RELATIVE-FUZZY

A Novel Approach for Handling Complex Ambiguity for Software Engineering of Data Mining Models

PhD Thesis

AYAD TAREQ IMAM

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Software Technology Research Laboratory (STRL)

Faculty of Technology

De Montfort University

Leicester - United Kingdom

June 2010

Declaration

I declare that the work in this thesis is original work undertaken by me for the degree of Doctor of Philosophy, at the Faculty of Computer Technology at De Montfort University, Leicester, U.K.

No part of the material described in this thesis has been submitted for the award of any other degree or qualification in this or any other university or college of advanced education.

This thesis is written by me and produced using word 2007

This thesis was self-funded.

Ayad Tareq Imam

Publications

1. Al-Zobaydi Ayad T., Al-Akaidi Marwan M. *MS/Ayad-Marwan Network: a new architecture of an Artificial Neural Network.* International Conference on Interactive Mobile and Computer Aided Learning (IMCL2006) conference, Amman, Jordan, April 2006.
2. Al-Zobaydi, Ayad T., Al-Akaidi Marwan M. and John R. I. *Data Mining for Generating Predictive Models of Automatic Speech Recognition*. MESM05 MESM'2005 International Middle Eastern Multiconference on Simulation and Modelling, Porto University October 2005, pp. 147-150
3. Al-Zobaydi Ayad T. and Al-Akaidi Marwan M. *Speech Recognition by Computer-A Survey Article*. MESM04, IEEE UKRI SPC, publisher EUROSIS, Amman, Jordan, September, 2004, pp. 102-107

To

"***The Wind beneath my wings***"

Dearest and most beloved mother

Dearest father

My wife … Hadeel

My beloved children

Omar, Mohammad, Anmar and Sarra'…

Those who ever reside in my heart

Acknowledgement

*Praise to **Allah** and may peace and blessing be on his most beloved messenger **Muhammad** (peace be upon him)*

I am very grateful for the manifold support I have received during the years of carrying this research. Special, due thanks and gratitude particularly go to the professional and noble scientist, ***Prof. Hussein Zedan, Head of STRL, De Montfort University /Leicester / United Kingdom***. I did learn beyond academic from him, and I truly prey to Allah to give him ever success.

My appreciation and thanks goes also to ***Prof. Ayman J. Al-Nsour, Dean of the Faculty of Information Technology/ Al-Isra University / Amman/ Jordan***, my local supervisor and true friend, whom I owe a lot and feeling speechless when I like to thank him for the support he gave to me.

I like also to register here my appreciation to ***Mrs Lynn Ryan***, the technical STRL coordinator for her kindness.

In addition, I like to thank all good friends whom I met at Leicester/U.K during the working in this thesis, for their generosity and truthfulness.

Leicester, United Kingdom
2010

Ayad Tareq Imam

Abstract

There are two main defined classes of uncertainty namely: fuzziness and ambiguity, where ambiguity is ‘one-to-many’ relationship between syntax and semantic of a proposition. This definition seems that it ignores ‘many-to-many’ relationship ambiguity type of uncertainty. In this thesis, we shall use complex-uncertainty to term many-to-many relationship ambiguity type of uncertainty.

This research proposes a new approach for handling the complex ambiguity type of uncertainty that may exist in data, for software engineering of predictive Data Mining (DM) classification models. The proposed approach is based on Relative-Fuzzy Logic (RFL), a novel type of fuzzy logic. RFL defines a new formulation of the problem of ambiguity type of uncertainty in terms of States Of Proposition (SOP). RFL describes its membership (semantic) value by using the new definition of Domain of Proposition (DOP), which is based on the relativity principle as defined by possible-worlds logic.

To achieve the goal of proposing RFL, a question is needed to be answered, which is: how these two approaches; i.e fuzzy logic and possible-world, can be mixed to produce a new membership value set (and later logic) that able to handle fuzziness and multiple viewpoints at the same time? Achieving such goal comes via providing possible world logic the ability to quantifying multiple viewpoints and also model fuzziness in each of these multiple viewpoints and expressing that in a new set of membership value.

Furthermore, a new architecture of Hierarchical Neural Network (HNN) called ML/RFL-Based Net has been developed in this research, along with a new learning algorithm and new recalling algorithm. The architecture, learning algorithm and recalling algorithm of ML/RFL-Based Net follow the principles of RFL. This new type of HNN is considered to be a RFL computation machine.

The ability of the Relative Fuzzy-based DM prediction model to tackle the problem of complex ambiguity type of uncertainty has been tested. Special-purpose Integrated Development Environment (IDE) software, which generates a DM prediction model for

speech recognition, has been developed in this research too, which is called RFL4ASR. This special purpose IDE is an extension of the definition of the traditional IDE.

Using multiple sets of TIMIT speech data, the prediction model of type ML/RFL-Based Net has classification accuracy of 69.2308%. This accuracy is higher than the best achievements of WEKA data mining machines given the same speech data.

Table of Contents

List of Tables

List of Figures

Glossary

AI	Artificial Intelligence
ANN	Artificial Neural Network
ARPASUR	Advanced Research Projects Agency Speech Understanding Research
ASR	Automatic Speech Recognition
BCM	Box Counting Method
CAP	Computer Aided Programming
CART	Classification And Regression Tree
CBCM	Continuous Box Counting Method
CHAID	CHi-square Automatic Interaction Detection
CMU	Carnegie-Mellon University
CSR	Continuous Speech Recognition
DB	Data Base
DFD	Data Flow Diagram
DFT	Discrete Fourier Transform
DIKW	Data Information Knowledge and Wisdom
DM	Data Mining
DMM	Data Mining Models
DOP	Domain Of Proposition
DSL	Data Description Language
DSS	Decision Supporting System
DTW	Dynamic Time Warping
EM	Expectation Maximization
ES	Expert System
FD	Functional Dependencies
FE	Feature Extraction
FFT	Fast Fourier Transformation
FL	Fuzzy Logic
FLS	Fuzzy Logic System
FOU	Footprint Of Uncertainty

GA	Genetic Algorithm
GUI	Graphical User Interface
GUIDE	Graphical User Interface Development Environment
HDL	Hypothesis Description Language
HMM	Hidden Markov Model
HNN	Hierarchical Neural Network
IDE	Integrated Development Environment
IE	Inference Engine
IS	Information System
ISR	Isolated Speech Recognition
IVFS	Interval Valued Type-2 Fuzzy Set
KBES	Knowledge Base Expert System
KDD	Knowledge-Discovery in Databases
LFCC	Linear Frequency Cepstral Coefficients
LPC	Linear Predictive Coding
LPCC	LPC-derived Cepstral Coefficients
LSH	Locality-Sensitive Hashing
MFCC	Mel-Frequency Cepstral Coefficients
MIT	Massachusetts Institute of Technology
ML	Multi Layer
MLE	Maximum Likelihood Estimation
MLP	Multi Layer Perceptron
NLP	Natural Language Processing
NNBMS	Neural Net-Base Management System
NNC	Nearest Neighbour Classifier
OLAP	On-line Analytical Processing
OOP	Object Oriented Paradigm
PDP	Parallel Distributed Processing
PSM	Power Spectrum Method
RF	Relative-Fuzzy
RFL	Relative-Fuzzy Logic

RFL4ASR	Relative-Fuzzy Logic for Automatic Speech Recognition
RFLS	Relative Fuzzy Logic System
SFSA	Statistical Finite State Automata
SOM	Self-Organising Map
SOP	State Of Proposition
SR	Speech Recognition
SRI	Stanford Research Institute
SSD	Software Design Description
TI	Texas Instrument
TIMIT	Texas Instrument - Massachusetts Institute of Technology
VQ	Vector Quantization
WDM	Walking- Divider Method

Chapter One

Introduction and Problem Definition

1.1 Data Mining and Uncertainty

The recent increase in data volume prompts the importance of the need for processing vast quantities, especially in retrieving certain types or in classifying the data obtained. A new computation field, which is Data Mining (DM), is now embarking on this problem via the inductive approach of Artificial Intelligence (AI) [30].

Discovering useful novel knowledge from data is the goal of DM, thus it can be defined as the process for pulling out of veiled extrapolative information from huge databases by using supervised or unsupervised techniques [121].

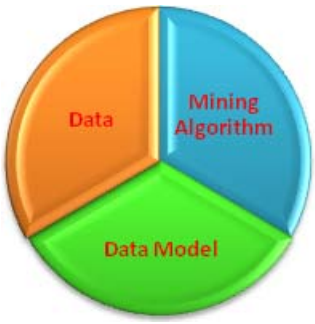


Figure 1.1 DM Elements

Figure 1.1 shows the block diagram of DM elements. DM is a new influential computational field with great possible facilitation of focusing on the most significant information in the database of companies for predicting future trends of their businesses. Since the tools of DM predict future trends and behaviours, we can say that the analysis of the forthcoming events would be automated with DM techniques, and differ from that performed in Decision Supporting System (DSS) that is usually built on past events leaving the expectation to human. Usually that DM uses large databases (known as warehouses) and performs its processing to extract hidden patterns of these data, and employs them to predict future events better than expert who may fall short in seeing those outside expectation-predictive information. DM is at its best, a method for

automatically penetrating huge assets of data and meaningful patterns. Knowledge-Discovery in Databases (KDD) is another term used occasionally for describing DM as well [23, 25, 42, 87].

With DM tools, rather than assuming certain relationships between different variables in a data set and studying these relationships one at a time, the most significant factors that influence the outcome can be determined. DM algorithms need no hypotheses about the data available, but they automatically discover both the hidden relationships between data and their patterns as well (using inductive approach of AI). Data representation plays an important role in DM. As the data they are structurally represented in a warehouse, this representation facilitates the process of DM to achieve its goal of defining the pattern of these data. DM has found numerous applications in banking, finance, marketing, telecommunications, and now in semantic webs [30, 87].

Practically, prediction and description models are the two types to create or to define, then to be used as a process for mining enormous data. Each of these two types of models has its own different function. Prediction type of Data Mining Model (DMM) involves the usage of some variables or fields in the database to predict mysterious or future values of other variables, whereas the description type of DMM concentrates on finding different patterns to describe the data that humans can easily interpret.

A new idea to visual DM had been proposed, which uses certain technology to apply some particular principles of the way that humans interpret data. The significance of visual data mining had been acknowledged. In visual DM, which is a strong sub-discipline of DM interfaces would be developed, which make visual presentations helps users to interpret the data that they see. The goal of visual DM is to merge traditional DM algorithms with techniques of information visualization to employ the advantages of both approaches. A number of modified techniques for visualizing the structure of certain information have been built. The collection of the information content resulted from searching the relationships between several information objects, is an important task in visual DM. These relations can be represented either explicitly or implicitly [42, 118].

As the goal of these two model types is to classify different types of data [23, 25, 42, 113], it is common knowledge that the lack of accurate results that are negatively influenced by the problem of uncertainty constitute the major problem of classification models [5, 46, 58].

In DM, the disclosing and handling of uncertainty in DM's environment are considered among the problems outlined by several recent research works in data mining applications [5]. This is due to the nature of DM strategy in creating its DMM, which seeks to classify huge amount of data into different sets, depending on some mathematical calculations. Generally, the classification comes in one of two types: crisp and fuzzy. The definite demarcation decision will mainly ensue from the calculation of membership strength value, which is either 0 or 1 of an element to a certain set via using crisp membership function that offers binary classification (in terms of belong / does not belong). This traditional classification has its limitations, mainly in situations where some basic information is missing or uncertain. In such situations, traditional inference procedures possibly will not be functional. One of the alternatives is fuzzy classification, which is able to describe these vague decisions using element's membership value. The membership value is the value used to express the strength of belonging of an element to a set, and it plays an important role in solving the problem of uncertainty. Apparently, there are different membership functions used to calculate membership value, which is in [0, 1] of an element to a set. There has been greatly interest in researching on the management of uncertain data in database; examples are uncertainty representation in databases and querying uncertain data. Yet, small research work has dealt with the problem of mining uncertain data [28, 42, 87, 102].

1.2 Uncertainty, Vagueness, and Decision-Making

One of the common features of the information presented to human experts is its indistinctness, which is termed as uncertainty. Berenji [6] believes that "uncertainty stems from lack of complete information." and, "Uncertainty may also reflect incompleteness, imprecision, missing information, or randomness in data and a process." Hubbard [49] defines uncertainty as "The lack of certainty, a state of having limited

knowledge where it is impossible to exactly describe existing state or future outcome, more than one possible outcome". As it reflects the inadequacy of the precise knowledge, uncertainty is a high probable cause of achieving an imperfectly consistent conclusion. There are different classifications of uncertainty, which actually involve a wider sense of uncertainty. Using mathematical language, uncertainty may range from a diminishing little certainty to an approximately entire lack of knowledge (particularly about a conclusion or result) [6, 49, 121]. Thus, uncertainty can be described as "the characterisation of the relative narrowness or broadness of the distribution function of a particular measurement, and it is sometimes referred to as error in the measurement" [135].

The cause of uncertainty involved in any problem solving, in term of these definitions, is information shortage. Information could be vague, disconnected, undependable, incomplete, conflicting, or lacking in some other way. These various scarcities of information, however, may result in different forms of uncertainty [46, 68, 75].

Klir and Wierman [68] elaborated on the description of uncertainty by defining three types, namely:

1. Fuzziness or vagueness, resulting from the imprecise boundaries of fuzzy sets;
2. Nonspecificity or information-based imprecision, associated with the size (cardinalities) of related sets of alternatives ; and
3. Strife or discord, which describes conflicts among the various sets of alternatives

Usually, the term 'imprecision' is used to describe the first two types of uncertainty, i.e. *fuzziness*, referring to linguistic imprecision, and *nonspecificity*, which is referral to information-based imprecision. The ambiguity in 'imprecision' term can be passed up by individualising between nonspecificity and fuzziness. Therefore, Klir and Wierman [68] define two main classes of uncertainty namely: *fuzziness* and *ambiguity*, where ambiguity (one-to-many relationship) includes nonspecificity and strife. Well, this definition seems that it ignores (many-to-many relationship) ambiguity type of uncertainty. In this thesis, we shall use complex-ambiguity to term many-to-many

relationship ambiguity type of uncertainty. Table 1.1 illustrates the types of uncertainty based on the source-forms (can be also syntax-semantic) relationships types.

Source	Form	Relationship	Ambiguity
One	One	One-to-one	None
One	Many	On-to-many	Simple
Many	One	Many-to-one	None
Many	Many	Many-to-many	Complex

Table 1.1 Source-Form relationships-based types of uncertainty

Anyhow, Uncertainty is usual happening, and it is really hard to keep away from it in dealing with real-world problems, where it appears in different forms. These forms are [68]:

1. Both measurement errors and resolution limits of measuring instruments,
2. Vagueness and ambiguity in natural language, and
3. The strategic use of uncertainty by people when it is created and maintained for different certain reasons (privacy, secrecy, propriety).

In spite of these appearances of uncertainty, a proficient can cope with them and can typically make correct judgments and right decisions.

1.3 Modelling of Uncertainty

The basic principle is that most, if not all, modelling of decision-making software deals with uncertainty in aspects such as data, knowledge, and rules. Modelling of uncertainty is reached mainly by using different types of logic, in which uncertainty is considered as challenge to all these types of logic. Luger [75] quoted from Bertrand Russell that "All traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life but only to an imagined celestial existence". This highlights the need for unusual types of logic to deal with uncertainty. Broadly speaking, dealing with uncertainty (modelling and managing its reasoning) is achieved by using the following types of logic: *Multiple-valued logics, Modal logics, Temporal logics, High-order logics and Logical formulations of*

definitions, Prototypes, and Exceptions [75]. It is important to underscore here the independence of the handling strategy from the appearance of uncertainty [46, 68, 75].

Fuzzy logic- a multiple-valued logic type is the dominant methodology used for effectively handling the uncertainty problem. It offers a way of taking this uncertainty into account to develop better computer systems [135, 69], which are termed as Fuzzy Logic Systems (FLS). Indeed, "Fuzzy logic is used heavily in control applications and the argument by some has not made as big in impact as expected in modelling human decision making" [84].

In type-1 FLS, uncertainty is to be quantified as a real number between zero and one by using fuzzy membership function, (Chapter Three illustrates the quantification of membership value of type-1 fuzzy logic, and type-2 fuzzy logic in detail). It has, however, been reported by Mendel et al. [83] that there are (at least) four sources of uncertainties in type-1 FLSs. They are:

1. Words mean different things to different people. Naturally, humans show dissimilarity in the understanding of words used in antecedents and consequents of rules (decision-making).
2. Dissimilarity may also occur among the decisions of a group of experts who do not all agree, as well as in the decisions of a single expert over time.
3. The noise that may exist in the measurements stimulates a type-1 FLS and hence is uncertain.
4. The possible noise of the data is used to harmonise the parameters of a type-1 FLS.

The terms: *different meaning* (in the first source of uncertainty), *decisions of a group of experts* (in the second), *noisy of measurements* (in the third), and *parameters* (of the fourth) clearly point to a shared property among these sources, namely *multiplicity*. Accordingly, we can say that whenever multiplicity occurs in data, measurements, or decisions, the uncertainty exists also. This multiplicity falls underneath the 'ambiguity' class of uncertainty defined by Klier and Folger [69], as shown in Section 1.2 above.

Because of the problems listed above, type-2 had been proposed to extend the ability of Fuzzy Logic (FL) to handle the uncertainty problem provided by type-1, precisely to address directly the three types of uncertainty: fuzziness, strife, and nonspecificity. This concurs with the common knowledge among researches that type-2 fuzzy logic is used more effectively as a method for handling more complex types of ambiguity and better than type-1 [73, 83, 82, 102, 135].

Type-2 fuzzy logic quantifies uncertainty as a fuzzy set in [0, 1], using a complex membership function (function of function) in which the internal function itself takes a type-1 fuzzy number. It maps this element to a set of type-1 fuzzy numbers [85, 84]. Due to the nature of type-2 fuzzy logic, Mendel and John [83] addressed number of the difficulties around it. These difficulties were:

1. The three-dimensional nature of type-2 fuzzy sets makes them very difficult to draw.
2. There is no simple collection of well-defined terms that let us effectively communicate about type-2 fuzzy sets, and then be mathematically precise about them (terms do exist but have not been precisely defined).
3. Derivations of the formulas for the union, intersection, and complement of type-2 fuzzy sets, all rely on using Zadeh's Extension Principle [139], which is in itself a difficult concept (especially for newcomers to FL) and is somewhat ad hoc. Hence, using it for derivation may be considered problematic.
4. Using type-2 fuzzy sets is computationally more complicated than using type-1 fuzzy sets.

The paper of Mendel and John [83] along with the research work of Coupland and John [22] advanced, indeed, the definition and usage of type-2 fuzzy sets. Both focus on overcoming difficulties mentioned above. Mendel and John had established, in their research, a small set of terms, namely: Type-2 membership function, Secondary membership function, Vertical-slice, Primary membership, Secondary grade, Footprint of uncertainty (FOU), Embedded type-2 fuzzy set, Embedded type-1 fuzzy set, and wavy-slice. These terms facilitate communication about type-2 fuzzy sets, and make the

definition of such sets very precise. In the paper of Mendel and John [83], a new 'Representation Theorem for type-2 fuzzy sets' has been presented and used to derive formulas for the union, intersection, and complement of type-2 fuzzy sets devoid of employing the extension principle [61, 83].

The principle behind defining type-2 fuzzy secondary membership function is that "Type-2 fuzzy sets are fuzzy sets whose grades of membership are themselves fuzzy" [28]. This means that type-2 fuzzy logic translates all uncertainties into *uncertainties about fuzzy set membership functions*. Admitting all these recorded achievements, we can easily note:

1. "The result of type-2 inferencing is a type-2 fuzzy set. Nevertheless, we usually need a crisp solution in order to reduce the type-2 fuzzy set to a type-1 fuzzy set and then defuzzify. Possibilities to do so are redundant: Centroid Type Reduction, Centre of Sums Type Reduction, and Height (various) Type Reduction" [58]. In that, there is a shortcoming in type-2 fuzzy logic to provide crisp membership value directly, and the apparent need to step further for crisping the type-2 fuzzy membership value.

2. A drawback in applying the principle of *fuzziness* to type-1 fuzzy logic to define type-2 fuzzy membership value set. In that, fuzziness can also be used to describe the nature of each element enclosed in the set resulting from type-2 membership function. Applying the same approach to each of the resulting membership values further complicates the description and evaluation of membership value. This means that the resulting values from type-2 fuzzy function can be considered as fuzzy values also!

3. Type-2 FL aimed to model and handle uncertainty caused by multi sources data or multi decision approaches [83 and others]. It means that this set maps multiple semantic to multiple judgments, which is shown by the nature of the type-2 semantic value (or membership value in set sense) that is a set rather than element. Klir and Wierman [68] described this form of uncertainty as ambiguity of type 'one-to-many' relationship, see Table 1.1 above. The point that needs to be clarified is: *What is about ambiguity of type 'many-to-many' relationship?*

These criticisms have encouraged this thesis to develop type-1 fuzzy logic in different way that type-2 did, by using another philosophy, namely *possible worlds*. In this thesis, the possible-worlds principle is to be applied to type-1 fuzzy logic to define a new type called Relative-Fuzzy logic (RFL). It means that RFL is aimed to model and handle the same types of uncertainties modelled by type-2 fuzzy logic and to overcome the criticisms defined against type-2 fuzzy logic as illustrated above.

1.4 Scope of the Thesis

Over the last decade, dramatic improvements in the quality of DM classification systems have been made. With the development and refinement of the Hidden Markov Model (HMM) approach, today's classification systems have proven to work effectively on various warehouses, noisy data, and various classification tasks. Nevertheless, despite the high quality of today's classification systems [26, 41, 74], there still be a significant improvement in performance that can be made. As will be shown in Chapter Seven, the reduction in a system's error rate and its complexity can handle types of problems that DM practitioners wrestle with everyday.

While developing a DMM, the importance of revealing and handling uncertainty should be recognised. A DM classifier (prediction or description model) should account for both the variability across syntax and the semantic of the data proposed. Currently, typical DM systems seek to achieve strong classification performance. Because complex-ambiguity is not defined yet, many systems don't clearly look the issue of complex-ambiguity constraint.

Classical modelling (but type-2 FL) deals mainly with simple (abstract) uncertainty of the data that cannot be further subdivided. Type-2 FL, which is able to deal 'one-to-many' relationship ambiguous data, has some difficulties as shown earlier, so it requires some sort of enhancement to overcome these difficulties without affecting its ability to model and handle the complex- ambiguity.

In this thesis, a new approach of modelling and handling the uncertainty is to be carried out and applied to a DM's classification method. This approach would requires

defining new type of membership value, and uses this new type of membership value in developing new type of fuzzy logic, and new type of Fuzzy Logic System (FLS), which will be used as a DMM. Figure 1.2 illustrates the scope of the thesis.

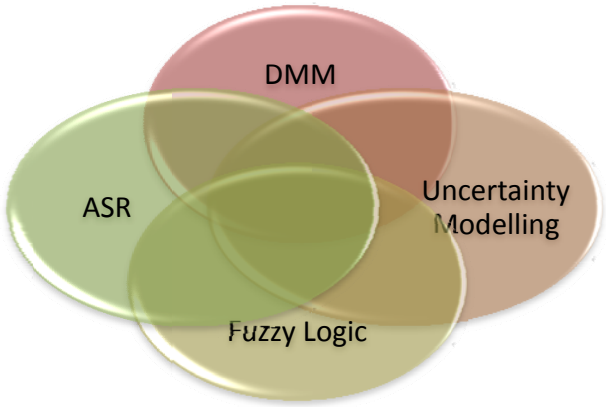


Figure 1.2 Scope of the Thesis

Relative-Fuzzy membership value set is to be defined to be the seed of defining RFL that provides a new framework for the characterisation and modelling of complex ambiguity found in data. RFL have provided us with a new method of characterising complex ambiguity in data by means of the possible world. This thesis will specifically examine the issues involved in utilising RFL that is able to model and handle the complex ambiguity.

Type-1 fuzzy set (that used to model vagueness of data) along with possible world philosophy, are going to be used for defining Relative Fuzzy (RF) membership value set, which is the core of RFL. Actually, that the theory of possible worlds became a core of numerous philosophical developments, since the 1960s. In addition, possible world has been used previously in some applications such as logic, Data Base (DB) [65, 74, 104] and we think it could be affinity found in AI and classification-based application's, i.e. DM.

Because RFL is new, it is expected that it will be some difficulties in designing new type of FLS that to be called Relative-Based FLS (RFLS) - the extended system type of traditional FLS. However, it is a generally held belief; that these problems may have solutions in the growing field of programming techniques.

Since speech sounds contain turbulence [53, 62, 78, 79, 105] that makes these sounds of complex ambiguity data type, this attribute of speech data motivated us to use the data of speech sounds to develop a case study in terms of DM prediction model to demonstrate the conceptual truthfulness of RFL as an approach to modelling and handling the uncertainty.

It is worth noting at this point that the scope and accuracy of classification that can reasonably be expected using the RFL essentially gives us a measure of the degree of success or achievement of the RFLS. In addition, the number of possible worlds (domains) to which data can truly be existed inherently limits the usefulness of RFL as an approach for modelling uncertainty.

In abbreviation, this thesis answers questions: How would the problem of classifying different elements, which have multiple meanings (knowledge sets) and multiple syntaxes of each element in each meaning modelled? How to build a relative fuzzy based computing machine? Also, how ***complex-ambiguity*** can be solved for building a successful DMM?

1.5 Approaches and Accomplishments

The investigation of this thesis proceeds from the position that current logics used in modelling and handling of uncertainty play an important role in obtaining more active approach of modelling and handling uncertainty in different classification-based applications like DM. The proposed approach of modelling and handling uncertainty will be useful in enhancing the performance of DM processes in order to create successful DMM. Therefore, the research examines the conceptual truthiness of the developed Relative-Fuzzy approach by using its membership function for calculating a new type of membership value (which is different from crisp and fuzzy values) and uses it in developing a new computational model. Hence, this thesis makes four new proposals as will be illustrated later.

The problem of complex ambiguity described earlier could not be solved by using the crisp, and type-1 Fuzzy logic, though both deal with data classification. While type-2

fuzzy logic can handle a type of the problem of uncertainty, its approach of processing involves drawbacks. One of these drawbacks is apparent in the applications of implementing type-2 fuzzy logic where new training data are applied to these applications with each new task domain, and a new classification model would be made, so that each mysterious element (with its set of different syntaxes) would assign its appropriate class name (semantic) from these training data. As a new training course is to be applied to these applications, a computational classification model, with new categorisation sets and functions, will be created. Of course, the new and old models would be able to deal with their respective task domains of data, but in no way can be combined.

Creating a computational model that is able to accumulate and handle multiple training courses (tasks domains), and using this knowledge sets to address the belonging of an unknown element to one of them would add a classification methodology to those already existing models. This model would be used for specifying the identity of the unknown element, despite its different attributes and its duplication in various previous sets of knowledge, i.e. classification of an element with complex ambiguity.

How does human brain distinguishing new data? [119]. To design a method that can handle complex ambiguity of data, our thinking started with the atomic level of the classification process, which is the measurement of the belonging of an element to a set (task domain). Measuring the belonging of unknown-element to a set is determined by using a scoring function, which would give a membership value (either crisp or fuzzy). But as this element may belong to multiple subsets of knowledge (tasks domains), which form a universal knowledge, such a function and its result would not be so helpful in terms of how to distinguish the belonging degrees of an element to each subset of data (tasks domain), which form a universal knowledge. Therefore, a further function that evaluates the belonging of an element to a certain subset of knowledge is required. One membership function would calculate the membership value of an unknown element to a set of circumstances that distinguish a task domain among other tasks domain in the universal set of knowledge, while a second one would calculate the membership value of

an unknown element to that certain task domain (the set of syntax that possibly form that unknown element).

1.6 Contributions

Four contributions resulted from current research. Figure 1.3 illustrates the contributions achieved by this thesis distinguished in gray boxes.

The first original contribution of this thesis consists of defining and using a new membership value set to handle the uncertainty problem. This contribution leads to define a new logic, called RFL. A pair real-value is the numeric expression used to measure the identity of an element in terms of the strength of belonging to a set of circumstances of a task domain, and to the task domain itself, (both sets form the relative-fuzzy universal set). This pair valued membership is to be generated by using two separated membership functions that may work in parallel in order to calculate each part of the Relative-Fuzzy membership value of an element.

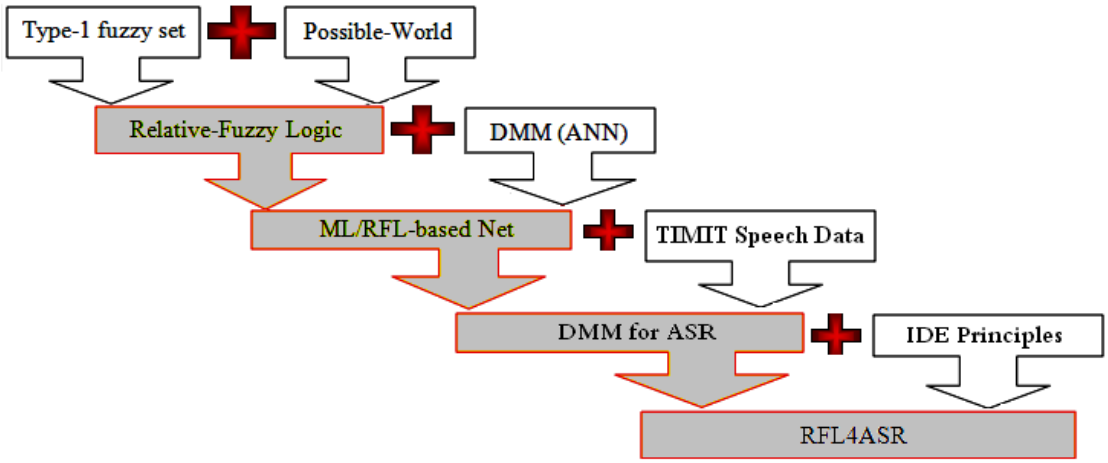


Figure 1.3 Contributions Achieved in the Research

The second contribution is the creation of a computational model that uses the new membership value set, proposed by this thesis, to classify elements. The computational model is a complex-structure neural network, called ML/RFL-Based Net.

The third contribution is the use of ML/RFL-Based Net (the proposed computational model) for building a new-structured DM predictive model to classify a set of

phonemes. The phonemes used in the experiments were different and were generated by various speakers, each speaker in his own accent, which represent a good example of data with complex ambiguity. To show the achievement of the ML/RFL-Based Net, these data were tested with 'WEKA', a well-known DM machine. This thesis records the highest accuracy performance of any 'WEKA' approach.

The last contribution to be reported here is that a new understanding of Integrated Development Environment (IDE) has been used when this thesis uses the principle of general-purpose software IDE to build an IDE of specific-purpose software-here Automatic Speech Recognition (ASR).

1.7 Thesis Outline

The ultimate goal of this thesis is to define a new approach to build a new classification model that is able to handle the uncertainty in DM, and as such to face some of the challenges of data mining, which will be outlined in Chapter Two. An ASR case study of this approach is used to provide a practical demonstration of its applicability. Figure 1.4 illustrates the strategy followed by this thesis

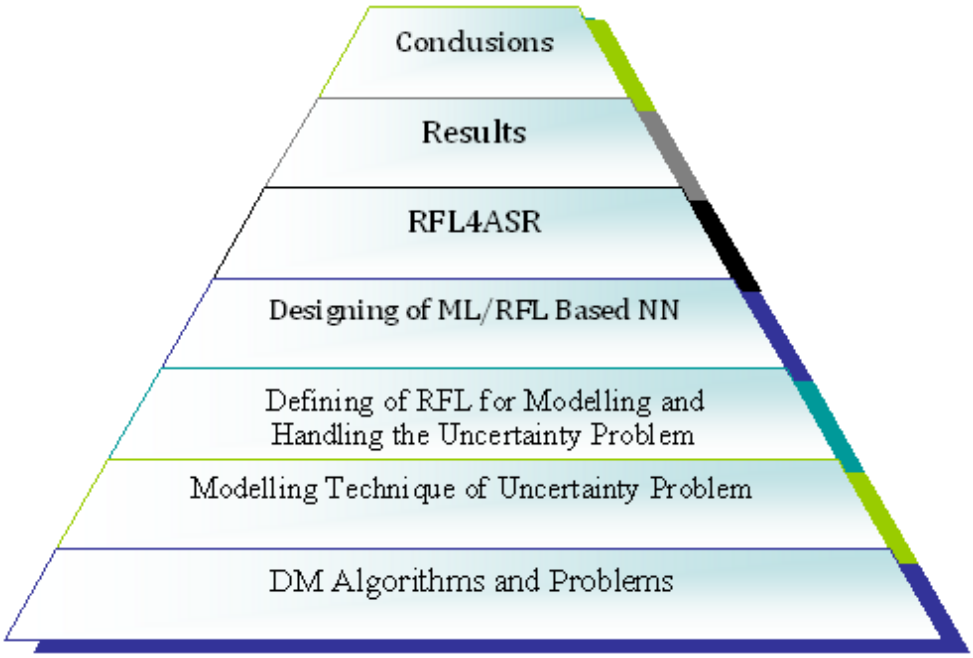


Figure 1.4 Methodology Followed by the Research

Chapter Two presents survey of literature about the algorithms of Data Mining (DM), including definitions, methods, and applications of DM. The role of DM in the current development of Information Systems (IS) is also presented, and the challenges of DM are reported. Finally, a number of conclusions are given.

Chapter Three includes literature survey about the uncertainty modelling and handling logics, focusing on fuzzy and possible-worlds of modal logic, which both is used to develop RFL.

Chapter Four includes the definition of Relative-Fuzzy logic, by means:

1) Of defining the States of Proposition (SOP), which allows the easy mapping of the type of uncertainty aimed to be handled by Relative-Fuzzy sets, and which also enables the precise definition of the sets
2) Of defining the Domain of Proposition (DOP) that shows the new parameter involved in calculating the membership of an element to a set, and
3) Of defining the membership value set of RFL

Chapter Five describes ML/RFL-Based Net**,** a novel architecture of Artificial Neural Network (ANN**)**. ML/RFL-Based Net has been designed and built to be RFL's computation machine. Description of its levels, layers, learning, and recalling approaches is given in this chapter.

ASR is used as a case study for the proposed DM approach. Chapter Six includes the development of an ASR using Relative Fuzzy concept. This DM classification system, which is called RFL4ASR, uses Relative-Fuzzy based classification principle- the proposed data mining methodology. A description of all constituent parts of RFL4ASR is also included. Chapter Six finished with a number of conclusions that have been drawn.

Chapter Seven shows the results of each part of RFL4ASR diagrammatically and numerically. The results of different versions of ML/RFL-Based Net are illustrated, highlighting the improvement on the proposed net during the course of its development.

Conclusions of this thesis, as well as recommendations for future work, are presented in Chapter Eight.

Appendix A illustrates the tools used in developing RFL4ASR. Appendix B presents a survey of literature, which includes previous and current methods of developing ASR's two components: front-end and back-end processing, speech signal, digitising, segmentation and parameter measurement (spectral analysis, coding, and encoding via quantisation).

Chapter Two

Data Mining Approaches

2.1 Introduction

The main goal of this thesis is to define a new approach to handle the problem of uncertainty, and show the conceptual truthiness of this new approach by using it to develop a computer application. The current research selects DM as a type of applications where the problem of uncertainty exists and harmfully affected DM's performance. The aim of this chapter is to give literature survey about DM's goals, main algorithms, and approaches to handle uncertainty, results, applications, and problems.

The mining or finding of implicit information, in terms of model, from vast amount of data is actually the goal of DM. The techniques, which are approved of by DM to achieve this goal, are to be adopted from such areas as machine learning, statistics, ANN and Genetic Algorithms (GA). The output knowledge is in form of various types of models [25, 30, 42].

The approaches used in DM are actually a result of the extensive development of both research and production. Historically, DM resulted from data storage evolving techniques. The first method used in storing data in computers was the flat files approach. Flat files contain numeric and/or character data, with no inter connection. The advancement in defining new methods for saving and retrieving data by using Data Base (DB) techniques pushed the methods of DM forward and made a good improvement in DM. Nowadays, On-Line Analytical Processing (OLAP) are the latest defined tools used to handle data processing (saving and retrieving) in real time. Being historical, a special data access called backward-looking gives DM an evolutionary process over these stored data. Accompanied with navigation, DM makes use of backward-looking for informative and upbeat liberation of required information [23, 25].

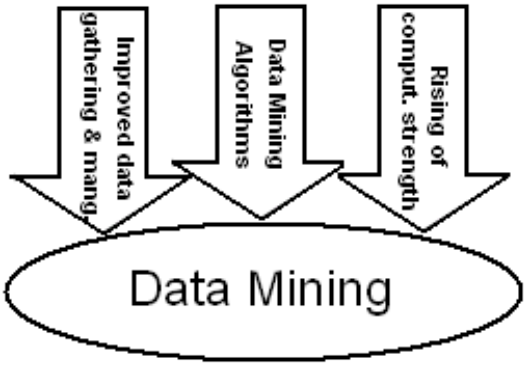


Figure 2.1 DM Supporting Technologies

Worth mentioning here is that DM is not engaged only in DB, but with some other technologies too, Figure 2.1 shows the technologies that support data mining for applications in the business society where they are adequately established now. They are [23, 25, 30]:

- Data mining algorithms
- Improved data gathering and management
- Rising of computation strength

There are many definitions of DM. Among others, DM is defined as the searching to find or extract (in terms of mining) hidden predictive patterns or rules which describe the information given in enormous quantities of data. In another word, data can be defined as a set of facts F, and a pattern can be defined as an expression E in a language L that describing the facts in a subset F_E of F. Another definition is that DM is a scheme that receives an input data in some form and produces knowledge in a model form [30] as illustrated by Figure 2.2.

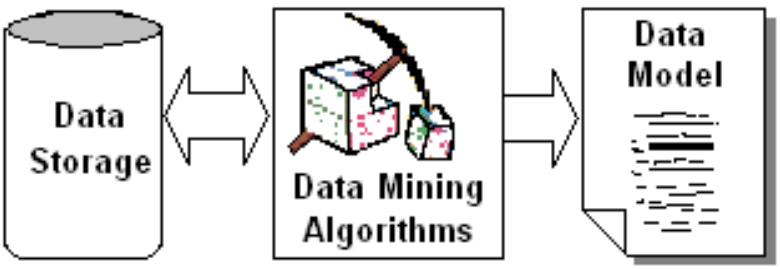


Figure 2.2 Definition of Data Mining

Recently, DM proved itself as an effective assistant tool for business centres looking backward on the most significant information in their data storage(s). DM is now of an influential technology with high probability for making this backward-looking. In detail, the DM tools are mainly used for predicting upcoming styles and behaviours, hence permitting companies to make proactive and knowledge-driven decisions. The computerised and potential analysis presented by DM goes beyond the analysis of historical events provided by backward-looking tools for participating building Decision Support Systems (DSS). Indeed, the tools provided by DM can answer business queries that were usually very high time-consuming to decide. These tools search databases for hidden patterns and discover predictive information, which experts may miss as it lies outside their expectations. The existing software and hardware platforms help quickly implement the techniques of DM for enhancing the value of resources of existing information. DM technologies can also integrate these resources with new products and systems as they are carried on-line. It is important to note that DM tools are able to analyse huge database, especially when they are employed at a high performance client/server or on parallel processing computers aimed at bringing replies to queries like '*Which goods are most likely to be consumed during September?'* [23, 30, 47].

DM has been applied in various fields like telecommunications, finance, banking, marketing, and nowadays in the semantic web. As a result, a wide range of companies has installed successful applications of DM in their every day business. Information-intensive businesses, like direct mail marketing and financial services, were the early adopters of this technology. Today, DM technology is applicable to any business looking to influence a hefty data warehouse to obtain superior management over their customer relationships. Each of these applications has a clear common ground. They influence the knowledge about customers implicit in data storage to decrease costs and advance the value of customer relationships [23, 42, 113].

In broader sense, the goals of DM fall into the following classes: prediction, identification, description, and optimisation. DM mainly focuses on prediction and description, which they can be achieved by means of an assortment of certain DM techniques. Some of these techniques are [23, 25, 30, 42]:

- Classification Hierarchies (Regression, ANN, and Genetic Algorithms)
- Clustering
- Association Rules
- Sequential Patterns
- Patterns within Time Series
- Summarisation
- Change and Deviation Detection

These techniques are integrated in huge data storage as well as with tools of flexible interactive business analysis to achieve maximum effect of these complex techniques. Worth noting here that the term: 'Knowledge Mining' is used to describe the process of extracting meta information from certain knowledge (upper level of data) in the form of IF-THEN rules, which can be modelled using Fuzzy Logic System (FLS)- since this knowledge may have uncertainty property [82].

Finally, researchers reported that there are four critical factors for success with standard DM system, which are [25, 113]:

- Customer's viewpoint,
- A large, well-integrated data warehouse,
- Preservation and operation management,
- A well-defined understanding of the business process to which DM is to be applied

2.2 Data Mining Algorithms

The algorithms of DM involve techniques, which have already been introduced for more than a decade, but have lately been implemented as an understandable, dependable, and mature tools [42, 47]. DM Methods are essentially based on standard statistical techniques as well as Artificial Intelligence (AI) methods. They are applied to huge databases for aiming to expose otherwise undiscovered data attributes, styles and patterns, thus, DM tools determines the most significant factor that influences the result. This determination approach is used as an alternative to assuming certain relationship

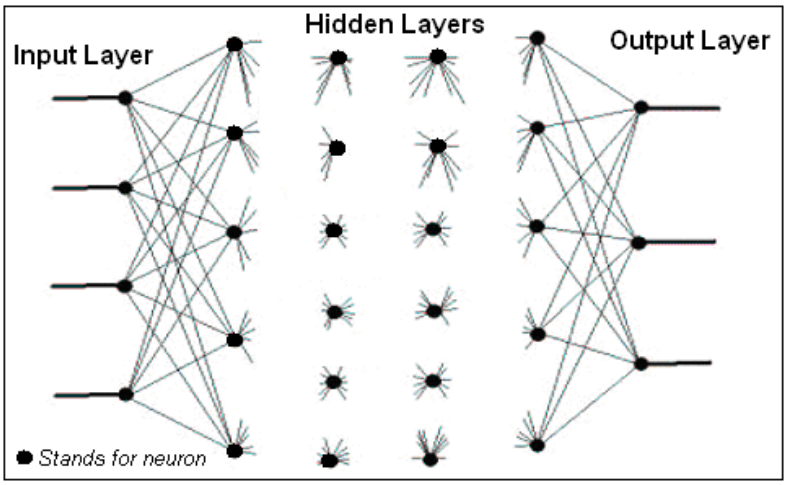
among different variables in a data set, and for studying these relationships at one time. In other words, the DM process has the ability to determine automatically the implicit relationships among the available data without need for any assumptions. Actually, as the data are represented structurally in a warehouse, the process of DM becomes easier [43, 86]. Note that there is a technique used to enhance the activeness of the defined DM algorithms by combining two or more of these algorithms to form what is so called hybrid data mining procedure [121]. There are so many methods of DM. Following paragraphs gives literatures about most known methods.

2.2.1 Artificial Neural Network

An Artificial Neural Network (ANN) can be defined as a paradigm of information processing which counterfeit the technique followed by biological nervous systems, e.g. brain. The new structure of information processing system provided by ANN is considered the corner stone of this paradigm. It consists of a big number of greatly interconnected neurones (processing elements) that are working in harmony to solve specific problems [15, 48, 98].

In recent years, ANN has emerged as a mature and practical framework with many applications in different areas of applications. The goal of developing ANN is to model the information processing and learning in the brain. ANN has many synonyms, which are: Neurocomputing, Neuroinformatics, Neural Information Processing Systems, Connectionist Models, Parallel Distributed Processing (PDP) Models, Self-organising Systems, Neuromorphic Systems [92, 106, 122, 128].

Figure 2.3 illustrates a typical structure of ANN. Very similar to human beings, ANNs learn by using examples. A specific application configures a specific ANN compatible to it, such as data classification or pattern recognition, throughout what is known learning/training phase. In fact, in biological systems, learning engages tuning values to the synaptic links that exist among the neurones, and this is what happens exactly in ANNs as well [15, 48, 106, 122].



Figurer 2.3 Neural Network Structure

ANNs can be used to recognise patterns and perceive styles that are too difficult to be observed by either human or other computer skills. This is thanks to the property of ANNs, which have an extraordinary ability to get meaning from complicated, inexact, or vague data. As such, a trained or learned ANN may be considered as an expert in the type of information it has been assigned to analyse during the learning/training phase. This very expert ANN can then be used later to provide answers when given a new data and to respond to questions of the type what if. Other criteria of ANN include [15, 48, 92, 128]:

- ***Self-Organisation***: The ability of an ANN to organise or represent the data it receives whilst in its learning / training course.
- ***Adaptive Learning or the Inductive Ability:*** The ability to learn (generalise processing) how to perform tasks based on the available data during the learning/training experience.
- ***Real Time Operation***: The computations of ANN can be carried out in parallel. To get used to this capability, special hardware devices have been designed and manufactured.
- ***Fault Tolerance via Redundant Information Coding***: Some ANN capabilities may be reserved even with major network damage.

