5	Name6	Name7	Name8	Postfix
captain retired	chaudhary	rehmat	ali						chairman golf club

Tokens formed can be many in number, however, ten fields are specified to hold them as shown in Table 4.2. Out of ten, eight are fixed to hold valid sub-names or those tokens that are assumed as valid sub-names (because they are not found in the dictionary of prefixes-postfix), each holding single token. That is, eight valid (or assumed to be valid) sub-names can be saved at the most. Among rest of the two fields, one is devoted to prefixes and postfixes each. Due to the fact that prefixes and postfixes contribute a small amount of information to the overall name, therefore, all tokens categorized as prefixes are treated as a single field. Same is true for postfixes.

Accuracy of the results significantly depends on richness of the dictionary. However the proposed algorithm for tokenization is designed to ignore or at least minimize the error even in the absence of a token in any of the three dictionaries.

The process of tokenization is simple as far as all tokens are found in any of the three dictionaries. Nevertheless, the process gets complicated if some or most of the tokens remain unidentified. In such situation, placement of token in appropriate field becomes an issue. To resolve it, first and last indexes of the valid sub-names (as given by (4.i)), provided they exist, are found initially. Refer to Table 4.3.

$$\begin{aligned} \text{firstIndexName} &= \text{tokensArray.IndexOf}(2) \\ \text{lastIndexName} &= \text{tokensArray.LastIndexOf}(2) \end{aligned} \quad (4.i)$$

Prefixes are segregated (as given by (4.ii)) by searching last index of 1 (*prefixes-postfixes*) that falls before the first occurrence of 2 (*valid sub-name*). It starts searching backward till index 0, having *firstIndexName* as starting index.

$$\text{lastIndexPrefix} = \text{tokensArray.LastIndexOf}(1, \text{firstIndexName}) \quad (4.ii)$$

Similarly, postfixes are separated (as given by (4.iii)) by finding first index of 1 (*prefixes-postfixes*) that falls after the last occurrence of 2 (*valid sub-name*).

$$\text{firstIndexPostfix} = \text{tokensArray.IndexOf}(1, \text{lastIndexName}) \quad (4.iii)$$

Finally, sub-name nm_a (valid or unidentified), where a denotes index of field specified for holding sub-names, sets of prefixes prf and postfixes poj , where n denotes length of *tokensArray*, are given as:

$$\begin{aligned} prf &= \sum_{i=0}^{\text{lastIndexPrefix}} \text{tokensArray}_i \\ nm_a &= \text{tokensArray}_i \quad \text{where } a < 8 \text{ \& } \text{lastIndexPrefix} < i < \text{firstIndexPostfix} \\ poj &= \sum_{i=\text{firstIndexPostfix}}^n \text{tokensArray}_i \end{aligned} \quad (4.iv)$$

On the other hand, the designed algorithm may encounter a situation in which no valid sub-name is found among the tokens, that is, *firstIndexName* & *lastIndexName* do not return any valid indexes. In such case, it is unfeasible to find postfixes without having the index(es) of valid sub-name(s) known. Nevertheless, *lastIndexPrefix* can be calculated by taking first index of unidentified token as a reference point, as given by (4.v):

$$\begin{aligned} \text{firstIndexUnidentified} &= \text{tokensArray.IndexOf}(0) \\ \text{lastIndexPrefix} &= \text{tokensArray.LastIndexOf}(1, \text{firstIndexUnidentified}); \end{aligned} \quad (4.v)$$

Hence, name nm_a (valid or unidentified), where a denotes index of field specified for holding sub-names, set of prefixes prf , where n denotes length of *tokensArray*, are given as:

$$\begin{aligned} prf &= \sum_{i=0}^{\text{lastIndexPrefix}} \text{tokensArray}_i \\ nm_a &= \sum_{i=\text{lastIndexPrefix}}^n \text{tokensArray}_i \quad \text{where } a < 8 \end{aligned} \quad (4.vi)$$

Table 4.3: Distribution of tokens into fields.

Tokens	Status
Retd	1 (Prefix/Postfix- Abbreviation)
Maj	0 (Unidentified)
General	1 (Prefix/Postfix)
Gull	0 (Unidentified)
M	2 (Valid Name - Abbreviation)
Faroq	0 (Unidentified)
Ali	2 (Valid Name)
Noore	0 (Unidentified)
Chairman	1 (Prefix/Postfix)
Golf	0 (Unidentified)
Club	1 (Prefix/Postfix)

In addition to producing the aforementioned results, this module isolates incorrect names from the correct ones that exist in the input data and forwards them to the next module for further processing.

4.1.2 Associative Matrix Memory (AMM) for Removing Spelling Mistakes

Using Associative Memory is helpful in removing noise out of input data and allows data recall at higher speed [45]. The purpose of implementing Associative Matrix Memory was to dig out possible matches against incorrect names, eventually, making the desired data error prone. This module can be viewed as illustrated in Figure 4.2:

TH-5096

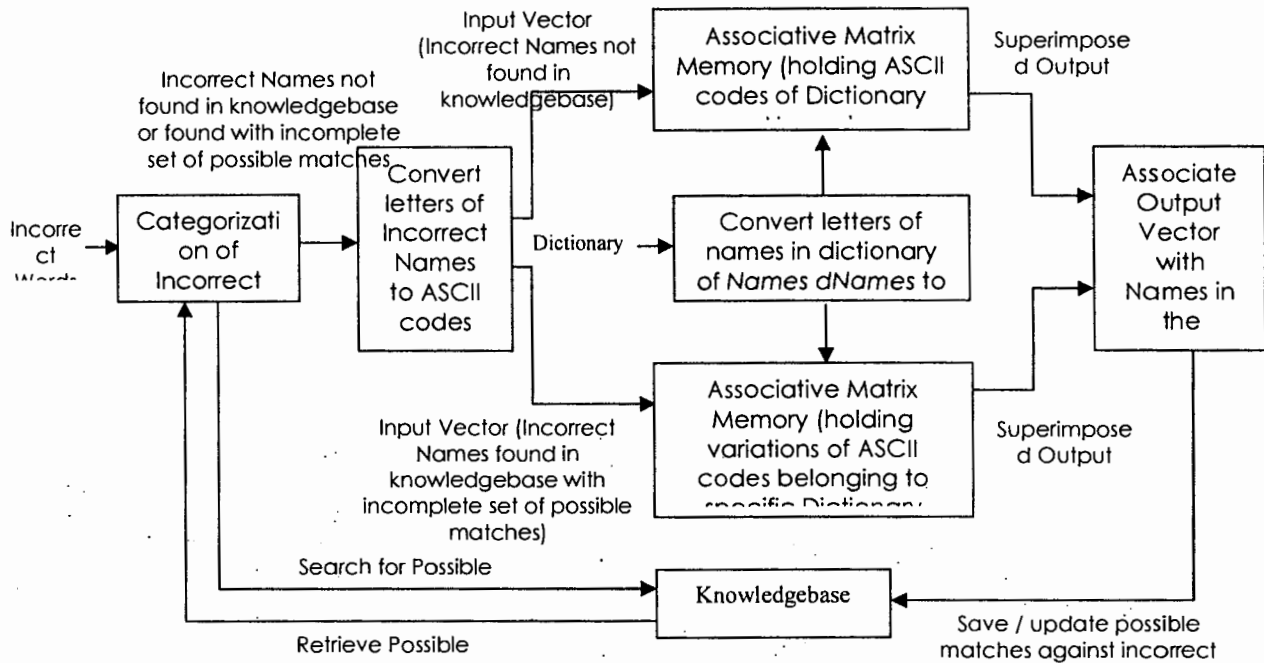


Figure 4.2: Architecture of the Sub-Module designed for the removal of Spelling Mistakes

4.1.2.1 Associative Matrix Memory

Associative Matrix Memory M of size $i * j$ maps input vector of length i to the output vector of length j . Incorrect Names are taken as input and matching names are generated as output. Input words are stored in the form of vector. To make comparison easy and avoid lengthy complex conversions, names of the dictionary are converted to ASCII codes, and incorporated into matrix M , such that ASCII codes of each name are placed horizontally / column wise (as shown in Figure 4.3). Number of rows i of Matrix M is equal to estimated length of the largest word in dictionary of names ($dNames$) (see Table 4.4). Analysis has shown that none of the sub-names, incorrect or valid, consists of greater than fifteen letters. Number of columns j of Matrix M is equal to the count of names existing in the dictionary.

Similar to the names belonging to dictionary ($dNames$), letters of the incorrect name are represented as ASCII codes and stored into the elements of input vector. All spaces preceding and following the incorrect name are removed. Moreover, names are shifted to lower case before converting their letters to ASCII codes. Input vector corresponds to the rows i of the matrix, whereas, output vector corresponds to columns j of the matrix [41]. The mapping function $f(m)$ is given as:

$$f(m) : \{\text{ASCII codes}\}^i \rightarrow \{0,1\}^j$$

Table 4.4: Dictionary of Names.

Index	ID	Names
0	1	Abdullah
1	2	Alexander
2	3	babar
3	4	Jamshaid
4	5	Naseer
5	6	Nasir
6	7	Nazir
7	8	Palwasha
8	9	Rajesh
9	10	Yousaf
10	11	Zahid

Input Vector
ASCII Codes of 'Naser'

0	110
1	97
2	115
3	101
4	114
5	
6	
7	
8	

ASCII codes of characters belonging to Dictionary Names form Associative Matrix Memory. Codes of each are placed horizontally / column wise

	0	1	2	3	4	5	6	7	8	9	10
0	97	97	98	106	110	110	110	112	114	121	122
1	98	108	97	97	97	97	97	97	97	111	97
2	100	101	98	109	115	115	122	108	106	117	104
3	117	120	97	115	101	105	105	119	101	115	105
4	108	97	114	104	101	114	114	97	115	97	100
5	108	110		97	114			115	104	102	
6	97	100		105				104			
7	104	101		100				97			
8		114									

Output Vector

0	0	0	1	4	4	3	1	2	0	1
---	---	---	---	---	---	---	---	---	---	---

Superimposed Output Vector

0	0	0	0	1	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---

Figure 4.3: Structure of Associative Matrix Memory

As shown in Figure 4.3, values in elements of output vector, corresponding to the column indexes of matrix M , are set equal to the count of ASCII codes (letters) found same between incorrect word (held in input vector) and dictionary names (held in matrix column wise). The

comparison is one to one, that is, only same indexes of input vector and matrix M are compared. Matrix M is processed horizontally to find the total number of common ASCII codes (letters) between input vector and matrix M [43]. Mathematically, it is denoted by (4.vii):

$$\text{outputVector}_j = \text{outputVector}_j + 1 \quad \text{where} \quad \text{inputVector}_i = M_{ij} \quad (4.vii)$$

Values in the elements of output vector are superimposed to 0, denoting *possible match*, or 1, denoting *non-match*. This is achieved by passing the values through a threshold function $f(t)$ given by (4.viii).

$$\begin{aligned} li &= \text{length of incorrect name} \\ ld &= \text{length of dictionary name} \\ \text{averageLength} &= (Li + Ld) / 2 \\ t &= \text{outputVector}_j / \text{averageLength} \\ r &= \begin{cases} 0.66; & li \leq 3 \\ 0.85; & li = ld \\ 0.75; & \text{otherwise} \end{cases} \end{aligned}$$

$$f(t) = \begin{cases} 0; & t < r \\ 1; & t \geq r \end{cases} \quad (4.viii)$$

Results have been analyzed against different values of t ranging from 0 to 1. However, outcome is much better when t is tuned to figures between 0.66 and 0.85.

Indexes of the values that are set to 1 in superimposed output vector S , finally, serve as references to the actual words in the dictionary (*shown in Table 4.4*) [neural spell checker].

4.1.2.2 Knowledgebase

In future perspective, associations between input vector and output vector are saved in knowledgebase for the purpose of reuse. The largest identity¹ *lid* found in dictionary of *names* (*see Table 4.3*), at the time of last comparison, is also saved in the knowledgebase along with the incorrect name and its possible matches. These values are updated, if required, each time that incorrect name is presented to Associative Matrix memory for processing.

Table 4.5: Fields of knowledgebase.

Incorrect Name	PossibleMatches	LastDictID
Naser	Naseer, Nasir, Nazeer	70000

¹ Dictionary is sorted by identity values that are auto increment. Existing entries of dictionary cannot be modified, yet, new items can be added into or deleted out of the dictionary.

The logic behind maintaining knowledgebase of possible matches found against an incorrect word is to avoid going through same process each time it is encountered. This practice, eventually, saves the time in terms of processing and resources.

$$clid = 4,clid = 10$$

Input Vector
ASCII Codes of 'Naser'

ASCII codes of characters belonging to Dictionary Names form Associative Matrix Memory. Codes of each name are placed horizontally / column wise

Processing starts from this point forward

0	110										
1	97										
2	115										
3	101										
4	114										
5											
6											
7											
8											

	0	1	2	3	4	5	6	7	8	9	10
0	97	97	98	106	110	110	110	112	114	121	122
1	98	108	97	97	97	97	97	97	97	111	97
2	100	101	98	109	115	115	122	108	106	117	104
3	117	120	97	115	101	105	105	119	101	115	105
4	108	97	114	104	101	114	114	97	115	97	100
5	108	110		97	114			115	104	102	
6	97	100		105				104			
7	104	101		100				97			
8		114									

Output Vector

0	0	0	0	0	4	3	1	2	0	1
---	---	---	---	---	---	---	---	---	---	---

Superimposed Output Vector

0	0	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---

Figure 4.4: processing of Associative Matrix Memory for incomplete set of possible matches retrieved from knowledgebase

Possible matches P for item r belonging to set of incorrect names (IN) are searched in the knowledgebase prior to performing comparison via Associative Matrix Memory. Set of names ($IN1$) with possible matches not found in knowledgebase K are separated out. They are processed through Associative Matrix Memory as explained earlier. The remaining ones ($IN2$) are further categorized into two sets. Identity $clid$ of name corresponding to the largest index of dictionary, in its current version, serves as criteria of categorization. If lid is equal to $clid$, it shows that no further word has been added into the dictionary since last processing performed for the incorrect name under consideration. Hence no further processing is required and set of matching words ($IN2a$) retrieved from knowledgebase encompasses maximum possible matches that could be computed through Associated Matrix Memory. In the other case, set of possible matches ($IN2b$) retrieved from knowledgebase may be incomplete. To complete the set, such incorrect names are forward to form input vector and Associative Matrix Memory processes them only for the dictionary names having identity greater than lid (as shown in Figure 4.4).

$$IN1 = IN - K$$

$(IN2a)_r$ is complete subset of K if complete Pr is found in K

$(IN2b)_r$ is incomplete subset of K if partial Pr is found in K

4.1.2.3.a Associative Matrix Memory - Novel Method of Matching by Changing Positions of Letters

According to the analysis, mentioned in the beginning of this chapter, anomalies occur at two positions within a name at the most. Although, increase in length of name increases probability of typing mistakes, however, proposed method is based upon the results of analysis performed.

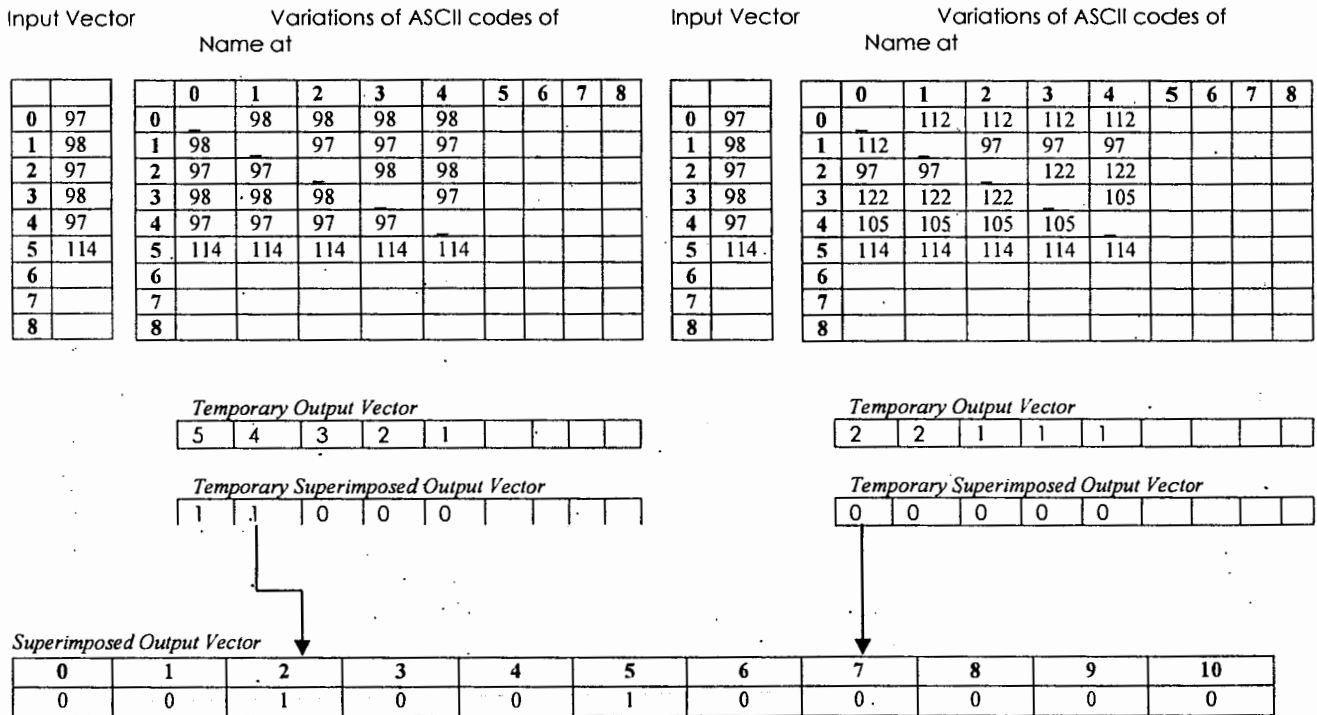


Figure 4.5: Structure of Associative Matrix Memory - Novel method of Matching

There are frequent cases in which desired results are achieved merely by shifting the positions of letters of the names that are compared for finding possible matches. For example, an incorrect word 'ababar' does not bear any match after getting processed in last sub-module. By adjusting position of letters forming 'babar', it can be seen as potential match for incorrect name 'ababar'. To bring it to practice, novel method of matching embedded with Associative Matrix Memory has been presented in this module.

As stated earlier, only specific names of $dNames$ are selected as input for the current Associative $i * i$ Matrix Memory. Length of largest word in $dNames$ denotes dimension i of matrix M . In first step, difference $diff$ (given by (4.ix)) between length l of the incorrect name r and length dl of

name a belongs to $dNames$ is calculated. If the difference is less than or equal to one, it is considered for further processing.

$$\begin{aligned} diff &= l - dl & (4.ix) \\ input &= \{ r \in IN ; 0 \geq diff < 2 \\ & \quad a \in dNames ; -2 > diff \geq -1 \end{aligned}$$

String with larger length is taken as input and stored in input vector, whereas, variations of the other string form Associative Matrix Memory. These variations are formed by changing positions of letters of name as shown in *Figure 4.5*. In *input vector* and matrix M , letters are represented in the form of ASCII codes. Each variation is stored in a separate column of matrix M . Special character '_' represents an empty cell that returns zero when matched with letter in the related index of *input vector*. In first column, first element consists of special character '_' and rest of the elements contain letters (ASCII codes) of the name considered to form matrix M . If we see it. In second column, '_' exchanges its position with first letter of the name. Remaining letters retain their position in the matrix with respect to i . In each column, '_' exchanges its position with the letter held in the next index of previous column. This process continues until z , given by (4.x), number of variations are formed and stored in upto z th column of the matrix.

Associative Memory Matrix is processed almost same as mentioned in the last sub-module, except for few alterations. No of similar letters between input vector and particular column of the associative memory matrix is stored in the respective index of temporary output vector. Threshold function $f(t)$, given as (4.xi), varies with length l of incorrect names. If length of incorrect name is less than or equal to four and incorrect name is smaller than dictionary name, threshold value is equal to the length of incorrect name. Threshold value z , is equal to one less than the length of incorrect name, otherwise. Threshold value z also determines the number of variations of name, either r or a , forming matrix M .

$$\begin{aligned} z &= \{ l ; l \leq 4 \\ & \quad l-1 ; l > 4 \end{aligned} \quad (4.x)$$

$$\begin{aligned} f(t) &= \{ 0 ; t < z \\ & \quad 1 ; t \geq z \end{aligned} \quad (4.xi)$$

If any index of temporary superimposed output vector is found equal to 1, it shows that the name of dictionary, which comparison was taking place for, is a possible match of the respective incorrect word. Hence, index of superimposed output vector S_k corresponding to that name is set to 1. To save the cost of storage and processing, single superimposed vector is used for storing results of output vectors generated by all three of the Associative Memory Matrices.

4.1.2.3.b Associative Matrix Memory - Novel Method of Matching by Sorting Letters

While typing a name, often does it happen that positions of the letters get exchanged. The error may not exist in two adjacent letters rather it can occur at any position in the word. For example, 'qureshi' is misspelled as 'quhresi'. Likewise, 'rathore' is correct form of 'ratohre'.

To provide solution for such anomalies, ASCII values of letters belonging to incorrect name and those of the names selected from dictionary are sorted and compared. Sorted ASCII codes of incorrect name form input vector. Whereas, ASCII codes of selected dictionary names form AMM (see Figure 4.6).

Input Vector
Sorted ASCII Codes
of 'Nsair'

0	97
1	105
2	110
3	114
4	115
5	
6	
7	
8	

Sorted ASCII codes of characters belonging to Dictionary Names form Associative Matrix Memory. Codes of each name are placed horizontally / column wise

	0	1	2	3	4	5	6	7	8	9	10
0	97	97	97	97	97	97	97	97	97	97	97
1	97	97	97	97	101	105	105	97	101	102	100
2	98	100	98	100	101	110	110	97	104	111	104
3	100	101	98	104	110	114	114	104	106	115	105
4	104	101	114	105	114	115	122	108	114	117	122
5	108	108		106	115			112	115	121	
6	108	110		109				115			
7	117	114		115				119			
8		120									

Output Vector

0	0	0	0	0	0	5	4	1	2	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Superimposed Output Vector

0	0	0	0	0	0	1	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---

Figure 4.6: Structure of Associative Matrix Memory - Novel method of Matching

Selection of the names from the dictionary is done by matching length l of the incorrect name with the length dl of dictionary names. Names having same length as that of the incorrect name ($l=dl$), are selected out of the dictionary to form AMM. Dimensions of AMM and their lengths are same as mentioned in section 4.1.2.1.

No of similar letters between input vector and particular column of the associative memory matrix is stored in the respective index of temporary output vector. Threshold function $f(t)$, given as (4.xii), is proportional to the length of Incorrect Name. In other words, if all the letters of two sorted strings are equal, the corresponding dictionary name is a potential match. Therefore, value in the respective index of superimposed vector is set to 1. The results contained in superimposed output vector, after processing previous AMMs, are retained and new values are added into them. This increases the priority of a matching name retrieved by more than one AMM.

$$f(t) = \begin{cases} 0; & t < l \\ 1; & t = l \end{cases} \quad (4.xii)$$

4.1.2.4 Set of Possible Matches

Set of possible matches PM for incorrect name k is sum of *output vector* S_k generated by Associative Memory Matrices and set(s) of possible matches retrieved from knowledgebase described in previous modules. It is given by (4.xiii):

$$PM_k = S_k \cup (IN2a)_k \cup (IN2b)_k \quad (4.xiii)$$

This module prepares data for duplicate elimination and widely helps improve the results produced by algorithms that identify duplicates.

4.2 Duplicate Detection

In various organizations, specifically the ones that maintain customer information, the database maintained holds similar records. These records are repeated exactly or with changes that, at times, get difficult but essential to remove. Three different techniques, mentioned below, have been designed for the said purpose.

1. Duplicate Detection technique based on *Associative Matrix Memory*
2. Duplicate Detection technique based on *Evolutionary Programming*
3. Duplicate Detection technique based on *Neural Networks*

Each sub-module is explained in the following sections.

4.2.1 Duplicate Detection technique based on *Associative Matrix Memory*

Set of records R (shown in Table 4.6) is considered for duplicate elimination. The proposed method uses Associative $i * j$ Matrix memory M , where i represents number of attributes in R and j is equal to number of records in R .

Table 4.6: Organization of Records considered for Duplicate Detection.
For simplicity, limited attributes are shown in the table

Index	Prefix	Name 1	Name2	Postfix	House	Street	Sector	Contact No	NIC No
0		wasim	qureshi		2	5	G-7/2	923049538853	6110192168121
1		faiz	khan		25	7	F-10/4	923045121357	6110122370281
2		ahmed	abdullah		8	37	F-6/2	923049029362	6110194182847
3		faiz	khan	retired	25	7	F-10/4	92304512135	6110122370281
4	captain	wasim			2A	5	G-7/2	923049538853	6110192168121

As shown in Figure 4.6, Records are placed in the matrix M , column wise, where each attribute of a record is placed in a separate cell of the column. A notable feature is that input vector is not defined separately to hold the records. Instead, record to be matched is selected within the Matrix M and compared with rest of the records falling towards right of it in the matrix M , according to the commutative law of algebra, as given by (4.xiv).

$$\text{If Record 1} = \text{Record 2, then Record 1} = \text{Record 2} \quad (4.xiv)$$

Records are matched attribute by attribute (one to one), which means that attribute at index 1 of second record (placed in column 2) is matched with the attribute at index 1 of the third record (contained in column 3). Matrix M is processed horizontally or row wise. In other words attribute at index 1 of an input record is matched with the same attribute (lying at the same position) of rest of the rows. Matrix M is processed as given by (4.xv):

$$r = \text{column index of input record}$$

$$\text{output vector}_i = \sum_{n=r+1}^j M_{ir} \cdot M_{in} \tag{4.xv}$$

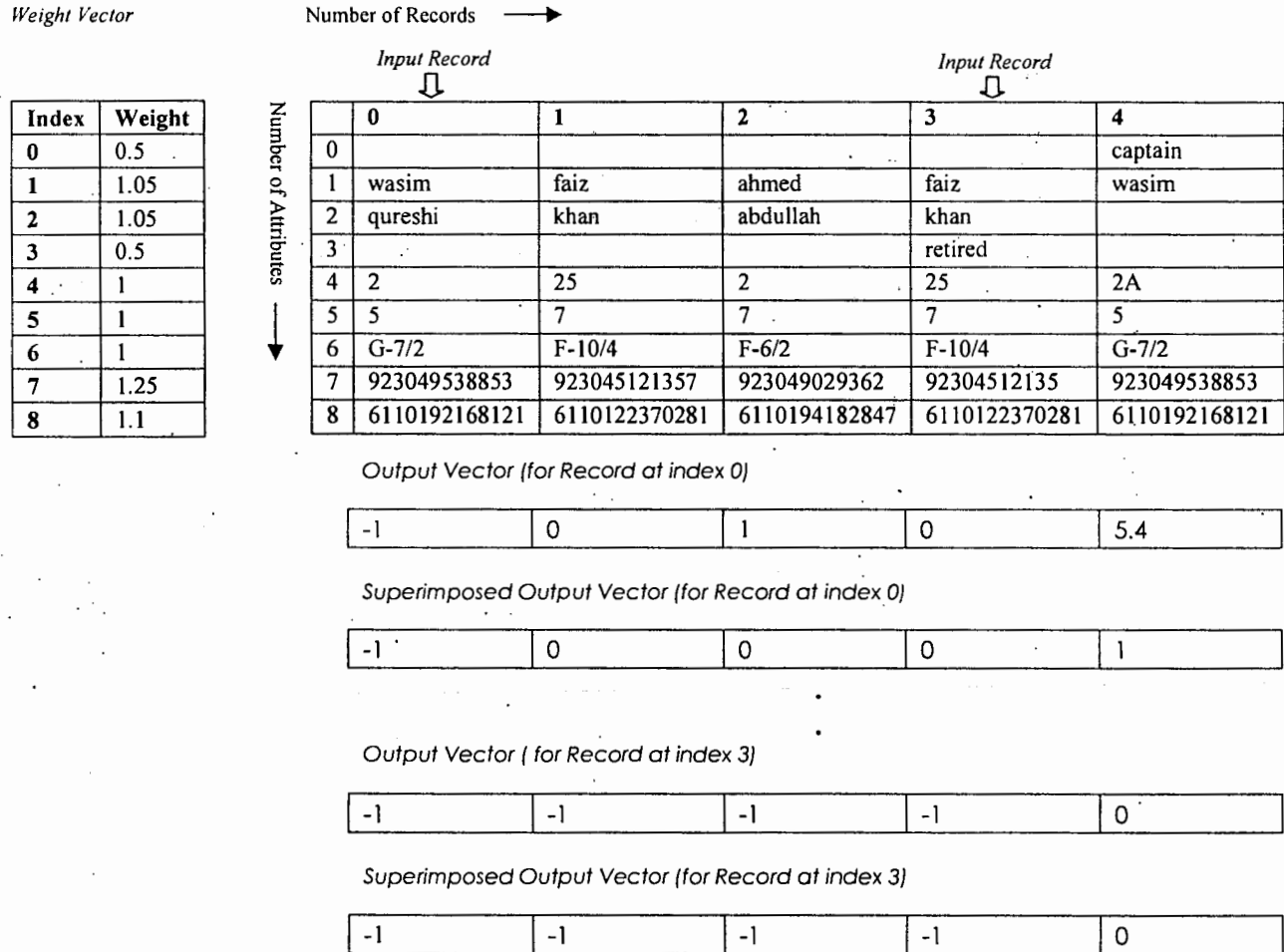


Figure 4.7: Structure of Associative Matrix Memory for Duplicate Detection

Weight vector W specifies the significance attached with an attribute of a relation. Extracting information out of data plays vital in finding hidden associations within a record set that, eventually, helps in duplicate detection. In other words, the degree of information that an attribute contributes to a relation is specified in weight vector W . For example, ones name provides more valid information rather than ones designation. To be more precise, postfix and prefix are often ignored as they do not provide enough information to identify an individual.