Because of these criteria, ANNs are mostly applicable wherever it is hard to define relationship between patterns that are to be dealt with. Applications and models have been developed in areas varied from speech recognition to stock market, time series prediction, and other new ones are appearing rapidly. Generally, ANN is good at [15,106,128]:

- Determining relationships and patterns that exist in the data,
- Pre-processing of data in a FLS, and
- Refining fuzzy rules to build a fuzzy adaptive system due to the ability of ANN to learn new relationships with new input data

ANN is a non-algorithmic approach of solving problems. This differentiates ANNs from those traditional algorithmic approaches applied to computers in solving problems. In a traditional algorithmic approach, the computer executes a set of commands, that is program, in which the computer cannot solve the problem unless these specific set of steps are earlier defined to be followed by the computer. This fact limits the ability of problem solving in traditional algorithmic approach that are already recognised and solved by human. Certainly, the value of computers would be increasingly enhanced if they could perform tasks (solving problems), whose processes are difficult to work out by human [122]. Any how, The problems to be solved using ANNs have two important characteristics [15, 48, 106, 128] :

- The problem required a nonalgorithmic type of solution. Nonalgorithmic solution is an opposite of step-by-step algorithm or logical formula.
- The data given on the problem is complex and may be noisy or incomplete.

Traditional procedural approach is a cognitive approach for solving problems in terms of induction. This algorithmic procedure must be identified and expressed in small definite human instructions. These instructions are to be implemented to an executable program by using either high-level or low level programming language. Hence, this procedural approach is ***very predictable.*** Unlikely, ANN's ***process cannot be predictable*** since the ANN finds out the way to solve the problem by itself without any human assistance during learning / training phase. It is agreed among ANN programmers that

the unpredictable property is the main disadvantage of ANN. It is worthwhile to say that there is no competition between ANN and algorithmic approach they rather complement each other. While the algorithmic approach is suited to certain tasks, such as arithmetic operations, ANN's approach is more suitable for solving other types of problems, whereas a third type of problems needs a hybrid approach of both algorithmic and ANN to be solved and eventually gain maximum efficiency [48, 122, 128].

ANN's areas of application generally include autoassociation and hetroassociation, optimisation, pattern recognition, data compression, data completion, signal filtering, image processing, and handwriting analysis. Currently, a new approach of data processing, which is called DM, implements ANN as one of the techniques used for mining pervious data [15, 34, 48, 122, 141].

2.2.1.1 The Structural and Functional Design of ANN

ANNs use the same way as that the human brain may do to organise synthetic neurons for solving the same kind of difficult and nonalgorithmic problems. ANN can be defined as a group of processing units where typically one subgroup (called layer) makes independent computations and passes the results to a next subgroup (or layer). The type of connection among layers are either feedforward (directed acyclic graph), or recurrent, i.e. networks with cycles. Figure 2.4 illustrates a typical diagram of ANN [15, 48, 122, 141].

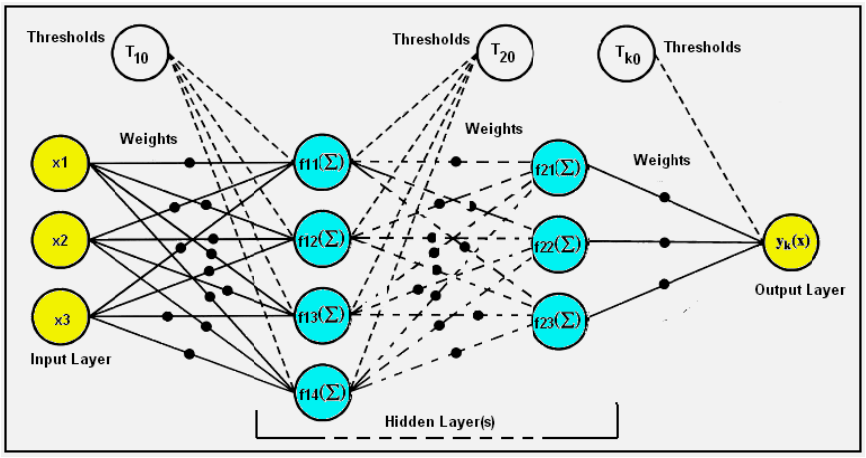


Figure 2.4 Typical Architecture of an ANN

The input of ANN is in the form of a pattern or vector, which needs to be designed while constructing the ANN itself. The two types of input pattern values that are used in ANN are either binary {0, 1}or bipolar {-1, 1} inputs [15, 48].

A threshold function is used to determine the output value from each processing unit (neuron). The value of a threshold may take one of two possible forms. In one, the calculated activation value is to be compared to a certain threshold value. Depending on that comparison, the neuron gives its reaction in that it fires if the threshold value is attained or exceeded, otherwise not. The second is to add a certain value to the activation itself (in which case it is called the bias), and then to determine the output of the neuron [15, 48, 92, 98, 122, 141].

The outputs are result from either a cooperation strategy; which is defined as the attempt between neurons that one neuron aids firing another neuron, or a competition strategy; which is the attempt between neurons to individually excel with higher output [15, 48, 92, 98, 122, 141].

The weights of the neuron's input, which are positioned on the connections between different layers, have much significance in the working of the neural network and the characterisation of a network. Initialising the network structure is part of what is known as the encoding phase of a network operation [15, 48, 92, 98, 122, 141].

The other essential issue of working with ANN is the learning of it. Learning can be defined as the process of changing – or rather refining - the weights. A network in which learning is employed is said to be subjected to training. Feedback information may be used in training. A network can be given supervised, unsupervised or reinforcement learning. The learning would be supervised if external criteria were used to be matched by the network output, and unsupervised if such criteria were not used. This is one way to classified broadly different ANN approaches. Unsupervised approaches are also termed as self-organising. Reinforcement learning is used in the applications of control problems, games and other sequential decision making application [34, 88, 106, 122, 141].

There are various existing algorithms for learning neural network models. Most of these algorithms can be viewed as a simple application of statistical estimation and optimisation theory. Most of the algorithms used in training of ANN utilise some form of gradient descent, which is achieved by simply getting the derivative of the cost function in respect to the parameters of the network and then adjusting those parameters in a gradient-related direction. Other frequently used methods for training ANN are: simulated annealing, evolutionary methods, and Expectation-maximisation and non-parametric methods [88, 89, 90, 122, 141].

The type of application imposed two restrictions to be considered when designing ANN for it, namely the type of connections in the ANN architecture and the type of learning (training) algorithm used with that architecture. ANN with lateral connections for example, can perform auto-association, while ANN of a vector matching type performs optimisation. Furthermore, the number of layers has an important role in selecting an ANN for a specific application. ANN of single-layer is able to perform autoassociation, while heteroassociation or other application types such as mappings, requires ANN of two layers at least. Last but not least, the reaction of the ANN to the existence of noise in data is a significant feature in determining certain ANN usability in a specific application, since a set of data used for training (or even testing) an ANN can simply have inherent noise [15, 34, 48, 106, 122, 141].

To design an ANN for a certain problem, there are three construction aspects that the designer should deal with, these are [15, 34, 48, 106, 122, 141]:

- ***Architecture***: relating to the number of layers (single, two, or multi) and type of their connections, the way to make interconnections between neurons in the network(lateral, forward, recurrent, feedback), and finally, the type of their functions

- ***Encoding and Learning***: that relates to the algorithms used for the changing of weights on the connections between neurons awaiting getting ideal values. The learning algorithms are of types: supervised, unsupervised, and reinforcement

- ***Recall***: related to reaching expected outputs from trained ANN with given specific input. ANNs are characterised as being either autoassociative, when ANN performs an association of the input data with itself as the output, or heteroassociative, when ANN remembers a related output given an input data. Both of autoassociation and hetroassociation are the types of recalling.

2.2.1.2 Hierarchical Neural Network

Multi classifier system has been suggested to overcome the limitations of single classifier systems. As every single classifier that could be used for a recognition process has its own strength and weakness, the purpose of multiclassifier systems is to utilise the strength of multiple classifiers while bypassing their weaknesses so that the recognition rate can be maximised. This approach is used after realising that the many great attempts that have been carried to optimise the performance of each layer in a single classifier system may not produced valuable enhancement in recognition by the entire system. Therefore, multiclassifier systems, which contain various single classifiers, have been developed to improve the recognition performance considerably. These systems organise multiple single classifiers in away that their weaknesses can be reciprocally rewarded by the strength of each other. The classification of the multiclassifier is to be made based on suitable judgement, and the recognition rate can be notably improved [72, 38, 133]. Four different types of multiclassifier systems have been acknowledged namely: *cascaded, vote-to-decide, decision enhancement, and hierarchical recognition systems* [19].

In the traditional hierarchical multiclassifier system, the classification stage (upper level) of the initial classification system generates new feature representations (inner pattern) for the original input pattern to be used as input to the further classification subsystem (lower level). This is not all, the further classification subsystem can be either a single classifier or a multiclassifier system, which takes the new generated feature as input for its classification stage to generate the results [19, 38, 71]. Figure 2.5 illustrates simple architecture of a HNN.

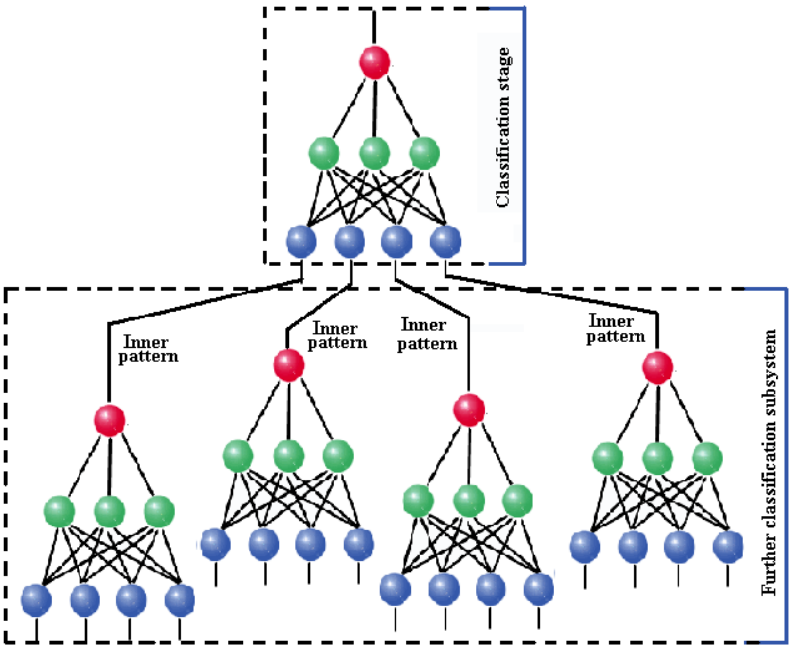


Figure 2.5 Simple Architecture of a HNN

A motivation behind implementing HNN in recognition is to simulate the hierarchy of the brain by using ANN. In such that simple NNs or neuron groups can be used as basic units to construct more complex NN, and the learning algorithm is selected at global and local levels. Such method is applied in both structural and functional design of ANN. The structurally hierarchical design concerns with assembling smaller networks into a big one. The sub network can be either homogeneous (like in ensemble learning) or heterogeneous (like in hybrid learning). Hence, a *network of networks* can be built and it can perform task better than any sub network, or, in some times, can accomplish a complex task that any of its sub networks can't do. The functionally hierarchical design concerns with dividing the learning task into smaller parts, which each of them is carried out by a single sub net. The result of learning will then be combined together to give an ultimate result. The functionally hierarchical design required investigation on how to decompose the tasks of pulling out principal component/feature, minimising interaction between subtasks, and dividing knowledge-base and learning parts. Comparing with classical ANN, hierarchical organisation has the following advantages [19, 38, 71, 119]:

- ***Robustness***: hierarchical organisation has excellent robustness due to the partial malfunction of sub net will not lead to serious failure of the entire system.
- ***Speed***: decomposition of learning task in hierarchical organisation structure can be divided to global learning and parallel local learning
- ***Complexity***: cooperation of function nets allows the network to perform more complex task than nonlinear mapping and dynamic evolution.

2.2.2 Decision Tree

Decision tree is considered a popular classifier, which does not necessitate any pre-knowledge or parameter setting. Decision tree is supervised learning approach. In that, by using a training data, we can develop a decision tree. The classification of hidden records can be easily predicted using decision tree. The decision tree algorithm represents sets of decisions, which are based on conditional probabilities, like Naive Bayes. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi-square Automatic Interaction Detection (CHAID) [54, 75, 113].

Actually, Decision tree have a hierarchical tree structure, which is used to classify classes by using a series of rules regarding the attributes of the class. While the classes are of qualitative type (categorical or binary, or ordinal), quantitative values (binary, nominal, ordinal) are possible types of variable that the attributes of the classes can be. Thus, the whole idea is that by given a data of attributes together with its classes to a decision tree, a sequence of rules (or series of questions) that can be used to recognize the class are to be produced [42, 75].

The decision trees are usually built from top to bottom. At each (non-terminal) node, a leaf decision is made pending a terminal node, also named leaf, is reached. Each leaf should contain a class label and each non-terminal node should contain a decisional question [75]**.**

The main problem, however, lies in building the tree classifier by using a training set, whose size is obviously limited. The overlapping between class areas, in case of noise, may complicate this problem [23, 25].

The tree is built by a process called splitting, which transforms a leaf into a decision-making node and swells the tree further down. In case of noise, the resulting splitting tree may be over fitted on the training set, so some pruning could be required [23,75].

As the pattern space is divided into decision areas by the decision boundaries, the tree-based classifiers may be seen a hierarchical way of describing the partition of input space. Usually, there are many possibilities of building a tree-structured classifier for the same classification problem, so a definitive search for the best one is not possible [25, 75].

In general, the tree is built with one considered feature (i.e. a component of the input vector). As for binary features, the choice is obvious, in continued ones, the problem is more difficult, especially if a small subset of features over simplifies the emerging tree [23, 25, 75].

2.2.3 Genetic Algorithms

Genetic Algorithms (GA) are considered as a class of adaptive stochastic optimization algorithms involving search and optimization used for searching heterogeneous environment of solution space. Prof. JohnHolland and his students developed GAs during the 1960s and 1970s at the University of Michigan. Fundamentally, GA can be thought as a method for reproduction computer programs and solutions to optimization or search problems by using simulated evolution. GA use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution. GA is considered as part of high performance evolutionary computing, which is a rapid area of interest for AI. Note that there are a large number of different types of GAs. [75, 117, 143].

Chromosomes are the representational units of solutions used in GA algorithm. GA starts its work with a set of chromosomes called population. Therefore, some solutions

from certain populations are to be taken and used in order to generate a new population (offspring). This step is an outlook to obtain an offspring that have properties better than its parents do (enhancing the chromosomes quality). This offspring would be considered as a parent, in the current step, and would be used to select solutions to form new solutions (or next offspring). It is important to note that selection is made according to solution fitness, which is based upon the principle, that the more suitable they are, the more likelihood they have to reproduce. This process is repeated awaiting certain conditions, e.g. or enhancement of the best solution fulfilled [39, 75, 117].

2.2.3.1 Outline of the Basic Genetic Algorithm

There are many forms for GA, the following one given by [39] is an example for it:

[Start] Generate random population of *n* chromosomes for the problem

[Fitness] Evaluate the fitness *f(x)* of each chromosome *x* in the population

[New Population] Create a new population by repeating the following steps until the new population is established

[Selection] Select two parent chromosomes from a population according to their fitness (the better the fitness, the bigger the chance to be selected)

[Crossover] The probability 'crosses over' the parents to form new offspring (children). If no crossover is performed, the offspring is the exact copy of parents.

[Mutation] With a mutation probability, mutate new offspring at each locus or position in chromosome.

[Accepting] Place new offspring in the new population.

[Replace] Use the newly generated population for a further run of the algorithm

[Test] If end condition is met, **stop** and return the best solution in the current population

[Loop] Go to step **[Fitness]**

2.2.4 Nearest Neighbour Classifier

Nearest Neighbour Classifier (NNC) rule gets continually high performance among the various methods of supervised statistical pattern recognition. NNC needs no priori assumptions regarding the distributions from which the training examples are drawn. NNC involves positive and negative cases training set. NNC can be defined as a process of classifying items by using the principle of closest training examples in a feature space. The training examples are to be addressed into multidimensional feature space, which is itself divided into areas by class labels of the training samples. Assignment of a point in the space to class c is done on condition that class c being the nearest class label among the k nearest training samples to that point, by using Euclidean distance as usual. While the training phase of the NNC classifying algorithm consists only of storing the feature vectors and class labels of the training samples, the classification phase is actually the computation of the test sample whose class is not known and is to be carried out by using the same features used in the training phase [25, 75].

The classification of a new sample is achieved by calculating the distance to the nearest training case. The principle of NNC is that the distances from a new vector to all stored vectors are computed and, then, the choice of the k closest samples, which belong to the new point, to the largely various classes within the set, is predicted (i.e. when the vote number k of a class is 1) [23, 43, 72].

The input data per se would determine the best choice of k. One of the well-known problems of data is noise, and usually the effect of noise on the classification process is reduced by larger values of k, but on the other hand, larger values of k make margins between classes less different. In addition, the use of parameter optimisation, cross-validation for example, would help in selecting a good k [23, 43, 72].

Noisy, unrelated features or inconsistent scales of features, though important, cause severe degrading of the accuracy of the k-NN algorithm. Many research efforts have been placed for selecting or scaling features to enhance classification. A predominantly well-liked method for optimising feature scaling uses evolutionary algorithms. Sharing

information of the training data with the training classes is another popular method for scaling features [25, 113, 117].

From the implementation point of view, the algorithm is easy. Yet its main disadvantage is that it is computationally intensive, particularly when the size of the training set grows. To reduce the number of distances already computed, numerous optimisations have been proposed. Some of them apply partitioning to the feature space, and then calculate distances within specific nearby volumes. A number of different types of Nearest Neighbour finding algorithms, like linear scan and Locality-Sensitive Hashing (LSH), make use of the kd_tree, Metric trees, and Balltrees data structures [42].

Finally, the NNC has a number of sturdy stability results. Since the quantity of data ever last, NNC is assured to produce an error rate better than twice of the minimum reachable error rate in producing the distribution of the data, which is known as the error rate given by Bayes. For a certain value of k, the K-nearest neighbour is assured to advance the Bayes error rate, where k increases as a function of the amount of data points [25, 42, 113].

2.2.5 Rule Induction

Rule induction is considered as one of the most machine learning significant techniques. Rule induction is one of the primary DM tools due to being hidden regularities in data are often expressed in terms of rules. Rule induction is the extract "of useful if-then rules from data based on statistical significance" [142].

The ease and transparent interpreting of rules, compared to say, a regression model or a trained ANN, make rule induction more attractive than other classification method. Usually rules are expressions of the form:

if (attribute − 1; value − 1) and (attribute − 2; value − 2) and .....and (attribute − n; value − n) then (decision; value):

As in other classification methods, the learning phase of classification rule can be defined as follows [42, 113]:

Given:

Set of training examples (instances in which classification is known)

Find:

Set of classification rules that can be used for prediction or classification of new unclassified instances, i.e., cases that have not been presented to the learner before.

Some rule induction based systems have rules that are more complex, in which the attributes may have reversal values of some other values or have a value that is a subset of the attribute domain. Data used to induce rules are usually presented in a table like form where examples are labels for rows and variables are to be labeled as attributes and a decision [35, 42].

The shortcomings of the language used to describe the data (Data Description Language-DDL) in addition to the language used to define the induced set of rules (Hypothesis Description Language-HDL) should be considered as an essential condition when defining a more formal definition of the classification rule-learning job. It should be noted here that the language bias refers to the restrictions imposed by the languages used for defining the format, and the range of both data and knowledge representation as well [35, 42].

As an example, consider the problem of binary classification to classify instances into two classes, namely positive and negative. The learning phase of a set of rules would define the class positive as follows [42, 113]:

Given:

- *DDL: that imposes a bias on the form of data*
- *Training examples: a set of classified instances described in the DDL*
- *HDL: that imposes a bias on the form of induced rules*
- *A coverage function: defines when an instance is covered by a rule*

Find:

- A hypothesis as a set of rules described in the HDL; that is

- *Consistent, i.e. does not cover any negative example*
- *Complete, i.e. covers all positive examples*

The definition, given above, distinguishes between the term examples, which usually refer to instances labelled by a class label, on the one hand, and instances, which bear no class label, on the other. Also in the definition above, another term appear, i.e. hypothesis, which is used to denote the output of learning.

Due to the hypothetical nature of induction, the output of inductive learning can be falsified by new evidence presented to the learner. A rule is an expression of the form *Head* ← *Body*, where *Body* describes the conditions under which the rule fires, and Head is typically a class label. In the simplified learning setting above, we learn rules for only one class, so the head of the rule is strictly speaking redundant. An *instance* is covered by such a rule if it satisfies the conditions in Body. An *example* can be either correctly covered (if it is covered and the class label in Head corresponds with the class label of the example), incorrectly covered (Head assigns a different class), or not covered [35, 42, 113].

Consistency and completeness are very strict conditions in this process of definition. They are unrealistic in learning from large datasets that may involve noise, (this term refers to random errors in the data, due to either incorrect class labels or errors in instance descriptions). It is also possible that the HDL is not expressive enough to allow a complete and consistent hypothesis, in which case the target class needs to be approximated. However, one more problem would develop from un-displaced targeted classes. To deal with these cases, the consistency and completeness requirements need to be relaxed and replaced by some other evaluation principle, such as sufficient reporting of positive examples, the high accuracy of prediction of a hypothesis or its super importance of the required, predefined threshold. These measures are to be used for a couple of reasons: as heuristics to manage definition of the rule and as measures for evaluating the quality of induced hypotheses [35, 43].

Noteworthy is the fact that the above description of the learning procedure assumes the learner has no previous knowledge of the problem and only learns from the examples. However, complicated learning problems normally need a considerable quantity of previous knowledge. The term 'background knowledge' is used to refer to a declarative previous knowledge. By using background knowledge, the learner is able to show the induced hypotheses in a more brief and accepted way [35, 89, 90, 113].

2.3 Data Model

DM is able to discover important implicit features or predict future evolutions by using modelling. The process of modelling can be defined as the act of building a model in one situation where the answer is known, and then applying it to another situation where it is not. Actually, the building of model as an approach is not modern. It has been used as a certain processing method of data long before computer was invented as a device, and before DM was invented as an information processing algorithms. As reported by many researchers, there is not much difference between the way of building models by computers and those followed by man to do so. The first step of building a model is known as the training phase. To do so, computers are to be loaded with lots of information about variant situations where an answer is known. Next, DM software runs through that data and distils the characteristics that should be applied to the model being built. Once it is built, the model can then be used in similar situations where the answer is not known [42, 43, 117].

Essentially, a system that is formally described as a model, may take the form of a mathematical expression or algorithm that provides a value based on input variables. DMM can be either descriptive or predictive. DM is conceived of as a process having two components: discovery, where meaningful patterns are detected in data and characterised formally as (descriptive models), and exploitations, where meaningful patterns are used to create useful applications (predictive models) [25, 86, 113].

Descriptive models are used for describing patterns in a given data, as well as for creating meaningful subgroups, such as a distribution group. On the other hand, predictive models are usually used to predict explicit results by making use of those

patterns built from known previous results. Taking a database of customers who have already acted against a certain offer, for example, a predictive model can be built to forecast which prospects are most likely be replied for the same offer in the future [36, 43, 111]. Predictive models can be so helpful in supporting decision making and solving complex planning tasks [121].

The ANN, decision tree, Naïve Bayes, or NNC algorithms are used usually to implement Classifiers (when a prediction relates to class membership), while ANN or decision trees are used to implement Regressors (when, the model predicts a number from a wide range of possible values) [36, 42, 43, 111].

2.3.1 Predictive Models

DMM is typically used for classifying or predicting, and so are called predictive models. Note that the term predictive does not mean that foretelling is implied, and as such, predictive models are *pattern classifiers*. The process of defining predictive models by using DM techniques and tools is known as predictive modelling. Since there are many ways to approach the problem of exploiting patterns in data, predictive models can be created by using different approaches. Patterns can be processed by using many techniques, including polynomial regression, Knowledge Base Expert System (KBES), and ANN [25, 43].

A predictive model seeks to forecast the value of a particular attribute to judge the identity or the belonging of an element to a certain set. For example [111], predictive models can predict:

- A long-distance customer's likelihood of switching to a competitor
- An insurance claim's likelihood of being fraudulent
- A patient's susceptibility to a certain disease
- The likelihood someone will place a catalogue order
- The revenue a customer will generate during the next year

The first four examples illustrated above demonstrate a particular type of prediction models, which are called classification models. These classification models are usually

used for predicting class membership; sometimes it is called classifiers. The class prediction in the first four examples might as follows be respectively:

- trusty versus untrustworthy,
- lawful versus falsified,
- vulnerable versus imprecise or unsusceptible, and
- Purchaser versus seller

Through the first four examples, it is obvious that in each case the prediction classes typically contain limited values. Another type of prediction model, which is called regression model or regressor in short, is illustrated by the fifth example [25, 26, 75].

2.3.2 Descriptive Models

The class of descriptive models encompasses important model types: clustering, association, and feature extraction [42, 43, 47, 86, 113].

"Clustering (also referred to as segmentation) collects together similar people, things, or events into groups called clusters" [111]. The importance of clustering is that *Clusters help reduce data complexity*. For example, it is obviously easier to design a different marketing plan for each cluster of few-targeted customers than to design a specific marketing plan for each of multi million individual customers [42, 43, 113].

As an example, the trade sector called Market Basket Analysis uses Association models. This type of models involve finding out of likeness, i.e. how often two or more things occur together. The result of this testing is in terms of creating rules like*: when people buy data structure books, they also buy Java programming language books 55 percent of the time*. Dealers usually use these rules to map shelf situation and promotional discount [111].

Researchers reported that data, which is the topic of analysis and processing in DM in many applications, *is multidimensional*, and is presented by a number of features. Many learning algorithms show pertinence, (called the curse of dimensionality), and denote the drastic rise in computational complexity and classification error. The curse of dimensionality usually comes up with data of large numbers of dimensions. Hence, the

attempt to reduce the dimensionality of the feature space before classification is undertaken. The attempt to face this curse is known as *Feature Extraction (FE) which is a one-dimension reduction technique*. *In abbreviation, FE extracts a subset of new features from the original feature set by means of some functional mapping and keeping as much information in the data as possible* [101, 74].

Approximately, all predictive models can be used descriptively. The relationship between descriptive and predictive models is a one-direction relationship. A descriptive model is not predictive. The converse does not hold: predictive models are often descriptive. Actually, the ability of predictive model to give descriptive aspect is sometimes more important than its ability to predict. For example, suppose a DM model has been built to predict the likelihood of a particular disease. The developer of this DMM might be more appealing, in exploring that disease- related features or their scarcity than using the DM model itself to predict if a new patient got disease. It is important to notice that the aim of descriptive models is not the forecast of a target value, but focuses more on the discovery of issues such as interconnectedness, relations, and the fundamental structure of the data [111, 113].

2.4 Difficulties in Data Mining

Traditional DM problems are solved by using classical optimization techniques, among which convex optimization has occupied the heart stage. However, new problems appear continually in DM community. Several of these newly appeared problems are more complex than traditional ones, which are formulated as nonconvex problems [138].

Researchers outline some of the current primary research and application challenges for DM. This list is by no means exhaustive, but it does give a feel for the types of problems that DM practitioners wrestle with everyday [42, 43, 45, 136]:

- Large Databases: Databases with hundreds of fields and tables, millions of records and of a multigigabyte size are common, and terabyte databases are emerging. Methods for dealing with large data volumes need algorithms, which are more

efficient, sampling, approximating, and perhaps parallel processing with a massive capability.

- High dimensionality: There can be a large number of fields, such as attributes and variables, as well as a large number of records in the database, thus elevating the dimensionality of the problem. A high-dimensional data set creates problems in terms of increasing the size of the search space for model induction, to the extent that the combination of the number of fields and records would be unmanageable at all. In addition, it increases the chances for a DM algorithm to find spurious patterns that are not generally valid. Solutions to this problem need to involve methods of reducing the effective dimensionality of the problem, and the use of prior knowledge to identify irrelevant variables.
- Over fitting: While searching for the best parameters for a particular model using limited set of data, DM algorithm can model any specific noise of the data set and not only general patterns in data. This model may result in poor performance on test data. Solutions suggested for solving this problem include cross-validation, regularisation, and other sophisticated statistical strategies.
- Evaluation of statistical importance: This problem (which is related to over fitting) usually happens when the system is searching for many possible models. Consider, for example, if a system tests models at a significance level of 0.005, then on average N/5000 of these models will be accepted as significant with entirely noisy data. This point is frequently missed by many initial attempts at DM. One way to deal with this problem is to use methods that adjust the test statistics as a function of the search.
- Shifting data and knowledge: Rapidly changing (non-stationary) data can make previously discovered patterns invalid. In addition, the variables measured in a given application database can be modified, deleted, or augmented with new measurements over time. Solutions proposed for this problem comprise incremental methods for updating the patterns and treating change as an opportunity for discovery by using it to cue the search for patterns of change only.

- Omitted and noisy data: This problem is especially acute in business databases. Important attributes can be missing if the database was not designed with discovery in mind. Solutions suggested for this problem embrace sophisticated statistical strategies to classify hidden variables and dependencies.
- Complex relationships between fields: The hierarchically structured attributes or values (relations between attributes) and more sophisticated methods for representing knowledge related to the contents of a database require effective algorithms that can use such information. Historically, DM algorithms have been developed for simple attribute-value records, although new techniques for deriving relations between variables are being developed.
- Comprehensibility of patterns: In many applications, it is important to make discoveries more comprehensible by humans. The most likely solutions for this problem are graphic representations, rule structuring, natural language generation, and techniques for the visualisation of data and knowledge. Strategies for rule-refinement can be used to address a related problem: The revealed knowledge might be implicitly or explicitly redundant.
- User interface and previous knowledge: Many current DM kits and schemes are not actually interactive, and are not able to incorporate conveniently prior knowledge about a problem in easy ways. The use of domain knowledge is important in all steps of the DM process. Bayesian approaches use prior probabilities over data and distribution as one form of encoding prior knowledge. Others employ deductive database (knowledge in AI) capabilities to discover knowledge that is then used to guide the DM search.
- Incorporation with other systems: It is not very useful for a discovery system to be stand-alone. Classic integration issues take in integration with a database management system (DBMS), through a query interface for example, integration with spreadsheets and visualisation tools, and accommodation of real-time sensor readings.

2.5 Uncertainty and Data Mining

What is the cause of the difficulties listed in Section 2.4 above? To answer this question we will start from the atomic process of DM's algorithms, i.e. classification.

Classification is a result of a decision-making process that can be defined as a choice among alternatives, like actions, hypotheses, and locations. The classification process typically uses either a statistical or a structural approach [75, 109].

In the statistical approach, each class is defined by weighted sum of values of isolated set of features that are relevant to the task domain. Each class is then defined by a scoring function, which is used as a mechanism of discovering the identity of the data. Scoring function comes in the form $c_1t_1+c_2t_2+c_3t_3+$ ..., where each t_i corresponds to a value of a relevant parameter, and each c represents the weight to be attached to the corresponding t. In the latter approach, each class is defined as a structure composed of isolated set of features that are relevant to the task domain [75, 109].

In the structural approach, the task of classification is realised by labelling the different sets of data if the existing set maintained a classified pattern known as a training set. The learning approach is regarded as a supervised learning as the classification process develops its own model linked to an already labelled set of data (patterns). Learning may also be unsupervised if the classification system is not supplied with a priori labelling of patterns, as an alternative, it sets up by itself, the classes on basis of the statistical habitual of the patterns [75, 109].

Undoubtedly, each approach has pros and cons. It has been reported that the statistical approach is often more efficient, and flexible, than the structural approach [37, 41]. Nevertheless, it is difficult to construct good class definitions by hand, especially in domains that rapidly change or are not well understood. Thus, the idea of producing a classification program that can evolve its own class definition is appealing. This idea is known as the concept of learning, or induction, which is indeed a type of inference in AI literature [75, 109].

The problem of uncertainty is viewed common in the classification process. The existence of more than one task domain is possible, in that each domain has its own scoring function to define or to assign different mysterious elements. In that case, defining of different classes names to the same mysterious element (each task domain has a class's name), would ensue and would create the problem of uncertainty of the identity (classification) of this element. Things could get worse when a single mysterious element comes with multi syntax, and evolves multi-meanings (or multi-semantics), that producing a whole case of uncertainty, involving misunderstanding in terms of computer's interpretation of the data.

This form of uncertainty is noted as multiplicity, which is found in the DM difficulties (all but the first one). Thus, since the multiplicity phenomenon causes the problem of vagueness or uncertainty [6] in DM as well as other applications, the way to improve the effectiveness of classification in DM is achieved by handling this DM's problem as agreed upon by [5].

2.6 Approaches of Handling Uncertainty in DM

Disclose and handling of uncertainty is an essential issue in DM research. The common use approaches of classification do not consider this aspect. Generally, fuzzy set, ANN, and GA are commonly applied in the DM to handle different challenges in it. Each of them contributes a different methodology for addressing problems in its domain, which is done in a cooperative, rather than a competitive, manner. Note that fuzzy sets give a normal framework for the process in dealing with uncertainty, ANNs are usually used for classification and rule generation, and GAs are concerned with various optimisation and searching processes [5, 29, 91].

Let us first describe the function and importance along with the mixing of each of these three approaches used by different systems developed for handling the different functional features of DM. It may be mentioned that there is no universally best DM method. Indeed, deciding particular computing tool(s) or various combinations with conventional methods (like fuzzy decision trees, fuzzy C-Means and fuzzy rules approaches that reported in literatures for dealing with uncertainty representation in DM)

is completely reliant on the nature of application and necessitates human dealings to settle on the aptness of such an approach [5, 54, 91].

2.6.1 Fuzzy Sets Approach

As stated earlier in this thesis, fuzzy sets are used for modelling and handling of uncertainty in addition to its use it in the modelling of qualitative knowledge. Fuzzy logic is able to support, to a reasonable scope, the natural form of human type reasoning.

DM is mainly concerned with recognising interesting patterns and describing them in a succinct and meaningful style. A careful filtering of data, qualitative opinions, and adjusting of commonsensical rules are to be represented using fuzzy models in order to set up meaningful and constructive relationships among the variables of a system. Note that there is a significant constituent of human collaboration that is normally required to knowledge representation, manipulation, and processing activities in spite of of the fact that there is an increasing in the adaptively property of knowledge discovery systems [5, 91].

Fuzzy sets are naturally tending toward functioning with linguistic domain of knowledge and generating more interpretable explanations. There is an indisputable increasing in the role of fuzzy set in the area of DM. Fuzzy set theory has been used to implement different DM systems. In DM, the analysis of data in real world frequently demands dealing with different types of data classes and numeric attributes simultaneously. The roles of fuzzy sets in the DM's techniques are found in [54, 91]:

- ***Clustering***: As DM aims at pulling out useful information in the form of new relationships, patterns, or clusters from huge volumes of data for supporting decision-making by a user, fuzzy sets support an accurate searching of data that expressed using linguistic terms. Furthermore, fuzzy sets facilitate discovering of dependencies among qualitative and semi-qualitative format of the data. This helps avoid searching for worthless or insignificant patterns in a database, since there are too many attributes to be considered and can ensue complicated errors.

- ***Association Rules***: One of important DM's research techniques is the discovery of association rules, which illustrate an interesting relationship among different attributes of given data. While Boolean association considers binary attributes, the widespread association considers hierarchically related attributes, and a quantitative association considers attributes that may exist in either quantitative or categorical values. For example, a crisp classification supposes that a kid belongs to its ancestor with crisp degree value of support and confidence equals one. In fuzzy classification, the partial belonging of an item is taken into account while computing this degree.

- ***Functional Dependencies***: The term functional dependency (FD) means that factor X [partially] determines the level of factor Y. FDs allow the intensive expression of real world properties, which are valid on a certain database. Regression is utilised to analyse the relation between two continuous variables, and it is the most suitable method for studying FDs between factors. The inference based on FDs among variables in database relations implements fuzzy logic, which generalises both imprecise and precise inference. Inference analysis is carried out using a particular abstract model, which sustains essential links to classical, imprecise, and fuzzy relational models in database. These links enlarge the value of the inference formalism in realistic applications like knowledge discovery and database security.

- ***Summarisation***: Summary discovery gives the user complete information for seizing the core representation from a huge amount of information in a database. Usually, fuzzy sets are used in an interactive top-down activity of data summarisation, which employs fuzzy IS-A hierarchies as domain knowledge. Representational structure of a database summary including fuzzy concepts comes as generalised combination of attributes in which the discovery process deals with more precise database summaries. The summarisation of large sets of data linguistic is derived as linguistically quantified propositions and corresponds to the favourite criterion coped in DM process. Data summarisation system involves a summariser (‘*old’* for example), a quantity in agreement (‘*most’* for example), and the truth

validity (say 0.83). The most interesting linguistic summaries are usually central and human-reliable concepts, which consist of compound grouping of attributes. Practically, linguistic summaries cannot be created automatically and it needs human support.

2.6.1.1 Fuzzy Logic System

Essentially, the computer, in its both hardware and software components, is a machine that works according to binary logic. Thus, the computer itself is classified as a crisp machine. Fuzzy logic, which is depicted in Figure 2.6, and its continual improvement are used during the internal process, so that the qualified (or fuzzy) data are to be quantified (or pre-processed) so they would be suited for further computation by computer-the crisp machine.

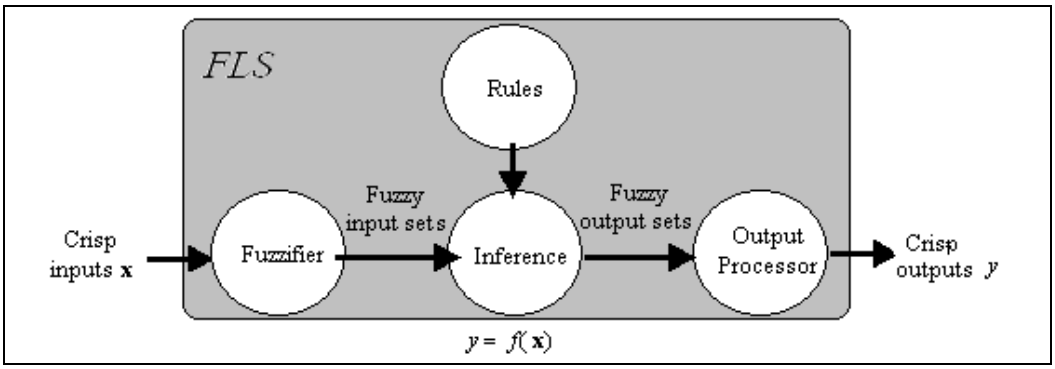


Figure 2.6 Architecture of Fuzzy Logic System

Logical inference is used as one of AI approaches to solve problems, when the data of these problems are vague or incomplete that affects defectively the calculations of problem's conclusion. The inference process or approach is performed to obtain concluding, which results from a series of principles that each one leads to the next depending on its logical crisp value (comes from crisp semantic description of a proposition) [75, 94, 109]. For non-crisp proposition (fuzzy proposition), a process of calculating fuzzy membership value using membership function, which is called fuzzification, should be performed firstly to convert the literals of fuzzy data to numerical values, as a required prior step for processing these fuzzy data numerically by

computer (i.e. converting the qualification terms into quantification values or membership values) [64, 82, 84].

The traditional FLS uses fuzzy membership value to perform the required inference. FLS consists of four interconnected portions [64, 82, 84, 99, 135]:

- Rules, which are the spirit of FLS, are coupled with membership functions in which each rule can be thought of as a subsystem. Rules do nothing unless inputs are applied to them. Traditional FLS implements fuzzification rules for either type-1 fuzzy set or type-2 fuzzy set.
- Inference Engine (IE) part, which maps each rule's fuzzy input sets into each rule's fuzzy output set. The mathematics of how the inference engine does this is very nonlinear.
- The fuzzifier component, which quantifies qualities, so that they can be treated as fuzzy sets.
- The output processor converts a fuzzy set into a number required in many of the applications of an FLS where numerical outputs are required to make informed decisions.

The function of FLS is to map the data from crisp inputs to crisp outputs, and is expressed as y=f(x), which can be elaborated as following:

1. Getting of single, or a multiple, of fuzzy measurement of conditions existing in some system aimed to be analysed or controlled.

2. Processing all these inputs according to human based, fuzzy 'If-Then' rules, which can be expressed in plain language words, in combination with traditional non-fuzzy processing.

3. Averaging and weighting the resulting outputs from all the individual rules into one single output decision or signal that decides what to do or tells a controlled system what to do. The output signal eventually arrived is a precise appearing, defuzzified, *crisp* value.

This kind of FLS is also identified as a fuzzy system, fuzzy controller, fuzzy model, or fuzzy expert system. It is widely used in many engineering applications of fuzzy logic, such as fuzzy logic controllers and signal processors [84, 99].

2.6.2 Neural Networks Approach

The use of ANN facilitates both integrating parallelism and tackling the problems of optimisation exist in DM. ANN models are typically appropriate in the environments where data are so heavy. Chapter Five gives technical description of ANN. The main contribution of ANN toward DM tasks is found in [91, 70, 106, 122]:

- ***Rule Extraction***: Usually, the main input to an algorithm of connectionist rule extraction is a representation of the trained ANN. This input comes in terms of ANN's neurons and links. The automatically derive of the rules is achieved by using one or more hidden and output neurons. Afterwards, combination and simplifying of these rules could be applied to get further comprehensible rule set (form an ANN model with set of outputs). These rules can give new views into the application domain. Pruning algorithm may used to remove the disused connections of the network. Typically, a network model is firstly trained to reach the necessary accuracy rate, where the weights and activation values of the hidden neurons in the network are justified. Researchers define quantitative measures to evaluate the achievement of the generated rules relates to the favourite criterion or integrity of fit chosen for the rules. These measures are: *Accuracy, User's accuracy, Kappa*, *Fidelity, Confusion*, *Coverage, Rulebase size*, *Computational complexity*, and *Confidence*.

- ***Clustering and Self-Organisation***: The organisation and retrieval of documents, which haven't unique structure, from annals are considered among the big challenges of DM. There are many examples for the using of self organisation ANN for clustering data are:

  - The using of a huge self-organising map (SOM) to partition multiple million documents

- The involving of SOM with partitioning the data in stepwise methodology
- The hierarchical clustering of SOMs, based on a widen factor, which is independent of the dimensionality of the data.
- Designing a DM method for clustering a set of pathological data by combining SOM with data visualisation
- Combining SOM and Sammon's nonlinear mapping to minimise the dimension of data representation for visualisation reasons

2.6.3 Neuro-Fuzzy Computing Approach

FLS may be developed using ANN, which results *neuro-fuzzy approach*. Neuro-fuzzy computation comprises an astute combination of the standard advantages of ANN (like enormous parallelism, learning, and robustness) and fuzzy approaches in data-rich milieu into one system, thus enabling the construct an intelligent decision-making systems. The rule generation feature of ANN is used to pull out rules that are more normal from fuzzy ANN. The features of data presented to fuzzy ANN are that the possibility of the data being incomplete, as well being in quantitative, linguistic, or a mixture of them. The input vector of ANN contains membership values to the linguistic properties corresponding to each input feature [8, 20, 54, 70, 91].

This approach is currently considered as one of the major areas of interest since it gets the benefits of both ANN and FLS. The merge of common features of both ANN's and FLS's leads to remove their individual disadvantages. Thus, while ANNs tolerate for noisy data, FLS tolerates the imprecision of data. Furthermore, the learning capability of an ANN provides a good technique by which the knowledge of an expert system can be automatically fine-tuned and automatically produce both additional fuzzy rules and membership functions to meet certain specifications. This learning capability reduces both the design time and cost. On the other hand, fuzzy logic approach perhaps improves the generalisation capability of an ANN by providing an output that is more reliable when extrapolation is required ahead of the restrictions of the training data [48, 81].

Researchers investigated a number of different architectures of neuro-fuzzy systems with a view to fixing an ideal architecture of a neuro-fuzzy system. These architectures is used in many applications, especially in the controlling type process. The common features of ANN and FLS include [81, 128]:

- The ability to handle data with ambiguity,
- Scattered representation of knowledge,
- Model-free estimation and vagueness

2.7 Summary

DM aims at creating data models through methods of research on data, which is deemed to store in data warehouses with a view to defining a model of predictive or descriptive type. The resulting models are the common attributes among the investigated data. The process of defining or building patterns from this data is to be performed by a variety of methods, namely classification, clustering and association rules. There are many algorithms for these methods, including ANN, GA and NNC. While it is an evolving technology, the DM's promising insights are often too compelling to ignore. DM is usually used in applications, such as business performance management, business intelligence, loyalty card, discovery science, bioinformatics, cheminformatics, and many other applications.

Although, some of DM's theoretical aspects need to be developed, DM adds a new category of data processing type to classical ones, such as sorting and searching data, data compression-decompression, data encryption-decryption. This new data processing type can be described as *classification-reclassification of data*.

Researches have already highlighted difficulties in DM (listed in Section 2.4 above). Careful studying of these problems implies that they fall into three categories: missing & changing data, attributes of data (i.e. large size and a high degree of complexity), and the accuracy & existence of multiple models of the same type of data. All but the first challenge remain to be well answered. It is obvious that some traditional DM classification approaches (like decision tree and rule induction) have been applied with

fuzzy logic to produce new approaches that are able to handle the uncertainty in DM. The rest of DM techniques reported in this chapter, namely ANN, GA, and NNC, are already defined as techniques to handle simple uncertainty. According to these approaches, each data value can be assigned to one or more categories with an attached degree of belief [8, 20, 54, 91].

The analysing and understanding of uncertainty, as the shared reason among DM difficulties, along with the methods and tools used to handle uncertainty in DM have lead this thesis to investigate an effective classification tool that is able to handle uncertainty wherever and whenever it exists in data.

Although no type-2 FLS has been defined to deal with the uncertainty in DM, the shortcoming of type-2 FL reported in Section 1.3 earlier was the motivation for seeking for an enhanced version of FL to be applied to an existing computation methodology for developing a new computation methodology that is able to deal with the complex ambiguity. This suggestion follows one of the approaches used to handle the uncertainty in DM namely neuro-fuzzy approach, which has been reported in this chapter.

Referring to studies in Cognitive Physiology, cognitive approach defines mental structure that describes the relationship between meaning and memory that is recognition [119]. This mental structure groups elements to clusters and then distinguishes each individual element according to its cluster. What supports this assumption is the fact that clustering of data helps reducing the complexity of the data (see Section 2.3.1). Actually, what makes recognition more difficult is that the element itself may come with different shapes and fall in different clusters. This means that a further recognition is required as well as cluster recognition is required.

Thus, we suggest a new complex classification approach that encompasses clustering of elements, as a super stage, and individual recognition of elements inside each cluster as a sub stage. Such computation approach of complex classification, would imitate the human approach of handling complex ambiguity of the data (which has been explained in Section 1.2). The suggested approach requires defining of computation methodology

that is based on a theoretical description of handling and modelling the complex ambiguity.

To implement the cognitive approach of recognition (the mental structure), which is described by cognitive physiology, we need to map each stage of this approach to an existing modelling technique used for handling and modelling the uncertainty. The super stage of the mental structure recognition approach can be mapped to possible-world modelling theory, and the sub stage can be mapped to fuzzy membership value set. Both of these techniques are to be explained in the next chapter.

Chapter Three

Uncertainty Modelling using Logics and Possible-world

3.1 Introduction

As pointed out in the previous chapter, uncertainty is the problem that negatively affects the accuracy of the classification-based processes. Expert, who maintains his own knowledge base, usually relies on common sense when coming to solve problems. Expert also uses vague and ambiguous terms, such as approximately, almost and partly. This type of description gives some degree of believability to a scenario and addresses the problem of uncertainty in data or decisions. Effective computerised classification-based systems must be able to handle uncertainty wherever and whenever it exists in the data. Classic two-valued logic, classic multi-valued logic, and fuzzy logic involve and cope with such type of uncertainty [75, 94, 109].

Sometimes there is a case that multiple knowledge bases are required to capture the different viewpoints of multiple actors concerned in solving a single problem. To do reasoning in such case, we would need a method for maintaining several parallel viewpoints spaces or domains, each of which would correspond to one actor's viewpoint. Upon their current *set of viewpoints and understanding of vague and ambiguous terms*, experts formulate valid conclusions using reasoning with uncertainty. Such type of uncertainty is dealt by using modal logic [29, 75, 109, 119].

Yet as more information becomes available, these viewpoints and understanding, along with their consequences, may vary. As experts use their common sense to solve problems with uncertain information, it would be difficult to give the computer the same level of such ability.

Thus, among the types of logics used to model and handle the uncertainty, which are reported in Section 1.2, this thesis focuses on both fuzzy logic (fuzzy logic is reasoning

using fuzzy sets), as a good approach to model certain type of uncertainty, and modal logic that provides possible-world as an approach to deal with conception in multiple domains. This is because these two approaches are going to be used to define a new approach for modelling and handling complex ambiguity (of many-to-many relationship type), which encompasses both vagueness and multiple viewpoints at the same time, namely Relative Fuzzy approach.

Generally, logics use an essential element, which is called proposition. A proposition is defined as "a statement in which something is affirmed or denied, so that it can therefore be significantly characterised as either true or false" [125]. "Propositions are the basic building blocks of any theory of logic, and can be described as the essential conceptual portion of understanding" [59]. Brain [16] formalised the proposition of knowledge representation as implying the supposition that knowledge will be propositionally symbolised. To keep away from ontological allusions and arguments, the word *sentence* is usually used instead of the word *proposition* with a view to denoting those strings of symbols of carry truth-value, i.e. being either true or false pending the interpretation.

Luger [75] gives full details about using proposition in mathematical logic. In short, propositions are statements that contain *atomic formulas*, or *symbols of proposition*, the five *logical connective* ($\wedge$,$\vee$,$\neg$,$\rightarrow$, and $\equiv$), *symbols of grouping* (( ), and [ ]), and *quantifiers* (there exist $\exists$ and for all $\forall$ quantifiers). Being a statement, a proposition comprises two components: syntax, and semantics. For example, the proposition symbol P may denote the statement 'the sky is blue'. A proposition may be either true or false. The truth-value assignment to propositional sentences is called *interpretation*, which is an assertion about their truth in some possible world. Formally, an interpretation is a mapping (or function) from the propositional symbols into the set {True, False}. Thus the proposition P, in the given example, may assign the truth-value (semantic value) True.

As it known, Logic is reasoning using membership set. For example, binary logic is reasoning using {0,1} membership value set, and fuzzy logic is reasoning using [0, 1].

On the other hand, operations on any membership value sets are alike to those of standard logic {AND, OR, NOT} but are defined in different way. Briefly, logic encompasses two components, namely semantic values and reasoning operations.

The mathematical illustration of a logic requires quantification of its qualified-semantic values (i.e. the use of numbers to represent logical values), and applying logical operations to these numbers. It should be clear that:

- 'AND' operation of two qualified semantic values means the selection of the lowest value between them,
- 'OR' operation of two qualified semantic values is the selection of the highest value between them, and
- 'NOT ' operation of a quantified semantic value is the absolute difference between it and the maximum possible quantified semantic value (which is 1).

Set theory is usually used to illustrate the operations of logic along its numerical representation values. It should be noted that 'AND' logical operation is opposite to 'intersection' set operation, 'OR' logical operation is opposite to 'union' set operation, and 'NOT ' is opposite to 'complement' set operation.

This chapter presents literature notations about fuzzy set along with the possible-world and its backgrounds of modal logic-the predefined approaches of modelling and handling the uncertainty problem. Strained conclusions are to be given in the end of the chapter.

3.2 Fuzzy Sets

Quantification of qualifications is the key for modelling the human description of real world, either crisp or fuzzy. This quantification is achieved by using finite set of numbers. The importance of set is due to its ability to organise, summarise, and generalise knowledge about elements. Sets are almost used unconsciously when talking about, for example, a set of even numbers, positive temperatures, fruits, personal computers, and the like [59, 75, 109].

Thus, for the crisp or dichotomous principle, a value of 1 could be assigned to quantities belonging fully to a membership (denoted as $\in$) of items, which in turn belong to a set, and 0 to those which do not (denoted as $\notin$). Thus the set of {0,1} is used as a crisp membership value set. In classical set theory, crisp set A of X is defined as a function μ_A (x) called characteristic function of A [75, 94, 137]:

$$\mu_A: X \rightarrow \{0,1\} \qquad (3.1)$$

where μ_A (x)=1 if $x \in A$ and μ_A (x) =0 if $x \notin A$.

Being naturally attractive, sets introduce a fundamental notion of dichotomy. In its essence, any process of dichotomisation imposes a binary, an all or none classification decision: either accept or reject an element (object) as belonging to a given set (group or category). For instance, consider set A in universe X, as depicted in Figure 3.1 [75, 83, 94, 102, 109].

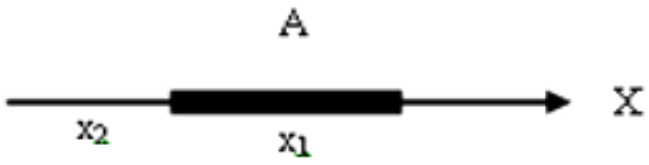


Figure 3.1 Sets and the Resulting Concept of Dichotomy [102]

In Figure 3.1, the element x_1, belongs to set A, whereas x_2 does not, that is, $x_1 \in A$ and $x_2 \notin A$. In general, if we denote the accepted decision by 1 and the rejected decision by 0, for short, then we may express the classification decision through a characteristic (membership) function μ_A (x), illustrated in Figure 3.2, by:

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

Clearly, the empty set $\varnothing$ has a null characteristic function, $\varnothing(x) = 0$, for all x in X, and the universe X has a unity characteristic function $\mu_X(x) = 1$, $\forall x \in X$. Also, for a set A= {a} with only one element a (that is, a singleton) $\mu_A(x) = 1$ if x=a, and 0 otherwise [75, 83, 102].

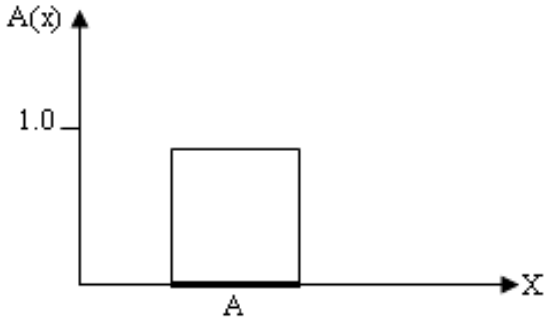


Figure 3.2 Example of a Characteristic Function [102]

It becomes harder and ultimately unachievable to formulate an accurate assertion about system's behaviour as system's complexity evolving towards growth, reaching at a point of difficulty where the method of fuzzy logic is the only way to solve this problem. The underlying concept of fuzzy sets is to admit intermediate values of class membership in [0, 1]. Fuzzy logic's breakthrough is to identify that there are alternative representations in between these two limits. This allows for an enhanced and more realistic interpretation structure by attaching statements with a partial quantification of belongingness [83, 137].

Fuzzy logic makes use of human common sense in that in reality most descriptions of real-world objects do not possess well-defined boundaries. A fuzzy set can be defined as a collection of things that are unable to be precisely defined. Consider, for instance, notions like *high* temperature, where is the separating border between high temperature and low temperature? Is a thirty centigrade a high temperature? How about forty centigrade? What about 39.9 centigrade? The assessment is in the sense of the speaker. Same thing is applied for other notions like *new* houses, *medium* pressure, *small* town, *low* speed, *big* car, and so forth, in which the italicised words identify the sources of fuzziness [75, 94, 109].

When analysis is a human-base, it must be a method to assign certain rational value to sensitive judgments of a fuzzy set's individual elements, a translation from human fuzziness to numbers that can be used by a computer. Numerically, whether an object

belongs to such a category is a matter of degree expressed, for example, by a real number in the unit interval [0, 1]. This is done by assigning judgment of situations a value from 0.0 up to 1.0. Consider for example, "how hot the room is" the human might rate it at 0.2 if the temperature were underneath freezing, and the human might tempo the room at 0.9, or even 1.0, if it is a summer hot day with a switching off air conditioner. As it is closer, to number 1, it is a higher grade of the object membership in a particular category, and vice-versa. Such fuzzy evaluation, with zero at the bottom of the scale and 1.0 at the top, gives a regulation for analysis the fuzzy logic method's rules. The outcome appear to result well for complex systems where the only base from which to carry on is the human experience, which is positively better than do nothing, where it would be if loath to proceed with fuzzy rules [83, 102].

The fuzzy membership values express the degrees to which each object is compatible to the properties or features distinctive of the collection. Fuzzy set has been defined as a set of elements with membership values between 0 (complete exclusion) and 1 (complete membership) [14, 102]. Zadeh defined Fuzzy sets formally as follows [70]:

> ***Definition:*** A fuzzy set A of the universe of discourse X is characterised by a membership function $\mu A: X \rightarrow [0, 1]$ mapping each point in X onto the real interval [0, 1], with $\mu A(x)$ representing "grade of membership" of x in A. A fuzzy set A can be mathematically represented as $A=\{(x,\mu A(x)) \mid 0 \leq \mu A(x) \leq 1 \; \forall x \in X\}$.

The previous definition is now referred to as type-1 fuzzy set. Clearly, a fuzzy set is a generalisation of the concept of a crisp set whose membership function takes on only two values {0, 1} [102, 137].

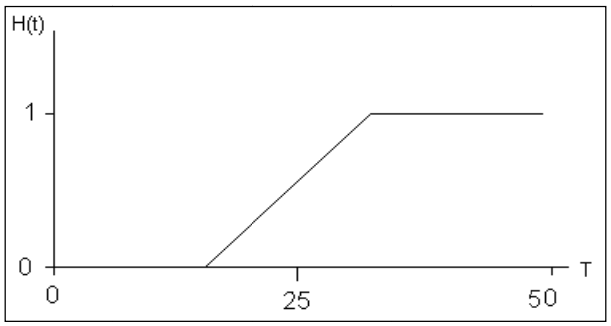


Figure 3.3 Example of Fuzzy Membership Function [102]

The value of $\mu_A(x)$ describes a certain degree of membership of x in A. For instance, consider the concept of high temperature in, say, an environmental context with temperatures distributed in the interval [0, 50] defined in °C. Clearly 0°C is not understood as a high temperature value, and we may assign a null value to express its degree of compatibility to the high temperature concept. In other words, the membership degree of 0°C in the class of high temperatures is zero. Likewise, 30°C and above are certainly high temperatures, and we may assign a value of 1 to express a full degree of compatibility to the concept. Therefore, temperature values in the range [30, 50] have a membership value of 1 in the class of high temperatures. The partial quantification of belongingness for the remaining temperature values through their membership values can be pursued as shown in Figure 3.3, which is actually a membership function H:T→[0, 1] characterising the fuzzy set H of high temperatures in T=[0, 50] universe.

Sum notation is used for describing the fuzzy set. This allows only the enumeration of elements of set X with nonzero grades of membership in the fuzzy set. For instance, if $X=\{x_1,x_2\ldots\ldots,x_n\}$, then the fuzzy set $A=\{(a_i/x_i)|x_i\in X\}$, where $a_i = \mu_A(x_i)$, i=1,….,n, may be described as [14, 102] :

$$A=a_1/x_1+a_2/x_2+ \ldots + a_n/x_n = \sum_{i=1}^{n} a_i / x_i \qquad (3.2)$$

The summation given by this notation should not be confused with the standard algebraic summation. The purpose behind using the summation symbol in the above expression is to symbolise the set of the ordered pairs. Also, note that when there is only

one point x in a universe for which the membership degree is not null, i.e. when A = {a/x}, than we get a fuzzy singleton. Hence, and using the same sense we may also interpret the summation symbol as a union of singletons. Equivalently, one can summarise A as a vector, meaning that A= $\{a_1, a_2, ., a_n\}$. When the universe X is continuous, we use, to represent a fuzzy set, the following expression [102]:

$$A = \int_x a / x, \quad \text{.....where a= } \mu_A(x). \qquad (3.3)$$

As stated earlier, sets and fuzzy sets can be defined in discrete and finite universes. Consider X = $\{x_1, x_2 \ldots x_n\}$, as an example, it has an interesting and transparent geometrical interpretation. Obviously, any subset of superset X is a member of the power set P(X) of X, which equivalently 2^n subsets of universal set X. Thus, we may correlate an n-dimensional vector with each of the 2^n elements of P(X), which include either one entry or none, which is a corner of the n-dimensional unit hypercube $\{0, 1\}^n$. For instance, if n=2, then P(X) have four elements: P(X) = $\{\varnothing, \{x_1\}, \{x_2\}, \{x_1, x2\}\}$[59, 41]. By using vector notation, we can define P(X) = $\{\lfloor 0\ 0 \rfloor, \lfloor 1\ 0 \rfloor, \lfloor 0\ 1 \rfloor, \lfloor 1\ 1 \rfloor\}$, whose elements are cornered in a square unit, as Figure 3.4(a) shows [63, 94].

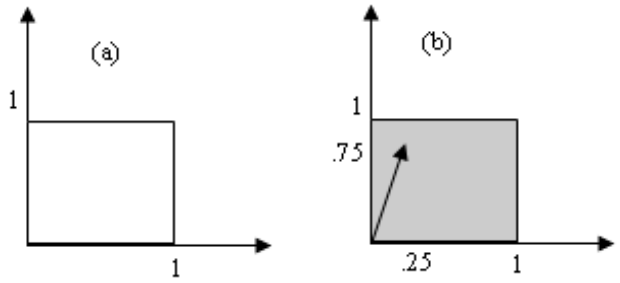


Figure 3.4 Geometric Interpretation of (a) Sets and (b) Fuzzy Sets [102]

As much as ordinary set, fuzzy sets form also n-dimensional vectors with entries in the [0, 1] range, except that the number of different vectors is not finite. Using the geometric interpretation, fuzzy sets occupy the entire n-dimensional unit cube, together with, specially, its corners. For n=2, we get a similar geometric illustration as shown in Figure 3.4 (b), which highlights the fuzzy set A= [0.25, 0.75].

There are two views associated with fuzzy sets, which should be highlighted. The first view concerns the fuzziness of a piece of information, expressed by the membership value $\mu_A(x)$ that quantifies how compatible x is by the concept conveyed to μ_A. The second view is concerned with the uncertainty about a portion of information, in that $\mu_A(x)$ represents how possible it is for x to occur, given that μ_A (sort of constraint) is present. This point of view is usually referred to as "an epistemic view of a fuzzy set representing the state of knowledge about a variable" [102].

3.2.1 Membership Function

In mathematics, a membership function can be described as a sign function or a distinguishing function. This function is defined on set X that point to the membership of an element in a subset Y of X. The membership function of a subset Y of a set X is a function: μ_A: X→ {0,1}. In fuzzy set, the membership function is a generalisation of the membership function in traditional crisp sets. However, in fuzzy logic this membership function represents the degree of truth as an expansion of judgment. Usually both degrees of truth and probabilities are frequently confused even though they are theoretically dissimilar, because fuzzy truth stands for membership in unclearly defined sets, not probability of some occasion or situation [14, 28, 85, 102, 140].

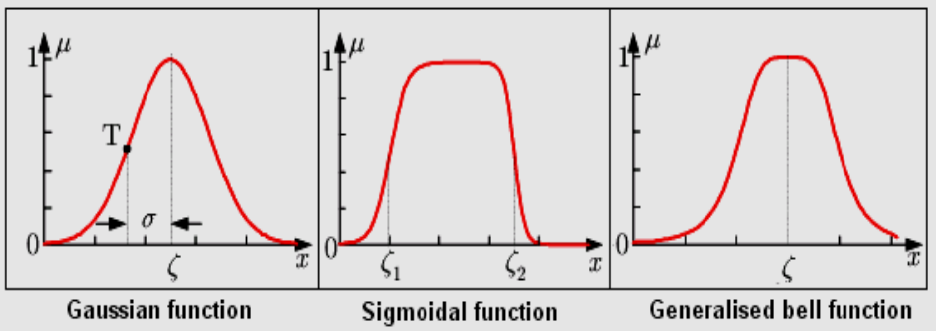


Figure 3.5 Examples of Membership Functions [114]

Generally, the membership function has a key role of creating a membership value set for either crisp or fuzzy types. The membership function can be scalar, vector, complex, or even real depending on the attribute of the membership values and the attribute of both elements and their set, which own this membership function as well.

Figure 3.5 shows examples of fuzzy set membership functions namely: Gaussian function, sigmoid functions, and generalised bell function [18, 114, 129]. Of course, there are many other different functions.

3.2.2 Crisp and Type-1 Fuzzy Set Representation Example

To illustrate the difference between crisp and fuzzy sets, as well as to show the ability of fuzzy set over the crisp consider the following example.

Name	Weight	Crisp Membership Deg.	Fuzzy Membership Deg.
Mark	108	1	1.00
David	105	1	1.00
John	98	1	0.98
Peter	81	1	0.82
C30s	79	0	0.78
Mike	72	0	0.24
Steven	67	0	0.15
Tom	52	0	0.00

Table 3.1 Real weight, and their membership values in crisp and fuzzy sets

The elements of the Fuzzy Set 'heavy men' are all men, but their degree of membership depends on their weight. Table 3.1 illustrates how a membership value is be assigned to each weight. Figure 3.6 shows graphically the values of membership sets in both Crisp and Fuzzy types of set 'weight men'.

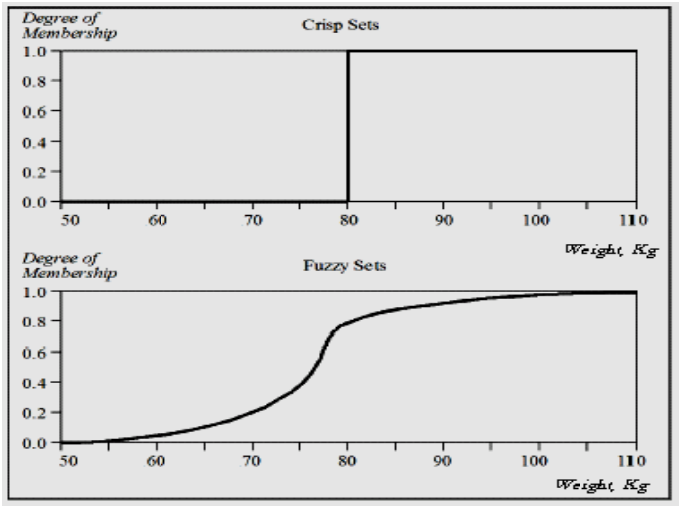


Figure 3.6 Crisp and Fuzzy Membership Values of Set 'Weight Men'

3.3 Type-2 Fuzzy Membership Value Set

"Type-1 fuzzy logic is restricting and we need to explore more vigorously other avenues to really contribute in a manner that is in more than (very very small) increments." [58]. To illustrate this, consider the perplexity surrounding the usage of English words and phrases where many are related to vagueness rather than randomness. This perplexity is a vital point for analysing language structures, and can be significant in creating a gauge of confidence in production rules. Hence, the interest in type-2 fuzzy sets stimulated by Zadeh [139]. Actually, a type-2 fuzzy set is an extension of the type-1 fuzzy set with an extra third dimension to give higher degree of freedom for better representation of uncertainty, compared to type-1 fuzzy sets.

The membership grade of each element of a type-2 fuzzy set is a fuzzy set in [0, 1]. The membership functions of type-2 fuzzy logic have been given different definitions from membership functions of type-1 fuzzy logic. In addition, special sets of operators have been defined for type-2, which leads to a derivation of the properties of type-2 fuzzy logic from those of type-1 fuzzy logic [58, 83]:

Definition: A type-2 fuzzy set, denoted $\tilde{A}$, is characterised by a type-2 membership function $\mu_{\tilde{A}}(x,u)$, where $x \in X$, $u \in J_x \subseteq [0,1]$, i.e.:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\}, \quad (3.4)$$

where $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$.

In the definition of type-2 fuzzy set given above, the first restriction that $\forall u \in Jx \subseteq [0, 1]$ is consistent with type-1 constraint that $0 \leq \mu A(x,u) \leq 1$, i.e. when uncertainties disappear, a type-2 membership function must reduce to a type-1 membership function, in which case the variable u equals $\mu A(x)$ and $0 \leq \mu A(x) \leq 1$. The second restriction that $0 \leq \mu\tilde{A}(x,u) \leq 1$ is consistent with the fact that the amplitudes of a membership function should lie between or be equal to 0 and 1. Figure 3.7 depicts $\mu\tilde{A}(x,u)$ for x and u discrete. In particular, X={1,2,3,4,5} and U={0.2,0.4,0.6,0.8} [83].

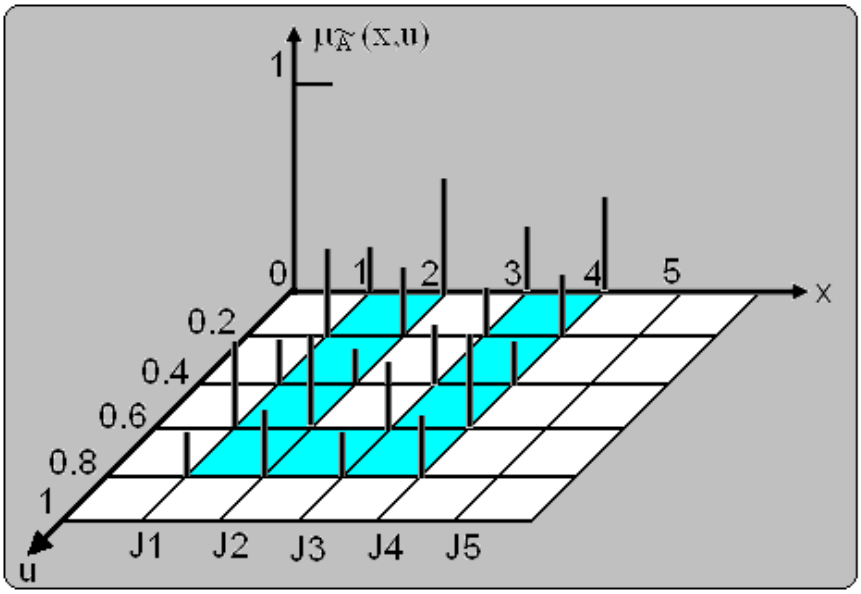


Figure 3.7 Example of a Type-2 Membership Function [83]
(*the blue area is the FOU*)

In that, u is an additional dimension associated with the membership value μÃ(x) for type-2 fuzzy sets. This u is a type-1 fuzzy set with the membership function Jx in a three dimensional space. Jx can be viewed as a vertical slice of μÃ(x,u). For a particular element say x', the membership function of the type-2 fuzzy set would give in multi-valued μA(x). Jx is denoted by [73, 82, 85, 135] :

$$\mu\tilde{A}(x=x',u) \equiv \mu\tilde{A}(x') = A(x) = \int_{u \in Jx'} fx'(u)/u, \quad \text{where } 0 \le fx'(u) \le 1, \qquad (3.5)$$

***where*:**

- fx'(u) is called a secondary grade, and it represents the amplitude of a secondary membership function.
- $\int_{u \in ju} fx'(u)/u$ means that the type-2 fuzzy set has a membership u associated with grade fx'(u) for x=x'.
- $\int u \in Jx'$ is a symbol that represents the collection of all points of u in Jx', and it is not an integration operator, exactly as in fuzzy logic notation. Likewise, fx'(u)/u means that the grade corresponding to the membership value u is fx'(u) and it is not a division operator.

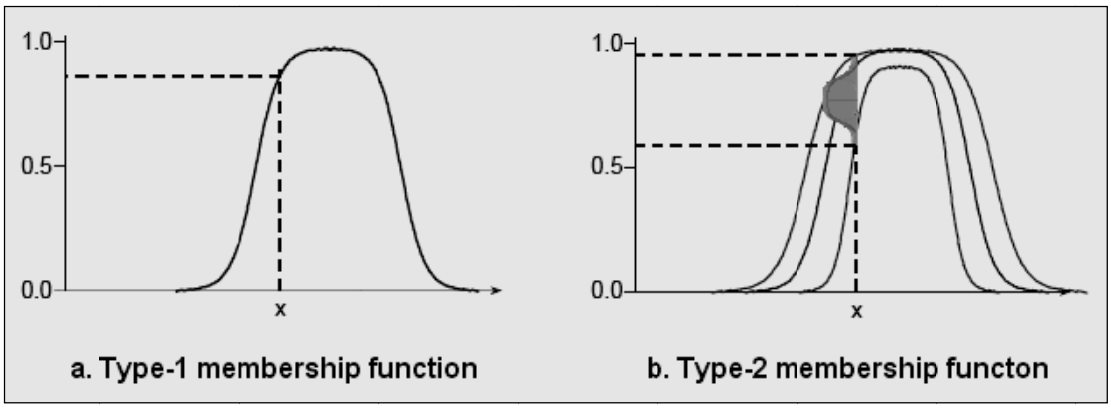


Figure 3.8 Type-1 and Type-2 Membership Functions [58]

Note that while type-1 fuzzy logic is two dimensional, type-2 fuzzy sets are three-dimensional because the value at each point is to be given as a function [83, 58, 18] as illustrated in Figure 3.8.

Type-2 inference approach results in type-2 fuzzy set. However, a crisp solution is required. It is achieved by reducing type-2 fuzzy set to a type-1 fuzzy set, and then defuzzifying it through a number of alternatives [58]:

- Centroid Type Reduction
- Centre of Sums Type Reduction
- Height (various) Type Reduction

3.3.1 Footprint of Uncertainty

Mendel and John first produced the term Footprint of Uncertainty (FOU). They adopted it from other terms to be used when working with type-2 fuzzy sets. FOU offers useful verbal and graphical description of the uncertainty captured by any given type-2 set [83, 58, 115].

As illustrated in Figure 3.9, FOU is a region of uncertainty for primary membership in a type-1 fuzzy set. When fx(x,u) = 1, $\forall u \in Jx \subseteq [0, 1]$ we have what so called an Interval Valued type-2 Fuzzy Set (IVFS). The uniform shading for the FOU represents the entire interval type-2 fuzzy set, and it can be described in terms of an upper membership function $\overline{\mu A(x)}$ and a lower membership function $\underline{\mu A(x)}$. Both of the upper

and lower membership functions are type-1 membership functions that delineate the FOU.

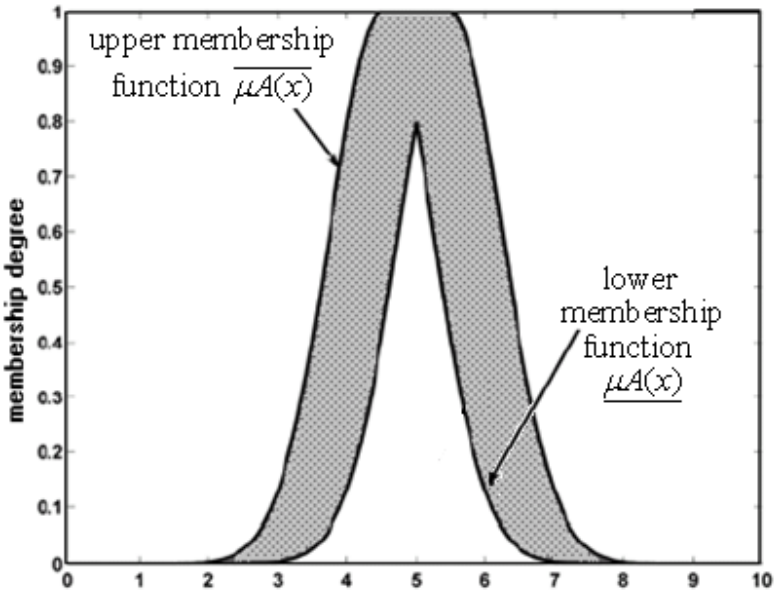


Figure 3.9 Interval Type-2 Membership Function [83]

3.4 Modal Logic and Possible-Worlds

In computer applications, classification of data necessitates an explanation of the data substance, in which information should be considered within the context, and where it used to be relevant and encompassed a purpose [100]. Boddy et al. [11] indicate that the notion of meaning is subjective; i.e. what someone assumes as valuable information, another may see it of no importance. Such cases form uncertainty of multiple-viewpoints type.

To model multiple viewpoints type of uncertainty, we need to move to a logic that supports reasoning about propositions (proposition is a primitive element of a sentence) of multiple viewpoints, which could be achieved by using modal logic type, which is one of approaches used for handling and modelling uncertainty (see 1.3).

There are two kinds of logical formations namely: absolute and relativist, which are used for clarification of the concepts of truth, proof, consequences etc. Of course, these two formations are associated, in that the relativist formation may be considered as a

way of carrying out the absolute one. Yet, the identifying logic with the absolute formation is considered practicable, unlike the relativist type, which is considered not so practicable to identify logic. This is why many logicians look to simply identify logic using absolute formation. Using the absolutist conception, logic provides the ultimate language for spelling out the facts of the world where semantics is beyond words; and therefore, any try for going outside of it is impossible. In contrast, in the relativist conception, logic provides several calculi, in which these calculi can be studied either individually, or as applied to something. Consequently, there exist a meta-language to every language, which has its own syntax and its semantics [103]. Relativist conception of logic is formally expressed using modal logic via using modalities such as necessarily contingent, possibly, and impossible. These modalities are important in the milieu of possible worlds or alternate universes. For example, providing '*possibly'* concept to proposition: '*Y is true*', results new formulation of this proposition that is: '*Y is possibly true*'.

Modal logics take their names from the modalities they have. Our daily natural and technical speech has copious broad variety of such modalities. As a matter of fact, our communicative aptitudes would be critically poor when excluding expressions like: *might be*, *can be*, *must be*, *is possible*, *is necessary*, and *would be*. Although these terms are importantly required to articulate the environment we are in, paradoxically, the speech involving possibility usually appears to be about something fictitious [75, 94, 109].

Modal logics concern themselves with variety of modes in which a statement may be true. Modal logics offers a set of powerful methods for understanding natural language speech, which habitually entail reference to other times, circumstances, and mental states of people. Generally, modal logics allow us to deal with [75, 94, 109]:

- Producing the truth of a set of propositions regarding their recent, past or future states of the real world, this type of logics is called temporal logics
- The truth of propositions under circumstances that possibly have been, this type of modal logics are sometimes called conditional logics

- The truth of proposition concerning viewpoint, knowledge, wishes, intentions, and obligations of actors, which possibly could be *false*, *unjustified*, *unsatisfied*, *unsatisfiable*, *irrational*, or *mutually contradictory*. This type of logic is referred to as possible world logic

The conception of possible worlds, which deals with the truth of propositions concerning viewpoint as stated above, is used to articulate modal claims in both philosophy and logic and has been already used to express modal assertions. Talk of possible worlds is very prevalent in modern philosophical discourse despite the fact that much about them is argued [104, 50].

There is an individual possible world for each different way the world could have been. The using of the principle of possible worlds imposes the consideration of the actual world to be one of the many possible worlds. The actual world is the world that corresponds to the way a universe in real, and it is one of possible worlds [104, 50].

Alternate universe is another term used in the literatures of possible worlds. Alternate universe has characteristics varies from our own universe. For example, works of imaginary tale usually depict some kind of alternate universe, which varies from our own universe to a superior or smaller in scope. Nevertheless, these alternate universes are required to be logically steady. For example, there may be alternate universes where hypothesise ‘Margaret Thatcher is not UK prime minister’ is true, but it hard to find alternate universes where 1 + 1 = 3 is a true hypothesis. One important thing to state here is that the scope to which alternate universes actually exist is a deep metaphysical subject, which has strongly connected to both theology and physics [103, 104].

Last but not least, the interpretation of many-worlds in quantum mechanics and the concept of possible worlds cause a slightly confusion and therefore have sometimes been compared. Among many difference between these two notions, possible worlds are logically, not physically possible, unlike many worlds of quantum-theoretical which are all physically possible [104, 55, 9].

3.4.1 Possible-Worlds Modalities

Modality is the major aim of possible world discourse. In a few words, a modality can be defined as a word, which may be applied to a certain statement S to develop a new statement that formulates an affirmation about the statement's (S) truthiness mode regarding the conditions under which S may be true, as well as about: *when*, *where* or *how* the statement (S) is true [50, 55, 9].

Philosophers have classically acknowledged four fundamental and consistent cases for modality, namely: Possibility, Impossibility, Necessity, and Contingency. As a standard, these cases are considered as interrelated in the following ways [56]:

- Possibility rules exposed impossibility and completely entails contingency or necessity.
- Impossibility rules out each of possibility, necessity and contingency.
- Necessity rules out both impossibility and contingency, and requires possibility.
- Contingency rules out impossibility and necessity, and requires possibility.

Yet, some logicians admit *Necessarily* and *Possibly* modal operators only, which are typically symbolised as □ and ◊ respectively, and describing modal logic as the logic of necessity and possibility. For example, let x stands for the statement: *G. B.*, and Y stands for the property: *lost the electoral vote of 2000*, then the statement: *it is possible that G.B lost the electoral vote of 2000,* is represented by: ◊**x**Y [50, 55, 9].

Logical quantifiers can be used with modalities. For example, using the existential (∃) logical quantifier, both of: (∃x) ◊Fx and ◊ (∃x)Fx can be read as: there is an x such that it is possible that x lost the electoral vote of 2000, and: it is possible that there is an x such that x lost the electoral vote of 2000, correspondingly [50, 55, 9].

3.4.2 Semantics of Possible Worlds

A functionality principle of semantics says that the meaning of discourse expressions is a function of the meanings of their component expressions. Generally, semantics is a component theory within a larger semiotic theory and it is related to meaningful,

symbolic, and behaviour. In turn, semiotic, which is wide-ranging science of signs and languages, encompasses of pragmatics (where reference is made to the user of the language), semantics, and syntax [75, 109, 56].

Attempt to formalise several pragmatics of natural languages has get significant exertion since the 1970s. Earlier logicians, who were principally interested in universal truths or mathematics, gave little interesting to the use of indexical expressions to include reference to the speaker's identity, location, or the time of either the speech or the mentioned events. The effort to formalise pragmatics has increased with the greater interest in linguistics. A formal answer of the problem of meaning has also been proposed for other topics [109, 50].

Pragmatics highlights the fact that local sense thinking of modality is the foremost alternative to thinking of it in a global sense. Therefore to illuminate everyday's talk of possibility, which the usual language is generally engages in, we get in possible worlds. Having possible worlds lets us consider relative and absolute conceptions in a uniform way. For example, the statement: *It is logically possible that UFO exists*, could be a claim that the proposition of UFO existing is true at certain world, exclusive of any constraint on which worlds are relevant with this claim [104, 50].

It is normally that the expressive of natural objects is habitually helpless to specify the type of the object (where it is belonging to) unless talking of circumstances that have dispositional properties to be actualised. Semantics in possible world logic can be viewed as the challenge to reduce modal notions to observable patterns in an actual world. By tradition, both philosophy and logic identified a set of properties of reference for a sentence to have a semantic value, (either true or false), which is so called referential dimension in the analysis of possible-worlds semantic discourse [104, 57, 9].

Semantics of possible worlds is the term often used as a synonym for Kripke semantics who and his colleagues were the first to introduce systematic theory- a consequence from the semantics of possible worlds- in 1950s. However, since that date improvements in the development of possible-world semantics have achieved, typically

ontological debate relating to whether that semantics provide a literally right explanation of modal propositions truth-conditions [104, 80, 57].

A proposition would have a truth value for being *exist in* an actual world w if that proposition is actually exist, or, equivalently, if w represents the universe *containing* that proposition. Likewise, a proposition is true at an actual world w if w represents the universe satisfying that proposition. A world corresponds to a universe by having its propositions true of that universe, and vice-versa, a proposition is true at a world if it is member of it. In other words, beside it is achievable; the definition of a truth notion by using the formal language of a science establishes the necessary and sufficient truth condition for each sentence. Thus, the detection of the meaning of a sentence is carried out using its truth condition. Logically, the semantic value of any proposition is either true or false at a given possible world [103, 104, 57].

A proposition is said to be logically possible if and only if there is some consistent approach for the world to be, where the proposition would be true. For example, '*the grass is green*' proposition (as well as all other actually true propositions) is logically possible, based upon the existence of a logically consistent way for the world to be an actual world, under which that the mentioned proposition is true [55, 56].

The semantics of possible worlds modalities is achieved by logically quantifying (assigning either true or false truthiness value) them over all worlds. This means that a way to conceptualise the notions of modality is to suppose a *possible world* that a universe could have been (with broadly understood of the word: *way,* avoiding prejudice the ontological issue of the definition of possible worlds). Important recalling here that different possible worlds are different forms in which our world could have been. Existence, ontological modesty, and modal modesty are the three restrictions required for achieving explanation of a possible world notion [57, 9, 40].

The rules for achieving semantics of possibility and necessity modalities of possible worlds are [103, 104]:

(1) A proposition is necessarily true if (and only if) it is true at all possible worlds (logician prefer to use at instead of in), including the actual world

(2) A proposition is possibly true if and only if it is true at one possible world, at least probably different from the actual world.

In other word, p is *necessary* true at every world, and p is *possible* true at a world w. Note that the set of all actual (of possible) worlds in which the statement is true is the logical range of it.

To illustrate that consider the following assertions that can be made for possibility, necessity and other logical modalities, which affect the semantic value of proposition in an actual world:

- True propositions have true semantic value in the actual world. For example: *Margaret Thatcher became the prime minister of U.K in 1979*, where: *Margaret Thatcher became the prime minister* is the proposition, and: *U.K in 1979*, is the actual world
- False propositions have false semantic value in the actual world. For example: *Conservative party won the U.K. elections in 2007*
- Possible propositions have true semantic value in at least one possible world. For example: *James Callaghan became the prime minister of U.K in 1976*
- Necessary propositions have true semantic value in all possible worlds. For example: *Man is mortal*
- Contingent propositions have true semantic in some possible worlds and false in other possible world. For example: *Labour party won the U.K. elections in 1999*, which is contingently true, and: *Labour party won the UK prime minister election in 2007*, which is contingently false.
- Impossible propositions (or necessarily false propositions) have false semantic value in all possible worlds. For example: *Labour party and Conservative party won UK prime minister election in 2007*

"Where is the wisdom we have lost in knowledge?" [T.S. Eliot in his 'The Rock', 1934]. Wisdom is viewed as prophetic in the sense that it considers human circumstances and incites a search for new understanding [1, 112]. The success to represent the semantic value of a proposition with complex ambiguity will open the door towards enabling the computer to deal with the highest level of knowledge that is wisdom. The first three elements of DIKW hierarchy, which is illustrated in Figure 3.10 namely: Data, Information, Knowledge, and Wisdom, had been treated by computer, wisdom still not there [60, 112]. Note that AI techniques are aimed to handle knowledge, the antecedent level of wisdom.

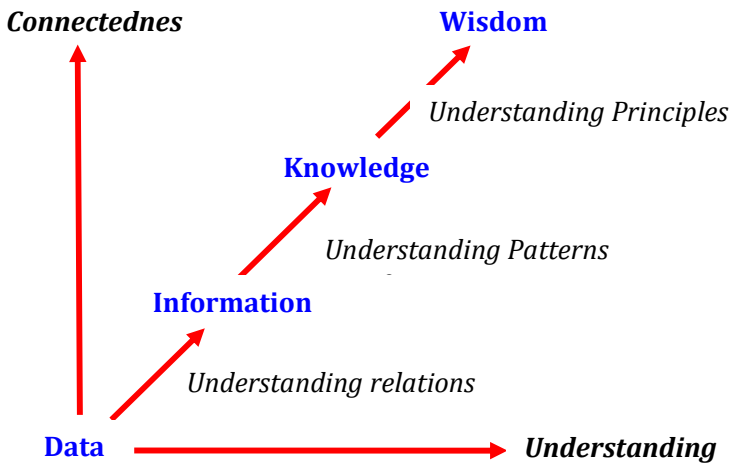


Figure 3.10 DIKW Hierarchy [60]

3.4.3 Mathematical Modelling of Possible World

As it has been shown previously, possible-world is a technique used to conceptualise modal notions related multiple viewpoints, where a domain of propositions could have been (the term domain is used to describe the collection of all propositions). Thus different possible worlds are different techniques in which a certain world could have been. Among those multiple possible worlds, actual world is the one that contains the propositions of true logical value [104, 80].

Based on fact that the domain of propositions has a close relation to possible worlds, one suggestion that has been made for theoretically modelling possible world is that the

maximal sets of composed sentences would represent possible worlds. A proposition will be included at a certain actual (possible) world if it is involved by the set of propositions resulted from the sentences of that world [55, 57].

Thus, the first approach for mathematically modelling of possible worlds is achieved via using of the approach of creating possible worlds as recombinations of multiple actual world entities. While modal realism is understood as the principle that an enormous collection of worlds is exist, each world in this collection is a closed system isolated from all others worlds and encompassing its own distinguishing semantic values for the properties set of possible worlds, where these properties are full with all their associations to each other [104, 80, 40].

As illustrated in Figure 3.11, we can mathematically say that a real possible (or actual) world (pW) is the set (D_k) of semantic values (v) for properties representing one *environment* in a set of possible real worlds (pR). The set of all possible worlds is denoted by [65, 104, 40]:

$$pR = \{pW \subseteq pR \mid pW \text{ is an actual world}\} \tag{3.6}$$

For each possible real world object there is only one environment (set of properties value) is contained in pW, denoted by:

$$(\forall v \in D_k \rceil pR) \rightarrow v(pW) \text{ have a semantic value} \tag{3.7}$$

Each set of properties value (pT), representing real world object in pR, is contained in pW at least once, denoted by:

$$(\forall pT \in pR)\ (\exists pT' \in pW) \rightarrow \text{semantic value } (pT) = \text{semantic value } (pT') \tag{3.8}$$

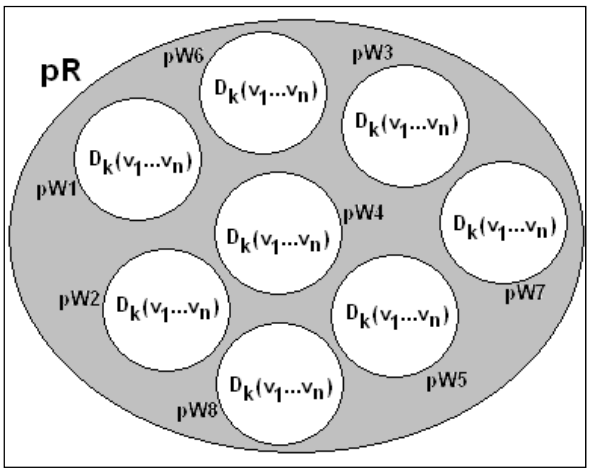


Figure 3.11 Venn diagram for Mathematically Modelling of Possible Worlds

Kripke-style semantics includes a function which maps each member of a set of possible worlds to the set of properties existing in that world properties, which need not to be exist in other world of this set. Thus, truth-conditions are given in terms of a totality of all possible worlds, including non-actual worlds [74, 80].