Also, such attributes are often left unfilled / empty. Therefore, amount of weight assigned to prefix and postfix is 0.5. However, higher weights are assigned to substantial information giving attributes, which include *Name1*, *Name2*, *NIC No* (*National Identification Card number*) and *Contact No*. As shown in *Figure 4.7*, elements of weight vector W hold values in accordance with position of the attributes in matrix M .

The elements of output vector correspond to the columns of matrix M . hence, length of output vector j is same as the number of input records. -1 is assigned to element of output vector which points to the index of input record (as shown in *Figure 4.7*). Due to the fact that records prior to input record are not considered for matching, therefore, values of all the elements of output vector on left side of input record are also set to -1. Values for rest of the elements are evaluated by (4.xvi).

$$\text{outputVector}_j = \begin{cases} -1 & \text{where } j \leq r \\ \sum_{n=0}^i W_n & \text{where } j > r \quad \text{where } M_{ir} = M_{ij} \end{cases} \quad (4.xvi)$$

Values in the weight vector corresponding to the attributes having same value between input record r and the other record j , are summed up to get the final value for element j of output vector.

Output vector is superimposed by threshold function $f(t)$ given by (4.xvii):

$n =$ number of non null or non empty attributes of input record

$t = \text{outputVector}_j / n$

$$f(t) = \begin{cases} 0; & t < 0.75 \\ 1; & t \geq 0.75 \end{cases} \quad (4.xvii)$$

The threshold value t is set after analyzing results of numerous record sets. 0.75, by and large, produces best results.

	0	1	2	3	4
0	-1	0 ^a	0 ^b	0 ^c	1 ^d
1	0 ^a	-1	0 ^e	1 ^f	0 ^g
2	0 ^b	0 ^c	-1	0 ^h	0 ⁱ
3	0 ^c	1 ^f	0 ^h	-1	0 ^j
4	1 ^d	0 ^g	0 ⁱ	0 ^j	-1

Figure 4.8: Matrix for saving superimposed output results

A square $u * u$ matrix S , shown in Figure 4.8, combines superimposed output vectors generated for all input records. The matrix is completed following the rule stated by (4.xviii). The matrix is interpreted to generate the results shown in Table 4.7.

$$v = u$$

$$S_{vu} = S_{vu} \quad (4.xviii)$$

Table 4.7: Set of Duplicate Rows

Index	Prefix	Name1	Name2	Postfix	House	Street	Sector	Phone	NIC No
0		wasim	qureshi		2	5	G-7/2	923049538853	6110192168121
4	captain	wasim			2A	5	G-7/2	923049538853	6110192168121
1		faiz	khan		25	7	F-10/4	923045121357	6110122370281
3		faiz	khan	retired	25	7	F-10/4	92304512135	6110122370281
2		ahmed	abdullah		8	37	F-6/2	923049029362	6110194182847
3		faiz	khan	retired	25	7	F-10/4	92304512135	6110122370281
1		faiz	khan		25	7	F-10/4	923045121357	6110122370281
4	captain	wasim			2A	5	G-7/2	923049538853	6110192168121
0		wasim	qureshi		2	5	G-7/2	923049538853	6110192168121

To find hidden associations, association law, given by (4.xix) is followed.

$$\text{If } a = b \ \& \ b = c \quad \text{then } a = c \quad (4.xix)$$

As an example,

$$\text{If } \text{Record 1} = \text{Record 3}, \text{Record 3} = \text{Record 2} \quad \text{then } \text{Record 1} = \text{Record 2}$$

4.2.2 Evolutionary Algorithms

In the field of search and optimization techniques, the development of Evolutionary Algorithms (EA) has been very important. Evolutionary Algorithms are a set of heuristics used successfully in many applications whose solutions are difficult to get.

Most of the present implementations of Evolutionary Algorithms come from any of these three basic types:

1. Genetic Algorithms (GA)
2. Evolutionary Programming (EP)
3. Evolutionary Strategies (ES).

Diagrammatically it can be shown as in figure 4.5:

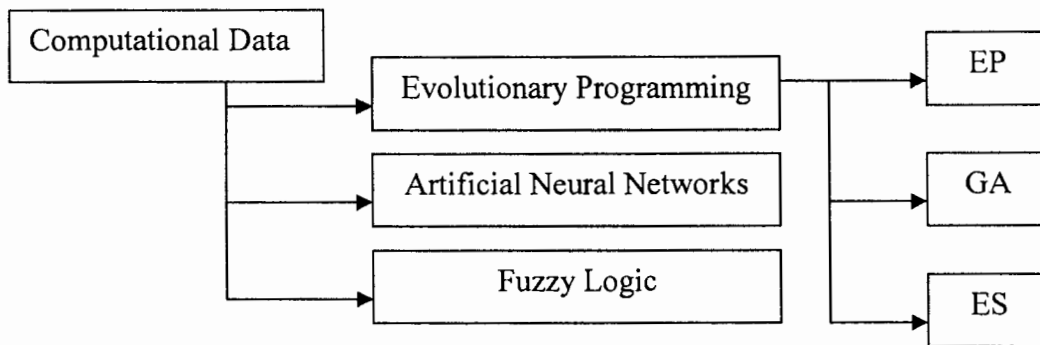


Figure 4.5: Location of the different families of evolutionary algorithms.

4.2.2.1 Execution of Evolutionary Algorithm

Diagrammatically execution of Evolutionary Algorithm can be shown as in Figure 4.9 below:

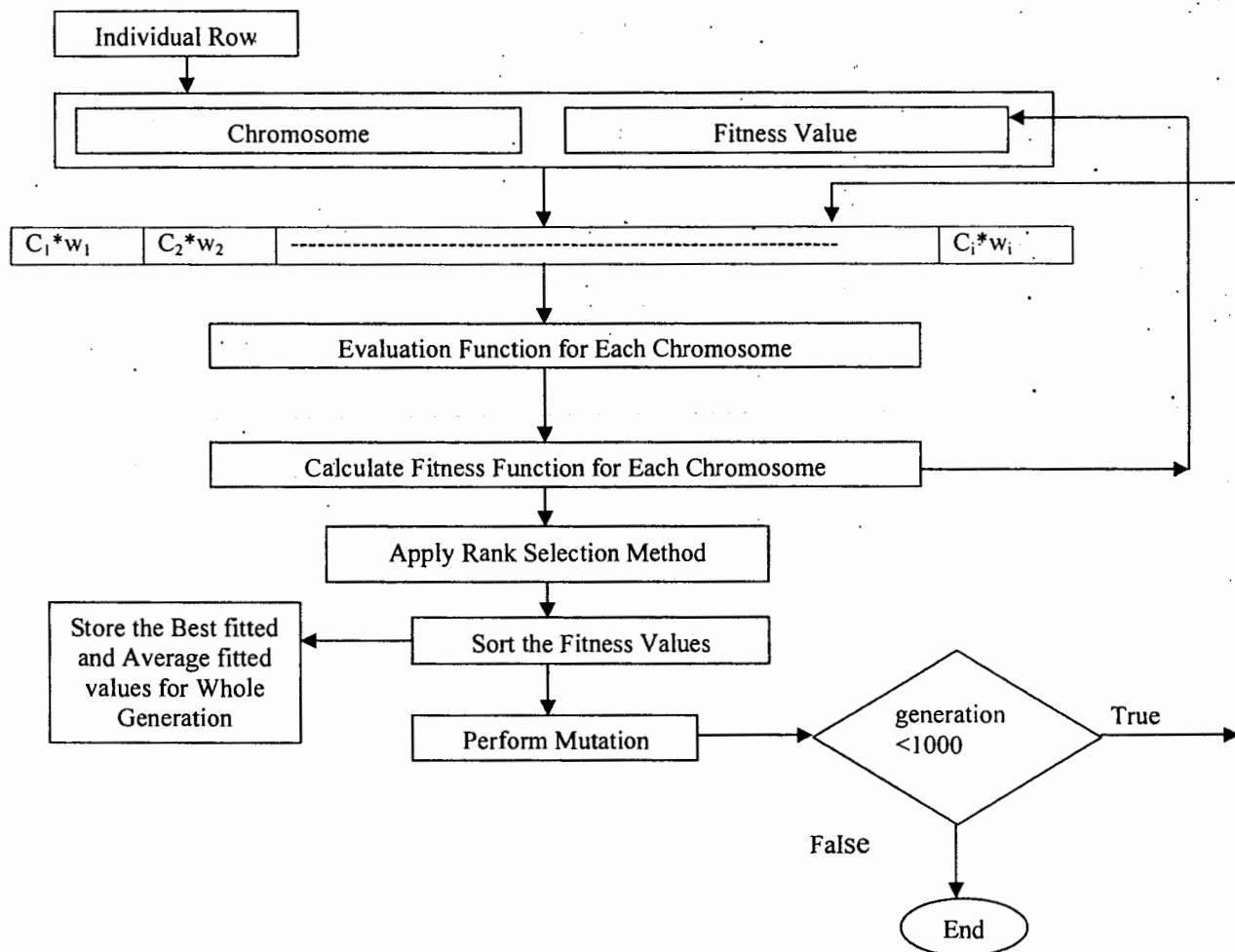


Figure 4.6: Execution Diagram of Evolutionary Algorithm

4.2.2.2 Components of Evolutionary Algorithm

In order to define Evolutionary Algorithm (EA), it is important to understand the components of EA to get the complete picture. Following are the components of EA:

Representation

In this component we connect the real world data to EA world data, that is we define a bridge between the original problem context and problem solving space where evolution will take place. The Evolutionary algorithm operates on a population of strings that represents tentative solutions. Every string represents an *individual row* and it is composed of number of attributes that define the row as it can be seen in the following figure 4.10:

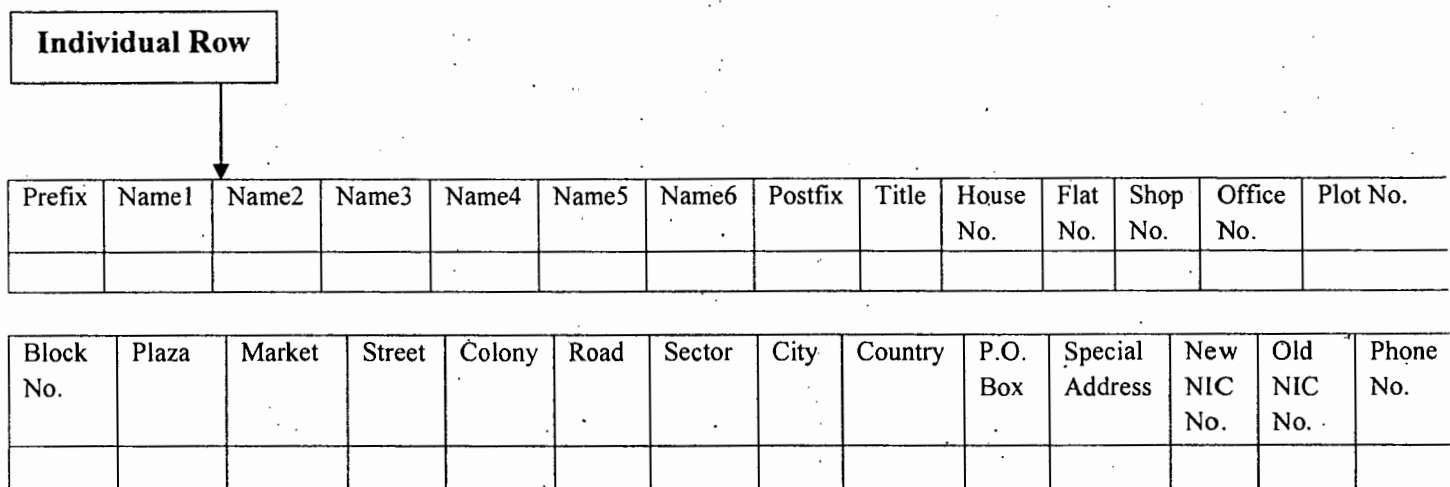


Figure 4.10: Representation of Individual Row

Population (Generation)

Population holds the representations of possible solutions. It forms the unit of evolution. Given a representation, defining a population is as simple as Specifying how many individuals are present in it that is setting the size of population.