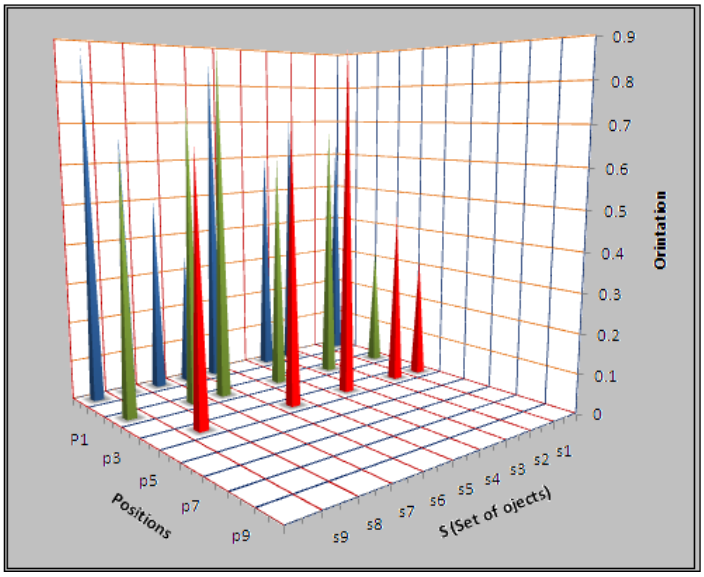


Figure 3.12 Euclidean 3D Space for Mathematically Modelling of Possible Worlds

The second approach for mathematically modelling of possible worlds can be achieved by using Democritean model, where a possible world is modelled as a function *f* defined on the product space $S \times R$. As shown in Figure 3.12, S is supposing to be a set

of objects and R is supposing to be the real line that assigns to each member of this space an ordered pair consisting of both a position in Euclidean three-dimensional space and an orientation [104, 40].

In other words, if S is a set of propositions and R is a possible worlds (a family of actual worlds), the point of view of this approach is to take the universe to be pairs (s,r) in the product S x R, and the membership functions to be maps from S x R into a semantic value set. In that, possible worlds are to be assembled as reorganisation of a subset of the propositions of the actual world and of arbitrarily many duplicates of these propositions as being in the other many possible worlds [57, 40].

The importance of possible worlds is that every proposition has a semantic value in any given possible world (actual world). *This means that a proposition would have multiple semantic values depending on the actual world(s) where it exists* [65, 75, 104].

The question here is 'what makes semantic values vary from one actual world to another?' The answer is there must be certain parameter(s) that affect the calculation of the semantic value in each actual world. This answer coincides with the explanation proposed by Beynon-Davies [7], who discerned that the sense of information is crucial and open to many explanations. His explanation is based on semiotics or semiology.

In 'possible worlds' principle, every element will obviously have multiple semantics (and hence multiple semantic values). So, unlike 'absolute conception of logic', relativist conception of logic provides multiple calculi that can be studied either 'in their own right' or 'as applied to something' [103].

As reported by Ruzhan [74], the semantic value of an element (a syntax of proposition) in an actual world of possible worlds is a pair (D,F) where D is the domain, non-empty set and F an interpretation function assigning a semantic value to each non-logical (syntax) of the language, which agrees with the Definition 4.2 of DOP. In other words, D is the domain of proposition, in which proposition's semantic value (membership value) of its syntax is to be calculated, see Lu Ruzhan [74] for more details.

Example 3.1:

Considering set of close similar handwriting shapes (syntaxes), and using the notion of relative semantic value given by Lu Ruzhan [74], each element of the syntax set of the given proposition has two semantic values. Table 3.2 shows that each semantic value is engaged with either O_char or Zero_digit syntax's domains, which is assigned by a certain interpretation-function. ■

Syntax (non-logical)	D	Semantic (truth) value
	O_Char	0.7
	Z_digit	0.64
	O_Char	0.67
	Z_digit	0.5
	O_Char	0.1
	Z_digit	0.7

Table 3.2 Possible Semantic Values of Close Similar Handwriting Shapes (Syntaxes)

In the above description of semantic value given by Ruzhan [74], where actual world of possible worlds is used, D is supposed to be identified, whereas calculation is required only to determine the semantic value of an unknown element (syntax) using the function F.

Since a proposition may have multiple different syntaxes (with the same semantic) [75], hence, proposition's syntax could be represented as a set. In addition, syntax of a proposition may have multiple semantics (interpretations) as stated by possible worlds logic [74], hence, the semantics of a proposition might be a set too.

3.5 Summary

This chapter has presented literature survey for both fuzzy set (including the most advance one that is type-2) and possible-world of modal logic, which each one is used to model and handle certain type of uncertainty. Although, only somewhat short literatures were presented, this chapter hopefully provides an enough succinct description of the types of logics which exist and which are not always taken advantage of each other.

The idea behind fuzzy logic is that most of the processing of information performing by human is not based on two-valued (true/false) logic. It is actually based on nebulous

perceptions, truths, inferences, etc., all ensuing in a typical, abridged, and normalised output, which is assigned a precise decision value by a spoken human. Humans are able to work with words like hot, tall, etc., which are fuzzy and can have various representations. They are just human judgments, not based on precise quantifications, thus called fuzzy variables. The big difference between humans and digital computers is the human's ability to deal with fuzzy sets and the consequent conclusion capability to arrive at an output. Dispute facing attempts to create artificial intelligence based computer is how to imitate this human ability; doing this is the goal of FLS. Although fuzzy set declare itself as a good approach in dealing with uncertainty, this thesis points to the shortcomings of this approach in dealing with uncertainty, as shown in Section 1.3 earlier.

Modal logics, which include multiple types of logics rather than possible-worlds logic, is considered in turn as an approach to handle certain types of uncertainty concerning multiple view points. Yet, there is an argument in quantifying its logical semantics. The literatures given in this chapter illustrate the approaches to mathematically modelling of possible worlds logic, but nothing is exist about the form of the numerical semantic value (quantification form) it should be. Note that crisp set quantifies the logical semantics as {0,1}, type-1 fuzzy set quantifies them as [0, 1], and type-2 fuzzy set as {[0, 1], [0, 1],…, [0..1]}.

Type-1 fuzzy set is unable to model multiple viewpoints (a sort of uncertainty) and modal logic has its shortcoming of missing a numerical description to quantify fuzziness in each viewpoint. Yet, each one of these two approaches has its pros that can be used in the development of new approach for modelling and handling of complex ambiguity (mentioned in Section 1.2). It is important recalling here that, the handling and modelling of complex type of uncertainty is currently achieved by type-2 fuzzy set.

As complex type of uncertainty encompasses of both fuzziness and multiple views, neither type-1, nor possible world is able to handle this type of uncertainty alone. But (as shown in this chapter) fuzzy set has the ability to model and handle fuzziness description of data, and possible world has the ability to capture multiple viewpoints to data. These

two properties are of great interest to define new membership value set that is able to handle and model these two components of complex ambiguity at the same time, thus improving the capability of type-1 fuzzy set that can overcome the drawbacks of type-2 fuzzy set, which have been illustrated in Section 1.3 of this thesis.

To achieve such goal, a question needs to be answered here, which is: *how these two approaches can be mixed to produce a new membership value set (and later logic) that able to handle fuzziness and multiple viewpoints at the same time*. Achieving such goal comes via providing possible world logic the ability to quantifying multiple view points and also model fuzziness in each of these multiple viewpoints and expressing that in a new set of membership value. Next chapter presents the definition Relative Fuzzy set resulted from combining these two approaches.

Chapter Four

Relative-Fuzzy Logic

A new approach for modelling uncertainty

4.1 Introduction

Definition of the new RF membership value set is a step goal of this thesis. This definition is to be based on a new explanation of the uncertainty problem called Domain Of Proposition (DOP), which relies on notes of possible world philosophy. Essentially Relative-Fuzzy membership value set targets the same goal of the type-2 fuzzy set, but differs in method of calculation and representation of the membership value. *Multiplicity* assumption is used to define this new membership value set rather than *fuzziness* assumption, which is applied to type-1 fuzzy membership value for defining type-2 fuzzy logic.

The aim of chapter is to present the definition of RFL. This definition is to be achieved via the definition of States Of Proposition (SOP), definition of DOP, definition of RF membership value set, and the description of union, intersection, and complement of RF sets. The chapter ends up with drawn conclusions.

Principally, logic, in mathematical language, requires the definition of its semantic values and the application of the basic logical operations (AND, OR, and NOT) to these semantic values as well. Set theory (including its intersection, union and complement operations) is usually used, as an approach to mathematically illustrate any logic in its two constituents, i.e. semantic values and logical operations. To clarify this, crisp logic, for example, has the {0, 1} as a semantic value set, while fuzzy has the [0, 1] set. Therefore, the principal definition of RFL needs to exact its semantic value set first (or as it is known the membership value set), and then demonstrate the set operation.

Definition of the membership value set grew from the definition of the SOP and the DOP, which specify the parameters to be used in calculating RF membership value. The

name given to the suggested novel membership value is Relative Fuzzy to distinguish it from Crisp (which is an integer 0 or 1) or Fuzzy (which is a real number in [0...1]). RF semantic value is to be a result of proposed functions that calculate this new membership value. The suggested Relative-Fuzzy logic presents a new philosophy of viewing ambiguity type of uncertainty, which is based upon the understanding of possible world logic. Figure 4.1 illustrates the strategy has been followed to develop RFL, where gray boxes are the contributions.

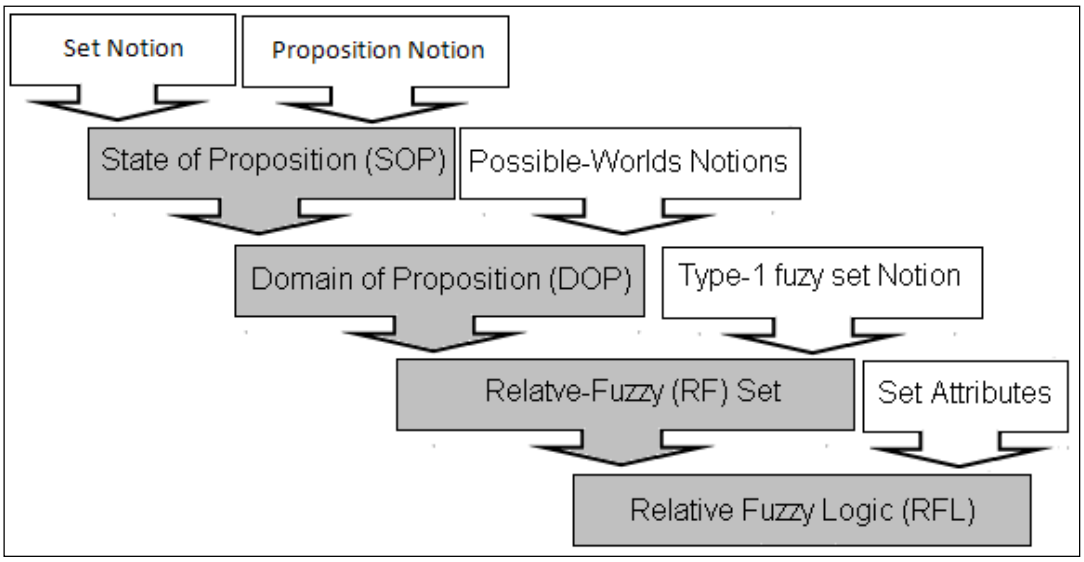


Figure 4.1 The Strategy of Developing RFL

4.2 The Definition of States of Proposition

The concept of proposition and set are used here to explain the problem of uncertainty of the data. To keep away from ontological allusions and arguments, the word *sentence* is usually used instead of the word *proposition* with a view to denoting those strings of symbols of carry truth-value, i.e. being either true or false pending the interpretation.

Since the two components of a proposition (syntax and semantic) are sets (see Chapter Three), it means that a binary relation between syntax set and semantic set of a proposition can be easily maintained to illustrate possible types of data uncertainty. This relation defines four possible states of a proposition, namely: single syntax/single

semantic, multi syntax/single semantic, single syntax/multi semantic, and multi syntax/multi-semantic. Thus, we are formally defining the SOP here as follows:

Definition 4.1:

As each of the syntax and semantic of a proposition comes as a non-empty set, the State of Proposition (SOP) defines a binary relationship between them. SOP can be one of the following four states: single syntax/single semantic, multi syntax/single semantic, single syntax/multi semantic, and multi syntax/multi-semantic. ■

Example 4.1:

Let ***'the shape 0 means digit zero'*** be a proposition.

- The syntax component of this proposition is a set of single element {0}
- The semantic component of this proposition is a set of single element {zero_digit}.
- The semantic value for the syntax is an element (since there is no other semantic to mapping) to be assigned, which results from a (crisp or fuzzy) membership function:

$$f(x) \rightarrow \text{zero\_digit} \qquad (4.4a)$$

- Such proposition is a *single syntax/single semantic* proposition. Uncertainty does not exist in this proposition since no multiplicity exists in both of its syntax and semantic. Using Venn diagram, Figure 4.2 depicts graphically the syntax and semantic of this proposition. ■

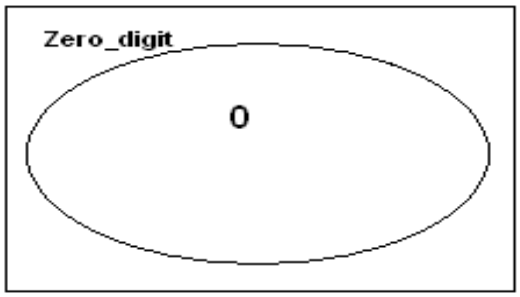


Figure 4.2 Venn diagram of Single Syntax / Single Semantic Proposition

Example 4.2:

Let ***' each of*** ○,○, and ▷ ***shapes means English O character'*** be a proposition.

- The syntax of this proposition is indicated by the set of elements {○,○,▷}
- The semantic in the proposition is a set of single element {O_Char}.
- Each syntax element has one semantic value that results from a membership function, which maps or interprets a syntax to its {O-Char} semantic. This function can be either Crisp or fuzzy:

  $f(x) \rightarrow$ O_Char (4.4b)
- Such proposition is a multi syntax/single semantic proposition. Uncertainty exists in this proposition due to the existence of multiplicity in its syntax. Using Venn diagram, Figure 4.3 depicts graphically the semantic and syntaxes of this proposition. ■

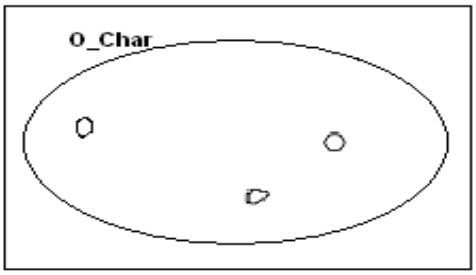


Figure 4.3 Venn diagram of Multi Syntax / Single Semantic Proposition

Example 4.3:

Let ***' the shape 0 means English O character or digit Zero', be*** a proposition.

- The syntax of this proposition is a single element set {0}
- The semantic of this proposition is a set of element {O_Char, Zero_digit}.
- Since there are two semantics, the semantic value of the syntax element is a set of two elements, each results from a certain membership function:

  $f(x) \rightarrow$ O_Char (4.4c)

  $f(x) \rightarrow$ Zero_digit (4.4d)

– Such proposition is a single syntax/ multi semantic proposition. Uncertainty exists in this proposition due to the existence of multiplicity in its semantic. Using Venn diagram, Figure 4.4 depicts graphically the semantics and syntax of this proposition. ■

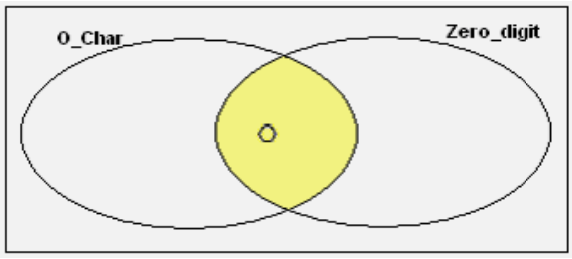


Figure 4.4 Venn diagram of Single Syntax/Multi Semantic Proposition

Example 4.4:

Let ***'each of*** ○,○, and ○ ***shapes means English O character or digit Zero',*** be a proposition.

– The syntax of this proposition is a set of the elements {○,○,○}
– The semantics of the proposition is a set of elements {O_Char, Zeor_digit}.
– The semantic value of each syntax element is a set of two elements, each results from a membership functions:

$f(x) \rightarrow$ O_Char (4.4e)

$f(x) \rightarrow$ Zero_digit (4.4f)

– Such proposition is a multi syntax/ multi semantic proposition. Uncertainty exists in this proposition due to the existence of multiplicity in both its syntax and semantic. Using Venn diagram, Figure 4.5 depicts graphically the semantics and syntaxes of this proposition. ■

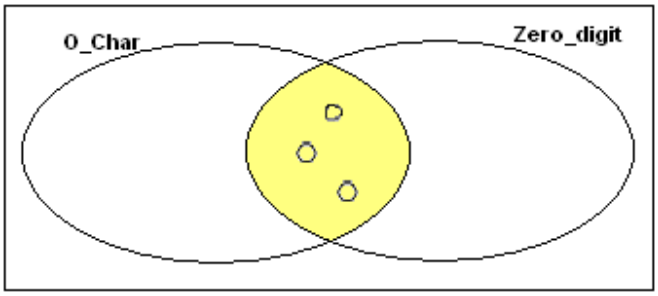


Figure 4.5 Venn diagram Of Multi Syntax/Multi Semantic Proposition

As a conclusion, our definition of the states of proposition along with the examples given in this paragraph:

- Distinguishes 'one to one' relationship (illustrated in Example 4.1), 'one to many' relationship (Example 4.2), 'many to one' (Example 4.3), and finally 'many to many' relationship (Example 4.4) between source of data (or syntax) and form of producing (or semantic).
- Explains the ambiguity class of uncertainty (reason of uncertainty is the existing of multiplicity).
- Extends the definition of ambiguity to a new dimension that is 'many to many' relationship (as illustrated in Example 4.4), in addition to traditional 'one to many' relationship.

Thus, our formalisation of SOP presents itself as a plain explanation of the second and third types of uncertainty as well as the four sources of uncertainty described earlier (see Chapter One).

4.3 The Definition of Domain Of Proposition

Among the four states of proposition illustrated above, the proposition's state of multi syntax/multi semantic is the most difficult in terms of ambiguity complexity. As 'many to many' relationship is a complex relationship, it should be broken into 'one to many' or 'one to one' relationship. Therefore, we need to break multi syntax / multi semantic SOP into at least 'multi syntax / single semantic' SOP in order to successfully handling it. Such an operation would produce multiple 'one to many' (or 'one to one') sub-relationships, and would make current methods of modelling uncertainty unable to

handle it without loosing any of these many 'one to many' sub-relationships. The present research makes use of relativist concept of logic to handle and explain the uncertainty problem.

As illustrated in Chapter Three, in possible world, a proposition is semantically true at a world in case it truly describes a state of affairs obtaining in that world. In addition, a proposition is true at a world if it is a member of that world. This means that the semantic value of a proposition is conditioned by its importance in an actual world. The actual world is defined as the set of propositions in which they have a semantic value. Existence of a proposition in an actual world is occurring if the proposition meets the conditions of that world.

Based on the above, this thesis views the ambiguity class of uncertainty as a property of proposition, which is exist in several actual worlds and its syntax involves different semantic values in each actual world. As mentioned in Chapter Three, certain parameter(s) affect the calculation of the semantic value in each actual world and makes it vary from one actual world to another. In such that, the information can be glimpsed as embodied in signs, and converses how the elements of semiotics, pragmatics, semantics, syntactic and empirics enlighten thinking about communication and information. In this thesis, we call such set of parameters as Domain of Proposition (DOP). Here it should be clear that the domain of element in set theory (semantic of element) is different from the DOP (actual world of proposition as termed in 'possible worlds' logic).

DOP is the term used in this thesis to describe the actual world of a proposition. In other words, domain of proposition is the environment that affects the syntax interpretation of a proposition. We define DOP as:

Definition 4.2:

For each proposition, there is a *Domain Of Proposition (DOP)*, which is the actual world where/when a proposition occurs. DOP is the circumscription parameter that affects the interpretation of a proposition's syntax. ■

Example 4.5:

The proposition, given in Example 4.1, has one domain, which can be denoted as 'digits' domain. Proposition, given in Example 4.2, has also one domain that is 'characters' domain. Proposition (set) given in Example 4.3 has two domains: 'digits' and 'characters', while the proposition given in Example 4.4, has two domains: 'characters' and 'digits', as illustrated in Figure 4.6. ■

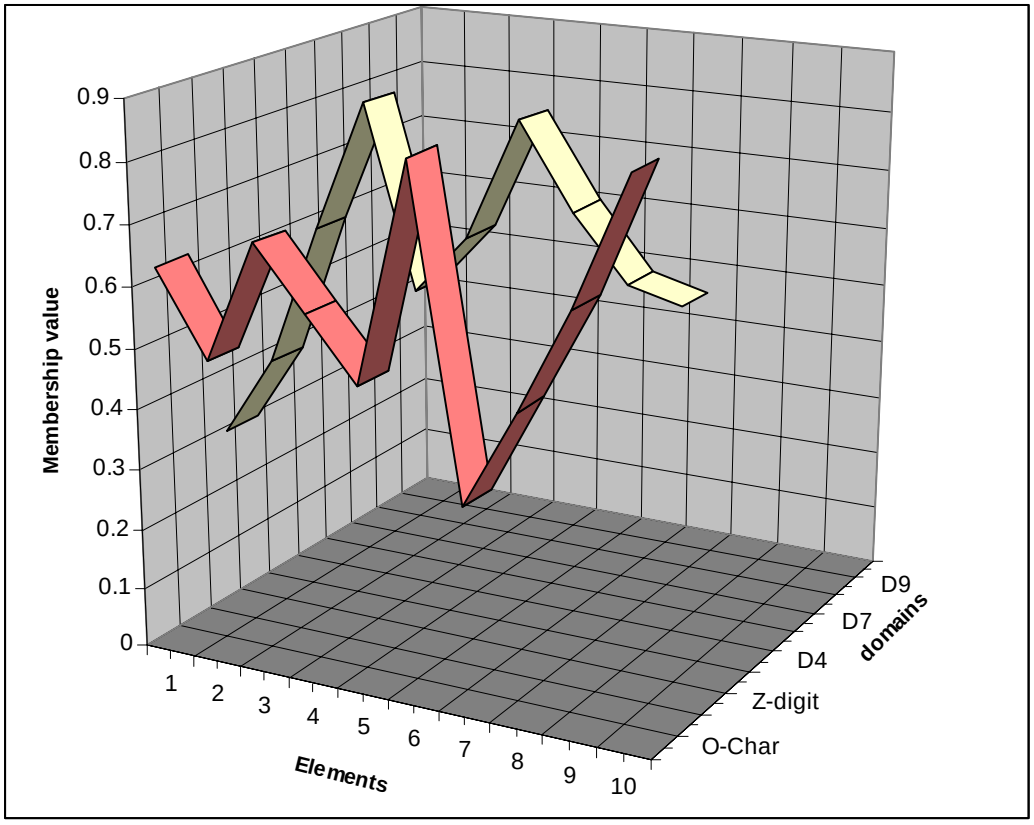


Figure 4.6 Proposition Representation Considering DOP

DOP can represent certain time/source of (for example): *generating data, decision-making, or environment (noise) surrounding the data generating operation*. Thus, instead of describing the proposition by using only its syntax and semantic, as crisp and FL do, the proposition using RF concept is going to be described by using its syntax, semantic, and domain (actual world of proposition), as modal logic does. Consequently, while the depiction of the membership value in both crisp and FL membership values

are positioned in a two-dimension plane (the 3rd dimension in type-2 FL is used to clarify FOU), the depiction of membership value in RF is positioned in a 3D geometry, as illustrated in Figure 4.6.

4.4 The Definition of the Relative-Fuzzy Membership Value Set

The shortcoming of relative semantic description of semantic value (which is given in Chapter Three) is that it stipulates the existence of the identity of the actual world, which is here (DOP). It does not offer an ability to calculate membership strength (semantic value) of syntax to DOP as much as the second part of it does that calculates the semantic value of syntax in its domain.

Therefore, while the suggested RF membership (semantic) value follows the same approach of describing the semantic value, RF slightly modifies this description to overcome its shortcoming. RF semantic value is represented as a pair (F_d,F_e), where:

1. Both F_d and F_e are interpreting fuzzy membership functions...
2. F_d is the function responsible for assigning the belonging strength of the syntax to DOP, and F_e is a function responsible for assigning the belonging strength of the syntax to syntax's actual world.

Now, we are going to formalise our description of the suggested RF membership value by giving the following definition of this novel membership value set, which we called it Relative-Fuzzy set:

Definition 4.3:

A Relative-Fuzzy (RF) set is characterised by a pair of Relative Fuzzy membership functions mapping the elements of a domain, space, or universe of discourse X, in a certain environment DOP, to a pair value where each of the two parts of the pair is a unit interval [0, 1]. That is RF: $X \rightarrow ([0, 1] \times [0, 1])$

Mathematically:

$$RF=\{(x,(\mu dRF(x), \mu eRF(x)) \mid \forall\, x \in X\}, \qquad (4.5)$$

where: $0 \leq \mu dRF(x) \leq 1$,

$0 \leq \mu eRF(x) \leq 1$ ■

The RF semantic pair value (0.0,0.0) means that the unknown element x (syntax of a proposition) is completely outside both the domain of proposition and the element's domain of a predefined proposition's domains. On the other hand, the pair (1.0,1.0) means that unknown element x (syntax of a proposition) is completely inside the domains (of proposition and syntax) of a predefined proposition. Any value between (0.0,0.0)<((μdRF(x),μeRF(x))<(1.0,1.0) addresses the partial strength belonging of an element x to both domains included in a predefined certain proposition. Therefore, a Relative-Fuzzy (RF) set is a set with two type-1 membership functions.

Example 4.6

Let the syntaxes of O_char domain have the following RF semantic values:

A={(0.12,0.7),(0.9,0.67),(0.78,0.1)}

Using Definition 4.3, A is an RF semantic value set of syntaxes in O_char domain, which encompasses the subset of domain membership value (i.e. dA) and the semantic membership value subset (i.e. eA), as illustrated in Table 4.1, and can be expressed as:

dA={0.12,0.9,0.78}
eA={0.7,0.67,0.1}

Syntax (non-logical)	O_Char Domain membership value (dA)	Semantic membership value (eA)
○	**0.12**	0.7
○	**0.9**	0.67
○	**0.78**	0.1

Table 4.1 DOP, Syntax and Possible Semantic Values of Example 4.4

While both have identical structure as a pair, RF semantic value differs from relative

semantic value in its first part of the pair in that RF semantic value describes a strength membership value of the syntax to D as a real number instead of naming D as in relative semantic. Figure 4.7 illustrates the positions of these RF membership values in syntax, semantic, and DOP geometry. ■

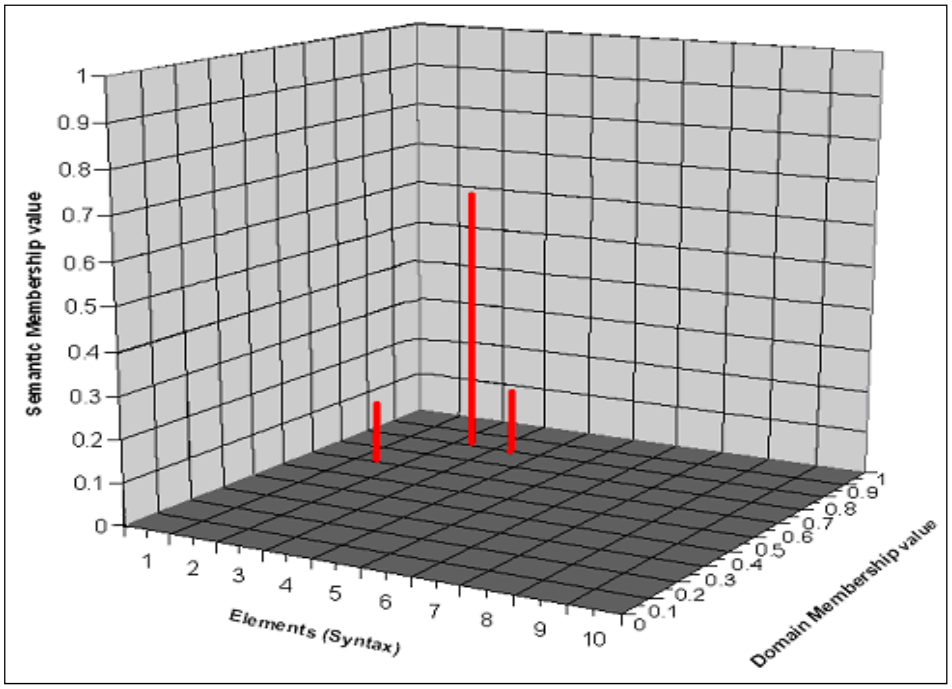


Figure 4.7 RF Semantic Value in Syntax, Semantic, And DOP Geometry

Imitating the way used to describe type-1 fuzzy membership, here are same suggested notations for the relative-fuzzy membership set:

$$\mathbf{RF = \{(\mu dRF(x_1),\mu eRF(x_1))/x_1, \ldots (\mu dRF(x_n),\mu eRF(x_n))/x_n\}} \qquad \mathbf{(4.6)}$$

$$\mathbf{RF=\int_{x \in X} (\mu dRF(x),\mu eRF(x)) /x \qquad 0\leq \mu dRF(x) \leq 1,\ 0\leq \mu eRF(x) \leq 1} \qquad \mathbf{(4.7)}$$

■

Example 4.7:

The RF in Example 4.6, which is depicted in Figure 4.7 could be rewritten using (4.6), as follows:

RF={(0.0,0.0)/0, (0.12,0.7)/1, (0.9,0.67)/2, (0.78,0.1)/3} ■

The Examples 4.1-4.4 above show that a proposition has more than one RF membership value (RF semantic value). The number of membership value depends on the number of actual worlds (semantics) where a proposition may exist. In that, the RF membership value should be calculated using two functions that produce pair value. This thesis suggests the use of two separate functions: the first is to calculate the membership value of syntax to DOP, and the second is to calculate the semantic membership value of this syntax in that possible world DOP.

4.5 Set Attributes and RF Set

We begin with several concept definitions associated to Relative-Fuzzy sets. These definitions are obviously extensions of the corresponding ones for both crisp and fuzzy sets but consider DOP. Of course, the RF semantic value given in the following examples are for elements in the same DOP with different RF membership value for each element (see Figure 4.7 above). This means that no relationship can be established between elements located in different DOPs.

Emptiness: A Relative-Fuzzy set is empty if, and only if, its membership function is identically zero on both μdRF(x) and μeRF(x).

Equality: Two Relative-Fuzzy sets A and B are equal, written as A=B, if and only if μdA(x)=μdB(x) and μeA(x)=μeB(x) for all x in X.

Complement: Given a Relative-Fuzzy set A, its complement ¬A is defined by its membership function for both the universe and discourse.

¬(μdRF(x),μeRF(x)) =(1-μdRF(x), 1-μeRF(x)) **(4.8)**

Example 4.9:

Let A= {(0.2,0.0)/0,(0.1,0.25)/1,(0.4,0.75)/2,(0.2,0.0)/3}

Then

¬A={(0.8,1.0)/0,(0.9,0.75)/1,(0.6,0.25)/2,(0.8,1.0)/3}

and for semantic given by DOP only:

¬eA={(0.2,1.0)/0,(0.1,0.75)/1,(0.4,0.25)/2,(0.2,1.0)/3}

and for DOP only:

$\neg dA=\{(0.8,0.0)/0,(0.9,0.25)/1,(0.6,0.75)/2,(0.8,0.0)/3\}$

■

Containment: A set can be completely contained as a subset within a larger set [59, 41]. Like Fuzzy sets, the membership values of a Relative-Fuzzy subset can have lower or equal membership values than their supersets in both DOP's membership value set and the semantic membership value set. That is, an item can belong less to a subset than it does to its parent superset. To illustrate this, given Relative-Fuzzy sets A and B, then:

A is completely contained in B if $(\mu dA,\mu eA) \leq (\mu dB,\mu eB)$. **(4.9)**

Example 4.10:

Let

$X=\{(0.2,1.0)/0,(0.6,0.25)/1,(0.4,0.75)/2,(0.2,0.55)/3\}$
$Y=\{(0.2,1.0)/0,(0.1,0.28)/1,(0.3,0.75)/2,(0.2,0.35)/3\}$

Then

$Y \subset X= \{(0.2,1.0)/0,(0.3,0.75)/2,(0.2,0.35)/3\}$

■

Union: The union of two sets has a membership that is the larger or equal of the two sets in both syntax and domain subsets:

$A \cup B$=Max [A (x), B (x)] in any component of A(x), B(x) **(4.10)**

Example 4.11:

Consider the two Relative-Fuzzy sets

long= (0.2,1.0)/0,(0.1,0.25)/1,(0.4,0.75)/2,(0.2,0.55)/3}
and
short={(0.5,0.7)/0,(0.51,0.35)/1,(0.42,0.55)/2,(0.62,0.57)/3}

Then

long$\cup$short(x)={(0.5,1.0)/0,(0.51,0.35)/1,(0.42,0.75)/2, (0.62,0.57)/3}

■

Intersection: Under classical set theory, the intersection of two sets is that which satisfies the conjunction of both concepts represented by those two sets. Under Relative-Fuzzy set, however, an item may belong to both sets with differing memberships without having to be in the intersection. Therefore, the intersection is defined as that the memberships are the lower of the two sets within both syntax and domain subsets.

$A \cap B$=Min [A (x), B (x)] for both components of A(x), B(x) **(4.11)**

Example 4.12:

Consider again the two Relative-Fuzzy sets

long= (0.2,1.0)/0,(0.1,0.25)/1,(0.4,0.75)/2,(0.2,0.55)/3}

and

short={(0.5,0.7)/0,(0.51,0.35)/1,(0.42,0.55)/2,(0.62,0.57)/3}

Then

long∩short(u,x)={(0.2,0.7)/0,(0.51,0.25)/1,(0.4,0.55)/2,(0.2,0.55)/3} ■

4.6 Comparison between Relative-Fuzzy and Type-2

Words represent different conceptions to different people. More precisely, all humans, including experts, show dissimilarity in their decision-making. Dissimilarity may occur in the decisions of a group of human experts, as well as in the decisions of a single expert over time. This dissimilarity forms a type of uncertainty that highlights the multiplicity on premises and consequences among people. This type of uncertainty has been classified as ambiguity, and in this thesis is described as complex ambiguity.

Ambiguity (including complex one) is the class of uncertainty aimed to be modelled and handled by RFL. Although it has not explicitly shown the class of uncertainty (but the sources of uncertainty) to be solved by type-2 FL, it is important to mention here that it seems that type-2 FL is the one supposed to handle ambiguity class of uncertainty. Thus, both RFL and type-2 FL share the same goal of modelling the ambiguity. RFL can

also model the fuzziness class of uncertainty since it uses fuzzy membership functions principle.

Multiple domains of proposition give clearer understanding, and may be a controlled approach to model uncertainty. As shown in Figure 4.6, each domain affects the way of calculating the semantic value of the proposition.

Property	Type-2 FL	RFL
Class of uncertainty	• Fuzziness, since it uses 'fuzzy' membership function type • Ambiguity of type 'one to many' relationship only	• Fuzziness, since it uses 'fuzzy' membership function type • Ambiguity, of all type of relationships including 'many to many' relationship
Simplicity of understanding	Difficult (see Chapter One)	Easy since it hasn't the complexity approach found in type-2 FL
Time complexity	Time consuming (see Chapter One) due to the complexity nature of its membership value function.	Possibly parallel execution of the RF's two membership functions makes the computation of RF membership value faster than type-2 membership value.
Accuracy	The ultimate semantic value (membership value) of an element results from two composed functions (type-2 membership function). In addition, an extra stage to crisp the type-2 membership value set is required. This gives three possible error sources (from two membership functions and a crisp stage).	RF has two independent functions only to calculate the ultimate semantic value of an element, which clearly means RF has less possible error sources than type-2 FL.

Table 4.2 Comparison between Type-2 FL and RFL

Modifying the approach of quantifying semantic values used in 'possible worlds', the RF membership function results, and consists of two sub- functions that may work independently. This independent feature of the RF membership sub-functions is different from the complex feature of the membership functions of type-2 FL. This property speeds up calculations and may give results that are more accurate.

Hence, a comparison between RFL and Type-2 FL is necessary to show the achievements of RFL and its properties as well. Table 4.2 lists the criteria, which are used to make comparison between RFL and type-2 FL.

4.7 Discussion

As noted earlier, 'Fuzziness' class of uncertainty has been solved by means of a certain type of membership function (understood by us as an inference function). The more testing (or learning in adaptive systems) of the membership function, the more accurate result could be produced, though to certain limit of course. However, in this thesis we have explained 'ambiguity', the second class of uncertainty using proposition and set notes. This explanation shows that ambiguity is not 'one to many' relationship only (as stated earlier in Chapter One), but can also be many to many relationship. RFL results from applying of 'possible worlds' philosophy to type-1 FL and represents a solution to both 'fuzziness' and 'ambiguity' classes of uncertainty.

Being crisp or fuzzy means that the semantic value of an element may come in one of the two following alternatives:

1. An individual number: This number is either an integer, which results from crisp membership function, or real number, which results from type-1 fuzzy membership function, as in crisp and Type-1 FL.
2. A set of membership values: in which the size of this set is engaged to the area of FOU as in type-2 FL.

As it has been shown, RF semantic value is neither an individual number nor set, but it is a pair of two real numbers quantifying the belonging strength of an element to semantic and DOP element of a proposition, both resulted from its own type-1 fuzzy

membership function. This pair fuzzy value has resulted from the normalisation of the complex ambiguity, i.e. converting it from many-to-many relationship to one-to-many relationship. This new membership value set can manage reasoning and uncertainty under conditions of multi syntax/multi semantics state of proposition by quantifying pair truth judgments for this proposition concerning its own domain and the domain of its syntax as well. Thus, we can say that the definition of SOP and DOP raise the level of knowledge that could be handled by the computer, which is wisdom (see Chapter Three).

A question may come in mind that: *what is the difference between inference and RF membership value?* The answer is that the former is a ‘process’ or ‘approach’ of a function to calculate a conclusion. The latter is a value or attribute that used to quantify a quality (which should come in terms of relative principle), so that this value makes the process of this quality able to be processed by a numerical machine that is computer.

Finally, we can report here the achievements of RFL as follows:

1. Ability to model the two classes of uncertainty namely 'Fuzziness' and 'Ambiguity' (including 'many to many' relationship type) simultaneously. As a philosophy, type-2 fuzzy logic seeks to reduce fuzziness attribute of semantic value rather than quantifying multiple concepts of multiple elements [28, 83]. This attempt limits the goals of type-2 to 'Fuzziness' class of uncertainty only.

2. Simplicity in calculation of membership value: the calculation approach of the Relative-Fuzzy membership value can be described as a 'direct approach' that uses the element to calculate two membership values. This approach differs from the indirect calculation approach of the fuzzy membership value of the type-2 fuzzy set, in which the inner membership function uses a fuzzy value resulting from the upper membership function to calculate the ultimate fuzzy value of an element.

3. The simplicity in calculation, as well as independence, of the two RF membership functions reduce the time required to calculate the semantic value of an element.

This makes RFL more suitable for those applications, where time is a critical factor, than type-2 FL.

4. Because it has less number of stages to calculate semantic value than type-2 FL, RFL seems to be more accurate as the accumulated error tolerance is reduced.

4.8 Conclusions

In this chapter, four novel definitions have been proposed namely: SOP, DOP, RF set, and RFL. RFL is defined to be a solution to model and handle the uncertainty problem that negatively affects the classification processing. The multiplicity (a type of uncertainty) in information has been studied using the low-level representation of information (i.e. proposition) together with the set theory, which helps to mapping the types and sources of uncertainty to proposition and leads to the novel definition of the four SOPs. Applying 'possible world' philosophy to SOPs addresses the way to understand the uncertainty problem, especially its complex type, which has been given as a second novel definition, namely the definition of DOPs. Finally, DOP has been used along with the concept of fuzzy membership value that yields the third novel definition that is the definition of RF membership value set. Successful application of the essential operations of the set, which are union, intersection, and complement, to RF membership value set converges with the fact that RF is a new logic. Each new definition is sustained by example(s) and these definitions are on proven mathematical theories.

Chapter Five

Software Design Description of ML/RFL-Based Net

5.1 Introduction

Chapter Four produced the description of RF membership value. Due to its novelty, no programming tool exists to accommodate the new RF membership value set. It is necessary to define a RF based computational tool in order to bring this principle into applications. Such computational tool will help investigating the ability of RF principle and investigating the ability of RF set for handling and modelling complex ambiguity in DM as such investigation is one of this thesis's aims.

ANN is one of the computational tools used to handle the uncertainty in DM either alone or as a part of neuro-fuzzy systems, as have been shown in Chapter Two. Since ANN is a non-algorithmic programming tool, it will be of great interest to automate creating the RF membership functions, which are required to calculate RF membership value.

While traditional ANN maps each element in an input data set to an element in the output result set, the set of output is not organised as a universal set of subsets (set of domains and subset of identifications within each domain as RF approach suggests). This means that ANN in its current structure is not suitable to be used directly for classification according to RF approach. Therefore, structural modifications are required to be applied for traditional ANNs in order to make them work according to the principle of RF. This chapter gives logical description for a RF based neural network that is called ML/RFL-Based Net, where ML stands for Multi-Level.

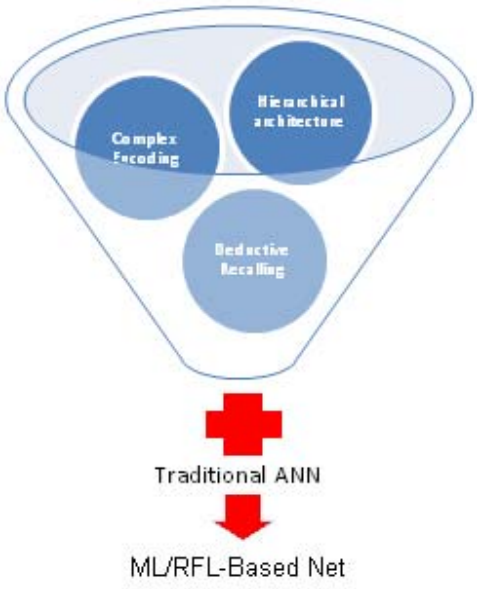
We are going to explain ML/RFL-Based Net Software by using Software Design Description (SDD) methodology, which includes the listing of functional requirements, system architectural design, and detail description of ML/RFL-Based Net components.

Next, the design of ML/RFL-Based Net is to be described considering its functional requirements. Details of its components are to be given also. Discussion about the relationships between ML/RFL-Based Net and RF membership value set, Hierarchical Neural Network (HNN), and Neuro-fuzzy system is given to highlight the dissimilarity between ML/RFL-Based Net and each of these three aspects . Finally, the chapter is to be concluded.

5.2 Software Design Descriptions of ML/RFL-Based Net

Generally, the aim of ML/RFL-Based Net is to use it for classification of data that have complex ambiguity. The problems to be solved by ML/RFL based Net have three important characteristics:

- The solution to the problems is nonalgorithmic, exactly as it is in traditional ANNs.
- There is complex-ambiguity (which is illustrated in Chapters One and Four) in data, i.e. an element may have more than one classification result.
- The input data to this modified ANN is a pattern just like the input pattern of traditional ANNs, so no modification would be applied for the structure of input pattern.



Figurer 5.1 Modifications Applied to Traditional ANN to get ML/RFL-Based Net

Since, ML/RFL-Based Net is going to be used for RF based classification; the output from this modified ANN is a decision to choose an element's identity from a subset (subcategory) among other elements of a universal set. Thus, unlike traditional ANNs, ML/RFL-Based Net will use multi-category knowledge rather than uni-category knowledge. This means that the structure's modification is the main one that will be applied to the traditional ANN to meet the principle of RF.

ML/RFL-Based Net can be defined as a hierarchical neural network (tree) of two levels that is able to find the identity of a mysterious element from subset in a universal set (from subcategory in a universal set of multiple categories). The upper level (root) has been called supernet and the second level (leaves) has been called subnets. Supernet level is simply a traditional ANN, and it forms the root in tree sense. The subnets level is a set of traditional ANN, which form leaves in tree sense. Consequently, the modification in the architecture requires modification in both learning and recalling algorithms to fit the new architecture. Therefore, a new learning (training) and testing (recalling) algorithms should be defined. Figure 5.1 illustrates the modifications required to be applied to traditional ANN to design ML/RFL-Based Net.