Evaluation Function (Fitness Function)

The evaluation function is called fitness function in Evolutionary Algorithm. It sets the basis for the selection of suitable (best fitted) chromosomes to be selected as a part of new generation. Mathematically it is given as:

$F(R) = \sum (f(R_i) * w_i)$; where R_i stands for the i th attribute value of row and w_i is the weight of the i th attribute.

$$F(R) = \begin{cases} \text{String Length}(R_i) & \text{if } i < 26 \\ 0 & \text{if } R_i = "" \\ \text{Ascii Encode } (R_i) & \text{if } i \Rightarrow 26, 27, 28 \end{cases}$$

Where String Length(R_i) represents a function that gives the length of the value of i th attribute of a row and a condition $i < 26$ is enforced as we calculate the string length of 26 attribute values of a particular row. If $f(R_i)$ is equal to zero than this means that value of i th attribute is null. We considered Ascii Encode of attribute 26th, 27th and 28th value of a row as it is a New NIC No., Old NIC No. and Phone No. whose length for every individual is same and hence if we had taken the string length than it will always show some similarity between all the rows. For this reason we considered ASCII Encode of 26th, 27th and 28th attributes. Mathematically it is shown as by considering example of two rows:

Prefix	Name1	Name2	Name3	Name4	Name5	Name6	Postfix	Title	House No.	Flat No.	Shop No.	Office No.	Plot No.
Mister	Safir	Hussain							6				
Mr.	Safeer	Hussain							6				

Block No.	Plaza	Market	Street	Colony	Road	Sector	City	Country	P.O. Box	Special Address	New NIC No.	Old NIC No.	Phone No.
			60			G-7/4	Islamabad	Pakistan	44000				
			60			G-7/4	Islamabad	Pakistan	44000				

Calculating the string length of Row1 and Row2 as shown below:

Prefix	Name1	Name2	Name3	Name4	Name5	Name6	Postfix	Title	House No.	Flat No.	Shop No.	Office No.	Plot No.
6	5	7	0	0	0	0	0	0	1	0	0	0	0
3	6	7	0	0	0	0	0	0	1	0	0	0	0

Block No.	Plaza	Market	Street	Colony	Road	Sector	City	Country	P.O. Box	Special Address	New NIC No.	Old NIC No.	Phone No.
0	0	0	2	0	0	5	9	8	5	0	0	0	0
0	0	0	2	0	0	5	9	8	5	0	0	0	0

Each string length of a particular row is than multiplied with a weight of i th attribute. The weight vector which is obtained by our algorithm is shown as under:

Prefix	Name1	Name2	Name3	Name4	Name5	Name6	Postfix	Title	House No.	Flat No.	Shop No.	Office No.	Plot No.
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}	w_{13}	w_{14}
0.03	0.08	0.08	0.05	0.08	0.14	0.07	0.15	0.13	0	0.17	0.06	0.09	0.05

Block No.	Plaza	Market	Street	Colony	Road	Sector	City	Country	P.O. Box	Special Address	New NIC No.	Old NIC No.	Phone No.
w_{15}	w_{16}	w_{17}	w_{18}	w_{19}	w_{20}	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	w_{26}	w_{27}	w_{28}
0.15	0.13	0.18	0.17	0.07	0.05	0.18	0.01	0.07	0.07	0.15	0.12	0.17	0.18

where w_1, w_2, \dots, w_{28} represents weights associated with each attribute.

In next step we will calculate Evaluation Function for each of the two rows by multiplying the string length of each value of attribute of a row with the respective weight if that attribute and than taking sum of it. Mathematically we show it as:

$$F(R) = \sum (f(R_i) * w_i)$$

So,

$$F(R1) = 3.64 \text{ and } F(R2) = 3.765$$

The evaluation function value of both the rows R1 and R2 is close which indicates that these two rows hold some similarity between each other.

Survivor Selection Method

The role of survivor selection method is to distinguish among individuals based on their quality. This method is called in evolutionary algorithm after off springs are created from the selected parents. The selection method being used by us is known as Rank Selection Method that first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1 , second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

Variation Operators

Variation operators are used in evolutionary algorithm to generate new individuals (off springs) from the old ones. There are two types of variation operators in evolutionary learning as follows:

- *Recombination or Crossover:* In this process, a new offspring is obtained by merging information obtained from two parents into one or two offspring. Te principle behind recombination is simple, that is by mating two individuals with different but desirable

features, we can produce an offspring that combines both of those features. It is a binary variation operator as it involves two operands to produce a result.

- **Mutation:** Mutation is a unary variation operator, as it is applied to one individual and a new mutant is produced with slight desirable differences. Mutation is supposed to cause random and unbiased change to produce new offspring. In our research, we have used mutation operator to produce new offspring randomly. Mathematically it can be explained as under:

Suppose S represents the set of fitness values of all chromosomes. Consider S' as a subset of S represented as:

$$S' = \text{Subset of } S$$

$$S' = \min_{i=1}^{n'} \{S\}$$

Where n' is the top 1/3rd of the population. We considered n'=5 as from experiments of David Fogel [42] it is seen that this value gives better results.

Perform mutation operation, S= mutation (S')

$$\text{Updated weight} = \text{previous weight} + \Delta w$$

$$w = w' + \Delta w$$

where η is mutation rate and; $\Delta w = (\eta * w')$

Experiments performed by us showed that η gave better results in the range 0.05 to 0.09. η is that small value which we have used to edit or bring a minor change in the weight associated with each attribute.

The new weights w for each attribute are obtained by multiplying η=0.05 with the previously obtained weights w' of every attribute which gives us Δw, than we add Δw in the previous weights w' to get updated weights. This process is shown below:

Prefix	Name1	Name2	Name3	Name4	Name5	Name6	Postfix
w ₁	w ₂	w ₃	w ₄	w ₅	w ₆	w ₇	w ₈
0.03+(0.05*0.03)=0.0315	0.08+(0.05*0.08)=0.084	0.08+(0.05*0.08)=0.084	0.05+(0.05*0.05)=0.0525	0.08+(0.05*0.08)=0.084	0.14+(0.05*0.14)=0.147	0.07+(0.05*0.07)=0.0735	0.15+(0.05*0.15)=0.1575

Title	House No.	Flat No.	Shop No.	Office No.	Plot No.	Block No.
w ₉	w ₁₀	w ₁₁	w ₁₂	w ₁₃	w ₁₄	w ₁₅
0.13+(0.05*0.13)=0.1365	0+(0.05*0)=0	0.17+(0.05*0.17)=0.1785	0.06+(0.05*0.06)=0.063	0.09+(0.05*0.09)=0.0945	0.05+(0.05*0.05)=0.0525	0.15+(0.05*0.15)=0.1575

Colony	Road	Sector	City	Country	P.O.Box	Special Address	New NIC No.	Old NIC No.	Phone No.
w_{19}	w_{20}	w_{21}	w_{22}	w_{23}	w_{24}	w_{25}	w_{26}	w_{27}	w_{28}
$0.07+(0.05*0.07)=0.0735$	$0.05+(0.05*0.05)=0.0525$	$0.18+(0.05*0.18)=0.189$	$0.01+(0.05*0.01)=0.0105$	$0.07+(0.05*0.07)=0.0735$	$0.07+(0.05*0.07)=0.0735$	$0.15+(0.05*0.15)=0.1575$	$0.12+(0.05*0.12)=0.126$	$0.17+(0.05*0.17)=0.1785$	$0.18+(0.05*0.18)=0.189$

These are the new weights obtained after mutation process. The process is then repeated for all the rows in the population or generation.

4.2.2.3 Pseudo Code

In this section we present the pseudo code developed by us as under:

1. Get the rows from database table
2. Count no. of rows
3. Get the columns from database table
4. Count no. of columns
5. Initialize chromosomes randomly
6. Consider 1000 generations
7. Write generations in a file
8. Calculate Evaluation Function for each chromosome

$$f(R) = \sum (f(R_i) * w_i)$$

$$F(R) = \begin{cases} \text{String Length}(R_i) & \text{if } i < 26 \\ 0 & \text{if } R_i = "" \\ \text{Ascii Encode}(R_i) & \text{if } i \Rightarrow 26, 27, 28 \end{cases}$$

Where R_i stands for the i th attribute value of row and w_i is the weight of the i th attribute.

9. Calculate Fitness Function for each chromosome as:

$$\text{Fit}(\text{chr}) = \{ \text{Absolute}[f(V_1) - f(V_2)] + \text{Absolute}[f(V_3) - f(V_4)] + \text{Absolute}[f(V_5) - f(V_6)] \} \div 6$$

Where $\text{Fit}(\text{chr})$ stands for fitness values of a chromosomes. We took 6 chromosomes as by experiments we observed that better results are obtained by considering 6 chromosomes. Experiments were performed by taking 50,30,20,10 and than 6 chromosomes.

10. Sort the Fitness values of Chromosomes. Minimum value is the best fitted value.

Mathematically:

S = Set of all fitness values of chromosomes

S' = Subset of S

$S' = \min_{i=1}^5 \{S\}$

11. Perform mutation operation, $S = \text{mutation}(S')$

i). updated weight = previous weight + Δw

$w = w' + \Delta w$; where η is mutation rate and,

$\Delta w = (\eta * w')$ where $\eta < 1$

12. If $g < 100$ than go to 5 else move to 13th step where g represents generation.

13. Calculate average fitness value for whole generation.

14. Reset chromosomes fitness values for next evolution.

15. Save tuned weights for the generation in a file.

4.2.3 Artificial Neural Network

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by biological nervous system. The key element of this paradigm is the novel structure of how the information is processed in a system. It is composed of a large number of highly interconnected processing neurons, to solve specific problems. ANNs, like people, learn by example. An ANN is widely used for applications, like pattern recognition or data classification. Learning in ANN systems involves adjustments to the synaptic connections that exist between the neurons. [46]

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation;

- **The Training Mode:** In the training mode, the neuron can be trained to fire (or not), for particular input patterns.[46]
- **The Using Mode:** In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not. [46]

Diagrammatically it can be shown as under in Figure 4.11:

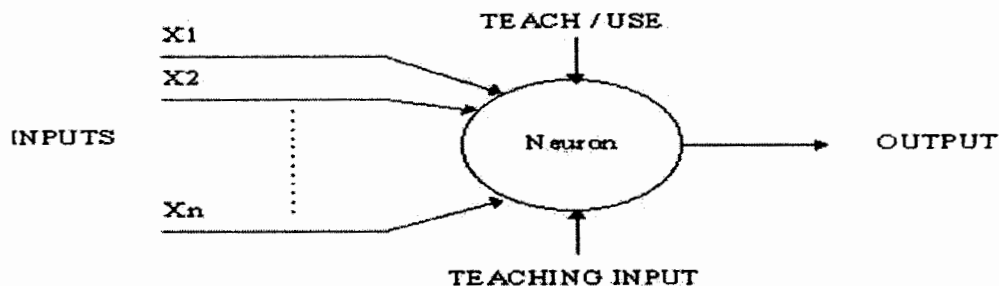


Figure 4.11: A Simple Neuron Processing [46]

4.2.3.1 Execution of Neural Network

Neural Network developed, works on two rows at a time. All the attribute values of row 1 (R_1) and row 2 (R_2) are passed through networks. First random weights are taken. Once data is processed then the out put of a complete network is calculated by taking the difference of outputs of both the networks N1 and N2. If the error is minimum the weights are back propagated and neural network learns it else the weights are discarded and the process is started again.

Diagrammatically execution of Neural Network Algorithm can be shown as in figure 4.12 below:

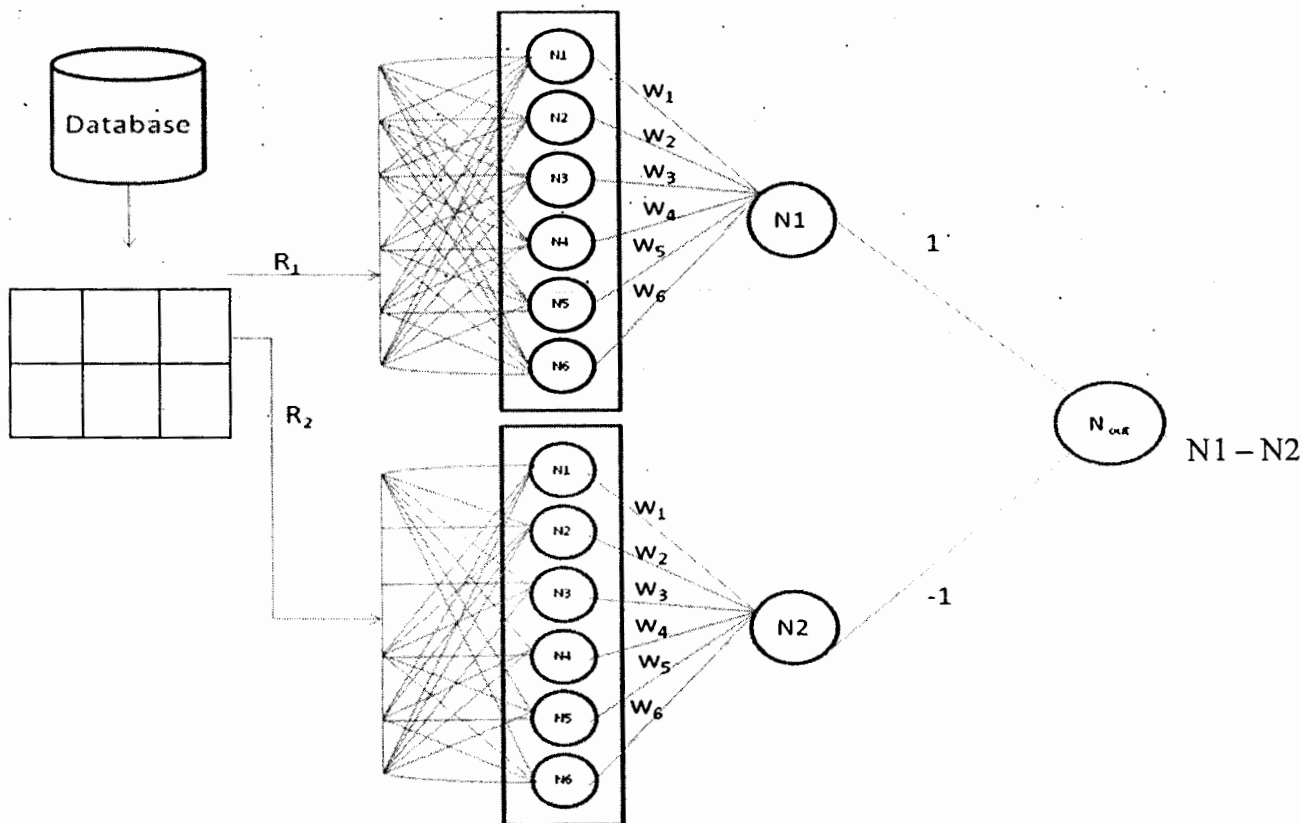


Figure 4.12: Execution of Neural Network

4.2.3.2 The Back-Propagation Algorithm

We used back-propagation algorithm to train our feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. The output response is then compared to the known and desired output and the error value is calculated. Based on the error, the connection weights are adjusted.

A rough picture of Back propagation algorithm can be given as under:

- Apply input to the network.
- Calculate the output.
- Compare the resulting output with the desired output for the given input. This is called the *error*.
- Modify the weights and threshold for all neurons using the *error*.
- Repeat the process until *error* reaches an acceptable value (e.g. *error* < 1%), which means that the NN was trained successfully, or if we reach a maximum count of iterations, which means that the NN training was not successful.

4.2.3.3 Pseudo Code

The algorithm for duplication detection developed by using Neural Network is shown as:

1. Get rows from Database
2. Count total no. of rows
3. Get columns from database
4. Count total no. of columns.
5. Initialize connection weights (from input layer to hidden layer) randomly.
6. Consider 100 Epochs
Where Epochs means population. Epoch is a term that is generally used while working with Neural Network.
7. Initialize the input.
8. Declare activation function for every neuron (here we consider 6 neurons because we carried out different experiments taking 8,7 and 6 neurons and results achieved by 6 neurons were better)
 - Activation function of *i*th Neuron:

$$V = 1 / (1 + e^{-\text{sum}})$$

$$\text{Where sum} = \sum_i (v_i * w[i])$$

and v_i is the *i*th input value of the attribute of a particular row and $w[i]$ is the weight associated with the output.

9. Calculate Network output for ith row.

$$\text{Output} = \sum_{i=1}^n (N_i * w'_i); \text{ where } N_i = v_i * w[i]$$

N stands for the network and i represents whether it's a N1 network or N2 network.

10. Calculate network error as:

$$e = f_j - f_i$$

$$e = \begin{cases} (f_j - f_i) = 0 & \text{if rows are similar} \\ (f_j - f_i) - 1 = 0 & \text{if rows are not similar} \end{cases}$$

Where e is error of the two networks. In figure 4.12 e is calculated at N_{out} .

11. Apply validation method using root mean square technique whose standard formula is given as:

$$\text{RMSE} = \sqrt{(1/n) \sum_{i=1}^n (T_i - A_i)}$$

where T_i is target value of ith row and A_i is the actual value of the ith row and n represents number of rows:

12. Use Back Propagation Algorithm to adjust weights
- Tune weights from hidden layer to output layer.

$$w_1 = w'_1 + \Delta w$$

$$\text{where } \Delta w = \text{Learn rate} * \text{error} * V_i$$

V_i is the value of ith neuron.

13. Take partial derivative of error due to hidden neurons with respect to weights of input layer to hidden layer.

$$\text{Partial derivative } (\partial) = V_i (1 - V_i) * \delta$$

Where δ represents the error of the hidden layer.

This equation is being used in back propagation algorithm presented by Tom M. Mitchell in his book Machine Learning.[49]

14. Tune weights of input layer to hidden layer

$$w = w' + \Delta w$$

where:

$$\Delta w = lr * \partial * X$$

lr is learning rate at which the network learns and our experiments shown that better results are obtained when lr is 0.08. X is the input vector

15. Connect weights with respected neurons.
16. Write root mean square (cross validation) value in a file.

ANALYSIS

5. Analysis

In this Chapter, we will discuss results obtained by different algorithms implemented and will do experimental study for analysis of the results obtained.

5.1 System Specifications

Experiments have been performed upon the system of following specifications:

- a) Intel Pentium M Processor 735A (1.7 GHz, 400 MHz FSB, 2MB L2 cache)
512MB (DDR2) RAM
- b) Intel(R) Pentium (R) M Processor 1.73GHz 794 MHz
504 MB RAM

Types of dictionaries (*as mentioned in section 4.1.1*) and their sizes considered for experiments are given in *Table 5.1*. To our knowledge, no standard list of South Asian names has been issued by any authority. To compile the dictionaries used in these experiments, however, reasonable amount of names and their abbreviations have been extracted out of Urdu dictionary and Telephone Directory of Pakistan, issued by National Language Authority and Pakistan Telecommunication Company Limited respectively.

Table 5.1: Sizes of dictionaries considered for experiments

Type	Size
Dictionary of Names	5521
Dictionary of Abbreviations	11
Dictionary of Postfixes-Prefixes	27

Analysis of the algorithms involved at different levels of the proposed model (*see Figure 4.1*) and results generated by them have been discussed in the following sections.

5.1.1 Tokenization Method

Time consumption with respect to the input data shows linear behavior (*see Figure 5.1*). Time increases as size of the input data increases, however, increase in the amount of time falls within few seconds. The results prove that the proposed algorithm is suitable even for those applications in which processing time is a vital concern. No major change in processing time is observed even upon increasing the data size two times. Difference of time between first node and second node is small. Likewise, change in time between second node and third node is minor. However, difference is comparatively large between first node and third node, which shows that variation in processing time increases in case size is increased three times. Experiments have shown that time consumption for a complete name depends upon number of tokens it is split into. Name consisting three tokens, for example, 'arif hameed makhdum', consumes thrice as much time as a

single token name 'imtiaaz'. It also depends upon the order in which dictionary is maintained. If dictionary is maintained alphabetically, tokens starting with 'z' will utilize more time than the ones starting with 'a', however, time consumption is minor. Most of all, sizes of the dictionaries affect time utilization to great extent. Number of comparisons increase with increase in the entries of the dictionaries. Lesser the size of dictionary, lower the time consumed.

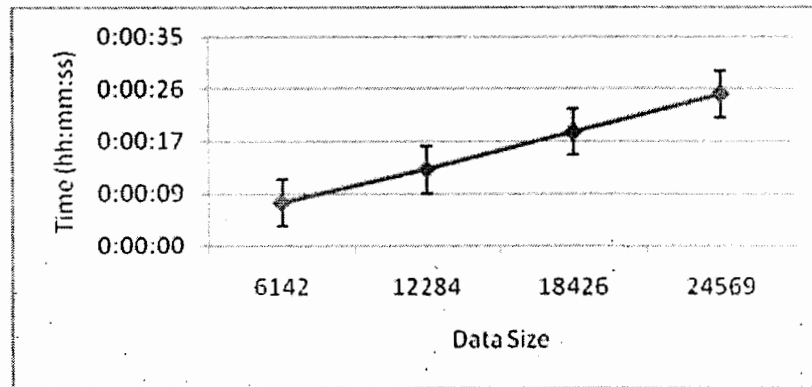


Figure 5.1: Time consumed by Algorithm for Tokenization

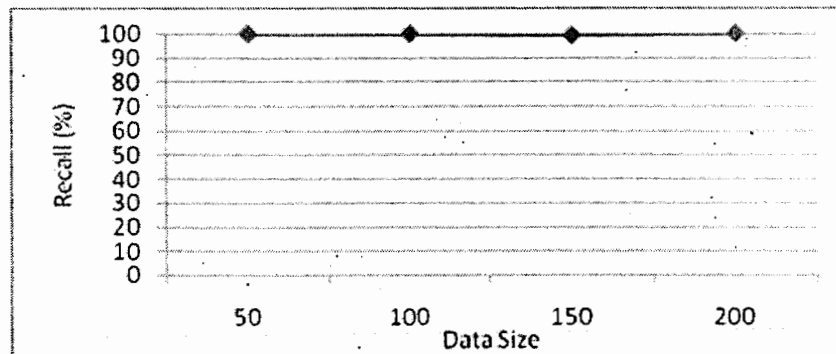


Figure 5.2: Performance in terms of Recall achieved by Algorithm for Tokenization

Performance of the proposed algorithm has been measured in terms of Recall R , which is computed as given by (5.i).

$$R = \frac{\text{Number of accurately tokenized Names}}{\text{Total number of Input Names}} * 100 \quad (5.i)$$

As shown in Figure 5.2, recall percentage touches the highest level of rating. Nevertheless, it doesn't remain at the same level and falls below by a fraction in certain cases. The lowest recall percentage value is 99.3. Tokens such as 'zainulabdeen' can be split into three tokens 'zain', 'ul', 'abdeen'. Such names cannot be tokenized using proposed solution due to the fact that names are parsed on the basis of spaces or special characters. Decrease in recall is minor, yet, the results

produced are near to perfect. The proposed algorithm performs well in terms of efficiency as well as accuracy.

5.1.2 Associative Matrix Memory (AMM) for Removing Spelling Mistakes

Although data stored in binary form consumes less time while comparison yet it increases the overhead (in terms of time and resources) of converting to and handling large binary patterns. The proposed methodology compares ASCII codes of the letters forming words, which eventually saves time and resources.

Combination of the algorithms produces better results [17]. The comparison of Associative Memory Matrix *AMM* and hybrid of BiGrams and Edit Distance shows that *AMM* consumes lesser time than BiGram and Edit distance together (see Figure 5.3). Time complexity of edit distance between strings of length m and n is given as $O(m*n)$ (47). Time complexity of bigram between the same strings is $O(m+n)$ (48). Hence, $O(m+1*n+1)$ is the total time complexity of both the algorithms. Whereas, time complexity of *AMM* is $O(\min(m, n))$, which is greater than the individual time complexity of edit distance and bigram. For dictionary of size d , time complexity for edit distance is $O((m*n)*d)$, bigram is $O((m+n)*d)$ and *AMM* is $O((\min(m, n))*d)$.

Increase in the time consumed by hybrid of edit distance and bigram with respect to data size is polynomial. Time starts from 26 seconds and reaches upto 1 minute and 26 seconds. In case of *AMM*, increase in time consumption is gradual as compared to the hybrid. Values remain between 7 and 21 seconds.

As a matter of fact, the proposed algorithm performs exhaustive search, that is, it compares an incorrect name with all names in the dictionary. Such approach has higher time complexity and consumption of time elevates with increase in data size. Nonetheless, it yields maximum and the best outputs possible. To balance time utilization, knowledgebase has been introduced. *AMM* incorporated with knowledgebase consumes minimal amount of time as shown in Figure 5.3. Therefore, time complexity is linear and given by $O(k)$ where k is size of the knowledgebase. Besides, increase in the time consumed by *AMM with knowledgebase* due to increase in the size of incorrect words is nominal. The time consumed by *AMM with knowledgebase* ranges from one second to three seconds for all sizes of the input data considered during the experiment.

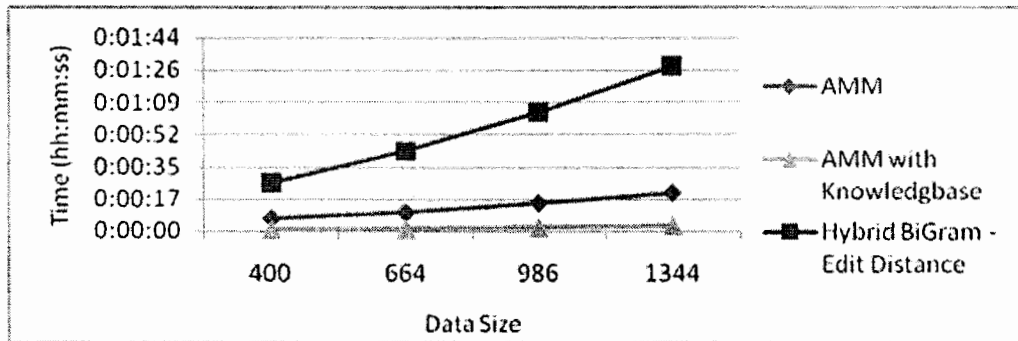


Figure 5.3: Time consumed by AMM for Removing Spelling Mistakes

Table 5.2: Recall for AMM & Hybrid – Qgram & Edit Distance

Data size	Hybrid Qgram - Edit Distance Correct Matches Found	AMM Correct Match Found	Hybrid Qgram & Edit Distance Recall (%)	AMM Recall (%)
400	278	374	69.5	93.5
664	450	610	67.771	91.867
986	675	903	68.458	91.582
1344	914	1237	68.005	92.038

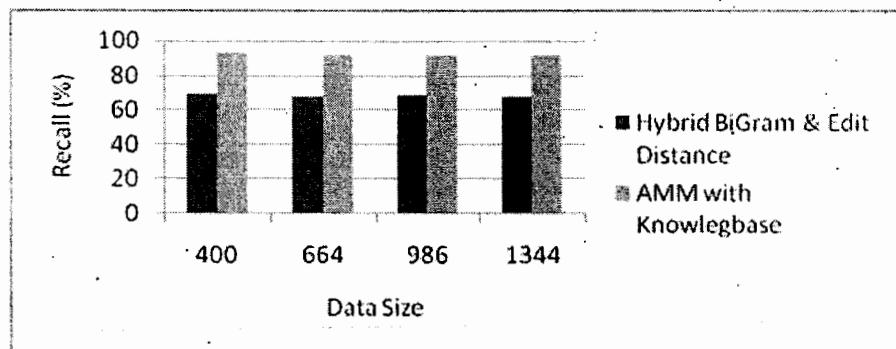


Figure 5.4: Comparison of the Performance in terms of Recall achieved by AMM & Hybrid- BiGram & Edit Distance

Performance of the aforementioned algorithms has been measured in terms of Recall R which is given by (5.ii).

$$R = \frac{\text{Number of Incorrect Names for which correct match(es) is/are found}}{\text{Total number of Incorrect Names}} * 100 \quad (5.ii)$$

AMM generates more accurate results than produced by edit distance and bigrams together (as shown in Table 5.2 and plotted in Figure 5.4). The highest recall percentage generated by AMM is 93.5. For all sizes of data, recall percentage of AMM does not fall below 91.5 showing that 91.5 percent of the incorrect names find their correct matches. As compared to that, hybrid of edit distance and bigrams produces 67.3 as highest recall percentage and reduces to lowest value

of 62.7. The difference between highest value of AMM recall percentage and the lowest recall percentage of hybrid is quite huge. Considering the largest data size, correct matches are found for 1237 names through AMM, whereas, hybrid of edit distance and bigrams identifies correct matches for only 914 names out of 1344 incorrect names. Matches for incorrect names such as 'atif' for 'afit', 'gohar' for 'ghorar' and 'aurangzeb' for 'orangzeb' are found by AMM only. Incorrect names mentioned above remain unidentified and hybrid of edit distance and bigrams does not provide any match for them.

Incorrect names for which no correct match is found by AMM in the dictionary, give rise to two interpretations. Either the dictionary is not complete or such incorrect names are noise and should be discarded. For example, 'aanboo', 'shshh' and 'dronrane' do not bear any possible match and are recognized as noise. Incorrect names for which the desired / expected matches are not retrieved, identifies that there is insufficient data in the dictionary. It emphasizes the need to enhance the dictionary. In other words, it pinpoints the names that should be added to increase the size of dictionary. For example, 'shujraa' points that the desired match 'shujaa' does not exist in the dictionary and should be added into it. Likewise, 'sameejo', 'haro' and 'asheesh' are the names that are not found in the dictionary.