5.2.1 Functional Requirements

1. Should working according to RF approach
2. Should be able to calculate the RF semantic value
3. Should maintain the pair functions that use to calculate RF semantic value
4. Just one input should be submitted to ML/RFL-Based Net
5. No internal re-representation of the input data
6. Able to be learned
7. Have clear structure
8. Performing classification job
9. Able to handle complex-ambiguity data
10. Simple to be developed
11. Able to do fuzzification

5.2.2 Architecture

Obviously, the process of classification, by using ML/RFL-Based Net, would be achieved via two steps: finding the sub knowledge (domain or pragmatics) of the input element by supernet, and then finding the identity of the input element (recognition regarding domain or pragmatics) by a subnet.

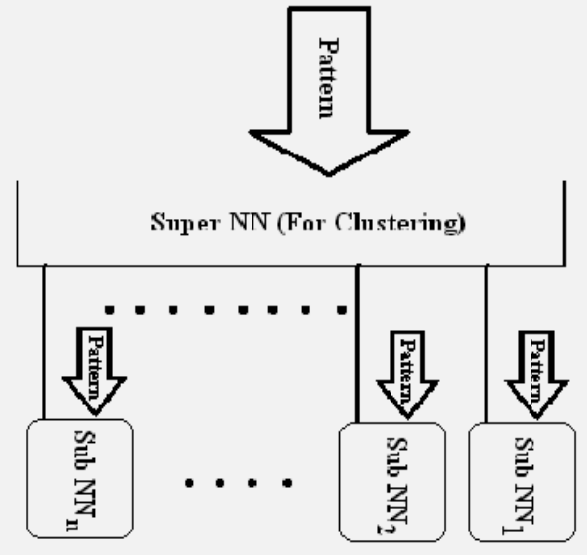


Figure 5.2 Logical Design of ML/RFL-Based Net Neural Network

The abstract unit in ML/RFL-Based Net is a traditional ANN, which could be either the supernet or any of subnets. As illustrated in Figure 5.2, the architecture of ML/RFL-Based Net is a tree structure composed of two levels, the first level has been called supernet and the second level, which is a set of nets, has been called subnets. Therefore, ML/RFL-Based Net is to be described as a multi-level ANN.

The connection between the supernet and subnet levels of ML/RFL-Based Net is of feedforward type due to the sequential nature of the processing followed by ML/RFL-Based Net as explained earlier. The number of layers in each level's net is not known, and it is changed regarding the problem being solved. In detail:

- Supernet: it is an ANN, which can be trained using either supervised/unsupervised learning approach. Unsupervised learning approach would make ML/RFL-Based

Net able to discover sub-knowledge (subnets) automatically. On the other hand, supervised learning aims to specify the number of sub-knowledge (subnets) prior to building the net, thus simplifying and speeding up learning process. The size of the input layer is the same as the size of the input data vector. The size of the output layer depends on the number of sub-knowledge or classes (which are represented as subnets) that may exist in the universal knowledge. The aim of the supernet is to find out the subnet (or knowledge subcategory) of the input pattern without specifying its identity, i.e. partial recognition.

- Subnets: they are the 2nd level of ML/RFL-Based Net. They are actually a set of ANNs. Only one net among these nets is to be activated (fired for recognition) by the upper level supernet. They can be trained by using either supervised or unsupervised learning approach. The size of the input layer of each net at this level is the same as that of the input data vector; this is because it takes the same input vector to be passed to the upper supernet. Each network in this set is responsible for specifying (identifying) the input pattern regarding subnet's set of elements.

Flexibility is applied to the procedure of creating ML/RFL_Based Net, in that no pre-specified number of subnets is required, but it has been left to be discovered while developing. Following is the procedure for creating ML/RFL-Based Net.

Procedure Create_MLRFLNet ( )

While Creating New Subnet

Create new SubNet and give it an index

Get training data and parameter

Train the new SubNet

Create Supernet

Train the supernet

Saving MS Network

The forward interconnection between the two levels of ML/RFL-Based Net performs link job, i.e. supernet is going to select one subnet from the 2nd level set of

subnets. Actually, the output of the supernet has nothing to be processed in the subnet(s) but to *enable* or *fire* one of the subnets for recognition.

5.2.3 Activation Function

The activation function behaves like hushing-up function. In that the neuron's output in a neural network is to be in certain set of values (either 0 and 1 combination, or -1 and 1 combination). Usually, three types of activation functions are defined to be used for developing ANN, which are Threshold Function, Piecewise-Linear function, and sigmoid function. Any of these three types can be used to create ML/RFL-Based. Of course, the choice is going to be made based on the nature of the data and problem to be solved.

5.2.4 Encoding (Complex Learning Paradigm)

The upper level (supernet) of ML/RFL-Based Net can be trained by using any ANN's learning algorithms or learning paradigm. The selection of the learning algorithm depends on the nature of the data of the problem being solved, same as the traditional ANN.

Due to the hierarchical structure of ML/RFL-Based Net, and the difference in function of supernet from subnets, the learning algorithm of ML/RFL-Based Net should be consisting of two portions. The first portion would responsible for training the supernet to find out the subnet (domain or pragmatics in the theory of semantic) of the input data, while the second portion would train the subnets to find out the identity of the input data (recognition regarding domain). It is important to recall here that the same input data pattern is used for both supernet and subnets, which clearly have been shown in Figure 5.2 above. Thus, the input pattern is going to be used for two different functions: identifying its domain and recognising the input pattern within this identified domain. Hence, a new learning paradigm for ML/RFL-Based Net can be defined, which is called: *the complex supervised learning paradigm*, as illustrated below:

Procedure Complex_Supervised_Learning (element)

Train supernet to be able to map an input element to a sub group (Finding the domain or pragmatics of the input element),

For each subnet do

Train each subnet to be able to map the input element to one of its output (recognition regarding element's domain/pragmatics),

Being supervised means that the number of subnets is going to be specified while designing. This property makes the design of ML/RFL_Based Net under controlled. Furthermore, it is expected to get an *enhancement* in the quality of learning phase in that, the hierarchical structure of ML/RFL-Based Net would help *maximising classification accuracy* and *minimising the risk of reaching over learning problem* (that makes ANN fans out its input). Another expected benefit is that once a subnet is trained, it could be amassed to a set of subnets belonging to a same supernet. When a new sub-knowledge is going to be included in future application, the previous subnets need no re-training, but the new one, and of course the supernet of ML/RFL-Based Net. This property could lead to define a *Neural Net-Base Management System (NNBMS)*, which will be responsible for storing trained ANNs (i.e. NN with the same-architecture, but with different recognitions), and retrieve the most suitable subnet according to supernet's result.

One may think that a drawback of this learning paradigm is *time consuming*. This is not true as parallel learning for both supernet and subnets can be performed instead of using sequential one.

5.2.5 Recalling (Deductive Autoassociative Recalling)

To decide which subnet is to be selected for the next step of recognition, ML/RFL-Based Net first discovers the attributes of the input testing pattern (attributes of the input's domain or pragmatics), which is the factor of selecting a subnet among multiple subnets. Thus, the testing algorithm of ML/RFL-Based Net, which is called *Deductive autoassociative recalling* should be composed of two portions as illustrated below:

Procedure Deductive_Autoassociative_Recalling (element)

Submit an unknown input element to the supernet,

According to the highest output value from supernet, activate the opposite subnet to supernet's decision.

Submit an unknown input element to the activated subnet, so the element would be classified.

Principally, while it is expected to be slightly slower than testing algorithm of the traditional ANN, due to its multi-level, the recognition result of ML/RFL based Net is expected to be more accurate since the ultimate goal results would be formed by double classifications.

5.2.6 The Interface of ML/RFL-Based

ML/RFL-Based Net is designed to be an embedded component in a DM system (or any other classification-based system). Therefore, its interface is in term of function header to be invoked for creating, training, or testing it along with suitable passing of parameters for each type of invocation.

5.3 Discussion

Important clarifications should be made to highlight the novelty of ML/RFL-Based Net among certain techniques that may share some features with ML/RFL-Based Net. First clarification is the relationship between ML/RFL-Based Net and the RF membership value set, which should be showed since it is assumed that ML/RFL-Based Net is implementing this novel membership value set. Regardless that ML/RFL-Based Net is a new design of ANN, it is somehow close to other known structure of ANN systems called Hierarchical Neural Network (HNN). Therefore, the second clarification is to show the difference between HNN and ML/RFL-Based Net. Finally, the relationship between ML/RFL-Based Net and Fuzzy Logic System (FLS) is to be revealed since ML/RFL-Based Net computational tool aims to develop fuzzy computerised systems. Figure 5.3 shows the relationship between ML/RFL-Based Net and those techniques close to its concepts.

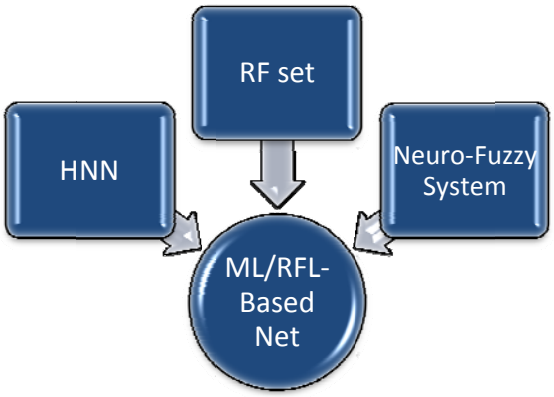


Figure 5.3 ML/RFL-Based Net and its Close Techniques

5.3.1 ML/RFL-Based Net and RF Set

As RFL membership value is composed of two parts that are the element's membership real value to DOP (dF) and the element's to one of DOP's identities (eF), ML/RFL based Net represents each DOP as a subnet, and the set of different identities in each DOP as a subset.

From structural point of view, supernet is responsible for recognising the DOP of the input pattern, i.e. selecting or picking up one of the subnets connected to the supernet. This is achieved by calculating the membership value of an input pattern to each available DOP (subnet connecting to the supernet), and selecting a subnet (domain), which offers the biggest membership value among other subnets (DOPs). The selected subnet is responsible for identifying (recognising) the mysterious input element by mapping it to one of its outputs via calculating the membership value of the input element to each output element in that subnet and selecting the output, which has the highest membership value. Thus, ML/RFL-Based Net uses the RF approach to recall the ultimate result of identifying an input of unknown pattern in two steps: find the domain (DOP) of the input pattern, using dF value- first portion of RF pair value, and then recognise the input pattern regarding this DOP by using eF value of RF.

From the above, it is clear that the supernet is the membership function that calculates dF portion of the RF membership value, and the subnet is the membership function that calculates the eF portion of RF membership value.

5.3.2 ML/RFL and Hierarchical Neural Network

Based on HNN literatures given in Chapter Three earlier, and from structure point of view, ML/RFL-Based Net can be classified as a hierarchical recognition system or HNN - a type of multiclassifier system mentioned above- *but with new features*. ML/RFL-Based Net has two important differences. The first one is that no new (inner) pattern is generated, but instead the same input pattern is used. This reduces the possible error that may be generated from the re-representation of the input vector (the approach followed by HNN). Secondly, the number of the subnets (further classification subsystem- as it is termed in the literatures of multiclassifier system) in ML/RFL-Based Net is tied to the number of domains involved in the classification, not to try and error principle as in the traditional hierarchical multiclassifier system.

HNN Feature	ML/RFL-Based Net	Benefit
Generate an inner pattern to be processed by sub net	No generation of inner pattern	• Reduces the time required for producing the new representation. • Reduces the error that could be generated by numerical calculations required to produce the new representation
Number of subnets is engaged to try and error principle	Number of subnets is known prior to construct the net	• Facilitates the designing • Reduces the designing time

Table 5.1 *C*omparison between Classical HNN and ML/RFL-Based Net

This difference facilitates the design of ML/RFL-Based Net and it makes it more controlled than traditional HNN. As a conclusion, these two features of ML/RFL-Based make it of less complex in designing (no try and error principle), and more accurate in classification results (no sub results or inner pattern is produced) as well. Table 5.1 abbreviates the differences between classical HNN and ML/RFL-Based Net and benefits resulted from these two differences.

5.3.3 ML/RFL and Neuro-Fuzzy Systems

Can ML/RFL-Based Net be considered as a *Relative FLS (RFLS)?* To answer this question we need to map the standard architecture of FLS to ML/RFL-Based Net, as Table 5.2 shows.

Standard FLS's Component	Implementation	ML/RFL-Based Net
Rules	IF-THEN rules that make the core of the IE, coupled with membership values	The threshold function located in each neuron in ML/RFL-Based Net
IE	The set of rules + the very nonlinear mathematics of how the inference engine does the decision	ML/RFL-Based Net, can be considered as a non-linear IE machine that used for building Expert System, since it contains both the threshold functions (stand for rules) and the nonlinear mapping of input to the output
The fuzzifier	Determination the level of membership of a measured input, which is required by IE (converts qualities to quantities)	No need for such unit, since ANN-the seed of ML/RFL- Based Net is getting its data as a vector representation of the fuzzy input data, without needing for explicit determination of the input pattern's fuzzification
The output processor	Converts a fuzzy set into a crisp number required in many decisions making applications	The ultimate identification achieved by ML/RFL-Based Net fit this description, which means that the output processor is embedded in ML/RFL-Based Net as a feature of ANN.

Table 5.2 Mapping standard FLS's features to ML/RFL-Based Net

Although ML/RFL-Based Net, as an ANN's type, is not used as a component in a FLS, but the mapping given in Table 5.2 shows that ML/RFL-Based Net can be considered as RFLS since it has the functions that the components of the standard FLS has. This shows that ANN can be used to build FLS, which could be easiest approach than defining IF-THEN approach that needs an explicit declaration of fuzzy membership values to each possible input, and explicit declaration of IE (set of IF-THEN rules

accompanied with fuzzy values). Of course, such conclusion needs more investigations to make a consistent comparison between the achievements of IF-THEN based FLS and ANN based FLS.

5.4 Conclusion

This chapter gave a description of a programming tool that follows RF approach and uses RF membership value defined in Chapter Four. This programming tool has been given a name ML/RFL-Based Net. The choice of ANN to be an approach to design a RF based computation model is made due to the non-algorithmic feature of ANN, which facilitates the building of such classification computation machine.

Actually, ML/RFL-Based Net imitates human brains in the way of distinguishing new data. Thus, ML/RFL-Based Net is a hierarchical structure of ANNs consisting of two levels, where typically the super level of ML/RFL-Based Net makes independent computations and its result is used to pick up one subnet from its sub level.

ML/RFL-Based Net has been presented it in this chapter in such a way that the description becomes a comparison. It is believed that this description approach clarifies the modifications applied to the traditional ANN, which are structure, learning, and recalling and leads to better understanding of the properties of ML/RFL-Based Net.

While ML/RFL-Based Net encompasses two levels (supernet level and subnet level), the size of each layer of both supernet and subnets is changeable depending on the nature of the data of problem being solved.

New terms consequently appeared in this chapter while modifying the traditional ANN. These terms are multi-level, complex learning paradigm, and deductive associative recalling. In addition, a suggestion for building a NNBMS has been introduced to improve the performance of the suggested ML/RFL-Based Net. Descriptions of both learning and testing algorithms have been given, along with the expected benefits and criticisms.

The discussion given in this chapter highlights the way to calculate RF membership value by ML/RFL-Based Net. The discussion also highlights the difference between

HNN and ML/RFL-Based Net. Furthermore, through this discussion a comparison between ML/RFL-Based Net and the traditional FLS is maintained. Through these two comparisons, a conclusion that ML/RFL-Based Net can be considered as a *novel HNN's design and* it is also an *RFLS* has been achieved.

One thing to mention here is that while it is important to ANN, the implementation of ML/RFL-Based Net has not been covered in this chapter. The implementation of ML/RFL-Based Net could be carried out either by using 4^{th} generation packages (like Matlab) or a 3^{rd} generation languages (like C++ or Java). Implementation using 4^{th} generation languages is much easier and takes less time, but needs to take into consideration the properties of the object units used in the implementation. Third generation language is the opposite as the programmer feels more free to code the design of ML/RFL-Based Net following the algorithms he is given. In Chapter Six, the description of the implementation of ML/RFL-Based Net is given, in which this design has been tested to create speech recognition model as part of a shell system called RFL4ASR. Implementation in this design is carried out by using Matlab development environment.

Chapter Six

‘RFL4ASR’ IDE for Creating RFL-Based Speech Recognition Predicting Model (Case Study)

6.1 Introduction

The definition of RFL and the ML/RFL-Based Net computation tool have been presented in the previous chapters. This chapter presents a description of an IDE or shell system that is able to create a DM prediction model for classifying speaker-independent phonemes (aiming recognition) on basis of the RF principle. It uses ML/RFL-Based Net as a classification model. RFL4ASR is the name given to this IDE system for developing, and testing a RF based classification model. The architecture of RFL4ASR system, the data, and the output results are to be explained in this chapter.

The goal of developing RFL4ASR system is to create a case study that shows the successful use of the novel RF as a methodology, and the novel ML/RFL-Based Net as a computational model, for classifying data having complex ambiguity.

The choice of speech recognition (SR) application to test the ability of RF membership value is made up because ASR system deals with speech data, which is very changeable locally by the speaker and globally by the circumstances surrounding the speech itself. Actually, speaker independent ASR is considered as one of many challenges aspects that Speech Recognition (SR) by computer faces because of the huge number of speech patterns that several people may produce for one speech term further to many different patterns for the same speech term that one person can produce. Thus, the speech data is considered as a good example of data having complex ambiguity.

Generally, to design and implement a successful ASR system, different types of problems have to be solved. These assorted problems are practically associated with a wide variety of disciplines, such as linguistics, psychology, phonetics, acoustics, signal

processing, pattern recognition, neuroscience, as well as computer science. Thus, speech recognition by computer is actually a very difficult task. In fact, important advances have been applied to ASR through many research works. Yet in comparison with human ability, ASR's current performance, at its best, maintains larger error rate than human's. Reddy [108] classified the problems, which have to be taken care of by the ASR's designer, and described them in terms of functions or attributes that certain ASR should have, as follows:

1. *Number of speakers*: More speakers means more difficulty; it creates a problem of speech variation (due to the variability among speakers), which ASR should be able to recognise. The enhancement of recognition ability of ASR is usually achieved by using large speech database as training data.

2. *Mode of the speech*: Generally, a speech mode is either isolated or continuous. Isolated speech recognition requires that a speaker should place an artificial break after each word in utterance. Nevertheless, in continuous mode of speech recognition, ASR can handle natural speech utterances in which words may be collectively joined, and probably be strongly affected by coarticulation. An ASR of natural speech permits the chance of break and wrong starts in a speech, the use of new words that do not exist in its lexicon, and other features.

3. *Size of vocabulary:* Usually, an increase in the size of the vocabulary would cause a decrease in the recognition achievement.

4. *Complexity of the language*: Decreasing the complexity of the language that is supposed to be handled by ASR is made by limiting the probable spoken statements, and imposing syntax and semantic of these statements.

5. *Circumstances*: The locations of real applications usually present difficult circumstances, such as interference signals, noised signals, and transmission media variability, which can severely corrupt ASR system performance.

From the challenges listed above, it is easily to note that the challenges posed by ASR application occur as they shared a property problem, which is the huge variety of

patterns that may be produced for the same speech term. This note agrees the fact that speech data is a good example of complex-ambiguity data.

The automatically generated ASR model from RFL4ASR IDE system is aiming to recognise multiple phonemes of different speakers in different accents and environments that is a data of complex ambiguity, thus, this model is supposed to treat challenges 1 and 5 listed above. Furthermore, its ability to handle some challenges of DM stated in Section 2.4, are to be studied as will shown in the next chapter.

Finally yet importantly, RFL4ASR has been implemented by using Matlab Ver7/Release14, and has been tested by using TIMIT data.

6.1.1 The Standard Architecture of ASR

As shown in Figure 6.1, existing ASR systems are made up of two stages. The first stage is called feature-extraction or front-end processing stage that seeks to extract the linguistic message and hold back non-linguistic sources of variability. The second stage is called the classification or back-end processing stage, which encompasses language modelling, and it aims at identifying the feature vectors with linguistic classes [62, 79, 105, 134].

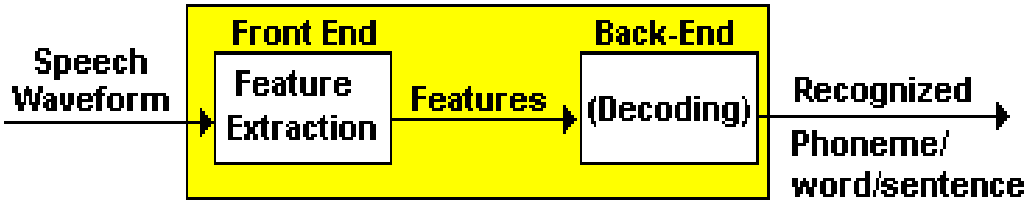


Figure 6.1 ASR Components

The processing approach of ASR system begins with the extraction level, which converts the input speech signal into a series of low-dimensional vectors, with each vector summarising the essential sequential and spectral performance of a short segment of the acoustical speech input. The generated feature from front-end processing stage is a vector of numbers that represents identification of the input speech wave. There are different techniques to calculate this vector, in which each of them has its pros and cons. The ultimate goal of ASR system is accomplished by the second stage by approximate

satisfactory statistics that are necessary for distinction among various phonetic units, while minimising the computational requirements of the classifier [53, 105]. Appendix B gives more details about each component of ASR.

6.2 RFL4ASR System

As stated earlier, RFL4ASR system, developed in this thesis work, is an IDE or shell, which aims to create and test a RFL-Based speech recognition model. IDE is defined as "a set of tools that aids application development. Among many other SW development tools, most IDEs usually have an editor for writing source code, a compiler (or interpreter), debugger, and currently GUI" [96].

Essentially, the suggested RFL4ASR IDE, which is intended to be used for developing and testing a special-purpose SW that is ASR here, is of a scope different from the traditional IDE software, which is currently used for developing general-purpose software. Of course, such major difference imposes important changes to the components of the usual IDE, in which these changes should support the developing and testing of the special-purpose SW targeted by this special-purpose IDE. Thus, instead of having compiler, debugger ...etc, which serve developing general-purpose SW, the special-purpose IDE should contain different components for create and test such special-purpose SW, which are in our case the tools for developing and testing each of front-end and back-end processing units of ASR, as well as some other supporting tools. The resulting ASR from RFL4ASR has the same components as the traditional ASR, but with the new implementation of back-end component, in that 'ML/RFL-Based Net' is to be used to develop this component. The Front-End (speech signal representation) of the resulting ASR, is to be implemented by using the traditional technique of Mel-Frequency Cepstral Coefficients (MFCC) with 13 features. Pre-processing is required in the form of phonemes data collection, which are to be gathered from different speakers in different dialects in various circumstances.

Thus, RFL4ASR can be considered as an example for using IDE principle as a framework for developing special-purpose types of software not limited to programming

languages environment, in which such conclusion can lead to an important enhancement of programming techniques.

RFL4ASR IDE system has two main jobs: the first is the construction (including training) of the recognition model while the second is the testing of this model. Input test-data (patterns) of different phonemes, which are produced by different speakers in different environments and different delegates, are to be prepared earlier and stored in files (file for each phoneme) in a certain folder. RFL4ASR shell provides the ability to select a certain phoneme file to be tested by an active ML/RFL-Based Net classification model.

The Data Flow Diagram (DFD) approach, which graphically characterise data processes (represented by rounded rectangles), data flows (represented by arrows), entities that communicate with the system (represented by rectangles), and data stores (represented by numbered open-end rectangle) in information systems [66], is used here to sketch the architecture of RFL4ASR. Figure 6.2 illustrates the logical design of RFL4ASR system using DFD (level 0 diagram). This system consists of the following eight main modules:

1. User Interface
2. Phoneme Extractor (get-phoneme)
3. Phoneme Selector
4. Phoneme's Features Extractor
5. Create ML/RFL-Based
6. ML/RFL-Based Loader
7. Phoneme Recogniser
8. Assess performance

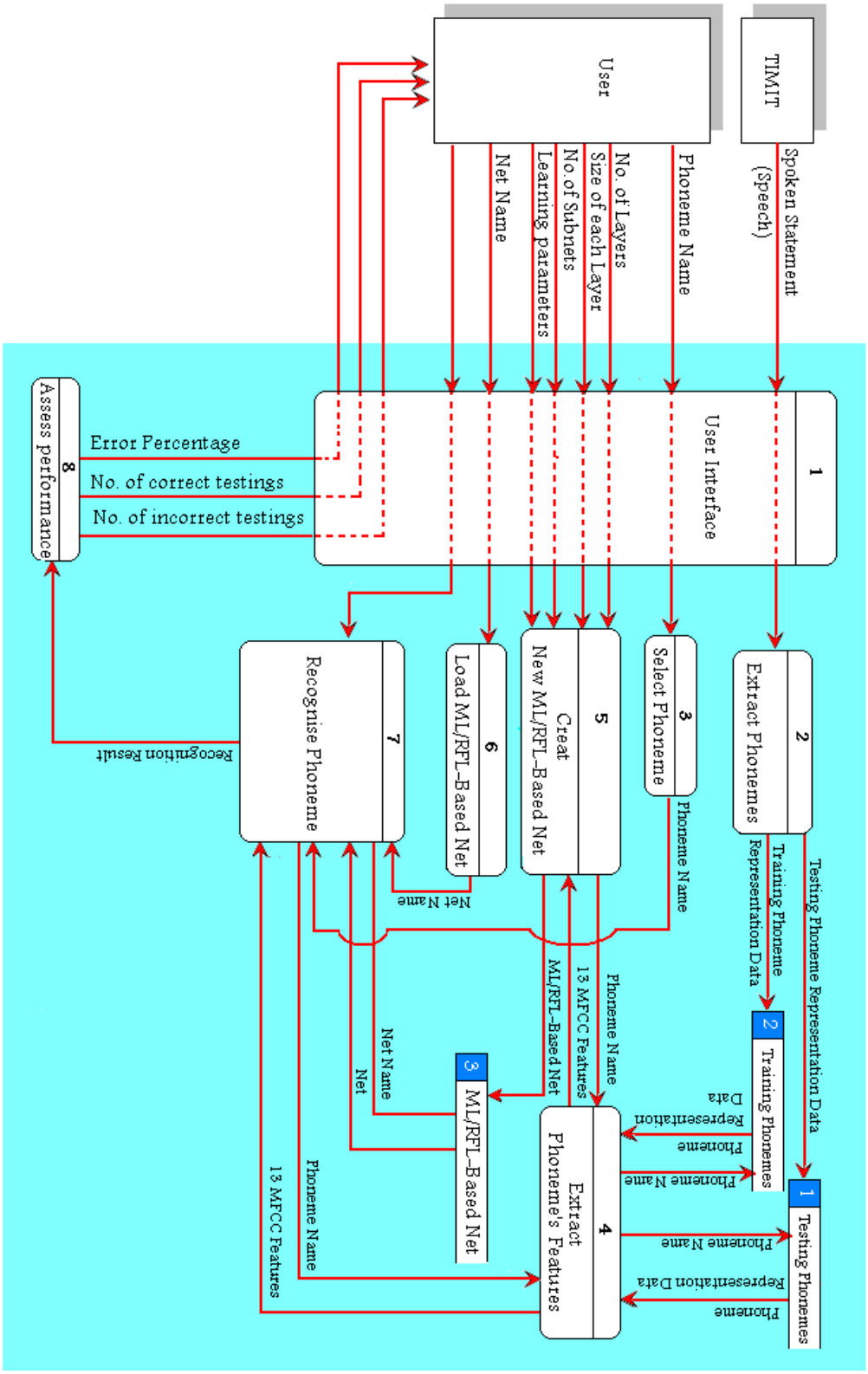


Figure 6.2 Logical Design of RFL4ASR System

RFL4ASR system also consists of three data stores, which are implemented as folders, the first folder is for collecting the testing phonemes, and the second is for collecting training phonemes, which are both to be extracted from spoken statements. The third folder is for the created ML/RFL-Based Nets, which have been called MLNets. The entities that communicate with RFL4ASR system are TIMIT files (the source of spoken statements) and a user. Following paragraphs illustrates each of these components. Following paragraphs illustrate each part of RFL4ASR system.

6.2.1 Speech File (Data)

It is an acoustic-phonetic speech corpus created by Texas Instrument (TI) and Massachusetts Institute of Technology (MIT), from whose acronyms this corpus gets its name TIMIT. The design of the corpus text was a cooperative work among the MIT, Stanford Research Institute (SRI), and TI. In fact, speech recording was done at TI, while transcribing was made at MIT. The National Institute of Standards and Technology (NIST) with sponsorship from the Defence Advanced Research Projects Agency's Information Science and Technology Office (DARPA-ISTO) maintained, verified, and prepared TIMIT CD-ROM [126].

TIMIT database contains three types of files for each utterance (each sentence's directory contains 3 sentence-related files). The file types are as follows:

- .adc – waveform with Carnegie-Mellon University (CMU) designed header, a binary file with sampled data
- .phn - an ASCII file containing a time-aligned broad acoustic-phonetic transcription
- .txt – an ASCII file containing an orthographic transcription (prompt form)

TIMIT database consists of the utterances of 630 speakers representing the major dialects of American English. Table 6.1 illustrates the codes of dialect region identifier.

Characters	*Meaning*
dr1	New England
dr2	Northern
dr3	North Midland
dr4	South Midland
dr5	Southern
dr6	New York City
dr7	Western
dr8	Army Brat (moved around)

Table 6.1 Dialect Region Identifier Codes

The speaker identifier is decoded by using a three-part code: the first character denotes the speaker's sex, the next two the speaker's initials, and a number differentiates speakers with identical initials (the first of more than one speaker with the same initials has an appended 0; the second has an appended 1, etc.)

Each speaker directory contains 10 sentence directories coded by two characters, which indicate the sentence type, and a number indicates the sentence number. The sentence types are coded as follows:

- '***sa***' stands for Stanford Research Institute (SRI) dialect calibration-shibboleth sentence (2 per speaker). (A shibboleth sound is one that gives away the nonnativeness of a speaker. This second criterion, then, measures how well a speaker's ability to pronounce a certain phoneme predicts the overall nativeness of that speaker.)
- ***'si'*** stands for Texas Instrument (TI) random contextual variant sentence (3 per speaker).
- ***'sx'*** stands for Massachusetts Institute of Technology (MIT) phonetically compact sentence (5 per speaker).

Two internal sets of data are to be developed using TIMIT data source in RFL4ASR system. The first one is the set of training phonemes, which are stored in TriningPhoneme folder that acts as a data store in RFL4ASR system. This set of

phonemes is used for training the generated ML/RFL-Based Net. The second set of phonemes is the testing phonemes saved in TestingPhoneme folder, which acts as the second data store in RFL4ASR system. This set of phonemes is used for testing the generated ML/RFL-Based Net. The developing of these two sets is governed by a phoneme extractor module, which will be illustrated in Section 6.4.3 later.

6.2.2 User Interface

Graphical User Interface (GUI) strategy has been used to make the RFL4ASR IDE system easy to use. This user interface includes the calling of processes, which the application presents such as getting a phoneme, creating, testing, as well as the results return from RFL4ASR system (like displaying classification's results and error messages). Thus, the user interface fans out information between user and RFL4ASR system. The callings and results are implemented as buttons, menus and scroll bars.

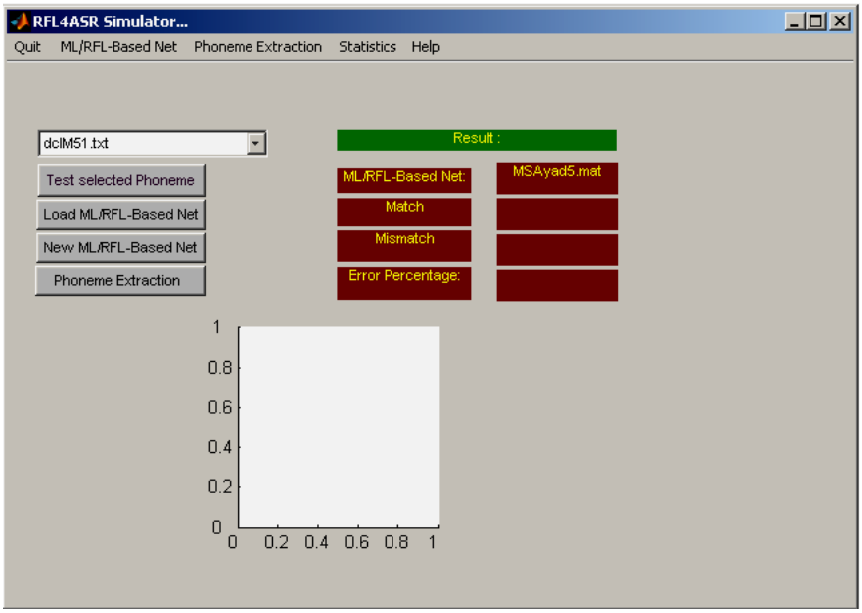


Figure 6.3 GUI of RFL4ASR System

Figure 6.3 illustrates the GUI of the RFL4ASR IDE system, which has been developed using Graphical User Interface Development Environment (GUIDE) supplied by Matlab. A number of ready-made functions provided by Matlab facilitate the implementation of option included in this module.

6.2.3 Phoneme Extractor (Get_Phoneme Module)

The data used by RFL4ASR are phonemes not sentences as TIMIT data source of this system supplies; this imposed a stage for extracting phonemes from TIMIT's files of spoken sentences, which is achieved via *Get_Phoneme* module. The input data to this module is a TIMIT file of spoken sentence, which belongs to either training or testing sets. The output of this module is a number of phoneme files in text format, which contain the frequencies of each phoneme. Each file of data phoneme is given a phoneme's name of these data. *Get_Phoneme* module plots the signal's representation of these phonemes on the screen along with the playing of the input spoken statement.

Technically, the (.adc) file of a waveform is organised to have a CMU-designed header followed by the digitised speech data. A binary representation methodology, called VAX, is used to represent each saved field. The data type of these fields is short, (or two bytes) size. On the other hand, the field's size of int data type is four bytes. This makes the size of the header twelve bytes (note that the sample frequency is 16 kHz or 16-bit samples). These fields should be clearly defined so various processing programs can use them effectively [126].

The header has been coded as a structure (or record) that contains number of fields, each to hold certain information related to the waveform itself. The header of the waveform file (of extension .adc), as defined by CMU, is coded in C language as following [126]:

```
struct ad_head
{
  short ad_hdrsize;     /* size of header in 2-byte words. */
  short ad_version;     /* version number. */
  short ad_channels;   /* number of channels recorded. */
  short ad_rate;         /* sampling rate. */
  int   ad_samples;     / * no. of samples of type short in the remainder of the file. */
}
```

The transcription files (.phn) are phonetically based, and saved as ASCII text files. These transcription files include a list of descriptors of phonetic label, and are separated by carriage returns character. CMU / ARPABET defines standard IPA phonetic symbol. Each phonetic symbol label encompasses two integer numbers (start and end sample-numbers for the phone), followed by an ASCII representation, e.g. 2200 4280 sh. Note that the ARPABET is a selection of symbols, used in the Advanced Research Projects Agency Speech Understanding Research (ARPASUR) project [126].

The orthographic transcription files, which have (.txt) extension, are ASCII text files containing the start and end sample-numbers of the complete utterance and sentence prompt type. These are sample-numbers, not milliseconds or other units of time [126].

The information reported above has been used to maintain the following algorithm that has been implemented as a Matlab function:

Function phon_extract( )

Select, and open (for read) a file of spoken statement;
Initialise number of phonemes in the selected file & eof marker;
Read the start-sampling integer & finish sampling integer for each phoneme & phoneme name up to the last phoneme in the file (marked with h#);
Read the wave file format of the selected file for playing purposes;
Read the text file format of the selected file to get the actual length of the wave file format (number of wave values), which is the second string in the file capture the sampling integers for each phoneme in the spoken sentence;
Plot each phoneme in the sentence;
Play the sentence;
Save Features of Phoneme by:

Get the current phoneme name to create a file holding phoneme name;
Open an output file or create it for each phoneme;
Save the wave sample for each phoneme;

End

A TIMIT file of spoken statement is to be selected prior activation of this module. While it tokenises phonemes from an input spoken statement, the module plots the signal wave of this spoken statement (in time domain), plays the spoken statement, and displays its phonemes ID. Each extracted phoneme will be saved in a separated file in phoneme folder that acts as a simple phoneme store, and each file holds the phoneme name and contains its time domain representation. Figure 6.4 shows the results of running Get_Phoneme module: extracting individual phonemes, and plots their sign waves.

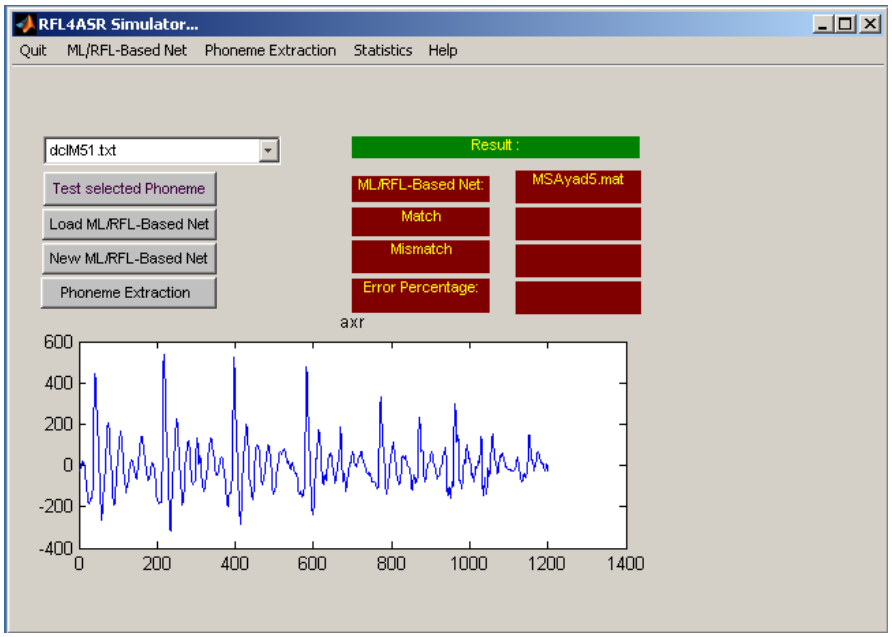


Figure 6.4 Output of Get_phoneme module

The playing of the phoneme files shows the good quality and intelligibility of the tokenising process. However, the phonemes extraction was reasonably well accepted in all the experiments when the phoneme's representation was forwarded to the next phase; i.e. recognition.

The phoneme identification is used as a name of the file of the representation values of that phoneme. This naming approach helps measure the accuracy of different ML/RFL-Based classification (recognition) models, which are generated by RFL4ASR IDE system.

6.2.4 Select a Phoneme

The aim of this module is to select or pick a phoneme from Testingdata folder,. The importance of this module is to feed certain phoneme data file to the feature extraction module to prepare it for recognition, as will be shown in Section 6.2.5. As it is shown by Figure 6.5, the selecting of a phoneme data file is achieved via a pulldown window that displays all phonemes files available in the Testingdata folder.

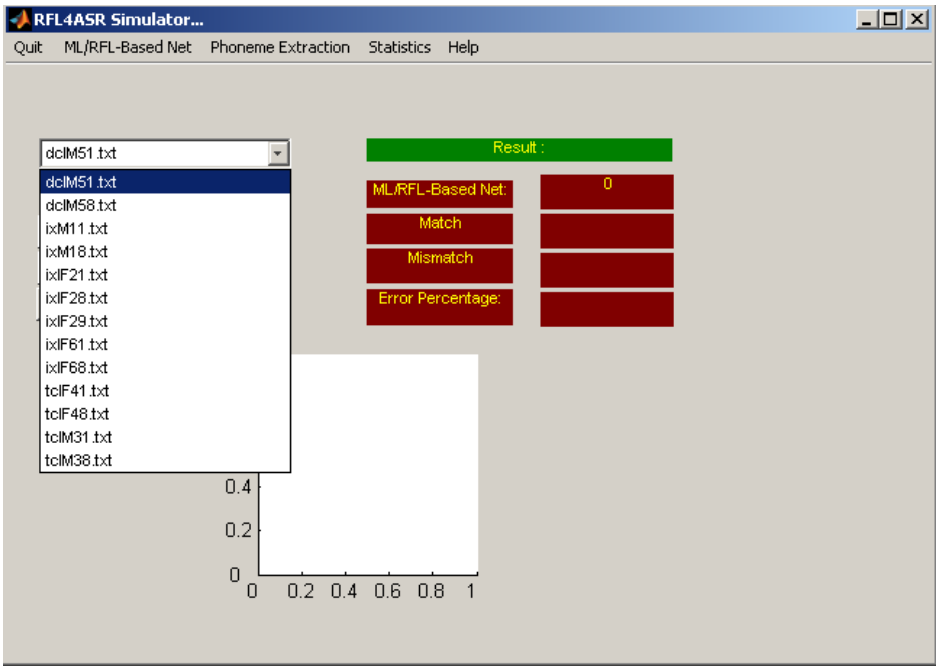


Figure 6.5 Pulldown window of select phoneme module

6.2.5 Extraction of Phoneme Features

Among multiple approaches for defining speech signal's feature set, RFL4ASR implements the Mel-scale Frequency Cepstral Coefficients (MFCC) technique to produce a feature set of the phoneme speech signal that will be used for recognition purposes. Generally, the selecting of certain feature set of the speech signal over another depends on some factors, such as the system's cost, speed, and accuracy of recognition.

Researchers reported that in the mel-frequency scale, the log-magnitude Discrete Fourier Transform (DFT) magnitude spectrum is frequency-warped to follow a critical

band scale (mel-scale) and amplitude-warped (logarithmic scale) before computing the inverse DFT parameters. Therefore, Q band pass filters are used to cover the required frequency range. Figure 6.6 illustrates the flow diagram of MFCC computation procedure [24, 131]. Get_MFCC[Cep] function has been built by using Matlab, which extracts MFCCs array of that phoneme, and stores this array in a file. This module has no window as other modules, but it internally serves both creating ML/RFL-Based module and testing ML/RFL-Based module as the DFD of RFL4ASR shows.

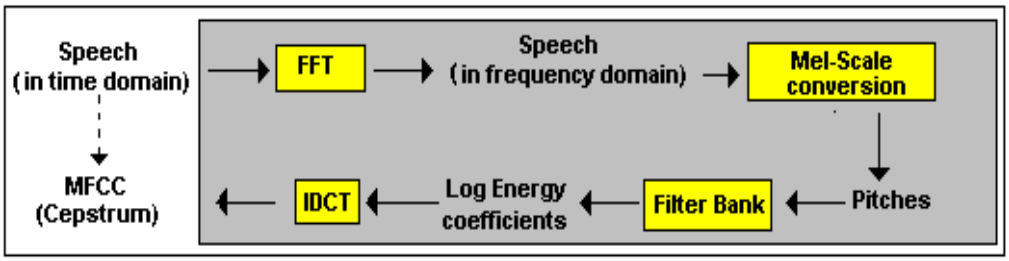


Figure 6.6 MFCC Computation Flow Diagram

6.2.6 Create ML/RFL-Based Net

This module gives RFL4ASR IDE system the ability to produce many classification models (as a predicting DM model) of type ML/RFL-Based Net for speech recognition, and saving each one in an external file in MLNets folder, thus maintaining a simple ML/RFL-Based Nets store.

The architecture of the ML/RFL-Based Net, which consists of two hierarchical levels, namely supernet and subnet, makes ML/RFL-Based Net different from traditional single level structure ANN. This results that ANN's developing platform provided by Matlab is not suitable to be directly used for handling creating, training, or even testing of the classification model of type ML/RFL-Based Net. Therefore, the programming environment of Matlab has been used for developing the ML/RFL-Based Net developer instead.