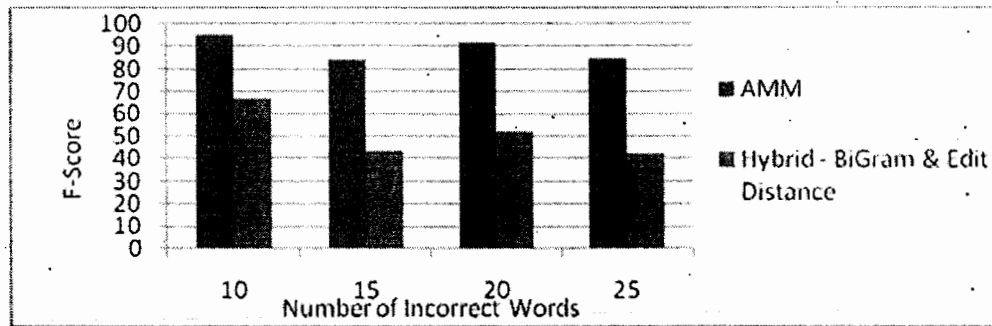


Figure 5.5: Comparison of the Performance in terms of F-Score achieved by AMM & Hybrid- BiGram & Edit Distance

Experiments have shown that accuracy of results significantly depends on size of the dictionaries. In addition, quality of results depends on their richness, completeness and correctness. Noise in the dictionary leads to decline in the ratio of accuracy. To analyze the accuracy of results produced by proposed algorithm, *F-Score* is computed, which is given by (5.iii), where R represents recall, P represents precision and n denotes total number of incorrect names.

$$R = \frac{\sum_{i=1}^n \text{Number of correct match}(es)}{\text{Total number of match}(es)}$$

$$P = \frac{\sum_{i=1}^n \text{Number of correct match}(es)}{\text{Total Number of possible match}(es) \text{ present in dictionary}}$$

$$F = \frac{2 \cdot P \cdot R}{P + R} * 100 \quad (5.iii)$$

To analyze behavior of the chart shown in *Figure 5.5*, data considered at all points of the chart is exclusive, that is, data at one point does not overlap the data at any other point on the graph. Matches generated by AMM are further shortlisted on the basis of their comparison value. This is done to keep the percentage of recall near the percentage of precision. If the set of matches s for an incorrect name w contains comparison value c equal to or greater than one, set of matches will be short listed to form subset s' of s such that only those matches are contained in it that have either comparison value equal to or greater than 0.77 or length equal to the length of w . Otherwise, all the matches produced by AMM are considered, that is, s' is equal to s . Initially, matches found against incorrect names 'aajaz', 'kaan' and 'zanib' are given in *Table 5.3*.

Table 5.3: Sets of Matchess found by AMM & Hybrid – Qgram & Edit Distance

Incorrect Name	Matches Found by AMM	Comparison Value	Matches Found by Hybrid Qgram & Edit Distance	Comparison Value
Aajaz	aijaaz	0.8		
	aijaz	0.8		
	ayjaz	0.8		
Kaan	jaan	0.75	kahan	0.76
	kahan	1		
	khan	0.75		
	kiani	0.75		
	kyani	0.75		
Zanib	zaib	1	zaib	0.76
	zaiba	0.8		
	zainab	0.8		
Oasto				
Shahim	hashim	1	shahid	0.773
	hashmi	1	shahin	0.773
	shahdia	0.8333333	shamim	0.773
	shahid	0.8333333		
	shahida	0.8333333		
	shahin	0.8333333		
	shahina	0.8333333		
	shamim	0.8333333		
	shah	0.8		

Table 5.4: Sets of Shortlisted Matches *s*' found by AMM & Hybrid – Qgram & Edit Distance

Incorrect Name	Matches Found by AMM	Comparison Value	Matches Found by Hybrid	Comparison Value
Aajaz	aijaz	0.8		
	aijaaz	0.8		
	ayjaz	0.8		
Recall	3/3		0/3	
Precision	3/3		0/3	
FScore	1		0	
Kaan	kahan	1	kahan	0.76
	khan	0.75		
	jaan	0.75		
Recall	3/3		1/3	
Precision	3/3		1/3	
FScore	1		0.5	
Zanib	zaib	1	zaib	0.76
	zaiba	0.8		
	zainab	0.8		
Recall	3/3		1/3	
Precision	3/3		1/3	
FScore	1		0.5	
Oasto				
Recall	0/0		0/0	
Precision	0/0		0/0	
FScore	0		0	
shahim	hashim	1	shahid	0.773
	hashmi	1	shahin	0.773
	shahdia	0.8333333	Shamim	0.773
	shahid	0.8333333		
	shahida	0.8333333		
	shahin	0.8333333		
	shahina	0.8333333		
	shamim	0.8333333		
	shah	0.8		
Recall	6/9		3/3	
Precision	6/6		3/6	
FScore	0.8		0.66	

Table 5.4 shows the shortlisted matches found against each incorrect name. Matches are sorted on the basis of comparison values in descending order. Match with highest comparison value is

placed at the top position. Matches with same comparison values are sorted alphabetically on the basis of starting letter of the incorrect name. For first three incorrect names, all matches identified by AMM are valid. Matches found by AMM for first incorrect name are spelling variation of the same name. Incorrect names at third and fourth position in the table can be replaced by any of the three matches identified by AMM. Therefore, recall and precision rate attain their highest value (that is one). Fourth incorrect word is noise, hence, none of the algorithms generate any matching name for it. However, the last incorrect name yields number of possible matches some of which are irrelevant. In such case, precision rate remains unaffected. Yet, there is impact on the recall rate that influences *FScore* eventually. It is due to such cases that *FScore* decreases with decrease in the recall rate as seen in the second and fourth point in the graph shown in *Figure 5.4*. Behavior of both the algorithms is periodic. However, *F-Score* percentage for hybrid – edit distance & bigrams rises and falls between 42.7 and 67.2, and that of AMM varies between 85.3 and 95.4. *FScore* for AMM is quite higher than the one generated by hybrid of edit distance and bigrams. Yet again, it is proved that AMM generates better results as compared to the combined results of edit distance and bigrams.

5.1.3 Duplicate Detection Technique based on Associative Matrix Memory

Number of comparisons N_c for the given data set S , comprising n number of rows and m number of columns is given by (5.iv).

$$\begin{aligned} N_c &= ((n-1) * m) + ((n-2) * m) + ((n-2) * m) + \dots + 1.m \\ &= n(n-1)/2 * m \end{aligned} \tag{5.iv}$$

As mentioned in *section 4.2.1*, number of comparisons reduces after each iteration. Number of comparisons is $(n-1) * m$, initially, which reduces to m number of comparisons till the last iteration (matching of second last record with last record). The proposed algorithm is not greedy indeed, yet, it generates results same as that produced by exhaustive approach. This saves time and resources to an extent besides generating optimum results.

Efficiency and accuracy are the parameters to judge performance of an algorithm. A tradeoff between the two produces better results, nevertheless, it depends on requirement of the application using it. *Figure 5.6* shows that time utilized by proposed algorithm increases as the data size increases. Increase in time is quadratic due to large number of comparisons involved. However, aim of the proposed algorithm was to achieve accuracy in the results rather than efficiency.

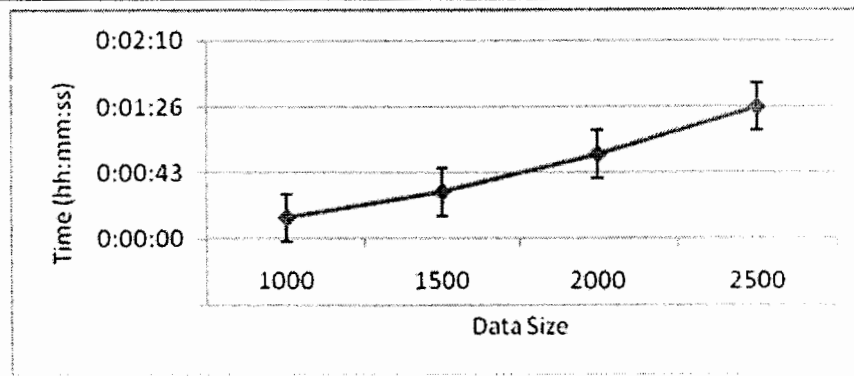


Figure 5.6: Time consumed by AMM for Duplicate Detection

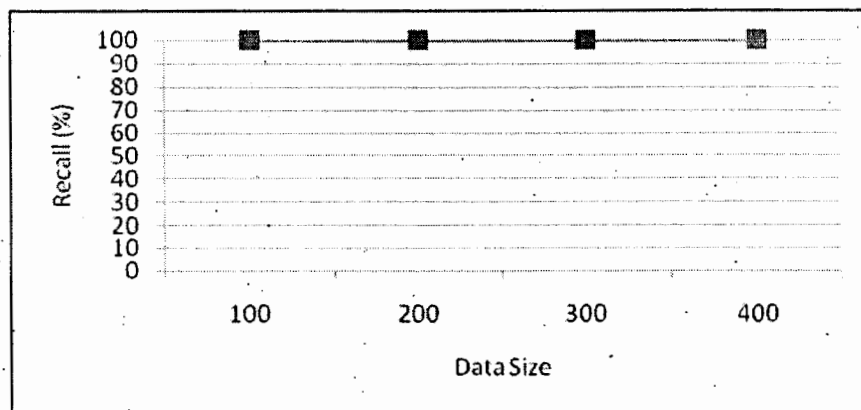


Figure 5.7: Performance in terms of F-Score achieved by AMM for Duplicate Detection

To analyse accuracy of the results produced by the same algorithm, recall percentage R is calculated as given by (5.v):

$$R = \frac{\text{Number of correctly identified Duplicate Records}}{\text{Total number of Duplicate Records}} * 100 \quad (5.v)$$

As shown in Figure 5.7, the results achieve highest level of accuracy and remain uniform for all sizes of data. The proposed model does not bring any search mechanism in use to short list rows for comparison for particular record. Although this approach increases consumption of time, however, it produces best possible results. All the attributes of two records are compared. Not only this, weights are assigned to the attributes on the basis of their significance in the record set. In other words, weights are proportional to the amount of information an attribute contributes to the whole data. It is due to this that matches are found even if an attribute with lower weight is null or contains null value. Also, desired results are generated even if values of the attributes with low weights are not same provided that values in the attributes with higher weights attached to them are equal.

Experiments have been performed using customer information data maintained by a cellular company. Duplicate detection is beneficial in number of ways depending on the scenario it is used in. For example, it relieves the system, may it be online transaction processing or data warehouse, from unnecessary data and helps extract useful information and provides grounds for valuable data analysis. Particularly, it helps in fraud detection.

Table 5.5: Duplicate Records in Customer information Data Set

Group	ID	Prefix	Name1	Name2	Name3	House	Street	Sector	NIC No	Contact No
1	1	doctor	saleem	akhtar		351	166	G-11/1	6110123059171	
	2		s	akhtar		351	166	G-11/1	6110123059171	2873992
2	3		abdul	nabi		156	69		1310179017743	
	4		abdul	nabi		1912	101		1310179017743	
3	5		irfan	rasheed		553	1	I-9	6110118380847	
	6		irfan	rashid	butt	553	1	I-9/1	6110118380847	
4	7		waqar	ahmad		5	38		6110191485705	4444995
	8		waqar	ahmed	siddiqui	5			6110191485705	4444995
	9		waqar	ahmad						4444995
5	10		ghulam	khan					6110151795617	2215545
	11		zain	ul	abdeen				6110151795617	2215545

Table 5.5 contains duplicate records identified by proposed algorithm. The first group is identified on the basis of same values of NIC number, sector, street, house and name2. In second group, values in the fields of name and NIC number are equal, hence, it is reckoned as a match. Third group contains duplicate records having same values in all attributes except name3 and sector. Consider first two records of group four, part of the name and part of address are the same, however, NIC number and contact number are exactly the same. Third record of forth group is also reckoned as a match even though NIC number, which contributes most significant information to the data, is not equal in the two records. Most interesting matches are enclosed in group five. Out of nine attributes, only two attributes, NIC number and contact number, contain equal values between the two records. Such identifications are effective in fraud detection.

5.1.4 Duplicate Detection Technique Based on Evolutionary Algorithm

In this section we discuss the results produced by Evolutionary Algorithm. These results were taken considering 1000 generations. Algorithm is trained first on a dataset and then used for further experiments. Following Table 5.6 shows the results for 1000 generations:

Table 5.6: Results with 1000 Generations

Generation	Chromosomes	η	Rank Selection	Average Time Taken(Millisecond)	Precision	F Score	Recall
1000	10	0.05	4	94.2	0.01695	0.033333	1
1000	15	0.05	4	95	0.01695	0.033333	1
1000	20	0.05	4	95.6	0.01695	0.033333	1
1000	25	0.05	4	96.8	0.01695	0.033333	1
1000	30	0.05	4	97.2	0.01695	0.033333	1
1000	30	0.05	4	97.6	0.01695	0.033333	1

Empirical evaluation plays a central role in estimating the performance of natural language processing (NLP) or information retrieval (IR) systems. Performance is typically estimated on the basis of synthetic one-dimensional indicators such as the precision, recall or F-score. We used the same mechanism to analyze the performance of evolutionary algorithm. Examination of Table 1.1 shows that the average of precision is 0.01695, and the average of recall is 1. Though precision is not very good but experiments showed that recall is 100%.

Chromosomes vs. Precision

The graph between chromosomes and Precision is shown in Figure 5.8 below for 1000 generations:

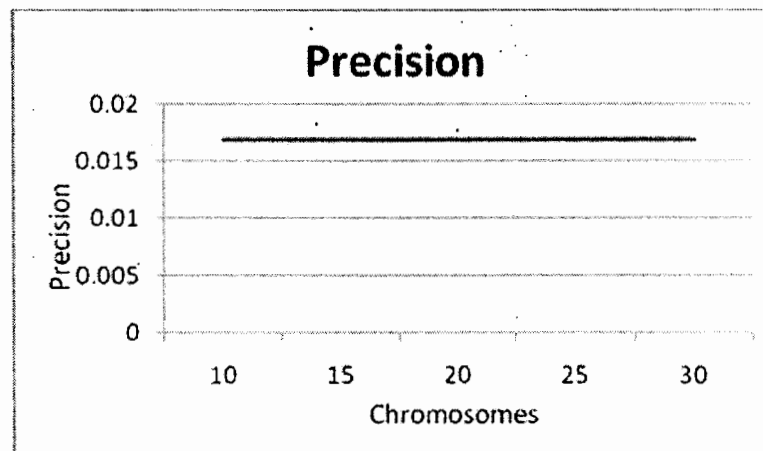


Figure 5.8: Chromosomes vs. Precision

For a given dataset, the precision does not change. It remains the same. Similarly as we increased the chromosomes size it does not have any effect on precision.