As the DM predicting model of type ML/RFL-Based Net has been built using the programming functions provided by Matlab, the traditional backpropagation ANN programming structure is used to build the supernet portion of ML/RFL-Based Net, and

the subnet portion has been implemented as an array of backpropagation ANN. Thus, the procedure for creating ML/RFL-Based Net used in RFL4ASR system, which is implemented using the programming environment of Matlab can be described as follows:

Procedure MLCreatting ( )

For each subcategory
Read Training Phoneme of this subcategory
Get the parameters of the subnet
Train the subnet
Get the parameters of the supernet

Train the supernet

Save both supernet and subnets

End

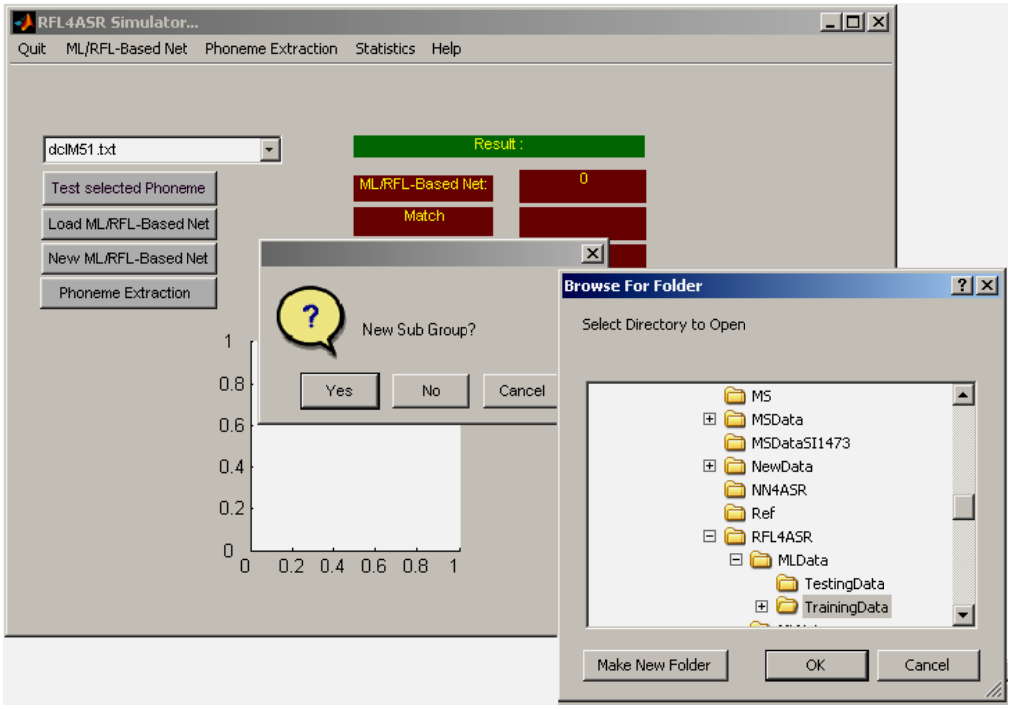


Figure 6.7 'RFL4ASR's creating window

The creating of a new instance of ML/RFL-Based Net is achieved by submitting the required initial data for developing a new net by using GUI interactive window(s)

strategy. These initial data are the number of hidden layers, the size of each layer, and the training parameters also as Figure 6.7 illustrates.

Technically, Matlab supports multiple versions of ANN's creating command: *new*. The choice of the most suitable one among these sets of '*new'* commands depends on the type of ANN that is intended to be created. The Matlab's *'new'* command of type backpropagation is selected to construct ML/RFL-Based Net since the types of both supernet and subnet of ML/RFL-Based Net is a backpropagation ANN.

Construction of a new classification model involves also the training of ML/RFL-Based Net. Complex_Learning procedure, which is defined in Chapter Five, is used to train ML/RFL-Based Net. The training of subnets of ML/RFL-Based Net type classification model is carried first, and that of the supernet is to be followed. This is because of the possibility of the unknown number of subnets to be included in the ML/RFL-Based Net type classification model under construction.

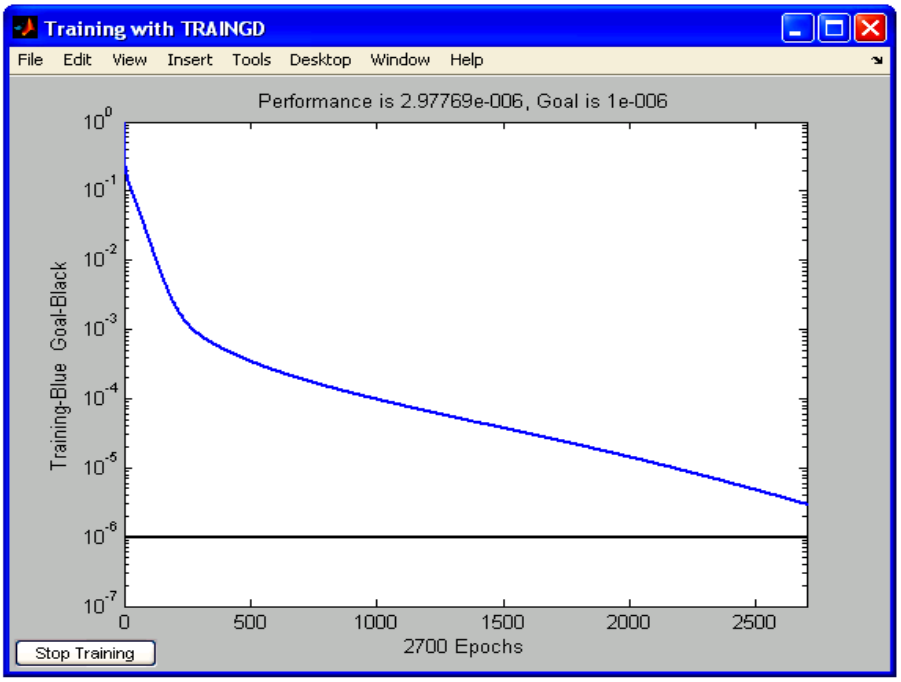


Figure 6.8 Training of ML/RFL-Based Net Neural Network

The training of ML/RFL-Based Net has been implemented by using the Matlab's command: 'train', which takes the training data and the accuracy level of matching,

which is termed as goal Matlab's ANN train command, along with maximum training iterations to reach this goal. It is important to mention here that the training may not reach the specified goal within the iterations specified. Figure 6.8 shows the window that appears while training an ANN (supernet or each of subnets) of the ML/RFL-Based Net type classification model.

In case of failing to reach an acceptable level of accuracy, a new structure (increasing/decreasing the size of a hidden layer or increasing/ decreasing the number of hidden layers) is going to be applied, which is the same approach followed to train the conventional ANN.

6.2.7 Loading Recognition Module

This module do loader job of a ML/RFL-Based Net from its file, which is saved in MLNets folder. This module is important to activate certain instance of ML/RFL-Based Net to be tested with set of phonemes.

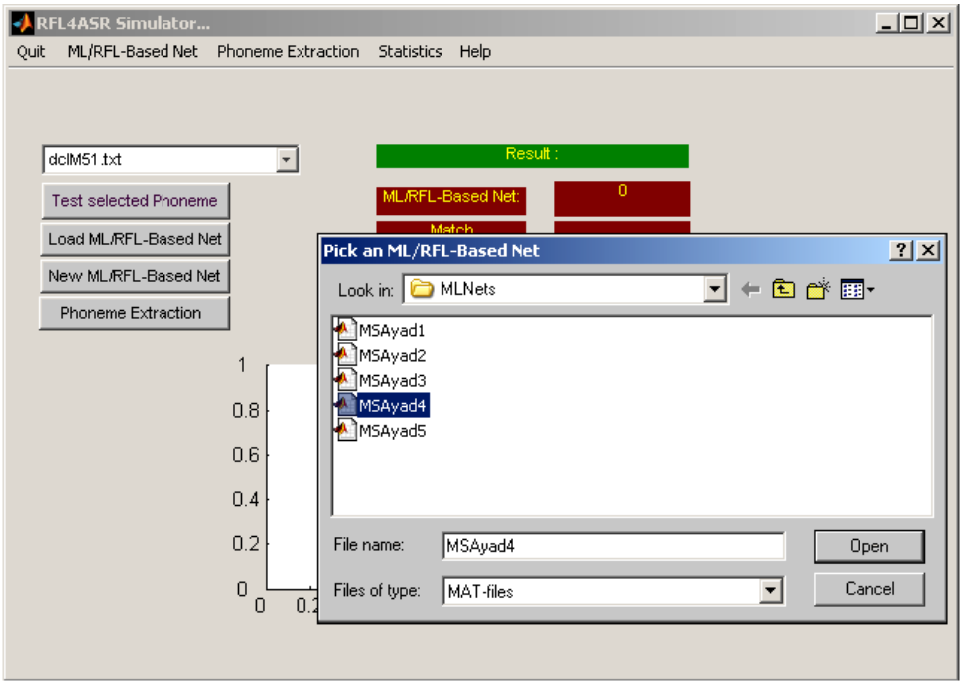


Figure 6.9 Loading ML/RFL-Based Net

As Figure 6.9 shows, picking a ML/RFL-Based Net is achieved via loading window, whose caption: pick a ML/RFL-Based Net. A loading window shows the

contents of MLNets folder, from where the user can simply select any of the displayed names to activate it. The name of the selected or activated instance of ML/RFL-Based Net is displayed on a label of the main interface. MLLoading module can be described by the following procedure:

Procedure MLLoading( )

Get the file name of ML/RFL-Based Net from MLNet folder

Read the contents of the file to a variable in memory

End

6.2.8 Testing of the Recognition Module

Many classification models of type ML/RFL-Based Net have been constructed and saved in an external file for each one. Therefore, loading a pre-trained model of type ML/RFL-Based Net from its external file is essential before carrying any testing. Also, the testing requires loading certain phoneme data by the user, which is to be read, and its contents (time domain representation of the phoneme signal) are to be converted to a 13 MFCC as it has been illustrated in Section 6.2.5. Figure 6.10 shows the window of testing a phoneme using ML/RFL-Based Net.

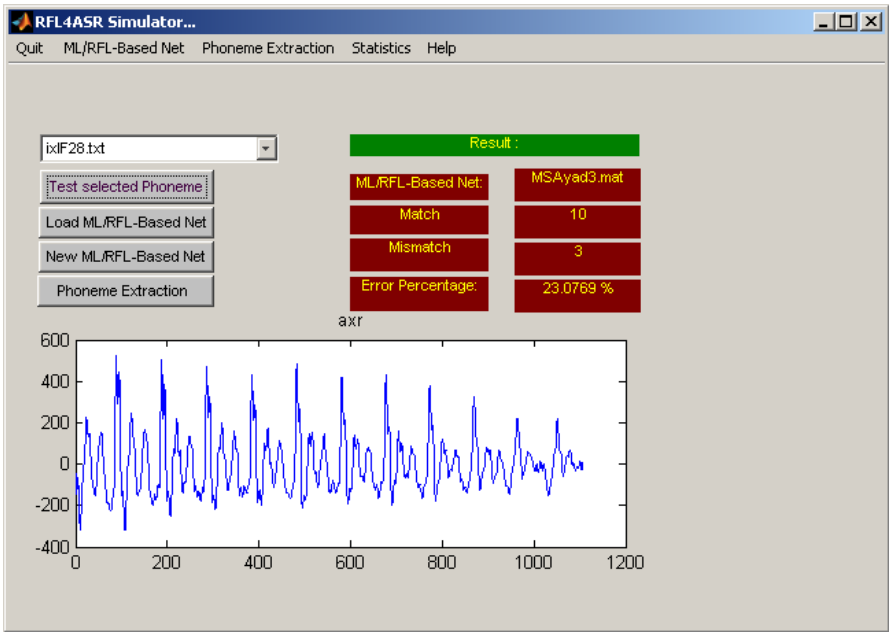


Figure 6.10 Testing of ML/RFL-Based Net Neural Network

The testing phase of ML/RFL-Based Net is performed according to deductive autoassociative recalling procedure defined in Section 5.3.3 earlier. The implementation of this recalling procedure has been achieved by using Matlab's library programming function namely: *sim* as a core one in this implementation. Of course, Matlab's *sim* recalling command for testing the uni-level traditional ANN alone is not suitable to be directly used for testing the classification model of type ML/RFL-Based Net due to the dissimilar structure of traditional ANN from the recognition model of type ML/RFL-Based Net.

6.2.9 Assessing performance

This module aims to give simple statistics that helps assessing the performance of the recognition module under test. It performs such job by producing counters for those successfully recognised phonemes and those are not, as well as the error percentage of each one. This statistics is important to make evaluation and rank each ML/RFL-Based Net instance. The results of testing an ML/RFL-Based Net classification (or recognition) model are to be displayed in the result labels of the main interface in terms of Match, Mismatch, and Error percentage, as shown in Figure 6.10. The way that, RFL4ASR shell judges whether ML/RFL-Based Net classification (recognition) model recognises the input test-data correctly or not is achieved by comparing the output of ML/RFL-Based Net classification model to the phoneme identification name provided by the input phoneme file name (which has been supplied during phoneme-extraction module). Thus, if the result calculated by ML/RFL-Based Net classification model is similar to that given by the file name of the input test pattern, the counter Match is increased by 1, otherwise the counter Mismatch is increased by 1. Accumulating results of multiple input test patterns would give a simple statistical measurement of the performance of the recognition model of type ML/RFL-Based Net generated by RFL4ASR system, which is expressed by Error percentage. This module can be described as follows:

```
Procedure  Assess( )

if the result of recognition == the name of phoneme file under test
    Then correct=correct+1
```

```
else
  incorrect=uncorrect+1;
end

error = incorrect/(correct + incorrect)*100

Display correct, incorrect, and error counters

End
```

Chapter Seven illustrates evaluation of the ML/RFL-Based Net performance using these statistics.

6.3 Experiments

RFL4ASR has been tested using different data set and different ML/RFL-Based Net classification instances, in which Table 6.2 shows the results of testing these cases of ML/RFL-Based Net.

Net Name	Testing Data	Training Data	Correct. %
Group2	80	80	92
Group3	80	80	91
MS	80	80	87.5
MS3	80	80	100
MS23	10	70	25
MSAyad3	36	36	100
MSAyad3	13	36	54
MSAyad4	13	36	62
MSAyad5	13	36	69.2308

Table 6.2 Results of Multiple Experiments on ML/RFL-Based Net

Each of Group2 and Group3 models has been constructed to identify one phoneme of the same speaker with different pronunciation. They account for 92 per cent and 91 per cent correctness respectively.

Each of the 4 subnets consists of one input layer of size 13 nodes, two hidden layers each of size 8 nodes, and one output layer of size 1 node. Their learning function has been selected to be '*traingd'* backpropagation training algorithm. Subnets of Group2 and Group3 models of type ML/RFL-Based Net has been assigned the following training parameters:

- Training Parameter. lr : Group2= 0.0455, Group3= 0.04,
- Training Parameter. epochs : Group2= 150000 , Group3= 160000
- Training Parameter error level: 1e-6 in both

On the other hand, the supernet level of Group2 and Group3 models of type ML/RFL-Based Net is composed of four layers. The input layer is of 13 nodes size (that is the MFCC input vector) with *tansig* sigmoid transfer function. The 2nd layer is of 9 nodes size with *tansig* sigmoid transfer functions. The 3rd layer is of 5 nodes size with *tansig* sigmoid transfer functions. Finally, the output layer is of 4 nodes size (the number of different pronunciations to be recognised), with *purelin* sigmoid transfer functions. The training algorithm (via *traingd* function) is used to train the supernet level of Group2 and Group3 models of type ML/RFL-Based Net with the following parameters:

- Training Parameter lr = 0.05
- Training Parameter .epochs = 300000
- Training Parameter error level = 1e-6

Using data of a phoneme spoken by multiple speakers with different pronunciation style, 'MS' net has a correctness percentage of 87.5 % and 'MS3' net 100% – i.e. it is error-free.

MS and MS3 models have been constructed to identify one phoneme of different speakers with different pronunciation. MS accounts for 87.5 per cent and MS3 accounts 100 per cent correctness.

Each of the 6 subnets (the number of different speakers with different pronunciations) consists of one input layer of size 13 nodes, two hidden layers each of size 9 nodes, and one output layer of size 1 node. Their learning function has been

selected to be '*traingd'* backpropagation training algorithm. The following training parameters have been assigned to train the subnets of MS and MS3 models of type ML/RFL-Based Net:

- Training Parameter. lr : MS= 0.05, MS3= 0.055,
- Training Parameter. epochs : MS= 150000 , MS3= 180000
- Training Parameter error level: 1e-6 in both

The supernet level of MS and MS3 models of type ML/RFL-Based Net is composed of four layers. The input layer is of 13 nodes size (that is the MFCC input vector) with *tansig* sigmoid transfer function. The 2nd layer is of 9 nodes size with *tansig* sigmoid transfer functions. The 3rd layer is of 6 nodes size with *tansig* sigmoid transfer functions. Finally, the output layer is of 6 nodes size (the number of different pronunciations to be recognised), with *purelin* sigmoid transfer functions. The supernet level of MS and MS3 models of type ML/RFL-Based Net had been trained using *traingd* function with the following parameters:

- Training Parameter lr = 0.055
- Training Parameter .epochs = 350000
- Training Parameter error level = 1e-6

The real advanced started with using the data of multiple phonemes for multiple speakers with multiple pronunciation of each phoneme for each speaker was tested using MSAyad3 net, and again gave a correctness percentage of 100 %. The supernet of this model consists of one input layer (of size 13 nodes), 3 hidden layers of size 8, 7,8 nodes respectively, and one output layer of size 3 (the number of speakers). This supernet level had been trained using *traingd* function with the following training parameters:

- Training Parameter lr : 0.055
- Training Parameter . epochs : 450000
- Training Parameter error level : 1e-6

Each of the 3 subnets (the number of different speakers with different pronunciations) consists of one input layer of size 13 nodes, two hidden layers of size 9, 8 nodes respectively, and one output layer of size 1 node. Their learning function has been selected to be '*traingd'* backpropagation training algorithm. The subnets of MSAyad3 model of type ML/RFL-Based Net had benn trained using the following parameters:

- Training Parameter. lr : 0.055,
- Training Parameter. epochs : 160000
- Training Parameter error level: 1e-6 in both

Testing data that have never been used in the training set (multiple phonemes for multiple speakers of multiple pronunciation of each phoneme in each speaker data) was performed on nets MSAyad3, MSAyad4, and MSAyad5. Actually, the complex ambiguity is quite clear in this set of data.

Training Set			**Testing Set**
DR2	DR5	DR7	dclM51
ixM12.txt	tclM32.txt	dclM52.txt	dclM58
ixM13.txt	tclM33.txt	dclM53.txt	ixlF21
ixM14.txt	tclM34.txt	dclM54.txt	ixlF28
ixM15.txt	tclM35.txt	dclM55.txt	ixlF29
ixM16.txt	tclM36.txt	dclM56.txt	ixlF61
ixM17.txt	tclM37.txt	dclM57.txt	ixlF68
ixlF22.txt	tclF42.txt	ixlF62.txt	ixM11
ixlF23.txt	tclF43.txt	ixlF63.txt	ixM18
ixlF24.txt	tclF44.txt	ixlF64.txt	tclF41
ixlF25.txt	tclF45.txt	ixlF65.txt	tclF48
ixlF26.txt	tclF46.txt	ixlF66.txt	tclM31
ixlF27.txt	tclF47.txt	ixlF67.txt	tclM38

Table 6.3 Phoneme Training Set and Testing Set Used in MSAyad5

The best achievement is drawn from MSAyad5, which produced 69.2308% error-free results (or 30.7692%-contained errors). As shown in Table 6.3, 36 training phonemes and 13 testing phonemes have been created to test the achievement of this ML/RFL-Based Net instance. It is important to note here that the 13 testing phonemes

are excluded from the training set. Note that the name of the phoneme file is coded here as following:

- The first two characters are the name of the phoneme.
- The next character denotes the gender of the speaker (M for male and F for female)
- The next numeric digits are used for distinguishing sentence, which this phoneme comes from, much like identification number.

The number of speakers of the phonemes involved in this experiment is 3 (coded as DR2, DR5, and DR7 respectively), thus the number of subnets in this experiment is 3 backpropagation nets of three layers for each one. The size of input layer is 13 that is the MFCC representation of a phoneme, with *tansig* sigmoid transfer function. The size of the hidden layer is 10 with *tansig* sigmoid transfer function. The size of output layer is 12 different phonemes with *purelin* sigmoid transfer function.

The supernet level of ML/RFL-Based Net is trained to find out the speaker's identity via selecting a subnet represents that speaker for each input phoneme, and each subnet is trained with the same set of phonemes to find out the identity of the phoneme itself by using speaker's subnet. Each subnet in this array has been trained individually.

The tansig, was selected to be the transfer sigmoid function of both input and hidden layers in both supernet and subnets of ML/RFL-Based Net. Among Matlab's library learning functions *traingd* backpropagation training algorithm has been used to train ML/RFL-Based Net. For subnets of MSAyad5 case of ML/RFL-Based Net, these training parameters are set to the following values:

- Training Parameter. lr = 0.0455,
- Training Parameter. epochs = 150000
- Training Parameter error level = 1e-6,

On the other hand, the supernet level of ML/RFL-Based Net is composed of four layers. The input layer is of 13 nodes size (that is the MFCC input vector) with *tansig* sigmoid transfer function. The 2nd layer is of 9 nodes size with *tansig* sigmoid transfer

functions. The 3rd layer is of 5 nodes size with *tansig* sigmoid transfer functions. Finally, the output layer is of 12 nodes size (the number of different phonemes to be recognised), with *purelin* sigmoid transfer functions. The training algorithm (via *traingd* function) is used to train the supernet level of ML/RFL-Based Net with the following parameters.

- Training Parameter lr = 0.0455
- Training Parameter .epochs = 400000
- Training Parameter error level = 1e-6

It is important to report here that the above parameters of ML/RFL-Based Net have been selected by using the try and error principle that is used in developing ordinary ANN. In addition, each of these models have been created using the function of Creating ML/RFL-Based Net, which its code is given in Section C.4.

6.4 Conclusion

Speech signals are highly changeable owing to their generating mechanism, which depends on the power of air that incites the vocal cords. The nature of the vocal cords is affected by many physical or psychological reasons such as tiredness, sickness, or anger. Thus, the speech as data is a good example of fuzzy and vague data type.

In ASR, classification is performed by either logical or statistical deduction or abduction. ASR also needs to be capable of knowledge acquisition or induction. Hence, these three types of inferences; i.e. deduction, abduction, and induction make ASR one of the AI's applications [3].

According to ASR's literature, which is given in Appendix B, speech sound features (data) are random and continuous that make 'statistics using probability' become the most suitable classification technique for building ASR. Moreover, because of the sequential nature of the speech events, parallel classification is limited. HMM is a statistically insensitive approach, and ANN is a knowledge insensitive approach. Therefore, HMM proves itself as a leading technique among classification techniques since it is a Statistical Finite State Automata (SFSA), and uses statistical deduction in search for a best solution, although ANN shows promising results in some cases. A

number of researchers tried to solve the problem of each of HMM and ANN by proposing a hybrid system that involves each of them, known as HMM/ANN hybrid system, which seems to be the most suitable ASR architecture defined as of yet. This hybrid system proves that it is the most accurate ASR resulted from mixing more than one classification technique [3, 127].

This chapter describes a speaker-independent ASR system using ML/RFL-Based Net, which is created by RFL4ASR IDE software system. RFL4ASR represents a new understanding of IDE targeted for developing specific purpose software modules as an extension of the traditional IDE, which is usually used for developing general-purpose programs.

Although the ASR resulting from RFL4ASR shell utilizes the standard architecture of ASR, it is actually based on the new recognition approach defined by a new ANN architecture namely ML/RFL-Based Net, which follows the novel RF set of membership principle. The design of RFL4ASR and its implementation have been described, with the latter being carried out using Matlab version 7/ Release 14.

The data used by RFL4ASR system, namely phonemes, has been extracted from TIMIT speech data, which come in form of spoken statements with different dialects by different speakers. This extraction is achieved by using the codes accompanying the files of TIMIT data to select certain phonemes to be produced to the RFL4ASR system. These phonemes were multiple, and each phoneme has multiple dialects as pronounced by either a single or multiple speakers in different circumstances. Thus, they meet the criteria of the problem to be solved by RFL4ASR IDE system.

Developing the Front-End portion of the ASR, which is generated by RFL4ASR IDE system, imposes the pre-processing of phonemes in order to calculate their MFCC features. The development of this stage has been carried out as a function to be called in each time a new input pattern is to be processed either for learning or for testing. High-level mathematical functions in Matlab, such as Matlab's FFT, facilitate the implementation of this function.

Recognition models of type ML/RFL-Based Net, which forms the Back-End portion of the ASR resulting from RFL4ASR IDE, has been implemented by using those ready-made programming functions of Matlab, which support the handling of ANN. Although it is possible to implement recognition models of type ML/RFL-Based Net using the object-oriented programming capabilities of Matlab, we believes that instead of re-inventing the wheel, it is better to use the already existing ANN functions provided by Matlab.

The interface of RFL4ASR IDE has been implemented using GUI methodology. GUI is presently the dominant interface technique. Matlab supports the creation of GUIs in spite of the limitations found in Matlab, which is quite understandable since Matlab is a simulation of 4^{th} generation language more than being software developer like visual programming languages.

The results obtained from testing different cases of ML/RFL-Based Net and different phonemes sets show the ability of ML/RFL-Based Net as a recognition model to successfully adapt the increasing in the complexity of data uncertainty.

Finally, we can report here that Matlab proves its ability to implement various algorithms, since it supports many different processing schemes, such as file processing, mathematical functions, the arithmetic of arrays and metrics, as well as many other complicated schemes such as ANN or GUI.

Chapter Seven

Results and Discussion

7.1 Introduction

The RFL4ASR IDE system has been implemented by using Matlab simulation software and has been tested by using a selected set of phonemes. While the implementation of RFL4ASR IDE software system shows the flexibility and capability of Matlab, the test has revealed one of its most successful features: i.e. the ability to recognise phonemes excluded from the set of training data. It is important to mention here that the tested data consist of various phonemes belonging to various speakers in various environments and various pronunciations, thus underlying the complex property of these phonemes (multi-syntax and multi-semantic from a proposition point of view). The results show the ability of the speech recognition model generated by RFL4ASR IDE to deal with such data, as well as highlighting the role of both RF principle and ML/RFL-Based Net computation model, both of which have been defined by this thesis to solve such a complex ambiguity problem.

DM is the theme of this thesis, which embarks on defining RF membership value set, and ML/RFL-Based Net computation model. The results of RFL4ASR have been compared to results produced by other DM machines using the well-known software 'WEKA', which gave the same data used by the recognition model generated by RFL4ASR IDE software system.

As mentioned earlier, the recognition model generated by RFL4ASR shell has been tested by using selected sets of phonemes. These phonemes are the data used to create and test a model of speech recognition. The data have been selected to reflect the difficulties of DM as reported in Section 2.4 of this thesis, i.e. it has certain properties as illustrated in Table 7.1. The results of this testing reveal the successes of both the defined RF membership value set and the computation tool ML/RFL-Based Net in

classifying complex and ambiguous data. They also prove that speech recognition models, in general, can be considered as one of the systems generated via DM.

DM Problem	Property of data used for testing the classification model produced by RFL4ASR
Dimensionality size	13 attributes for each variable
Over fitting	Different sources of speech phonemes*
Evaluation of statistical importance	Multiple sub-models
Shifting data and knowledge	Different source speech phonemes
Omitted and noisy data	13 omitted data, Different sources of speech phonemes*
Complex relationships between fields	Speech data
Comprehensibility of patterns	Automatic Comprehensibility using ANN
Previous knowledge	Multiple knowledge
Incorporation with other systems	RFL4ASR speech recognition system

** Different speakers, Different accents, Different circumstances*

Table 7.1 Properties of the Speech Data Used To Test the Classification Model Produced By RFL4ASR

Early experiments on ML/RFL-Based Net (reported in a published paper [4]) were carried out to investigate DM principle on phoneme recognition task, which showed promising results. At that paper, confusion matrix is the method used to show the accuracy performance of ML/RFL base Net. This method is used because the set of the data phonemes is small and simple compared with the data used later, as the new method has been extended to be more sophisticated and involves more complexity.

In this chapter, the description and discussion of the results obtained from RFL4ASR system are given. An evaluation and analysis of the results are also presented. The evaluation has been conducted by making comparison between RFL4ASR's results and WEKA's results.

7.2 Error Analysis

The handling of complex ambiguity in DM (or by classification models in general) is the goal of this thesis, so accuracy is the most important factor in focus. Since

ML/RFL-Based Net has been essentially developed to test the ability of the RF principle for making classifications as a DM methodology, comparing the results of ML/RFL-Based Net classification model is made with the existing DM approaches rather than with ASR systems. (Actually, ASR involves complex of model that increases its accuracy)

The ability to recognise complex-ambiguity data (phonemes) of different speakers by using recognition models of ML/RFL-Based Net type is demonstrated in terms of error percentage appearing in the interface of the RFL4ASR IDE system. As shown in table 6.3, the data set used for training and testing was complicated from one experiment to another, but led to improvement in the classification construction (recognition) model of ML/RFL-Based Net type in terms of the structure of both supernet and subnets. The best achievement of ML/RFL-Based Net classification model was 69.2308 % of correctness (30.7692% as error percentage) via MSAyad5 instance. Hence, it is easy to conclude that this accuracy can be further improved, not only by improving mainly the supernet, but also by improving the subnets.

Although it is somehow difficult to evaluate this achievement, in as much as it is hard to find identical machines, WEKA DM package of Waikato university-New Zealand (see Appendix A) has been used as a benchmark to evaluate the achievement of this RFL4ASR. The following section will show WEKA's classification performance using the same testing data used to test the best achievement recognition model generated by RFL4ASR namely MSAyad5.

7.2.1 WEKA's Results

WEKA contains the following classification collections of machine learning algorithms for DM tasks [67]:

1. **Bayes**: contains bayesian classifiers, e.g. NaiveBayes
2. **Functions**: e.g., Support Vector Machines, regression algorithms, neural nets
3. **Lazy**: no offline learning, that is done during runtime, e.g., k-NN

4. **Meta**: Meta classifiers that use a base classifier as input, e.g., boosting or bagging
5. **Mi**: classifiers that handle multi-instance data
6. **Misc**: various classifiers that don't fit in any another category
7. **Trees**: tree classifiers, like decision trees
8. **Rules**: rule-based classifiers, e.g. ZeroR

Each group consists of many classification schemes. Among the number of alternatives for test option, 'Cross Validation' with 13-folds has been selected. This is because the testing of ML/RFL-Based Net has been carried out using 36 instances as training data, and 13 instances as testing data. Thus, the environment of testing the data in both WEKA and ML/RFL-Based Net become similar.

All classification machines of WEKA have been investigated. The testing shows that some classification schemes are not able to treat the data given to them, (Figure 7.1). A summary list of those successful classification schemes with their results is shown in Table 7.2.

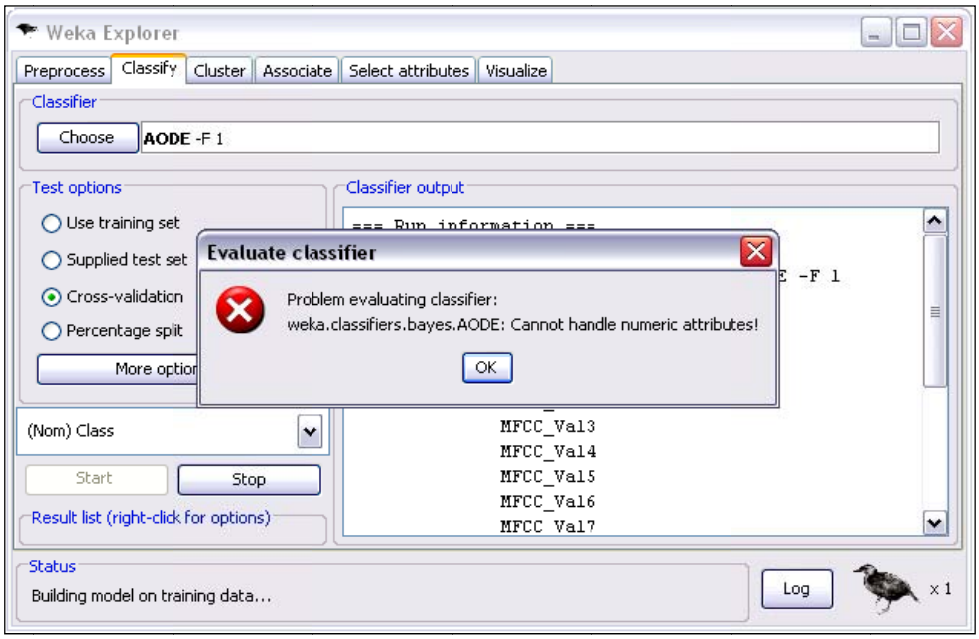


Figure 7.1 Error Message from Some WEKA Classifiers

The descending sorted list in Table 7.2 shows that NaïveBayes was the best achievement of the classification approach of WEKA package. Although the accuracy of classification obtained from other classification schemes of WEKA was close to the accuracy of the results obtained from the ML/RFL-Based Net classification model, the comparison between the best achievement of WEKA's scheme and ML/RFL-Based Net shows a clear difference. The correctness percentage of NaïveBayes was 63.2653%, while it was 69.2308% using MSAyad5 model of ML/RFL-Based Net, see Table 7.2, which is graphically illustrated in Figure 7.2. Obviously, this fact highlights the success of RF as a principle, the success of ML/RFL-Based Net as a computation model, and the success of DM as a strategy for creating predictive models for ASR (although, there is need to make some further improvement to reach the same accuracy attained by the current state of other ASR systems).

Scheme	Correctly Classified Instances (out of 49)	Correctness %	Error %
NaiveBayes	31	63.2653%	36.7347%
NaiveBayesSimple	31	63.2653%	36.7347%
NaiveBayesUpdateable	31	63.2653%	36.7347%
RBFNetwork	31	63.2653%	36.7347%
SMO	31	63.2653%	36.7347%
FLR	30	61.2245%	38.7755%
LWL	30	61.2245%	38.7755%
RandomForest	30	61.2245%	38.7755%
AdaBoostM2	29	59.1837%	40.8163%
DecisionStump	29	59.1837%	40.8163%
END	29	59.1837%	40.8163%
EnsembleSelection	29	59.1837%	40.8163%
AttributeSelectedClassifier	28	57.1429%	42.8571%
ClassificationViaRegression	28	57.1429%	42.8571%
LogitBoost	28	57.1429%	42.8571%
MultiBoostAB	28	57.1429%	42.8571%
RandomCommittee	28	57.1429%	42.8571%
ConjunctiveRule	27	55.1020%	44.8980%
FilteredClassifier	27	55.1020%	44.8980%
MultilayerPerceptron	27	55.1020%	44.8980%
NNge	27	55.1020%	44.8980%
ClassBalancedND	26	53.0612%	46.9388%

Table 7.2 Summary List of WEKA's Successful Classification Schemes

(***to be continued***)

Scheme	Correctly Classified Instances (out of 49)	Correctness %	Error %
DataNearBalancedND	26	53.0612%	46.9388%
NBTree	26	53.0612%	46.9388%
PART	26	53.0612%	46.9388%
IB1	25	51.0204%	48.9796%
IBk	25	51.0204%	48.9796%
J48	25	51.0204%	48.9796%
KStar	25	51.0204%	48.9796%
Logistic	25	51.0204%	48.9796%
nestedDichotomies.ND	25	51.0204%	48.9796%
REPTree	25	51.0204%	48.9796%
SimpleLogistic	25	51.0204%	48.9796%
BFTree	24	48.9796%	51.0204%
Decorate	24	48.9796%	51.0204%
LMT	24	48.9796%	51.0204%
Bagging	23	46.9388%	53.0612%
HyperPipes	23	46.9388%	53.0612%
RandomSubSpace	23	46.9388%	53.0612%
MultiClassClassifier	22	44.8980%	55.1020%
OneR	22	44.8980%	55.1020%
Ridor	22	44.8980%	55.1020%
SimpleCart	22	44.8980%	55.1020%
BayesNet	21	42.8571%	57.1429%
Dagging	20	40.8163%	59.1837%
Grading	20	40.8163%	59.1837%
JRip	20	40.8163%	59.1837%
MultiScheme	20	40.8163%	59.1837%
RacedIncrementalLogitBoost	20	40.8163%	59.1837%
Stacking	20	40.8163%	59.1837%
StackingC	20	40.8163%	59.1837%
UserClassifier	20	40.8163%	59.1837%
VFI	20	40.8163%	59.1837%
Vote	20	40.8163%	59.1837%
ZeroR	20	40.8163%	59.1837%
RandomTree	18	36.7347%	63.2653%
OrdinalClassClassifier	17	34.6939%	65.3061%

Table 7.2 Summary List of WEKA's Successful Classification Schemes

For speech as data, the results prove the success of MFCC as an approach to representing speech items - phonemes in the case of this thesis. Storage of speech using this methodology may result in the reduction of the size required to store speech, since speech has intensive data that should be recorded. MFCC reduced the size of the data

used for recognition purposes, but still gave slight high property dimensionality to the speech pattern.

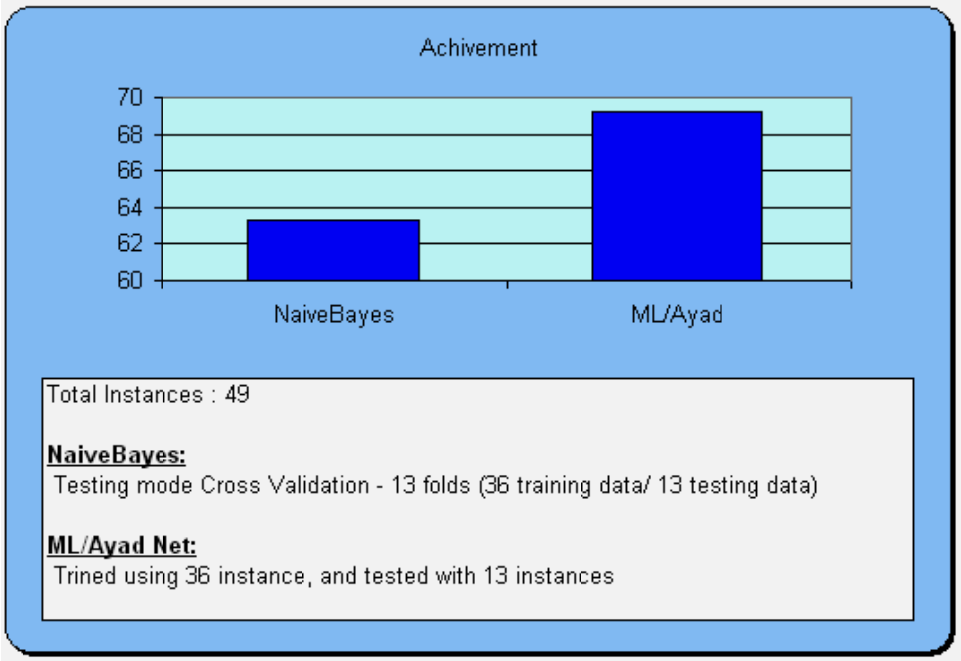


Figure 7.2 Comparison between Best Achievement Schemes of WEKA and ML/RFL-Based Net

7.3 Time Analysis

Multiple different structures for both supernet and subnets of ML/RFL-Based Net recognition model type have been built and examined. The reason behind this investigation is to obtain the best choice of the size of hidden layers (number of hidden layers as well as number of neurons in each hidden layer). Thus, the model that is able to give a certain level of accuracy within a reasonable level of training time has been selected. Actually, this shows that the best internal structure of both supernet and subnets is tied to 'trial and error' principle as much as in the traditional ANN. Note that subnets have a identical structure, but different from that of the supernet. As it is obvious, accuracy is the interesting factor of the experiments. Therefore, during the experiments no investigation to minimise the training time has been carried out due to the known fact that ANN is a time cost during training phase.

The comparison between the training times required by WEKA and ML/RFL-Based Net recognition models generated by RFL4ASR shows that the time required for training recognition model of type ML/RFL-Based Net was obviously longer than that of WEKA's.

7.4 Discussion

DM methods provide good tools for the classification of huge data. The ability to deal with huge data with a view to extracting classification models in terms of prediction or description have led to their widespread use. This confirmed record of achievements with applications speaks for itself. The DM classification methods, which usually lack the ability to deal with true uncertainty, frequently produce a set of shortcomings (listed in Section 2.4). Note that current classification methods are mainly based on either crisp or fuzzy logic.

In fact, Type-1 fuzzy methods have proven useful in employing behaviours and in judging them as well. Classification models of DM are discrimination systems that have to deal with uncertain understanding of these data given by imprecise description. In other words, DM classification methods should be able to tackle the drawbacks caused by imprecise description. Type-2 methods are more effective in dealing with uncertain data. However, they currently entail coping flexibility and simplicity to the computation required by classification models since these models are intended to process huge data. Developing a fuzzy approach that implements this new flexible and simple computation method represents a significant, non-trivial challenge. The prize from such effort would be noteworthy for sure. Thus, this thesis comes up with the definition of RFL. The field of DM based on RFL will open the door for the application of RFL in the commercial and industrial sectors. Developing, implementing, and evaluating RFL classification approach was the central objective of this thesis.

The investigation of the practical results reported in this thesis clearly shows that providing RFL to a DM system can help improve the performance of a classification system. This is confirmed by the performance enhancement obtained by the computation model introduced in this thesis. In addition to accuracy, the other factor that determines

the effectiveness of the RF approach is the ability of the RF-based classification approach to capture new explanation of the data whenever information is available about this new explanation. However, the comparative achievements evolved should be addressed.

It is good to understand, in particular, why ML/RFL-Based Net is time consuming. One would wish less training time of ML/RFL-Based Net. However, there are two aspects of this question. The first, ANN itself is a time consuming computational model. This is a principal shortcoming that has been recorded in this concern. Due to the nature of ML/RFL-Based Net as a complex ANN, composed of hierarchically organised backpropagation ANN, the time required for training recognition model of ML/RFL-Based Net type was obviously longer than that of WEKA. The second is that the implementation of ML/RFL-Based Net has been carried out using Matlab, which is a 4^{th} generation language as compared to WEKA algorithms, which have been implemented using Java. This fact highlights the need to develop a new computation model that works according to RFL approach by using a lower level programming language than Matlab. Part of this development is the defining of new algorithms for training ML/RFL-Based Net. Another drawback of ANN, which appears also in ML/RFL-Based Net, is the designing of ANN. As it has been shown in Table 7.2 many architectures of ML/RFL-Based Net have been investigated until the best (namely MSAyad5) is reached. However, it is still difficult to achieve a perfect architecture for each ANN in ML/RFL-Based Net (To be precise, hidden layers pose a problem). It is important to consider the effects of the size of the training data on the results.

The choice of the complex-ambiguity data, for the experiments presented in this thesis, reflects the challenges of DM stated in Section 2.4 earlier (Table 7.1 illustrates those certain properties exist in the used speech data). Furthermore, the data involve phonemes produced by males and females of different accents and different circumstances of a speech. This means that the phoneme recognition performance of the classification model of type ML/RFL-Based Net would be largely determined by its ability to recognise phonemes. Despite these advantages, the testing data also have one major shortcoming, which may have adversely affected the results obtained from the

ML/RFL-Based Net classification model. The amount of training data is relatively small in comparison to the data that might be produced to WEKA. The data were minor in size to allow for faster turn-around time during stages of training and development of the testing of ML/RFL-Based Net. In particular, WEKA classification machines are able to handle bigger size data in lesser time. In retrospect, it is clear that the experiments conducted on 49 data items in this thesis would have required more effort to construct, and more time to run had they been performed on these TIMIT source phonemes. On the other hand, it is entirely possible that the relative performance gains, observed by the various different instances of ML/RFL-Based Net, would have been greater if the system had had the benefit of additional training data. Unfortunately, this belief cannot be confirmed unless ML/RFL-Based Net is to be tested with a larger data size of those properties listed in Table 7.1.

In this thesis, the ML/RFL-Based Net classification model utilised techniques, which were developed and refined for the task of multi-classification of input phonetic data. While the techniques of the standard classification model have been under development for years, these same techniques may not be the most effective techniques for complex-ambiguity data. Any classification model aims to differentiate between different items. Based on that differentiation, either a descriptive or a predictive model is to be built. The purpose of the RFL based classification model, presented in this thesis, is mainly designed to build up a predictive model that is able to classify complex-ambiguity data. This difference between the traditional DM models and the proposed ML/RF based model is propelling our expectation towards new improvements of the capabilities of the DM classification approach.

The choice for using the technique of phoneme error rate (correctness term in RFL4ASR IDE) as the decision metric in this thesis should be discussed. The technique of calculating phoneme error rate is the standard evaluation metric for many phoneme-level recognition tasks as a method of evaluation. Because the RFL based techniques introduced in this thesis are designed around the classification (recognition of phonemes) model, the phoneme error rate metric could be more useful for error analysis.

It is sensible to question whether an analysis of the RFL approach presented in this thesis on the task of phoneme recognition would have been profitable on other data type or not. Actually, this thesis started with the aim of using DM technique for speech recognition. However, this aim was slightly shifted because the preliminary investigation of DM problems leads to define RFL as a method to encounter the classification of complex-ambiguity data (like speech data).

Finally, it is worth discussing the choice of selecting IDE technique to develop the case study of this thesis, which is RFL4ASR. Actually, traditional software architectures have fallen by the wayside, replaced by object oriented (class-based) architecture consisting of data acquisition portion and processes developer. With Matlab's IDE, implementation becomes easier than traditional ones since it provides many library functions, which are able to define the structure, train and simulate or test an ANN.

Chapter Eight

Conclusions and Future Work

8.1 Introduction

A careful study of data mining and analysis of classification approaches towards creating data models underscores a number of shortcomings. This thesis has found out that these shortcomings share one reason that is complex ambiguity. While type-2 fuzzy logic is already defined to handle such problems, this thesis gives a clearer definition for complex ambiguity hence found a case that is not considered by type-2 FL. This thesis provides also a new method to quantify the membership of an element to a set under condition of complex ambiguity. Thus, the research has worked on an atomic level of classification, which is novel membership value set, rather than crisp and fuzzy ones. This novel membership value set is called *Relative-Fuzzy*, along with functions to calculate this new membership value. A comparison between type-2 FL and the defined RFL is presented to show the progression made in handling the uncertainty using RFL. In addition, a novel computational model for classification has been developed, which works according to the defined RFL. This computational model is called 'ML/RFL-Based Net'. A comparison between traditional HNN and the proposed ML/RFL-Based Net is given. Last but not least, a case study on the new membership value and new computational model has been presented by this thesis, which is a prediction (classification) model for speech recognition. Due to its properties, speech is used as the data by which the novel computational model, as proposed in this thesis, would be tested. Thus, a novel automatic speech recognition model, based on the relative-fuzzy principle of DM, has been defined.