Chromosomes vs. F Score

The graph between Chromosomes and F Score for 1000 generations is shown in Figure 5.9:

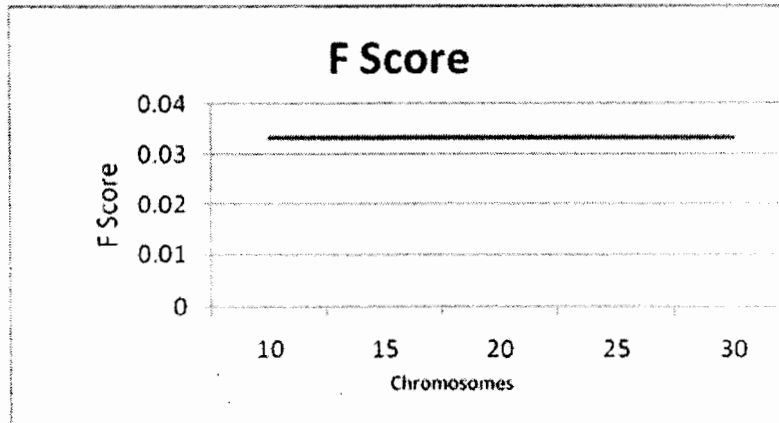


Figure 5.9: Chromosomes vs F Score for 1000 Generations

Looking at the results above, it can be seen that constant behavior is shown by algorithm when number of chromosomes is changed. This means that number of chromosomes does not affect the computation of algorithm.

Chromosomes vs. Time

The graph between Chromosomes and Time for 1000 generations is shown in Figure 5.10:

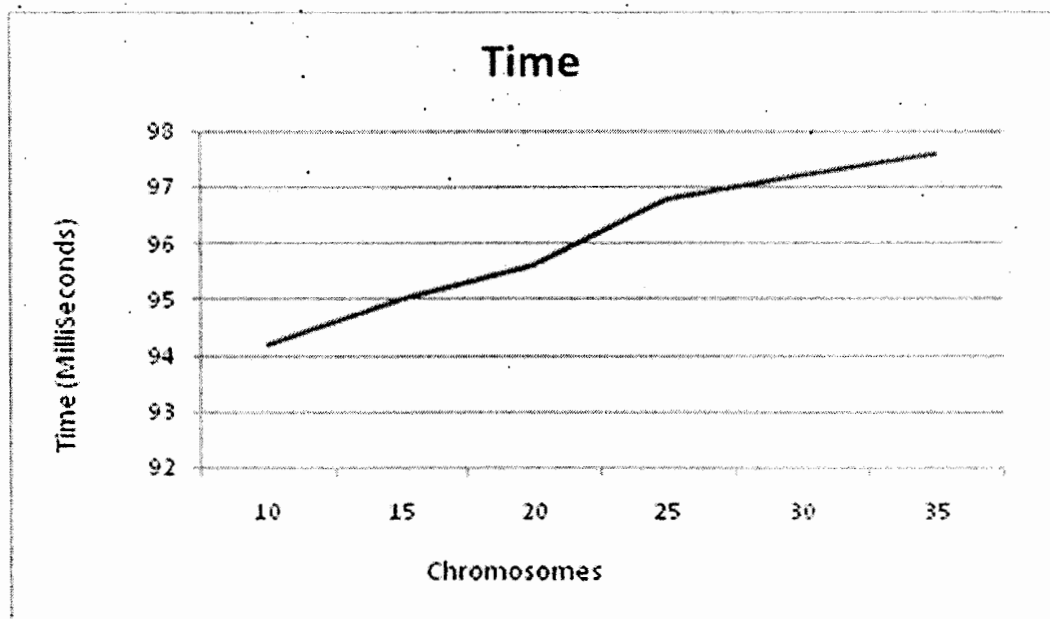


Figure 5.10: Chromosomes vs. Time for 1000 Generations

Above mentioned results depict that chromosomes are directly proportional to computation time. If chromosomes are more in number, time taken for processing is increased.

In order to get the curve is somewhat smooth taken average time for computation; we have indicated that the average execution time for the given dataset increases as the chromosomes are increased.

Based on the values obtained, relation between eta and best fit cannot be determined in a clear manner as eta and best fit are neither directly proportional nor inversely proportional to each other. These results have a different behavior with different values of eta.

5.1.5 Neural Network Algorithm for Duplicate Detection

In this section we discuss the results obtained by ANN. To assess the performance of the network, Root Mean Square Error (RMSE) is chosen as an indicator for analysis of network. Following are the results obtained with different Epochs as shown in Table 5.7.

Table 5.7: Results of Neural Network

Epochs	Topology	Time Taken	RMS error	Learning Rate
100	20-6-1	1 sec	5.66E-10	0.08
200	20-6-1	1 sec	1.00E-10	0.08
500	20-6-1	2 sec	1.01E-11	0.08
500	20-6-1	2 sec	1.01E-11	0.05
500	20-6-1	3 sec	1.01E-11	0.8
1000	20-6-1	4 sec	1.79E-12	0.08
1000	20-6-1	4 sec	1.79E-12	0.05
1000	20-6-1	5 sec	1.79E-12	0.8

In comparison with the Evolutionary Results, neural Network performed better and we can see from following Figure 5.11 that the RMSE is reduced as we increase the Epochs

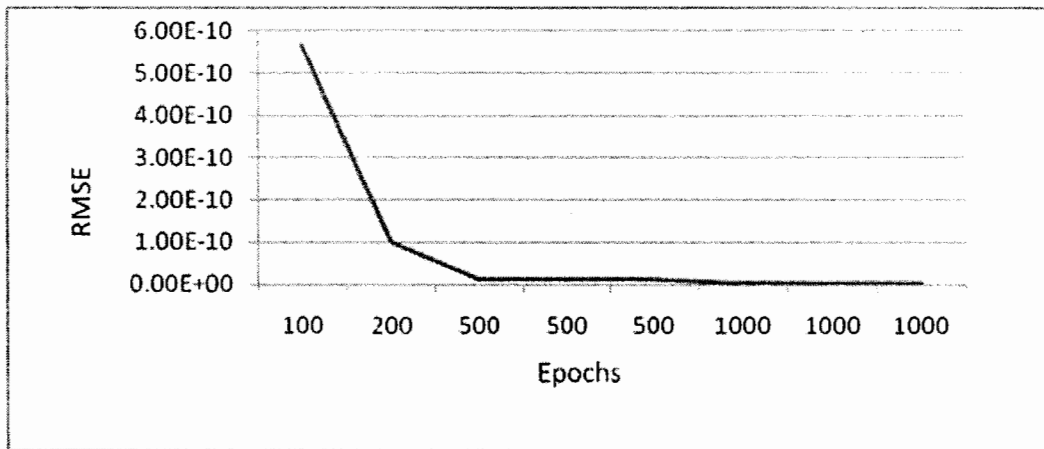


Figure 5.11: Epochs vs. RMSE

Hence we can say that greater the Epochs, less will be the RMSE.

Similarly as we proceed, we also observe from Table 5.7 that as the number of Epochs increases, the computation time also increases but the RMSE value reduces.

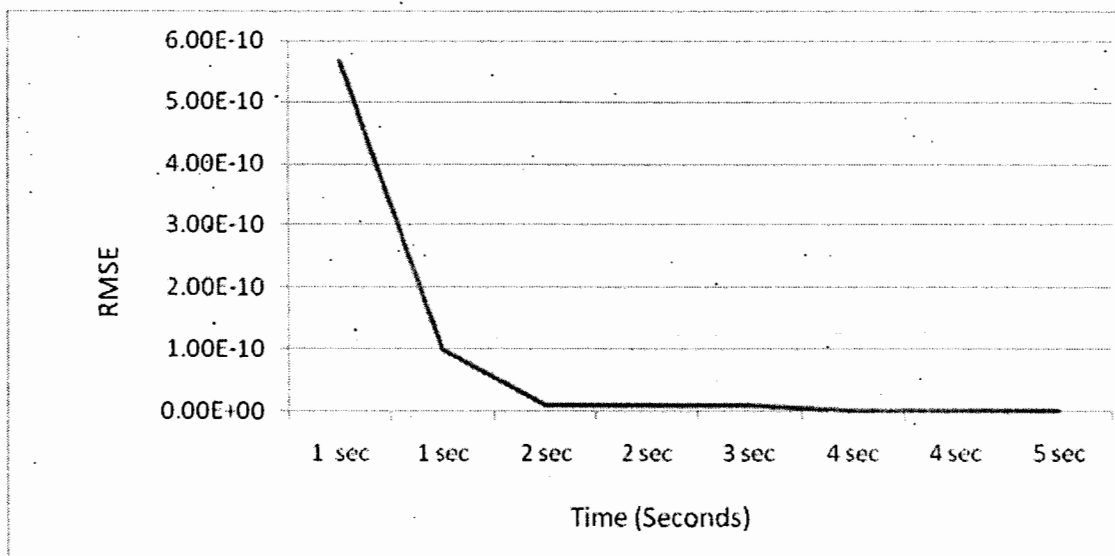


Figure 5.12: Time vs. RMSE

From the results above in Table 5.7, the average RMSE is 8.78E-11.

CONCLUSION

6. Conclusion

Data cleansing is practiced in number of applications. Most of the leading organizations of the world are in the need of quality data; especially in Telecom Sector, it is very important to have cleaned data for processing. However, it is not as simple as it seems to get perfectly cleaned data for further processing and decision making. In data cleansing techniques, some of the cleansing methods are implemented. But the existing techniques are good only in some part of cleansing process.

We have proposed a new framework in which we have used three different techniques for Data Cleansing Associative Memory Matrix (AMM), Evolutionary Learning and Neural Networks. However this framework can be enhanced further with more complex techniques that are emerging in the world of Information Technology. Phonetics can be embedded into Associative memory matrix to generate higher value of FScore.

Duplicate Detection task performed using Evolutionary Learning can be improved by using Adaptive Mutation Technique, for example; Gaussian Mutation or Cauchy Mutation. Furthermore, Evaluation optimization can also be achieved by adding more features of the data set.

Duplicate Detection task performed by using Neural Networks can be implemented using different architectures of Neural Networks for example; Hopfield [1] and rt Neat. Similarly advanced gradient based learning methods can be used for achieving better results.

REFERENCES

References

1. http://en.wikipedia.org/wiki/Artificial_neural_network
2. <http://www.machinelearning.net/unsupervised-learning/>
3. <http://www.machinelearning.net/supervised-learning/>
4. <http://www.acm.org/crossroads/xrds8-2/evolution.html>
5. <http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,57545,00.html>
6. http://en.wikipedia.org/wiki/Feedforward_neural_network
7. http://en.wikipedia.org/wiki/Radial_basis_function_network
8. <http://www.comp.nus.edu.sg/~pris/AssociativeMemory/DetailedDescription.html>
9. <http://www.comp.nus.edu.sg/~pris/AssociativeMemory/AssociativeMemoryContent.html>
10. "A Theory for Record Linkage"; Fellegi, I. P., and A.B. Sunter, Journal of the American Statistical Association, 64, (1969), 1183-1210, 1969.
11. "Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida"; Jaro, M.A., Journal of the American Statistical Society, 84(406):414-420, 1989.
12. "The Merge/Purge Problem for Large Databases"; Hernandez, M.A., and S.J. Stolfo. In Proc. of 1995 ACT SIGMOD Conf., pages 127-138, 1995.
13. "Methods for Linking and Mining Massive Heterogeneous Databases"; Pinheiro, J.C., and Don X. In Fourth Int. Conf. on Knowledge Discovery and Data Mining, 1998.
14. "Efficient Clustering of High Dimensional Data Sets with Application to Reference Matching"; McCallum, A., K. Nigam, C. Mellon. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000.

15. "Preparation of name and address data for record linkage using hidden Markov models"; Churches, T., P. Christen, K. Lim, and J. X. Zhu. *BMC Medical Informatics and Decision Making*, 2, 2002.
16. "Data Cleansing: Beyond Integrity Analysis"; Jonathan I, Maletic and A. Marcus. In *Proceedings of the Conference on Information Quality (IQ2000)*.
17. "Record Linkage: Current Practice and Future Directions"; Gu, L., R. Baxter, D. Vickers, and C. Rainsford. *CMIS Technical Report No. 03/83*, 2004.
18. "Advanced Methods for Record Linkage"; Winkler, W.E., In *Proceedings of the Section of Survey Research Methods, American Statistical Association*, pages 467–472, 1994.
19. "Matching and Record Linkage"; Winkler, W.E., In Cox et al, editor, *Business Survey Methods*. J. Wiley & Sons Inc., 1995.
20. "Data Cleansing Methods"; Winkler, W.E., *Proceedings of the ACM Workshop on Data Cleansing, Record Linkage and Object Identification*, Washington, DC, August 2003.
21. "Integration of heterogeneous databases without common domains using queries based on textual similarity"; Cohen, W.W. In *Proceedings of AC SIGMOD-98*, pages 201–212, 1998.
22. "Reasoning about textual similarity in information access"; Cohen, W.W. *Autonomous Agents and Multi-Agent Systems*, pages 65–86, 1999.
23. "Data integration using similarity joins and a word-based information representation language"; Cohen, W.W. *ACM Transactions on Information Systems*, 18(3):288–321, July 2000.
24. "Learning to Match and Cluster Entity Names"; Cohen, W.W., and J. Richman. In *Proc. of the ACM SIGIR-2001 Workshop on Mathematical/Formal Methods in Information Retrieval*, 2001.
25. "Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration"; Cohen, W.W. and J. Richman. In *SIGKDD'02*, 2002.