In this thesis, the need for a new RF membership value is underlined, which is the quantification of description of belonging of an element to more than one set at the same time. The suggested RF membership value set presents a new view and new approach to handle ambiguity (a type of uncertainty) belonging of an element to a multiple set. RFL

is based on the understanding of 'possible worlds' logic. Principally, each set of numbers is generated via a function. Therefore, this set of membership values are to be generated via two separated functions as shown in Chapter Four.

Since RFL is new, no previous RFL based computational model or system is defined. Actually, the definition of the relative-fuzzy membership function makes the definition of computation model easy. Chapter Five suggests the novel 'ML/RFL-Based Net' computation model that uses Relative-Fuzzy principle in its work.

The proposed 'ML/RFL-Based Net' computational model is used to create a speech recognition model. The selection of speech as a data for testing this thesis work is due to the nature of the speech itself, which could be affected by many physiological and psychological causes. The property of speech signals as data is that no speech item (phoneme, word ….etc) comes with the same values (representation) all the time. This is what makes the number of the forms for each speech item unlimited (huge data). Hence, complex-ambiguity property of speech data provides the challenge of DM defined in Section 2.4 of this thesis. It is clear from this property of speech that it is impossible to define a unique value (as a threshold) for each speech item in order to be used for classification (recognition) purposes. So, it is impossible to make a classification by using comparison in the sense of identical matching (crisp principle). On the other hand, the use of fuzzy comparison would be more suitable, even though a need for other techniques to enhance the performance of fuzzy system is essential.

A novel Automatic speech recognition system based on new Relative-Fuzzy approach has been developed using the code generation strategy (a 4^{th} generation technique of programming) as described in Chapter Six. RFL4ASR IDE system presents a side novelty of this thesis, which is the extension of the IDE principle. In that, it illustrates the ability to develop IDE tool for specific purpose programs rather than general-purpose programs as in its usual definition. Such a property can be useful toward improving current programming development tools, like visual programming tools that gets use of object oriented paradigm (OOP).

8.2 Conclusions

1. As data becomes more meaningful and increase in complexity, variety and amount, there is a great need for developing new processing techniques that can deal with such a revolution in data property. The development of RFL will provide the capability of handling better linguistic uncertainties. This thesis successfully defines a new processing approach for the classification of multi-meaning, multi syntax data rather than using traditional techniques, most of which have been created some time ago when data and information technology were both simpler than they are now. This classification technique could be described as multi-semantic classification.

2. Applying the 'point of view' principle of possible worlds as a modal logic to type-1 fuzzy set produces one of the most interesting results of this thesis, which appears as Relative-Fuzzy Logic. Since it is fuzzy, RFL may answer the well-known contradictory phrase paradox of Pythagoras, as well as Bertrand Russell's paradox, both of which highlighted the limitation of classical logic. Relative-Fuzzy also improves Type-2 fuzzy logic by producing a new explanation of the uncertainty by using the notion of both propositions and sets further to other improvements as shown by the comparison made between Type-2 FL and RFL.

3. The Relative-Fuzzy concept can be used for the quantification of concepts in the shadow of point of view. This gives adaptability feature to software. There are a number of applications where the adaptive property is required, in which this logic may play an important role, such as Robotics, Natural Language Processing (NLP), and control systems of machines under changeable environments.

4. The definition of RFL raises the level of knowledge that the computer can deal with, current inference algorithms of crisp and fuzzy logic are both used to handle knowledge, while RFL brings the computation model along with wisdom.

5. Traditional computation is based on crisp logic, and neurocompution uses fuzzy logic. Relative-Fuzzy principle may open the way to a new type of computation, which is an extension of neural computation, that may be called multi-level

neurocomputing. Its aim is mainly to deal with the problem of many-to-many ambiguity and multi-meaning of data in the DM field. Hence ML/RFL-Based Net is a programming tool that has been designed and implemented in this thesis to support multi-level neurocomputing. A comparison between ML/RFL net and the traditional HNN shows the sort of improvements that could be made to enhance the design and the computation approach of HNN.

6. Extending the principle of IDE to include development of specific-purpose programs have been used to develop RFL4ASR. As it is defined, standard IDE contains those components used for developing general-purpose programs, which are not suitable for a specific purpose programs (like ASR). Of course, such use requires redefinition of IDE software in terms of its contents such that it fits to specific programs.

7. The feature presented by data mining, which is the ability to create a processing model (either descriptive or predictive), could be used in the constitution of a new paradigm in programming. The researcher suggests 'Computer Aided Programming – CAP' as a name for it. Actually, this ability opens the door for developing a new programming approach that the computer itself will at least help create programs.

8. The ability to add knowledge to the already existing one leads to enhance and strengthen the ability of the already running system. In that, it will make the next generation of programs *automatically adaptive software* to new environment without loosing the ability of utilising its previous knowledge. In other words, it will lead to the creation of mechanisms to allow the system to adapt to individual experts decision-making. Such a system would truly be a smart adaptive fuzzy expert system. This ability also provides the non-algorithmic solutions, i.e. ANN, a new property, namely 'inheritance'.

9. Since the principle followed in creating RFL4ASR speech recognition system in this thesis is based on multi knowledge, it is possible to develop one speech recognition system that is able to deal with multiple languages. Such a system gives

a new type of ASR, which is 'Multi-Language ASR', to the already defined types of ASR.

10. Speech data is very variable for various reasons. This makes the computation of speech recognition highly difficult when using one computation approach, especially in terms of accuracy. The currently working ASR systems have been constructed by using hybrid techniques that are a mix of more than one computation technique to produce a successful ASR.

8.3 Future Work

There are many difficulties in processing algorithms, especially classification algorithms, which traditional techniques have failed to address satisfactorily. It is a generally held belief, however, that these problems may have solutions in the growing field of Data Mining. RFL and ML/RFL-Based Net, both resulting from the present research, are largely inspired by this promise. In addition, the past decade has seen a remarkable growth in the theory of fuzzy logic systems in Data Mining. One reason of this interest has been the realisation that deterministic mathematical models with little scope for freedom can generate extremely complex behaviour. Thus, data with complicated uncertainty may be well processed by the relative approach of computation, which is presented here as Relative Fuzzy.

The research handled by this thesis can be enhanced by further investigation and practical implementation if the following factors are taken into consideration:

1. Apply further study to RFL with a view of establishing new axiomisation and inferencing approaches that integrated with the mathematics of RFL.

2. Although three comparisons were made in this thesis (between RFL and type-2 FL in Chapter Four, between ML/RFL-Based Net and HNN and RFL in Chapter Five, and between WEKA and RFL4ASR DMM in Chapter Seven), additional evaluation study of all these approaches is suggested as a future work. Furthermore, RFL needs detailed evaluation study to rank it among the other approaches of handling the uncertainty in DM.

3. Multi-layer Neurocomputing: Traditional computation theory, based on crisp logic, and Neurocomputing, which follows fuzzy logic, have inspired studies bridging the gap between mathematical theory and computation techniques, making these computing approaches applicable to various areas. Likewise, there is need to study a new type of computation based on Relative Fuzzy. Multi-layer Neurocomputing aims to define techniques for building computation models to solve problems with a certain type of data (i.e. multi-meaning data in particular). Such studies will extend the ability of current computational approaches to be able to handle the revolutionary 'data' itself.

4. Using Relative Fuzzy approach in new applications: This study uses Relative Fuzzy approach and the programming tool based on it in the ASR system. Actually, there are many areas of application that the Relative Fuzzy approach could be applied to, especially those having several meanings in their data. Examples include Information Security, NLP, Programming Languages processors (compilers, interpreters etc.) and databases. Defining programming tools that operate according to relative fuzzy approach are very important to some computing areas including Data Mining.

5. Speech unit used in recognition: Speech recognition based only on phoneme units will not be enough. Phonemes will however be good essential units for the enhancement of larger speech units - for example, spoken words that are more interested in application, or even spoken statements. This could extract new types of application that specifically distinguish different languages and improve recognition. Relative fuzzy techniques will allow the merging of different types of data and classification problems, as well as the inclusion of the dynamics of the speech signal in the model. This is likely to lead to significant improvements over current methods, which are inherently static.

6. Other speech processing techniques: Speech processing (e.g., characterisation, compression, recognition, and analysis) could be carried out using Relative fuzzy approach, which is capable of being used in important speech processing aspects

(decomposition, representation, and characterisation). By multilayer analysis, speech can be decomposed into various segments, which are used in speech characterisation through its features. This approach could be used to develop more accurate speech recognition schemes.

7. Developing IDE definition: IDE is traditionally used in developing programs. So, its components are the editor, compiler, linker, and nowadays GUI maker. Specific purpose software (like ASR) can have their own IDE, which will actually speed up and facilitate the development of such specific purpose programs.

8. Computer Aided Programming: The ability of current techniques in data mining to create models for processing data automatically can lead simply to a new approach in programming that the computer itself will be a tool in program creating. Of course, a detailed study is required to make such paradigm mature and include all programming techniques and problems.

Appendix A

Tools Used to Develop RF4ASR

A.1 Matlab

The implementation of RFL4ASR uses Matlab Ver7 / Release 14. This is because Matlab supports many programming schemes from simple array and metrics calculations up to more complicated ones such as handling ANN, and creating Graphical User Interface (GUI). Matlab consists of many components, such as [95, 98]:

- Interactive environment: This feature is given through a set of graphical interfaces (GUI principle), which supports the fast solution of both mathematical and engineering problems. Such interfaces permit finding files, performing functions given in Matlab, viewing earlier executed commands (known as history), viewing data structures, and plotting.

- The programming environment: This environment of Matlab provides two programming facilities, namely an interpreted scripting language and a set of programming support tools. Other tools used for programs development in this environment include an editor, debugger, and code profiler. Furthermore, supporting for access to source-code management system from Matlab is available.

- Built-In functions: This set of functions is considered as core of the Matlab engine. Built-In functions provide a large number of optimised, high-level functions, which allow the fast prototyping of scientific and engineering programs. The essential engine of Matlab consists of functions for input-output operations, matrix manipulation, plotting, interpolation, data analysis, linear and non-linear methods, sound processing, and numerical solutions.

- Toolboxes: A big number of add-on toolboxes and other components are available, in addition to the primary Matlab engine stated above. The importance

of these special purpose applications is to supply sets of different functions for specific purposes, such as the control system, data acquisition, symbolic mathematics, fuzzy logic, signal processing, neural networks, and statistics.

- Supported platforms: Matlab can work under a variety of operating systems, like Microsoft Windows (with its different versions), Mac, Linux, and a variety of UNIX systems including Sun Solaris, Silicon, Graphics, and Compaq Alpha.

Matlab is considered as one of the best SW tools for developing ANN based applications. This developing may be achieved either by its NN interactive mode, or as Matlab code via calling its library NN functions. Creating, learning, and testing are the most important among Matlab's functions for dealing with ANN.

For developing an ANN, Matlab gives a number of transfer sigmoid functions like tansig, logsig, or purelin. The tansig, which was selected to be the transfer sigmoid function of both input and hidden layers in both supernet and subnets of ML/RFL-Based Net, is a hyperbolic tangent sigmoid transfer function that calculates a layer's output from its net input. Purelin is a linear transfer function.

The library learning functions of Matlab include trainlm, trainbfg, trainrp, and traingd. The traingd learning function, which is backpropagation training algorithm, has been used to train ML/RFL-Based Net. *Traingd* is used to calculate derivatives of performance (perf) of both weight and bias variables (Xs) respectively. Each bias variable (X) is adjusted according to gradient descent given by: $dX = lr * dperf/dX$, where (lr) variable is the learning rate. Training stops when any of the following conditions occurs, the maximum number of repetitions (epochs) is reached, the maximum amount of time has been exceeded, or the performance has been minimised to the goal [95, 98]. Traingd is a network training function that updates weight and bias values according to gradient descent. Traingd can train any network as long as its weight, net input, and transfer functions have derivative functions [95, 98].

A.2 WEKA

WEKA**,** Figure A.1, is a software developed by Waikato University-New Zealand. WEKA contains the following classification collections of machine learning algorithms for DM tasks [67]:

Figure A.1 WEKA Software

1. **Bayes**: contains bayesian classifiers, e.g. NaiveBayes
2. **Functions**: e.g., Support Vector Machines, regression algorithms, neural nets
3. **Lazy**: no offline learning, that is done during runtime, e.g., k-NN
4. **Meta**: Meta classifiers that use a base classifier as input, e.g., boosting or bagging
5. **Mi**: classifiers that handle multi-instance data
6. **Misc**: various classifiers that don't fit in any another category
7. **Trees**: tree classifiers, like decision trees
8. **Rules**: rule-based classifiers, e.g. ZeroR

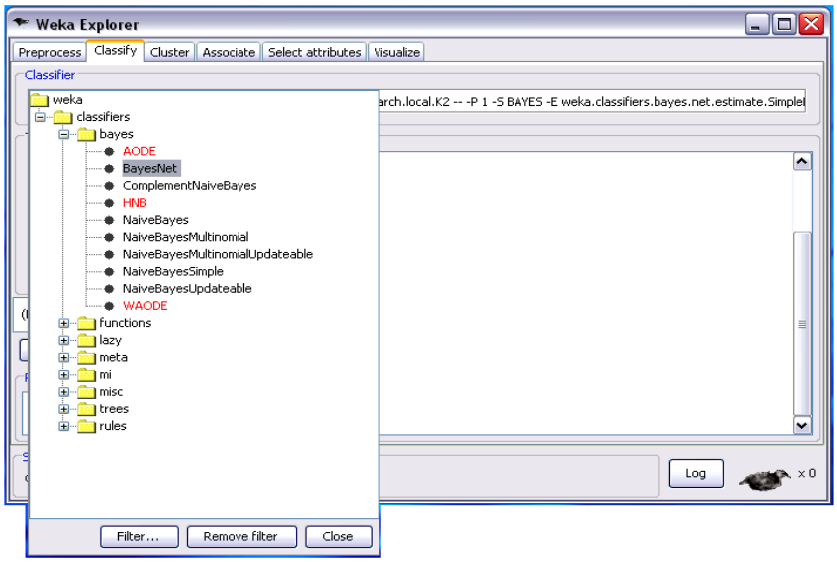


Figure A.2 WEKA Classification Groups

Each group consists of many classification schemes, Figure A.2, and a number of alternatives for test option, Figure A.3.

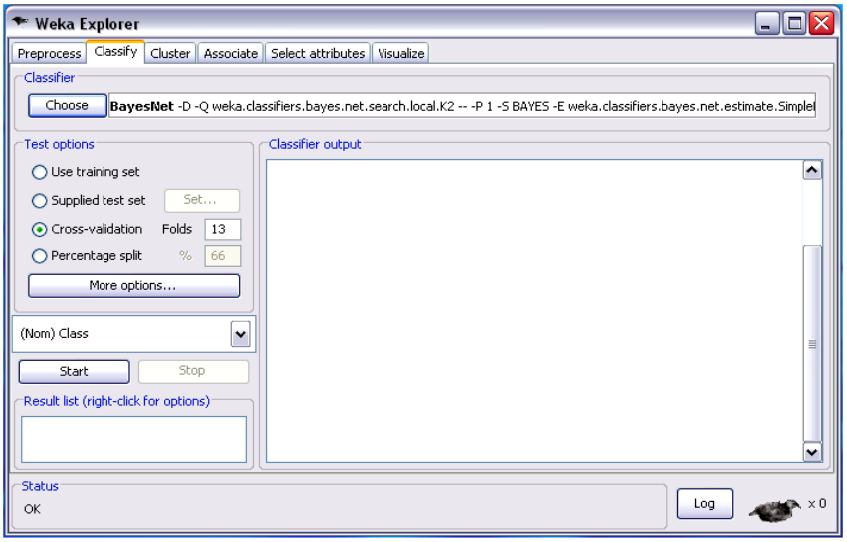


Figure A.3 Setting Test Option

After loading the data, selecting the classification method, and setting test option, WEKA becomes ready for producing classification results of these data. An error message could be issued if the test option accompanied by the classification method is not suitable, as shown in Figure A.4.

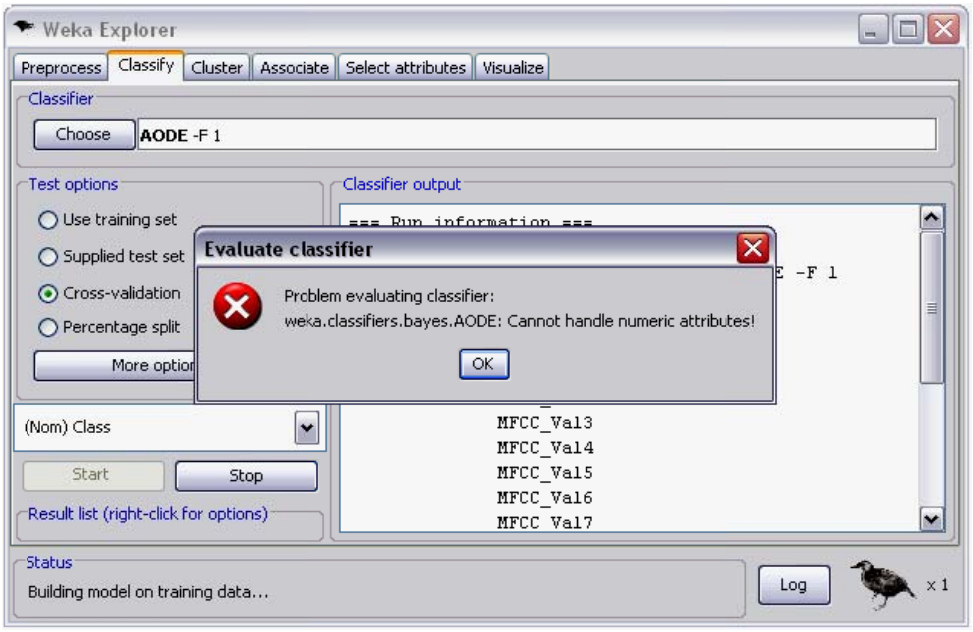


Figure A.4WEKA Error Message

Appendix B

Automatic Speech Recognition

B.1 Speech Recognition by Computer

Speech recognition model is considered part of the ASR, which can be defined as the independent, computer-driven dictation of spoken language into readable text in real time. ASR can also be defined as the computer application software which aims at the processing of extracting meaningful words of speech signal by using a number of different approaches. Speech recognition is one category of speech processing; other categories are [53, 79]:

- Speaker recognition, where the aim is to recognise the identity of the speaker
- Enhancement of speech signals, e.g. noise reduction
- Speech coding for the compression and transmission of speech
- Voice analysis for medical purposes, such as analysis of vocal loading and dysfunction of the vocal cords
- Speech synthesis: the artificial synthesis of speech, which usually means computer generated speech
- Speech compression, which is important in the telecommunications area for increasing the amount of info that can be transferred, stored, or heard, for a given set of time and space constraints

The applications of a tool that helps people with hard hearing, to read in both modes of interaction (interactive and text mode), have also increased in interest long time ago. Historically, In 1883, the editor of the (American Annals of the Deaf) reported that “while the mechanical device used to automatically transcribe human speech was inappropriate, it is not unreasonable to hope that some instrument will yet be contrived that will record ordinary human utterance without annoyance or discomfort to the speaker" [31]. In addition, the overcoming of limitations in using technology imposed

greater focus on human-to-computer interaction to facilitate the use of these machines in various daily applications. The challenging goal was mainly the identification of how to improve speech recognition technology so that it could be employed to support enhancing human interaction with machines. These machines, which need an ASR system as an interface, are [44, 76, 79]:

1. Voice Dialling
2. Voice-Controlled Answering Machine
3. Call Routing
4. Airline Information System
5. Data Entry, Form Filling
6. Network Agent
7. Keyboard Replacement
8. Internet Access

Research on speech recognition spanned in more than 40 years. It has been reported that SR began in 1952 by the achievement of isolated digits recognition for a single speaker at Bell Laboratories. SR research encompasses many other research areas like physics, mathematics, computer science, electrical engineering, linguistics, biology, and psychology. Within these research fields, related work is being carried out in different areas, like information theory, artificial intelligence, computer algorithms, linear algebra, linear system theory, pattern recognition, syntactic theory, phonetics, physiology, probability theory, acoustics, and signal processing [79, 87, 123, 124]

Usually, ASR systems (which compose speech recognition model) are classified, according to the nature of the input data (the speech), as discrete or continuous systems, and as being speaker dependent, independent, or adaptive systems as follows [62, 79, 105, 134]:

- Isolated Speech Recognition (ISR): This class of ASR deals with developing models that handle split acoustic unit of speech. The speech unit here can be at word level, combination of words, or even phrase. This kind of ASR is sometimes known as discrete system

- Continuous Speech Recognition (CSR): deals with speech units as if they were linked together, i.e. deal with speech rather than speech unit.
- A speaker-dependent system: this mode of ASR is able to recognise the speech of a certain person. The SR model is usually trained by recording examples of a certain person's spoken word, sentences, or phrases. So this type of ASR could be used for recognising people via their speech (although this technique for identifying people is not dependent due to the possibility of speech falsification).
- A speaker-independent system: this type of ASR does not require certain person speech as training data; instead, the training data could be the speech of different people. It is obvious that this mode of SR is used for speech (rather than speaker) recognition.
- A speaker adaptive system: in which ASR is developed to respond to the characteristics of new speakers.

CSR provides a more difficult task than that of ISR system for too many technical reasons. Comparing Speaker-independent and Speaker-dependent systems, the latter is considered simpler than the former from development point of view and from its highest recognition accuracy also. This property has resulted in focusing on speaker-dependent isolated word of limited vocabulary by early ASR systems [10, 21, 44, 105, 123, 134].

Towards the end of 1950s and the early 1960s, speech and language processing had been obviously separated into two paradigms namely, symbolic and stochastic. The symbolic concept took off from two types of research work. The first was the formal language theory and generative syntax of Chomsky and others. The second was the work of many linguists and computer scientists on parsing algorithms, initially top-down and bottom-up, and then via dynamic programming. The stochastic paradigm took hold mainly in departments of statistics and electrical engineering. It should be noted here that the term stochastic refers to the process of creation a sequence of non-deterministic selection from among sets of alternatives. They are non-deterministic for that the picking during the recognition process is governed by the characteristics of the input, and are not specified in advance [62, 105, 124].

Currently, probabilistic and data-driven models become quite standard throughout Natural Language Processing (NLP) and its related ASR. Therefore, the algorithms for part-of-speech-tagging, resolution, parsing, reference and discourse processing are all starting to use probabilities and to utilise evaluation methodologies, which are rented from SR, and information retrieval fields. Hence, the current ASR system uses mainly probabilistic models to represent a series of sounds. The current probabilistic model used in ASR is Hidden Markov Models (HMM), especially for words. Aiming at achieving a more accurate ASR system, a new type of models is used, i.e. language model, to get the attribute of combined certain words, and to eventually achieve the improvement of context-based recognition [3, 62, 105, 124].

B.2 Front-End (Feature Extraction Module)

Continuous speech can be defined as a set of complicated audio signals. It is difficult to reproduce them artificially. Not all speech signals are voiced. Actually, they are generally considered as voiced, unvoiced, and some times a combination of the two. Talking about the voiced sounds, they consist of an essential frequency called fundamental frequency (F0), and many harmonic components that are produced by the vocal cords. These voiced sounds are to be modified by a vocal tract which creates a formant (pole), and occasionally antiformant (zero), frequencies. As a property, each formant frequency has its own amplitude and bandwidth. Sometimes it is difficult to define some of F0's parameters correctly. Generally speaking, the fundamental frequency and formant frequencies are probably the most important concepts, which are defined in speech processing [51, 52, 92, 105, 107, 132].

A discrete time model that simulates the process of generating and modifying speech signal is depicted in Figure B.1. The unit of impulse train generator is used to simulate vibrations resulting from the vocal cords. This generator creates pulses p at a pitch period. The airflow of unvoiced speech can be seen as a white noise created by an appropriate random noise generator. The discrete series, produced by impulse train generator is multiplied by an amplitude value of the signal. This amplitude value stands for the loudness of human speech. The human vocal tract is artificially simulated by

using a linear filter. Eventually, this simulation system cannot be considered as being static, though, it is dynamic as it changes with time. Since the vocal system has slow movement of speech signal, the model of speech production can be understood as a static system for only short-time intervals (15ms for example) [32, 92].

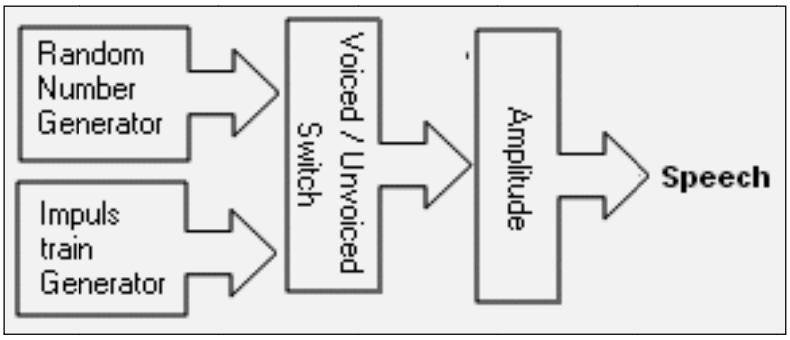


Figure B.1 Model of Speech Production

Researches reported that speech signal is highly redundant signal. It carries information about linguistic messages as well as information about the speakers themselves in certain aspects such as the speaker's physiology and psychology. It is obvious that the latter information can be used to recognise the speaker's identity. Returning to feature extraction, the first stage of ASR, which is called feature measurement, can essentially be viewed as a data reduction procedure. To make a speech signal able to be processed by computer it should be digitised first. The digitised speech signal then is transformed into a smaller set of features, which faithfully describe the relevant properties of the acoustic waveform. It is worth noting that the data reduction rates (or compression ratios) in the range of 10 up to 100 are generally considered practical. Through a long period of research, a number of different feature sets have been proposed, ranging from simple sets, such as energy and zero-crossing rates, to complex representations. These sets are [62, 105]:

- *Short-time spectrum (Discrete Fourier Transformation-DFT or filter bank)*: The output of a filter bank is considered the popular set of features used in many ASRs. The filter bank approach passes the speech signals through a set of band pass filters, which cover the speech bandwidth area (20-20000 MHz). Each channel produces an

energy output concerned to be from each individual filter. The importance of the set of energy values at each time interval (frame) is that it represents an N-dimensional feature vector. The prototype of a speech utterance is to be defined by time variation of these N-dimensional feature vectors. Broadly speaking, there are two approaches to space the band pass filters that are either linearly approached at low frequencies (below 1000 Hz), or logarithmically at high frequencies. Finally, ASR research works reported that 13 filters spaced along a critical-band frequency scale (or bark scale), are enough for high recognition accuracy, and that using 15 filters spaced uniformly in frequency give the same result as critical-band filters in a template matching method of ASR [27, 97, 107, 130].

- *Linear Prediction Coding (LPC):* The coefficients of linear prediction coding (LPC) are capable of modelling the spectral envelope well, and are broadly used. The principle of using LPC for modelling speech wave is that a given speech sample (stationary) can be approximated as a linear combination of past speech samples. For a short gap, (say M samples of speech), the coefficients of LPC are to be computed to yield an N-dimensional feature vector, where N equals p (which is the degree of the model). This vector is usually taken to be (8 to 14). Finally, the time variation of these feature vectors defines a pattern for the speech sound. The frequencies of Formant and their bandwidths can be extracted from the transfer function of the vocal tract by using peak picking procedure. Computing the Fast Fourier Transformation (FFT) over the set of LPC parameters and taking the inverse of the result yield the transfer function of the vocal tract [2, 77, 78].

- *Cepstral Parameters (homomorphic model):* There are three defined types of cepstral parameters, which are used in the homomorphic model of speech recognition systems. These types are Linear Frequency Cepstral Coefficient (LFCC), Mel-Frequency Cepstral Coefficients (MFCC), and LPC-derived Cepstral Coefficients (LPCC). Following is an abbreviation of each [2, 77]:

    - The LFCCs are computed from the log-magnitude Discrete Fourier Transform (DFT) directly.

- In mel-frequency scale, the DFT magnitude spectrum is frequency-warped to follow a critical band scale (mel-scale) and amplitude-warped (logarithmic scale) before computing the inverse DFT parameters. Therefore, Q band pass filters are used to cover the required frequency range. For more, see [24, 131].
- The LPCCs are obtained from the LPC parameters directly

The set of N parameters (LFCCs, MFCCs, or LPCCs) constitute an N-dimensional feature vector. The time variation of these feature vectors defines a pattern of a speech utterance.

- *Articulatory Parameters*: The position of the tongue, lips, jaws, and the velum can be used to define another set of features for describing speech sounds. These features come in terms of parameters giving the position of each of the above organs as functions of time. These parameters can be estimated from the speech signal. The theory of speech production, based on distinctive areas along the vocal tract, has been advanced to introduce a new concept in relation to acoustic-articulatory-phonetics. By getting the inversion of acoustic-articulatory, the function of area can be used as an articulatory parameter for speech recognition [17, 51, 116] .

- *Auditory Model*: One more approach for measuring a feature of speech waveform is the model of auditory utilisation. The auditory model concerns those psychophysical aspects of critical bandwidth, loudness, timbre, and subjective duration. Another design, which tries to detain the time-varying nature of the auditory model by combining the psychophysical critical-band and loudness estimation with a ring-rate model, has enhanced the accuracy of the speech recognition model compared to earlier method of filter-bank of calculating the speech feature measures [10, 21, 130].

- *Fractal Dimension of Speech Signal*: In this approach of speech wave representation, a spectral representation of speech unit based on fractals is to be defined. The fractal principle is used to measure the degree of irregularity or roughness of complicated-structure objects, which is called fractal dimension D. This dimension is used to

characterise of one object from another. The fact that single numbers can characterise such complicated-structure objects has led to its application in the areas of acoustic and speech science. As it is known, speech waveforms are highly irregular patterns that can be quantified by using fractal mathematics. The fractal dimension can be defined as a real number, which generally falls in the range 0 up to 5, and can be calculated in a number of ways [32]:

- Box Counting Method (BCM)
- Continuous Box Counting Method (CBCM)
- Walking- Divider Method (WDM)
- Power Spectrum Method (PSM)

B.3 Back-End Processing

Broadly speaking, there are three approaches to speech recognition, namely the acoustic-phonetic, the pattern recognition, and the artificial intelligence [12, 13, 62, 105, 120], as illustrated in Figure B.2.

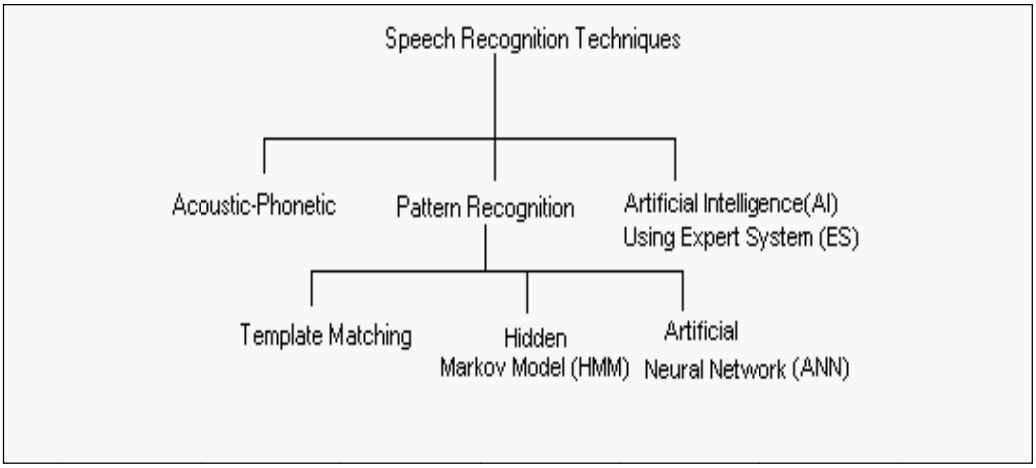


Figure B.2 Speech Recognition Approaches

- *Acoustic-Phonetic Recognition*: The approach of acoustic-phonetic recognition is usually applied at the phoneme level. Theoretically, this approach is considered a smart approach to speech recognition since it confines the number of representations that have to be stored relative to the total number of phonemes required for a certain language. Acoustic-phonetic recognition normally engages

three steps. They are feature-extraction, segmentation & labelling, and word-level recognition. The Acoustic-phonetic recognition system scans the input of spectral patterns (such as the formant frequencies) required to distinguish phonemes from each other. There are acoustic-phonetic interpretation rules that are used for interpreting those groups of extracted features. These rules categorise label phonemes and establish positions of phoneme end and the start of the next phoneme (segmentation). An uncertainty regarding to potential phoneme labels results from high level of acoustic similarity among phonemes collective with phoneme variability resultant from coarticulation property and other sources. Thus, the segmentation and labelling phase gives a set of hypotheses. These hypotheses can be organised into a type of structure like phoneme lattice, decision tree, or any other structure. As soon as the process of segmentation and labelling is completed, the system looking through the vocabulary of application for words similar to the phoneme hypotheses. The term best matching a sequence of hypotheses is identified as the input item [105, 120].

- *Template Matching*: This form of pattern recognition is based on a processing unit called template. The template is actually a set of feature/parameter vectors of the speech data. Each template contains speech waveform of a word or phrase, thus it stores these speech units separately. These spoken inputs of end users are organised and structured into templates prior to performing the recognition process. Comparison of unrecognised input with the stored template is to be applied, and the most closely stored template matches to the incoming speech pattern is identified as the input word or phrase. The recognition task, using template matching, is performed at word level, and it includes no reference to the phonemes of the word at all. Template matching process requires a frame-by-frame comparison of spectra patterns to generate an overall similarity judgment (usually called distance metric) of each template. Note that due to the nature of the speech signal itself (individual utterances of the same word, even by the same person, often differs in length), comparison between templates of speech units is expected to produce an approximate (not identical) match. Regardless of the

cause, it should be a way to decrease temporal differences between templates so that both fast and slow state of utterances of the same word will not be recognised as different words. This process of decreasing the length of temporal /word is called temporal alignment. Dynamic Time Warping (DTW) is pattern-matching technique, which is considered the most commonly used approach for performing temporal alignment in template matching. Most ASR systems, which use template-matching approach, should have a predetermined threshold of acceptability. The function of this threshold is to avoid including noise and words in vocabulary application, thus making sure of their correct identification as acceptable speech inputs. Template matching approach of speech recognition has been reported as a very good approach with small vocabularies of phonetically distinct items. Unfortunately, this is not true with applications of large vocabulary. This approach has difficulty in making fine distinctions necessary for recognition in such larger vocabulary, as well as recognition in vocabularies containing similar-sounding words called confusable words. Of course, it is not expected to have small vocabulary since one stored template, at least, is required for each word in the application vocabulary. One more problem is that the linear expansion of storage requirements increases as the size of vocabulary grows. This, in turn, results in a comparable increase in computational complexity [13, 52, 105].

- *The Stochastic Processing (HMMl):* Stochastic processing demands the creation and storage of models of each of the items to be recognised, as much as in the template matching technique. This is the only similarity that exists between these two approaches. Beyond this point, the two approaches diverge. Unlike template matching approach, stochastic processing has no direct matching between stored models and input models. Alternatively, it uses complex statistical and probabilistic analysis. The statistical and probabilistic analysis is best understood by investigating the network-like structure in which those statistics are stored, namely the HMM, which is a Probabilistic Finite State Automata (PFSA) [120, 127].

- *ANN:* ANN used for ASR applications can be classified in terms of ANN's ability to handle data of time-dimension and their inclusion of non-network structures. The ANN of static classification focuses on the frequency domain of speech waveform, but rarely attempts to handle temporal information. ANNs usually use stationary unit of speech (pre-segmented). The segmentation of speech is often performed manually, and the speech input would be in terms of speech items (units) rather than as time sequences. Researchers reported that the biggest problem of using ANN in ASR system is time alignment, due to the static nature of the patterns (input data structure of ANN) [105]. They also showed that there is an ability to use ANN with dynamic pattern of speech via certain type of ANN called current networks. These current networks analyse slices or streams of data of a real speech and are usually referred to as dynamic classification networks. Current networks got such name because they deal with processing components of time dimension speech, and are concerned with creating systems related to operational speech understanding. Time alignment unit is attached to some dynamic classification networks as part of the network architecture. An example of these networks is the Time Delay Neural Network (TDNN) developed at Carnegie Mellon University (CMU), which is one of the most popular of these networks. The feed forward network structure, with a sigmoid transfer function, was used to build TDNN, while backpropagation approach is used to train it. The TDNN was trained and tested to recognise phonemes b, d, and g. The TDNN was constructed by using module methodology to enhance extensibility, where a full TDNN is to be constructed from many modules. A few researchers to capture the overlapping temporal information in speech have used Recurrent Neural Networks. Recurrent networks add information to the analysis of the current slice about prior time slices. Recurrent Neural Networks are able to process shifts of time without requiring the creation of multiple copies of speech slices, like time delay networks. This ability makes recurrent networks natural candidates for real-time analysis of speech. Recurrent networks have shown reasonably good performance on small, well-defined tasks [12, 13, 33, 105].

Reference to Section 2.2.1 and Section 4.2 of this work for more information about ANN itself might be useful.

- *Artificial Intelligence (AI) using Expert System (ES):* Many researchers have investigated and used ES approach of AI for developing ASR systems. As a principle, ES approach requires knowledge gathering (in ASR application, knowledge is related to speech, speaker, and other topics). Knowledge could be complex and have multi-resources, which should be abided by the problem at hand. As an example, the AI approach to segmentation and labelling would be to augment these different knowledge sources [105]:
    - Acoustic knowledge – facts concerning the identity of sounds (predefined phonetic units) on basis of spectral measurements and presence or absence of sound features
    - Lexical knowledge – that is combining acoustic proof to suggest words, as specified by a dictionary, which maps sounds into words, or equivalently decomposes words into sounds
    - Syntactic knowledge – the grouping of words with a view to forming strings that are grammatically correct in accordance to a language model, such as phrases or sentences
    - Semantic knowledge – is the job of understanding the task area to be able to validate sentences or phrases, which are consistent with the task being performed or consistent with previously decoded sentences.
    - Pragmatic knowledge – inference ability necessary in resolving ambiguity of meaning based on ways where words are generally used.

- *Hybrid Systems*: Several authors have shown that the output of ANNs used in a classification mode can be interpreted as estimates of posterior probabilities of output classes dependent on the input to HMM. A combination of ANNs and HMMs into what is now referred to as hybrid HMM/ANN speech recognition systems has thus been proposed. To derive probability, ANN has several potential advantages over conventional techniques [13, 37, 110]:

- Enhanced model accuracy: The estimation of probabilities as calculated by ANN does not involve elaborated assumptions related to the structure of the statistical distribution that is going to be modelled. Yet, ANN's acoustic models are more accurate.
- Availability of contextual information: For an ANN estimator, multiple inputs can be used from a range of speech frames in which the ANN will learn something about the correlation linking the acoustic input data. This is not similar to ordinary approaches, which suppose that consecutive acoustic vectors are not correlated. No doubt, that such assumption is wrong.
- Increased Discrimination: ANNs can easily hold discriminated training. In the training phase, the speech frames that characterise a given acoustic item are going to be used for training the related HMM to recognise these speech frames, and for training the other HMMs to reject them.

Appendix C

The Matlab Code of RFL4ASR

C.1 Introduction

RFL4ASR consists of functions and data files. In order to work with it, its folder and files should be created first according to the structure illustrated by Figure C.1. Then from Matlab interactive screen, simply by using matlab's command cd, change the active directory to RFL4ASR and then issue the command project1.