26. "Approximate String Joins in a Database (Almost) for Free"; Gravano, L., P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srinivasta, In Proc. 27th Int. Conf. on Very Large Data Bases, pages 491–500, 2001.
27. "Efficient Record Linkage in Large Data Sets"; Jin, C. Li, and S. Mehrotra. In Int. Conf. on Database Systems for Advanced Applications (DASFAA), Tokyo, Japan, March 2003.
28. "An Extensible Data Cleansing Tool"; Galhardas, H., D. Florescu, D. Shasha, and E. Simon. AJAX: In SIGMOD (demonstration paper), 2000.
29. "Declarative Data Cleansing: Language, Model and Algorithms"; Galhardas, H., D. Florescu, D. Shasha, E. Simon, and C. Saita. In Proc. of the 27th VLDB Conf., 2001.
30. "An Extensible Framework for Data Cleansing (extended version)"; Galhardas, H., D. Florescu, D. Shasha, and E. Simon. INRIA Technical Report, 1999.
31. "A Distance-Based Approach to Entity Reconciliation in Heterogeneous Databases"; Dey, D., S. Sarkar, and P. De. IEEE Transactions on Knowledge and Data Engineering, 14(3), 2002.
32. "Learning to Combine Trained Distance Metrics for Duplicates Detection in Databases"; Bilenko, M., and R.J. Mooney. Technical Report AI-02-296, University of Texas at Austin, Feb 2002.
33. "A Bayesian decision model for cost optimal record matching"; Verykios, V., G.V. Moustakides, and M.G. Elfeky. The VLDB Journal, 2002.
34. "The Field Matching Problem: Algorithms and Applications"; Monge, A.E. and C.P. Elkan. 2004.
35. "Artificial Intelligence"; J.F.Lugar. Fourth Edition, 2001.
36. "Machine Learning, Information Retrieval, and Record Linkage"; Winkler, W.E., In Proc. of the Section on Survey Research Methods, American Statistical Association, 2000.
37. "A Re-Examination Of Text Categorization Methods"; Yang, Y. and X. Liu, Association of Computing Machinery, Proceeding of the SIGIR, 42-49, 1999.

38. "TAILOR: A Record Linkage Toolbox"; Elfeky, M.G., V.S. Verykios, and A.K. Elmagarmid. In Proc. of the 18th Int. Conf. on Data Engineering. IEEE, 2002.
39. "Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration"; Cohen, W.W. and J. Richman. In SIGKDD'02, 2002.
40. "String Edit Analysis for Merging Databases"; Zhu, J.J. and L.H. Ungar. In KDD 2002 Workshop on Text Mining, 2002.
41. "A Comparison of Standard Spell Checking Algorithms and a Novel Binary Neural Approach"; Hodge, V.J. and J. Austin, IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 5, September/October 2003.
42. "A Self-Learning Evolutionary Chess Program"; Fogel DB, Hays TJ, Hahn SL, and Quon J (2004) Proceedings of the IEEE, Vol. 92:12, pp. 1947-1954.
43. "Efficient Implementation of Correlation Matrix Memories on SNO based ccNUMA Parallel Computers"; Sam Clegg, submitted for the Degree of Bachelor of Science at the University of York. May 1999
44. "The State of Record Linkage and Current Research Problems"; Winkler W.E., Technical Report RR/1999/04, 1991.
45. "Associative Memories and the Application of Neural Networks to Vision"; Jim Austin, Handbook of Neural Computation, Ed. R Beale, published by Oxford University Press, 1-1-1995.
46. http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#What%20is%20a%20Neural%20Network
47. "An efficient uniform-cost normalized edit distance algorithm"; Arslan, A.N., Egecioglu,. String Processing and Information Retrieval Symposium and International Workshop on Groupware, 1999.
48. www.hig.no/index.php/content/download/21441/211654/file/Bakke%20%20presentation.pdf
49. "Machine Learning"; Tom M. Mitchell, Carnegie Mellon University.

APPENDIX A

A. User Manual - Data Cleansing

A.1 Import Data Screen

File containing erroneous data is selected from directory and shown on the screen. In, addition the selected data is saved into the respective table of database.

Import Data

Import Data Source A

Name	Address	IDCard	Phone
CAPT RETD CH REHMAT C...	H NO 4/3 C STREET 2 CILV...	6110119690837	923044566006
CAPT RETD NASER AHME...	H NO 4/3 C STREET 2 CILV...	6110119690837	923044566006
YEAHHH RETD ZONGGH ...	H NO 4/3 C STREET 2 CILV...	6110119690837	923044566006
IQBAL JAVED	H NO 4/3 C STREET 2 CILV...	6110119690837	923044566006
CAPT. AHMED FAROOQI (R...	H 27 STREET 44 SECTOR F...	6110119690837	923044566006
OMER ZAHOOR	MAKAN NO 11 MUHALA AZI...	6110119690837	923044566006
OMER ZAHOOR	MAKAN NO 11 MUHALA AZI...	6110119690837	923044566006
QUAID E AZAM MOHD. ALI ...	MAKAN NO 11 MUHALA AZI...	6110119690837	923044566006
R ABABAR	MAKAN NO 11 MUHALA AZI...	6110119690837	923044566006
OMER ZAHOOR	HOUSE 56 C ST5 CIVIL CO...	6110119690837	923044566006
AIN ABA	H NO 4/3 C STREET 2 CILV...	6110119690837	923044566006
NAEEM AHMAD KAYANI ...	HOUSE NO.1148/4 NASEE...	6110120550245	923045121211
SIRTAJ NABI KHAN AFRIDI ...	MOH AL HIDE TOWN SATR...	1730136321681	923045121227
MOHAMMAD SABIR	HOUSE NO-423 STREET-N...	6110122370281	923045121295
FAZULLAH KHAN	H NO. 25 B ST 6 SECTOR ...		923045121326

A.2 Standardize Data Screen

A.2.a Data before Standardization

The imported data is presented for standardization.

Standardization

Original Data

Name	Address	ID Card	Phone
CAPT RETD CH REHMAT CHAIR...	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
CAPT RETD NASER AHMED	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
YEAHHH RETD ZONGGH	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
IQBAL JAVED	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
CAPT. AHMED FAROOQI (RT)	H 27 STREET 44 SECTOR F 8/4 ...	6110119690837	923044566006
OMER ZAHOR	MAKAN NO 11 MUHALA A787 B...	6110119690837	923044566006

Standardise Data

Data After Tokenization

Name	Address
------	---------

View Possible Matches

A.2.b Standardized Data

Standardized data is displayed on the screen after running methods & algorithms for cleansing on the original / imported data.

Standardization

Original Data

Name	Address	ID Card	Phone
CAPT RETD CH REHMAT CHAIR...	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
CAPT RETD NASER AHMED ...	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
YEAHHH RETD ZONGGH ...	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
IQBAL JAVED ...	H NO 4/3 C STREET 2 CIVIL C...	6110119690837	923044566006
CAPT. AHMED FAROOQI (RT) ...	H 27 STREET 44 SECTOR F 8/4 ...	6110119690837	923044566006
OMER ZAHOR	MAKAN NO 11 MUHALA 272 DI	6110119690837	923044566006

Data After Tokenization

Prefix	Name1	Name2	Name3	Name4	Name5	Name6	Pc
captain retired	chaudhry	rehmat					ch
captain retired	naser	ahmed					
	yeahhh	retd	zonggh				
	iqbal	javed					
captain	ahmed	farooqi					reti
	omer	zahoor					
	omer	zahoor					
	quaid	e	azam	muhammed	ali	jinnah	
retired	ababar						
	omer	zahoor					
	ain	aba					

View Possible Matches

A.3 Possible Matches Screen

A.3.a Q-gram & Levenshtein Distance

For comparison, generated by Q-gram & Levenshtein Distance after removing spelling errors are displayed on the screen.

Possible Matches

QgramsEDThreshold CMM Details

Time Consumed by program as calculated by stopwatch: 0 : 0 : 2 : 908

Matching_Words	ED	Qgrams	Threshold
a			
abu			
edal			
ebdal	1	0.727272749	0.763636351
edaf	1	0.727272749	0.763636351
badal	1	0.727272749	0.763636351
amanallah			
amanullah	1	0.8	0.844444454
asjed			
butal			
fiazullah			
iqbal			
ibal	1	0.727272749	0.763636351

Save Identify Duplicate Records

A.3.b Associative Matrix Memory

Results generated by Associative Matrix Memory after removing spelling errors are shown on the screen.

Possible Matches
- □ ×

QgramsEDThreshold

CMM Details

Starting Time as provided by Timer
6/24/2008 1:57:44 AM 625

Associative Memory Matrix

For

Finding Possible Matches

Time Consumed by program as calculated by
Timer : 6/24/2008 1:57:46 AM 890

Time Consumed by program as calculated by
stopwatch: 0 : 0 : 2 : 258

No of comparisons for Novel method of
comparison: 12273 6

Without Recall		Recall Only		Semi Recall	
Matching_Wo	Comparison_V	Names	Comparison Value	Names	Comparison Value
abu		a		siraj	
butal		ca	2	sartaj	0.8333333
jhanzeb		fa	2	siraj	2
jhanzeb	2	ma	2	* [REDACTED]	
jhanzeb	2	sa	2	[REDACTED]	
jhan	0.8	adal		[REDACTED]	
jhanzeb	0.8571429	abdal	2	[REDACTED]	
nabi		adala	0.8	[REDACTED]	
abid	2	adalet	0.8	[REDACTED]	
nabil	1	adal	1	[REDACTED]	
nabila	0.8	badal	2	[REDACTED]	
naim	2	amanallah		[REDACTED]	
nain	2	amanallah	0.8888889	[REDACTED]	
* [REDACTED]		asadallah	0.7777778	[REDACTED]	
[REDACTED]		asjed		[REDACTED]	
[REDACTED]		amjed	0.8	[REDACTED]	
[REDACTED]		arshed	2	[REDACTED]	
[REDACTED]		asjad	0.8	[REDACTED]	
[REDACTED]		asud	0.8	[REDACTED]	
[REDACTED]		assed	0.8	[REDACTED]	

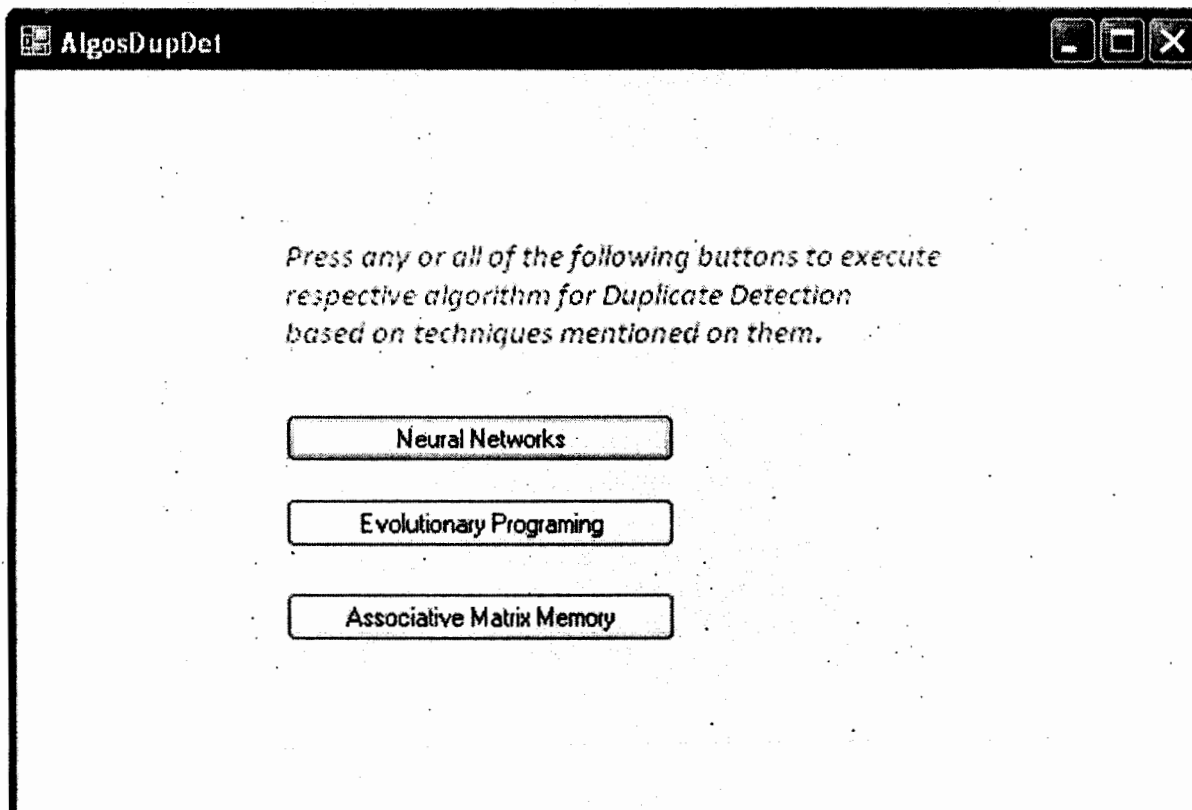
Save

Identify Duplicate Records

A.4 Duplicate Detection

A.4.a Algorithm(s) for Duplicate Detection

The screen shows buttons to execute proposed algorithms for identifying duplicate records in imported data.



A.4.b Duplicate Records

Duplicate records are presented on screen after undergoing procedures defined in proposed model.

ST_ID	Prefix Name	First Name	Middle Name 1	Middle Name 2	Middle Name 3	Middle Name 4	Postfix	Title
1		MOHAMMAD	SABIR					
7664		MOHAMMAD	SABIR					
2		FAIZULLAH	KHAN					
5		FAIZULLAH	KHAN					
7665		FAIZULLAH	KHAN					
3		AMANALLAH	KHAN					
4		AMANALLAH	KHAN	MALIK				
7666		AMANALLAH	KHAN	MALIK				
4		AMANALLAH	KHAN	MALIK				
3		AMANALLAH	KHAN					
7666		AMANALLAH	KHAN	MALIK				
5		FAIZULLAH	KHAN					
2		FAIZULLAH	KHAN					
7665		FAIZULLAH	KHAN					
7608		NAEEM	R	JAVED				
7682		IQBAL	JAVED				RETIRED	
7685		IQBAL	JAVED					
7710		IQBAL	JAVED					
7713		IQBAL	JAVED					
7662		NAEEM	AHMAD	KAYANI				
7663		SIRTAJ	NABI	KHAN	AFRIDI			
7664		MOHAMMAD	SABIR					
1		MOHAMMAD	SABIR					
7665		FAIZULLAH	KHAN					
2		FAIZULLAH	KHAN					