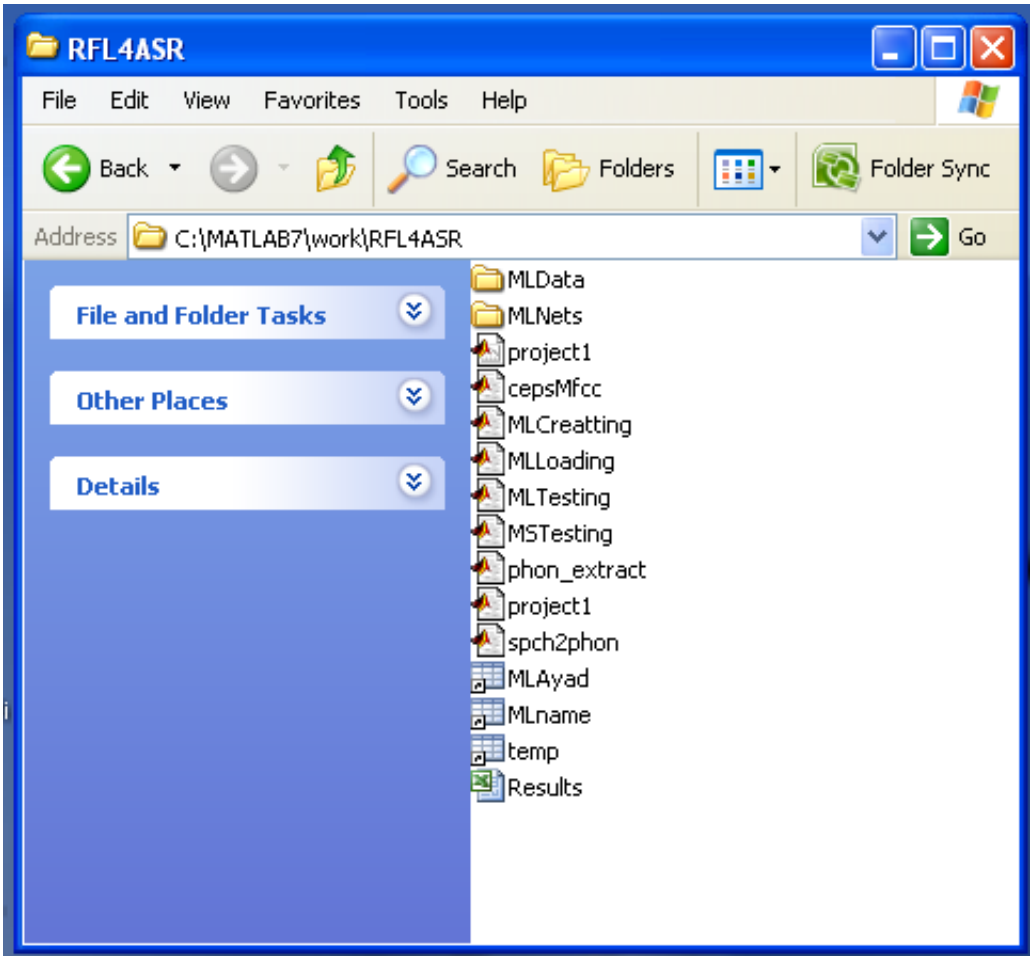


Figure C.1 RFL4ASR Files and Folders

C.2 The GUI

```
function varargout = project1(varargin)

% PROJECT1 M-file for project1.fig
% PROJECT1, by itself, creates a new PROJECT1 or raises the existing
% singleton*.
%
% H = PROJECT1 returns the handle to a new PROJECT1 or the handle to
%     the existing singleton*.
%
% PROJECT1('CALLBACK',hObject,eventData,handles,...) calls the local
%           function named CALLBACK in PROJECT1.M with the given input
%           arguments.
%
% PROJECT1('Property','Value',...) creates a new PROJECT1 or raises the
%           existing singleton*.  Starting from the left, property
%           value pairs are applied to the GUI before
%           project1_OpeningFunction gets called.  An
%           unrecognized property name or invalid value makes
%           property application stop.  All inputs are passed to
%           project1_OpeningFcn via varargin.
%
%  *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%   instance to run (singleton)".
%

% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help project1

% Last Modified by GUIDE v2.5 02-Oct-2006 14:08:09

% Begin initialization code - DO NOT EDIT
%------------------------------------------------------------------------
cd C:\MATLAB7\work1\RFL4ASR;
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @project1_OpeningFcn, ...
                   'gui_OutputFcn',  @project1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);

if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

% End initialization code - DO NOT EDIT

% ------------------  Executes just before project1 is made visible.
```

```
function project1_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to project1 (see VARARGIN)

% Choose default command line output for project1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  U S E R   D E F I N E D   D A T A
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SETUP THE RLN

SetRlnRef(handles);

% SETUP THE CONTENTS OF THE PULLDOWN MENUE (TEST FILES)....
SetPullDnMenu(handles);

% UIWAIT makes project1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%--------------------------------------------------------------------
% --- Outputs from this function are returned to the command line.
function varargout = project1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------
% -------------------- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              T E S T I N G   A   P H O N E M E
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load('MLAyad');
load('C:\MATLAB7\work1\RFL4ASR\MLname');
set(handles.text4,'String',filename);
```

```
load('temp');

set(handles.text16,'String',correct);
set(handles.text17,'String',uncorrect);
% default='C:\MATLAB7\work1\MSData\Testing';
default='C:\MATLAB7\work1\RFL4ASR\MLData\TestingData';

files = dir(default);
index=3; % skip the first two files of the folder(. and ..)

element=0;

fone{1,1}='ix' ; fone{1,2}='ixl';
fone{2,1}='tcl'; fone{2,2}='tcl';
fone{3,1}='dcl'; fone{3,2}='ixl';

x=get(handles.figure1,'UserData');

popup_sel_index = get(handles.popupmenu1, 'Value');
files=get(handles.popupmenu1, 'UserData');
TV=files(popup_sel_index+2).name;
fn=strtok(TV,'.');
fl=length(fn);

[vector,phnm1]=cepsMfcc(TV,default);
 x='';
 for tt=1:(fl-3)
    x=strcat(x,phnm1(tt));
 end;
an = sim(Mnet,vector);
[val1,grp]=min(abs([1]-an));
a= sim(net{grp},vector);
[val2,ele]=min(abs([1]-a));

x1=fone{grp,ele};

m=strcmp(x,x1);
if (m)correct=correct+1;
else
   uncorrect=uncorrect+1;
end

Err=num2str((uncorrect)/(correct+uncorrect)*100);
Err=strcat(Err,' %');

set(handles.text18,'String',Err);
set(handles.text16,'String',correct);
set(handles.text17,'String',uncorrect);

save('temp','correct','uncorrect');

%--------------------------------------------------------------------
% -------------------- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Creating Refrence Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes1);
cla;
spch2phon('Ref');
SetRlnRef(handles);

%--------------------------------------------------------------------
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Creating Test Data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes1);
cla;
spch2phon('Test');
SetPullDnMenu(handles);

% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------

function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end

% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Loading ML/Ayad Net                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
MLLoading(handles);

%-------------------------------------------------------------------
----
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                          Selecting Test Phoneme                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject  handle to popupmenu2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

% --- Executes on selection change in popupmenu2.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
% as cell array
%  contents{get(hObject,'Value')} returns selected item from popupmenu2

% -------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                            Creating ML/Ayad Net                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

MLCreatting();

% -------------------------------------------------------------------
function Untitled_3_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       Creating Data for ML/Ayad Net                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

axes(handles.axes1);
```

```
cla;
spch2phon('Ref');
SetRlnRef(handles);

% --------------------------------------------------------------------
function Untitled_4_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_7_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_8_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_9_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_10_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_11_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_12_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_13_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_13 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_14_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_15_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_16_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_17_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_18_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_19_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_21_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_22_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_22 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_23_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function Untitled_24_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%--------------------------------------------------------------------
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

%--------------------------------------------------------------------
% --- Executes on selection change in listbox1.

function listbox1_Callback(hObject, eventdata, handles)

% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: contents = get(hObject,'String') returns listbox1 contents as
% cell array
% contents{get(hObject,'Value')} returns selected item from listbox1

%--------------------------------------------------------------------
% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

%--------------------------------------------------------------------
% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)
% hObject    handle to listbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox2 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox2

%--------------------------------------------------------------------
% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

%--------------------------------------------------------------------
% --- Executes on selection change in popupmenu1.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents
% as cell array
%  contents{get(hObject,'Value')} returns selected item from popupmenu1

%--------------------------------------------------------------------
% --- Executes on mouse press over figure background.
function figure1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over
pushbutton1.

% --------------------------------------------------------------------
function pushbutton1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------- Executes during object creation, after setting
all properties.
function figure1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% --------------------------------------------------------------------
% --- Executes during object creation, after setting all properties.
function listbox3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.

if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

%--------------------------------------------------------------------
% --- Executes on selection change in listbox3.
function listbox3_Callback(hObject, eventdata, handles)
% hObject    handle to listbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox3 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from
listbox3


%--------------------------------------------------------------------
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
% called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'))
;
end

%--------------------------------------------------------------------
% --- Executes on selection change in popupmenu2.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
% as cell array
% contents{get(hObject,'Value')} returns selected item from popupmenu2

% --------------------------------------------------------------------
function Untitled_25_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Untitled_26_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --------------------------------------------------------------------

function Untitled_27_Callback(hObject, eventdata, handles)

% hObject    handle to Untitled_27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Untitled_28_Callback(hObject, eventdata, handles)

% hObject    handle to Untitled_28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%          Exiting the simulator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end
correct=0;
uncorrect=0;
save('temp','correct','uncorrect');
delete(handles.figure1);

%--------------------------------------------------------------------
% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%--------------------------------------------------------------------
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
MLCreatting();
%--------------------------------------------------------------------
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
phon_extract();

%--------------------------------------------------------------------
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                           MLLoading();
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

MLLoading(handles);

function SetRlnRef(handles)
```

```
load('MLAyad');
load('C:\MATLAB7\work1\RFL4ASR\MLname');
set(handles.text4,'String',filename);
x=0;
set(handles.text16,'UserData',x) ; % Correct counter
set(handles.text17,'UserData',x);  % uncorrect counter


% --------------------------------------------------------------------

function SetPullDnMenu(handles)
% hObject    handle to NewTestData (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

default='C:\MATLAB7\work1\RFL4ASR\MLData\TestingData';
files = dir(default);
index=3;
if length(files)>2
    more=files(index).name;
    while (index<length(files))
      index=index+1;
      more=strvcat(more,files(index).name);
    end
    set(handles.popupmenu1,'UserData',files);
    set(handles.popupmenu1,'String', more);
end


% --------------------------------------------------------------------
function NewTestData_Callback(hObject, eventdata, handles)
% hObject    handle to NewTestData (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

MLCreatting();

% --------------------------------------------------------------------
function Untitled_29_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

phon_extract();
```

C.3 The Function of Calculating the MFCC

```
function [cepstral,phonm]= cepsMfcc(FileName,PathName)

global CCArray FilterParameters

FileID = fopen([PathName '\' FileName],'rt');
if FileID==-1
    errordlg('OOPS! Cannot open the specified file name...');
end

InputPhoneme = fscanf(FileID,'%5d');
Index=1;
while (~ feof(FileID))
   InputPhoneme (Index,1) = str2num(fscanf(FileID,'%s',1));
   Index = Index +1;
end
fclose(FileID);

[Row Column] = size(InputPhoneme);
if (Row > Column)
      InputPhoneme=InputPhoneme';
end

% Recommonded initialization of the filter bank
FrameRate=31.25;
SampleRate=16000;
SizeOfWindow = 400;          % Function of the SampleRate and FrameRate
CCSize = 13;                 % No. of Cepestral Coefficients
NoFilters = 13;              % Linear Filters No. = No. of CC
FilterLog = 27;              % Logarithmic Filters No.
AllFilters = 40;             % that is the sum of NoFilters + FilterLog
SpacingLog = 1.0711703;
Spacing = 66.66666666;
LowerFreq =133.3333;
FFTvolume = 512;

Frequency =(0:NoFilters-1)*Spacing+LowerFreq;
Frequency(NoFilters+1:AllFilters+2)= Frequency(NoFilters) *
SpacingLog.^(1:FilterLog+2);

FilterParameters = zeros(AllFilters,FFTvolume);
FrequicesOfFFT = (0:FFTvolume-1)/FFTvolume*SampleRate;
LowerFrequencies = Frequency(1:AllFilters);
CenterFrequencies= Frequency(2:AllFilters+1);
UpperFrequencies = Frequency(3:AllFilters+2);
TrngleHight = 2./(UpperFrequencies-LowerFrequencies);

for Channel=1:AllFilters          % Find Filter parameter of all
channels in the all filters
   FilterParameters(Channel,:) = (FrequicesOfFFT >
LowerFrequencies(Channel) & FrequicesOfFFT <=
CenterFrequencies(Channel)).* ...
TrngleHight(Channel).*(FrequicesOfFFT-LowerFrequencies(Channel))/...
(CenterFrequencies(Channel)-LowerFrequencies(Channel)) + ...
```

```
(FrequicesOfFFT > CenterFrequencies(Channel) & FrequicesOfFFT <
UpperFrequencies(Channel)).* TrngleHight(Channel).* ...
(UpperFrequencies(Channel)-FrequicesOfFFT)/(UpperFrequencies(Channel)-
CenterFrequencies(Channel));
end

HammingWindow = 0.54 - cos(2*pi*(0:SizeOfWindow-1)/SizeOfWindow)* 0.46;

if 0
   x = .54;
   y = -.46;
   z = pi/SizeOfWindow;
   windowStep = SampleRate/FrameRate;
   HammingWindow = 2*(sqrt(windowStep/SizeOfWindow))/sqrt(4*x*x+2*y*y)*
   (x + y*cos(2*pi*(0:SizeOfWindow-1)/SizeOfWindow + z));
end

CCArray = 1/sqrt(AllFilters/2)*cos((0:(CCSize-1))' * (2*(0:(AllFilters-
1))+1) * pi/2/AllFilters);
CCArray(1,:) = CCArray(1,:) * sqrt(2)/2;

if 1
  % The filter is a "Direct Form II Transposed"
  % implementation of the standard difference equation:
  % a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
  % - a(2)*y(n-1) - ... - a(na+1)*y(n-na)

  BeforeCalculation = filter([1 -.97], 1, InputPhoneme);
else
      BeforeCalculation = InputPhoneme;
end
SteppingWindow = SampleRate/FrameRate;
Column = 1;
CepsCofficients = zeros(CCSize, Column);

freqresp = zeros(FFTvolume/2, Column);

% Start Caculating:

for i=0:Column-1
% 1. Windowing the data using hamming window technique
    left = 1+i*SteppingWindow ;
    right = SizeOfWindow-1 + left;

% 2. Move the windowing data into FFT order
   FFTParameter = zeros(1,FFTvolume);

% 3. Calculate the filter bank outputs from FFT parametrs
  FFTParameter(1:SizeOfWindow)=
            BeforeCalculation(left:right).*HammingWindow;

% NOTE: Window size is 256 which may bigger than InputPhoneme data!!!

% 4. Calculate the magnitude of the fft,
    MagOfFFT = abs(fft(FFTParameter));
```

```
% 5. Calculate base_10 logarithm of the product of filter parameters
%    by FFT magnitude   ,

    Reduced = log10(FilterParameters * MagOfFFT');

% 6. For reducing dimensionality, calculate the cosine transform
    CepsCofficients(:,i+1) = CCArray * Reduced;
   end

cepstral= CepsCofficients;
phonm1 = strtok(FileName,'.');
phonm = strtok(phonm1,'_');
```

C.4 The Function of Creating ML/RFL-Based Net

```
% This function creats ML/RFL-Based Net of one supernet and 3 subnets

function [] = MLCreatting()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                 Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
echo off

InputSize=13;
OutputSize=2;
MOutputSize=0;

Mdata=[];


for k=1:12  % 12 elements(phoneme)/group
    MTarget(1,k)=1;
    MTarget(1,k+12)=0;
    MTarget(1,k+24)=0;
end;

for k=1:12  % 12 elements(phoneme)/group
        MTarget(2,k)=0;
        MTarget(2,k+12)=1;
        MTarget(2,k+24)=0;
end;
for k=1:12  % 12 elements(phoneme)/group
     MTarget(3,k)=0;
     MTarget(3,k+12)=0;
     MTarget(3,k+24)=1;
end;
T1=[];
for i=1:6
    T1=[T1;1 0];
end;
for i=1:6
    T1=[T1;0 1];
end;

  T1=[1 1 1 1 1 1 0 0 0 0 0 0;...
      0 0 0 0 0 0 1 1 1 1 1 1];

DR2={'ixM12.txt','ixM13.txt','ixM14.txt','ixM15.txt','ixM16.txt','ixM17
.txt',...

'ixlF22.txt','ixlF23.txt','ixlF24.txt','ixlF25.txt','ixlF26.txt','ixlF2
7.txt'};
DR5={'tclM32.txt','tclM33.txt','tclM34.txt','tclM35.txt','tclM36.txt','
tclM37.txt',...
```

```
'tclF42.txt','tclF43.txt','tclF44.txt','tclF45.txt','tclF46.txt','tclF4
7.txt',};
DR7={'dclM52.txt','dclM53.txt','dclM54.txt','dclM55.txt','dclM56.txt','
dclM57.txt',...

'ixlF62.txt','ixlF63.txt','ixlF64.txt','ixlF65.txt','ixlF66.txt','ixlF6
7.txt',};

button='Yes';
while (strcmp(button,'Yes'))
button = questdlg('New Sub Group?');
if (strcmp(button,'Yes'))
    MOutputSize=MOutputSize+1;
    DataFolder=uigetdir('C:\MATLAB7\work\RFL4ASR\MLData\TrainingData');
    files = dir(DataFolder);

    % Create and Train Neural Net for a group...
    Cindex=0;
    Indata = [];
    index=3; % skip the first two files of the folder(. and ..)

    while (index<=length(files))
        more=files(index).name;
        fn=strtok(more,'.');
        fl=length(fn);
        d=fn(fl%1);
        [vector,phnm]=cepsMfcc(more,DataFolder);
        Indata=[Indata,vector];
        index=index+1;
    end;

    Mdata=[Mdata,Indata];


    net{MOutputSize} = newff(minmax(Indata),[InputSize 10
OutputSize],{'tansig','tansig','purelin'},'traingd');

    net{MOutputSize}.trainParam.show   = 50;
    net{MOutputSize}.trainParam.lr     = 0.0455;
    net{MOutputSize}.trainParam.epochs = 150000; %times
    net{MOutputSize}.trainParam.goal   = 1e%6;
    [net{MOutputSize},tr]=train(net{1},Indata, T1);

    end ; % of Big If
end; % of while


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                        Mnet
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

OutputSize=3;
```

```
Mnet= newff(minmax(Mdata),[InputSize 9 5
MOutputSize],{'tansig','tansig','tansig','tansig','tansig','purelin'},'
traingd');

  Mnet.trainParam.show   = 50;
  Mnet.trainParam.lr     = 0.0455;
  Mnet.trainParam.epochs = 400000; %times
  Mnet.trainParam.goal   = 1e%6;

  [Mnet,tr]=train(Mnet,Mdata, MTarget);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                Saving MS Network
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  NetName=inputdlg('Name of the Net:');
  save(NetName,'Mnet','net','default');
  msgbox('End of Training...');
```

C.5 The Function of Loading a ML/RFL-Based Net

```
function MLLoading(handles)
cd MLNets;
[filename,pathname,filterindex]=uigetfile({'*.mat','MAT-files'},'Pick
an ML/Ayad Net','MultiSelect', 'off');
loadednet=strcat(pathname,filename);
load(loadednet);
save('C:\MATLAB7\work1\RFL4ASR\MLAyad','Mnet','net','default');
save('C:\MATLAB7\work1\RFL4ASR\MLname','filename');
cd '..';
set(handles.text4,'String',filename);
correct=0;
uncorrect=0;
set(handles.text16,'String',correct);
set(handles.text17,'String',uncorrect);
save('temp','correct','uncorrect');
```

C.6 The Function of Extracting Phonemes from Spoken Statement

```
function [] = phon_extract()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Select, and open (for read) a spoken statement File %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[fname pname] = uigetfile('C:\MATLAB7\work\TIMIT Data\*.phn','Choose
file');
fid_phn = fopen([pname fname],'rt');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initilize number of phonems in the selected file & eof marker %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Phonem_Count = 1;
EOF = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read the start sampling integer & finish sampling integer for each
% phoneme & phoneme %
% name up to last phoneme in the file (marked with h#)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

while ~ EOF
   start(Phonem_Count,:) = str2num(fscanf(fid_phn,'%s',1));
   finish(Phonem_Count,:) = str2num(fscanf(fid_phn,'%s',1));
   pho = fscanf(fid_phn,'%s',1);
   phoneme(Phonem_Count,1:length(pho)) = pho;
   if strcmpi(pho,'h#') & Phonem_Count > 1; EOF = 1; end;
   Phonem_Count = Phonem_Count + 1;
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read COMPLETELY the wave file fromat opposit the phoname file format
for playing purposes %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid_wav = fopen([pname fname(1:length(fname)-3) 'wav'],'r');
wav = fread(fid_wav,inf,'int16');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read the text file fromat opposit the phoname file format to get the
% actual ength of the %%
% wave file format (number of wave values) which is the second string
% in the file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fid_txt = fopen([pname fname(1:length(fname)-3) 'txt'],'rt');
fscanf(fid_txt,'%s',1);
len_wav = str2num(fscanf(fid_txt,'%s',1));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% capture the sampling integers for each phoneme in the sentence  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

start = start + length(wav) + 1 - len_wav;
finish = finish + length(wav) + 1 - len_wav;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot each phoneme in the sentence  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=2:length(start)-1
   plot(wav(start(i):finish(i)));
   title(phoneme(i,:))
   pause(.1)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Play the sentence %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

soundsc(wav(start(2):finish(length(finish)-1)),16000,16);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Phoneme Features Saving ....
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ph_name1 = ['C:\MATLAB7\work\RFL4ASR\MLData\'];
delete(strcat(ph_name1,'*.*'));
for i=2:length(start)-1
   clear pho;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% get the current phoneme name to creat a file holding phoneme name
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
      for n=1:size(phoneme,2)
        pho(n)=phoneme(i,n);
      end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % open an output file or creat it for each phoneme  %%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   ph_name = [ph_name1 pho];
   ph_name = strcat(ph_name,'.txt');
   fid = fopen(ph_name,'wt');

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Save the wave sample for each phoneme %%
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   for n = start(i):finish(i)
      fprintf(fid,'%g\n',wav(n));
   end
   fclose(fid);
   clear ph_name;
end
```

C.7 The Function of Extracting Features of Phonemes

```
function [] = spch2phon(where)

% Select, and open (for read) a voiced statement File
[fname pname] = uigetfile('C:\MATLAB7\work\TIMIT
Data\TRAIN\DR1\FCJF0\*.phn','Choose file');
fid_phn = fopen([pname fname],'rt');

% Initilize number of phonems in the selected file & eof marker
count = 1;
eof = 0;

% Read the start sampling integer & finish sampling integer for each
phoneme & phoneme
% name up to last phoneme in the file (marked with h#)
while ~ eof
   start(count,:) = str2num(fscanf(fid_phn,'%s',1));
   finish(count,:) = str2num(fscanf(fid_phn,'%s',1));
   pho = fscanf(fid_phn,'%s',1);
   phoneme(count,1:length(pho)) = pho;
   if strcmpi(pho,'h#') & count > 1; eof = 1; end;
%   length(pho) == 2 &
   count = count + 1;
end


% Read COMPLETELY the wave file fromat opposit the phoname file format
for playing
% purposes
fid_wav = fopen([pname fname(1:length(fname)-3) 'wav'],'r');
```

```
wav = fread(fid_wav,inf,'int16');

% Read the text file fromat opposit the phoname file format to get the
actual
% length of the wave file format (number of wave values) which is the
second string
% in the file
fid_txt = fopen([pname fname(1:length(fname)-3) 'txt'],'rt');
fscanf(fid_txt,'%s',1);
len_wav = str2num(fscanf(fid_txt,'%s',1));

% capture the sampling integers for each phoneme in the sentence
start = start + length(wav) + 1 - len_wav;
finish = finish + length(wav) + 1 - len_wav;

% Plot each phoneme in the sentence
for i=2:length(start)-1
   plot(wav(start(i):finish(i)));
   title(phoneme(i,:))
   pause(.1)
end

% Play the sentence
soundsc(wav(start(2):finish(length(finish)-1)),16000,16);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                              Phoneme Features Saving ....
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ph_name1 = ['C:\MATLAB7\work\' where '\'];
delete(strcat(ph_name1,'*.*'));
for i=2:length(start)-1
   clear pho;
   % get the current phoneme name to creat a file holding phoneme name
     for n=1:size(phoneme,2)
       pho(n)=phoneme(i,n);
     end

   % open an output file or creat it for each phoneme
   ph_name = [ph_name1 pho];
   ph_name = strcat(ph_name,'.txt');
   fid = fopen(ph_name,'wt');

   % Save the wave sample for each phoneme
   for n = start(i):finish(i)
      fprintf(fid,'%g\n',wav(n));
   end
   fclose(fid);
   clear ph_name;
end
```

References

1. Ackoff R. L., From Data To Wisdom. *Journal of Applied Systems Analysis,* Vol. 16,1989, pp.3-9.

2. Alotabi Yousef and Shahsavari M. M., Speech Recognition: What It Takes for the Computer to Understand you. *IEEE Potentials*, Vol. 17, (Feb/Mar), 1998, pp.23-28.

3. Al-Zobaydi Ayad T. and Al-Akaidi Marwan M., Speech Recognition by Computer - a survey article. *International Middle Eastern Multiconference on Simulation and Modelling (MESM'04) September 2004*, IEEE UKRI SPC, publisher EUROSIS, Amman, Jordan, 2004, pp. 102-107.

4. Al-Zobaydi, Ayad T., Al-Akaidi Marwan M. and John R. I, Data Mining for Generating Predictive Models of Automatic Speech Recognition. *International Middle Eastern Multiconference on Simulation and Modelling (MESM'05) October 2005*, Porto University-Portugal, 2005, pp. 147-150.

5. Amanatidis Christos, Halkidi Maria, and Vazirgiannis Michalis, UMiner: A Data Mining System Handling Uncertainty and Quality. Advances in Database Technology. *8th International Conference on Extending Database Technology (EDBT) March 2002*, Prague, Czech Republic, 2002, pp 762-765.

6. Berenji H. R., Treatment of Uncertainty in Artificial Intelligence. *Machine Intelligence and Autonomy Aerospace Systems, American Institute of Aeronautics and Astronautic (AIAA)*, Vol. 115, 1988, pp. 233-247.

7. Beynon Davies P., *Information Systems: An Introduction To Informatics In Organizations*. Palgrave Macmillan, Basingstoke, ISBN-10:0333963903, 2002, pp31-43.

8. Bezdek James C., Ehrlich Robert, and Full William, FCM: Fuzzy C-Means Algorithm. *Computers and Geosciences*, Vol. 10, Issues 2-3, 1984, pp. 191-203.

9. Blackburn, P., Van Benthem, J., and Wolter F., *Handbook of Modal Logic*. Elsevier. ISBN-13: 978-0-444-51690-9, 2007, pp 1-84.

10. Blomberg M., Carlson R., Elenius K., and Grantsrom B., Auditory Models In Isolated Word Recognition. *IEEE International Conference on Acoustics, Speech, and Signal Processing* (*ICASSP'84) Mar 1984*. Vol. 9, Part 1, 1984, pp33 – 36.

11. Boddy D., Boonstra A. and Kennedy G, *Managing Information Systems: An Organizational Perspective*. FT Prentice Hall, Harlow, ISBN 10: 0273686356, 2005, pp. 195-197.

12. Bourlard H Hervé and Morgan Nelson, *Connectionist Speech Recognition - A Hybrid Approach*. Kluwer Academic Publishers, ISBN: 0792393961, 1994, pp 4-7, 71-95.

13. Bourlard H. Hervé and Morgan Nelson, Hybrid HMM/ANN Systems for Speech Recognition: Overview and New Research Directions. *Adaptive Processing of Sequences and Data Structures, Lecture Notes in Artificial Intelligence* (1387), Springer, 1998, pp 389-417.

14. Brandtberg Tomas, *Introduction to Fuzzy Sets With Application To Image Processing And Pattern Recognition*. Centre for Image Analysis (CBA) a collaboration between Uppsala University and the Swedish University of Agricultural Sciences, www.cb.uu.se/~ingela/Teaching/ImageAnalysis/Fuzzy/TomasCompendium.pdf, 2004, pp. 1-12 [Accessed 2004].

15. Callan Robert, *The Essence of Neural Networks*. Prentice Hall Europe, ISBN: 0-13-908732-X, 1999, pp. 1-12, 20-26,59,84100,115.

16. Cantwell Smith Brian, Prologue To Reflection And Semantics In A Procedural Language. *Readings in Knowledge Representation*, Los Altos, Morgan Kaufman, 1985, pp. 31-40.

17. Carre R. and Mrayati M., New Concept in Acoustic-Articulatory-Phonetic Relation Perspectives And Application. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-89), 23-26 May 1989*, pp. 231 – 234 .

18. Castro J. R., Castillo O., and Martínez L. G, Interval Type-2 Fuzzy Logic Toolbox. *IEEE International Fuzzy Systems Conference (FUZZ-IEEE 2007)* (Online version available), 2007, pp 1-6.

19. Chai Sin-Kuo, *Multiclassifier Neural Networks for Handwritten Character Recognition*. PhD Dissertation, Ohio University, 1995, pp. 4-6, 13-19, 126-134.

20. Chiu S., Extracting Fuzzy Rules from Data for Function Approximation and Pattern Classification. *Fuzzy Information Engineering- A Guided Tour of Applications*, (ed. D. Dubois, H. Prade, and R. Yager), Chapter 9 John Wiley & Sons., 1997, pp. 149-162, (http://chius.homestead.com/files/ExtractRulesFromData.pdf).

21. Cohen J., Application of An Adaptive Auditory Model To Speech Recognition. *Journal Acoustic Society of America, November*, Vol. 78, Issue S1, 1985, pp. S50-S50.

22. Coupland S. and John R. I., An Approach To Type-2 Fuzzy Arithmetic. *UK Workshop on Computational Intelligence, Bristol, UK, September 2003*, pp.107–114.

23. *Data Mining Erudition Home*. URL: http://datamining.eruditionhome.com. [Accessed on 2009].

24. Davis S. B. and Mermelestein P., Comparison Of Parametric Representation For Monosyllabic Word Recognition In Continuously Spoken Sentence. *IEEE Transactions on Acoustic Speech Signal Processing*, Vol. 28, Issue 4, 1980, pp. 357-366.

25. Delmater Rhonda and Hancock Monte, *Data Mining Explained*. Digital Press, Newton, MA, USA, ISBN: 1-55558, 2001, pp. 31-55,129-134,140-146, 192,212-216,245,254,315.

26. Division of Information Technology, Engineering and the Environment, *Oracle Data Mining Concepts, 11g Release 1 (10.1).* Part No. B10698-01, 2007, pp. 1.1-1.4,3.1,4.1-19.1.

27. Doutrich B. A., Rabiner L.R. and Martin T., On The Effect of Varying Filter Bank Parameters on Isolated Word Recognition. *IEEE Transactions on Acoustic Speech Signal Processing*, Vol. 31, Issue 4, Aug., 1983,pp. 793-807.

28. Dubois D. and Prade H., *Fuzzy Sets and Systems: Theory and Applications*. Academic Press; First Printing edition October 28, ISBN: 0122227506, 1980, pp. 9-10, 158-185, 285-295, 317-330,358.

29. Eastman J. Ronald, Uncertainty and Decision Risk in Multi-Criteria Evaluation: Implications for GIS Software Design. *International Institute for Software Technology Expert Group Workshop on Software Technology for Agenda'21: Decision Support Systems,* Section 8, 1996.

30. Elmasri Ramez and Navathe Shamkant B., *Fundamentals of Database Systems-Edition 5*. Addison Wesley, ISBN: 9780321415066, 2007, pp. 960-991.

31. Fay Edward Allen, The Glossograph. *American Annals of the Deaf*. 1883, pp. 28, 67-69.

32. Fekkai Souhaila and Al-Akaidi Marwan M., *Fractal Based Speech Recognition and Synthesis*. PhD thesis, De Montfort University- Leicester-U.K., 2002, pp. 36-48, 68-80.

33. Fekkai Souhaila, P.Uniwin, and Al-Akaidi Marwan M., Neural Network Techniques for Speech Recognition Speaker Independent. *European Concurrent Engineering Conference (Euromeida 2001), Universidad politecnica de Valencia-Spain, April 2001*.

34. Fine L. Terrence, *Feedforward Neural Network Methodology*. Springer-Verlag New York, Inc., ISBN: 0-387-98745-2, 1999, pp.1-16, 53-58.

35. Flach Peter and Lavra Nada. *Rule Induction*. Chapter in Intelligent Data Analysis: An Introduction. ISBN 3-540-43060-1, 2003, pp. 229–267.

36. Frawley W., Piatetsky-Shapiro G., and Matheus C., Knowledge Discovery in Databases: An Overview. *Knowledge Discovery in Databases,* AAAI/MIT Press, ISBN 0-262-62080-4, 1991, pp. 1-30.

37. Fukunaga K., *Introduction to Statistical Pattern Recognition*. Academic Press, London, UK, 1991, pp. 1-9, 508-549.

38. Ghosh Joydeep, Multiclassifier Systems: Back to the Future. *The 3d International Workshop on Multiple Classifier Systems, MCS'02*, *June 2002, Cagliari, Italy, also in Lecture Notes in Computer Science, Springer-Verlag*, Vol. 2364, 2002, pp 1-15.

39. Goldberg E. David, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing company ISBN: 0201157675, 1989, pp.1-14, 92-106.

40. Goldblatt, R., Mathematical Modal Logic: a View of its Evolution, *in D. Gabbay and J. Woods (eds.), Handbook of the History of Logic*, vol. 6. Amsterdam: Elsevier, 2006.

41. Halmos P. R., *Navie Set Theory*. Reprinted, Springer-Verlag, New York, NY, ISBN 0-387-90092-6, 1974, pp.1-38.

42. Han Jiawei and Kamber Micheline, *Data Mining: Concepts and Techniques*. School of Computing Science - Simon Fraser University, Canada, http://www.statsoft.com/textbook/stdatmin.html, 2009 [Accessed 2009].

43. Hand J. David. Mannila Heikki and Smyth Padhraic, *Principles of Data Mining (Adaptive Computation and Machine Learning)*. MIT Press, Cambridge, MA, ISBN:9780262082907, 2001, pp. 1-24, 93-140, 271-366.

44. He Xing, Scordilis Michael and Li Gongjun, A Novel VQ-Based Speech Recognition Approach for Mobile Terminals. *Southeast Conference, IEEE, April 2005*, pp. 433-436.

45. Hegland M., Computational challenges in data mining. *Australian & New Zealand Industrial and Applied Mathematics Journal ANZIAM Journal* Vol. 42, (http://anziamj.austms.org.au/ojs/index.php/ANZIAMJ/article/view/588/456, 2000 pp. C1- C43.

46. Ho Shen-Shyang and Wechsler Harry, On the Detection of Concept Changes in Time-Varying Data Stream by Testing Exchangeability. *The 21st Conference on Uncertainty in Artificial Intelligence***,** Edinburgh, Scotland, 2005, pp. 267-274.

47. Hong Seokmin and Leckenby D. John, *How Data Mining Can Help Advertising Management*. Working paper, the university of Texas at Austin, 2000, pp. 157.

48. Hristev R. M., *The ANN Book*. Institute for Theoretical Computer Science, Graz University of Technology, Graz, Austria URL: http://www.igi.tugraz.at/lehre/NNA/WS04/downloads/ANN.pdf, 1998, pp. 1-26, 110-116, 129-137, 153-164. [Accessed 2005].

49. Hubbard Douglas, *How to Measure Anything: Finding the Value of Intangibles in Business- Edition 2* John Wiley & Sons, ISBN: 978-0-470-11012-6, 2010, Chapter 4, 5, and 6.

50. Hughes G. E. and Cresswell M. J., A *New Introduction to Modal Logic*. Rutledge, London, UK. ISBN: 0-203-02810-4, 1996, pp. 252-270.

51. Hunt J. Melvyn, Signal Representation. *Survey of the state of the Art in Human Language Technology*. Centre for Spoken Language Understanding (CSLU) URL:http://www.cslu.ogi.edu/HLTsurvey, Oregon Graduate Institute of Science and Technology, 1996 [Accessed 2004].

52. Ilah Fleming, Communication analysis: A Stratificational Approach. *A Field Guide for Communication Situation, Semantic, and Morphemic Analysis*, Vol. 2, Dallas: Summer Institute of Linguistics, 1988, pp. 369.

53. Ilyas Potamitis, Speech recognition based on feature extraction with variable rate frequency sampling. 4th *International Conference on Text, Speech, and Dialog*. *Zelena Ruda, Czech Republic, Sept.,2001,* 2001, pp.329-333.

54. Janikow Z Cezary, Fuzzy Decision Trees: Issues and Methods. *IEEE Transactions on Systems, Man, and Cybernetics*, Part B, Vol.28, Issue 1, 1998, pp.1-14.

55. Jaroslav Peregrin, Possible Worlds: A Critical Analysis. *Prague Bulletin of Mathematical Linguistics* 59-60, 1993, pp. 9-12.

56. John Divers, *Possible Worlds*. Routledge, London, UK. ISBN: 0-415-15555-X(hbk). 2002, pp. 3-40.

57. John Perry. *Semantics, Possible-Worlds (e-book).* URL: http://www.pdfgeni.com/book/John-Perry--philosopher--pdf.html. pp.4-7 [Accessed 2006].

58. John R. I., *Type-2 Fuzzy Logic Modelling Uncertainty*. http://www.cci.dmu.ac.uk/podcasts/CCIseminarBob.pdf, Centre for Computational Intelligence, School of Computing, De Montfort University – UK, 2006 [Accessed 2007].

59. Johnsonbaugh Richard, *Discrete Mathematics*-5th *e*dition. Prentice-Hall Inc., ISBN: 0-13-089008-1, 2001, pp. 1-52,56-114,427-434.

60. Jonathan Hey, *The Data, Information, Knowledge, Wisdom Chain: The Metaphorical link*. http://best.me.berkeley.edu/~jhey03/files/reports/IS290_Finalpaper_HEY.pdf. Intergovernmental Oceanographic Commission (UNESCO), 2004, pp.2-11 [Accessed 2008].

61. Jonathan M., Salang M., and Ozen Turhan, The association between non-stationary and interval type-2 fuzzy sets: a case study. *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2005), Reno, Nevada, USA, May 2005*, pp.224-229.

62. Jurafsky Daniel and Martin J. H., *Speech and Language Processing- Edition: 2*. Prentice-Hall Inc, ISBN: 0131873210, 2009, pp. 2-16,489-529,545-580,681-709.

63. Kandel Abraham, *Fuzzy Mathematical Techniques with Applications*. Addison-Wesley, ISBN-10: 0201117525, 1986, pp. 13-28, 196-210, 421-430,447-467.

64. Karnik Nilesh and Mendel J. M., An Introduction to Type-2 Fuzzy Logic Systems. *IEEE International Conference on Fuzzy Systems. Anchorage, AK, May 1998*, pp. 915-920.

65. Keijzer Ander De and Keulen Maurice Van, A Possible World Approach to Uncertain Relational Data. *International Workshop on Supporting Imprecision and Uncertainty in Flexible Databases (SIUFDB 2004), 3 Sep 2004, Zaragoza, Spain.* 2004, pp. 922-926.

66. Kendall, Kenneth E., and Kendall, Julie E., *System Analysis and Design*. Prentice-Hall International, Inc. ISBN: 0-13-042365-3, 2002, pp. 241-247.

67. Kirkby Richard, Frank Eibe, and Reutemann Peter, *WEKA Explorer User Guide*. http://www.docstoc.com/docs/2140760/WEKA-Explorer-User-Guide-for-Version-3-4-3. The University of WAIKATO, 2004, pp. 3-10. [Accessed 2004].

68. Klir G. J. and Wierman M., *Uncertainty-Based Information: Elements of Generalized Information Theory*. Springer Verlag, Heidelberg, ISBN: 3-7908-1242-0, 1998, pp. 1-10,44,100-104,131-134.

69. Klir George J. and Folger T. A., *Fuzzy Sets, Uncertainty and Information*. Englewood Cliffs, Prentice Hall, ISBN: 0133459845, 1988, pp.73-74.

70. Kosko Bart, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, ISBN:0-13-611435-0, 1992.

71. Kumar S., Ghosh J., and Crawford M.M., Hierarchical Fusion of Multiple Classifiers for Hyperspectral Data Analysis. *Pattern Analysis and Applications, special issue on Fusion of Multiple Classifiers*, 2002, pp. 210-220.

72. Kuncheva Ludmila, *Combining pattern classifiers: Ideas and Methods*. Wiley-IEEE, ISBN 0471210781, 2004, pp. 45-80,101-111, 189-200.

73. Li L., Lin W. H., and Liu H., Type-2 fuzzy logic approach for short-term traffic forecasting. *IEE Proceedings Intelligent Transport System, Vol. 153, Issue 1, March 2006*, pp. 33 – 40.

74. Lu Ruzhan, Dynamic logic with possible world. *Proceedings of the 15th conference on Computational linguistics*, Kyoto-Japan 1994, by: Association for Computational Linguistics Morristown, NJ, USA, 1994, pp. 1267 – 1269.

75. Luger George F., *Artificial Intelligence, Structures and Strategies for Complex Problem Solving- 6th edition*. Pearson Education limited, ISBN: 0321545893, 2009, Chapter 2 and 7.

76. MAES SAM, MEGANCKAND STIJN, and MANDERICK BERNARD, Inference in multi-agent causal models. *International Journal of Approximate Reasoning,* Volume 46, Issue 2, 2007, pp. 274-299.

77. Makhoul J., Linear Prediction: A tutorial review. *Proceedings of IEEE, Vol. 63, Issue 4.* 1975, pp. 561-580.

78. Markel J. D. and Gray A. H., *Linear Prediction of Speech*. Springer-Verlag, NewYork, 1976.

79. Markowtz A. Judith, *Using Speech Recognition*. Prentice Hall Incorporated, ISBN-13: 9780131863217, 1996, pp. 1-53, 127-172.

80. Mcmichael A., A New Actualist Modal Semantics. *Journal of Philosophical Logic*, Springer Netherlands, Vol. 12, NO. 1, 1983, pp. 97.

81. Medsker L. R., *Hybrid Intelligent Systems*. Springer Kluwer Academic Publishers, Boston, ISBN: 0792395883, 1995.

82. Mendel J. M., *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Prentice-Hall, ISBN-13: 9780130409690, 2001, pp. 3-110, 517-525.

83. Mendel J. M. and John R. I., Type-2 fuzzy sets made simple. *IEEE Transactions on fuzzy systems*, Vol. 10, NO. 2, 2002, pp. 117-127.

84. Mendel J. M., Hagras Hani, and John R. I., Standard background material about interval type-2 fuzzy logic systems that can be used by all authors. http://ieee-cis.org/_files/standards.t2.mac.pdf. *IEEE computational Intelligence Society* , 2006 [Accessed 2006].

85. Mendel, J. M., Type-2 Fuzzy Sets: Some Questions and Answers. *IEEE Connections, Newsletter of the IEEE Neural Networks Society*, Vol. 1, 2003, pp. 10-13.

86. Meta Group Application Development Strategies, *Data Mining for Data Warehouses: Uncovering Hidden Patterns,* 1995.

87. Michael Berry and Gordon Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Support*. John Wiley and Sons, ISBN: 0471482994, 2003, pp. 51-68, 112-144,187-216,243-360.

88. Michalski R., Carbonell J. and Mitchell T., *Machine Learning: An Artificial Intelligence Approac*h (*Volume I)*. Morgan Kaufmann Publishers Tioga, Palo Alto, CA, ISBN: 0934613095, 1983, pp. 3-14, 331-345.

89. Michie D., Spiegelhalter D. and Taylor C., *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Chichester, ISBN:0-13-106360-X, 1994 .

90. Mitchell, T., Does Machine Learning Really Work, *AI Magazine*, Vol. 18, No. 3, AAAI Press, 1997, pp. 11-20.

91. Mitra Sushmita and Pal Sankar K., Data Mining in Soft Computing Framework: A Survey. *IEEE Transactions on Neural Networks*, Vol. 13, No. 1, 2002, pp. 3-14.

92. Modi Bansal and Sharma, *Neural Networks*. URL: http://www.academic.marist.edu/~jzbv/architecture/Projects/S2002/NeuralNet/INDEX.HTM, 2002 [Accessed 2009].

93. Mrayati M., Carre R., and Guerin B., Distinctive Regions and Modes: A New Theory of Speech Production. *Speech Communication*, Vol. 7, 1988, pp. 257 – 286.

94. Negnevitsky Michael, *Artificial Intelligence - A Guide to Intelligent Systems-2nd edition.* ISBN: 0321204662, Addison Wesley, 2005, pp. 1,18,87-94.

95. Neuman Edward, *Programming in Matlab*. http://www.math.siu.edu/matlab/tutorials.html. Department of Mathematics, Southern Illinois University at Carbondale, 2005 [Accessed 2005].

96. Nourie Dana, *Getting started with an integrated development environment (IDE)*. http://java.sun.com/developer/technicalArticles/tools/intro.html. Sun Developer Network (SDN), Sun Microsystems, 2005 [Accessed 2005].

97. Oppenheim A.V. and Schafer R. W., *Digital Signal Processing*. Prentice-Hall, ISBN 0-13-214635-5, 1975, pp. 1-44, 571.

98. Otterlo Martijn Van, *Neural Networks and Neurocomputing Practical Exercises in Matlab*. http://mathforum.org/kb/profile.jspa?userID=67946, 2004 [Accessed 2004].

99. Ozen Turhan And Garibaldi J. M., Investigating Adaptation In Type-2 Fuzzy Logic Systems Applied To Umbilical Acid-Base Assessment. *European Symposium on Intelligent Technologies, Hybrid Systems and Their Implementation on Smart Adaptive Systems, Oulu, Finland, June 2003*, pp. 289 – 294.

100. Pearlson K.E. and Saunders C.S., *Managing and Using Information Systems: a Strategic Approach*. John Wiley & Sons Incorporated, New York, ISBN-13: 9780471320012, 2004, pp. 33-37, 189-191.

101. Pechenizkiy Mykola, Puuronen Seppo, and Tsymbal Alexey, Feature Extraction for Classification in Knowledge Discovery Systems. *International Journal on Information Theories and Applications*, Vol. 10, Issue 3, 2003, pp. 271-278.

102. Pedrycz Witold and Gomide Fernando, *An Introduction to Fuzzy Sets*. MIT Press, ISBN-13: 9780262161718, 1998, pp. 1-84, 181-204.

103. Peregrin Jaroslav, Absolute and Relative Concepts In Logic. *Logica yearbook 2000* Praha, Filosofia, 2001, pp. 71-77.

104. Pruss Alexander Robert, *Possible Worlds: What They are Good for and What They are Not.* PhD Thesis, University of Pittsburgh, Pittsburgh, Pennsylvania, 2001, USA.

105. Rabiner Lawrence and Juang Biing Hwang, *Fundamentals of Speech Recognition*. PTR Prentice-Hall, Inc., ISBN: 0-13-015157, 1993, pp. 1-13, 20-130, 242-244, 321-322.

106. Rao Valluru B. and Rao Hayagriva V., *C++ Neural Networks and Fuzzy Logic*. M & T Books, ISBN: 1-55828-298-x, 1993, pp.1-14, 25-41, 67-91.

107. Rappaport Theodore, *Wireless Communications Principle and Practice*. Prentice Hall PTR, ISBN-13: 9780130422323, 2002, pp. 361-394.

108. Reddy D.R., Speech recognition by machine: a review. *Proceeding of IEEE*, Volume 64, Issue 4, 1976, pp 501 – 531.

109. Rich Elaine and Knight Kevin, *Artificial Intelligence-2nd edition*. McGraw Hill, ISBN 0-07-052263-4, 1991, pp. 246, 418-419, 452-453.

110. Richard D. and Lippman R. P., Neural Network Classifiers Estimate Bayesian Posteriori Probabilities. *Neural Computation*, 1991, pp. 461-483.

111. Rob Gerritsen, Assessing Loan Risks: A Data Mining Case Study. *IT Proceeding by IEEE*, Vol. 1, Issue 6, 1999, pp. 16 – 21.

112. Rowley Jennifer, The Wisdom Hierarchy: Representations of the DIKW Hierarchy. *Journal of Information Science, Vol. 33, No. 2*, 2007, pp. 163-180.

113. Rud Olivia Parr, *Data Mining Cookbook: Modeling Data for Marketing, Risk, and Customer Relationship Management*. John Wiley & Sons publishing company ISBN: 0-471-38564-6, 2001, pp. 51-124.

114. Schmid Christian, *Course on Dynamics of Multidisplicinary and Controlled Systems*. http://www.atp.ruhr-uni-bochum.de/rt1/syscontrol/main.html, 1999 [Accessed 1999].

115. Sepulveda Roberto, Melin Patricia, Rodriguez Antonio, Mancilla Alejandra, and Montiel Oscar, Analyzing the effects of the Footprint of Uncertainty in Type-2 Fuzzy Logic Controllers. *Engineering Letters*, 2006, pp.1-10.

116. Shirai K. and Honda M., Estimation of articulatory parameters from speech waves. *Electronic and Communication in Japan*, Vol. 61, 1978, pp. 1-8.

117. Silva M. Marcelo, Maia T. Thiago and Braga A. P., An evolutionary approach to transduction in support vector machines. *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, - HIS'05, Rio de Janeiro-Brazil, November 2005,* pp. 329 – 334.

118. Simoff Simeon, Böhlen Michael H., Mazeika Arturas, *Visual Data Mining: Theory, Techniques and Tools for Visual Analytics*. Springer, ISBN-13: 978-3540710790, 2008, pp.1-12.

119. Solso S. Robert, Maclin M. Kimberly, and Maclin H. Otto, *Cognitive Psychology*. Seventh edition, Pearson Education, ISBN: 0-205-41030-8, 2005, pp. 110-115.

120. Sproat R. and Riley M. D., Compilation of weighted finite-state transducers from decision trees. *The 34th Annual Meeting of the Association of Computational*

Linguistic (ACL), June 1996, University of California, Santa Cruz, California, USA. 1996, pp. 215-222.

121. Stahlbock Robert, Crone F., and Lessmann Stefan, *Data Mining: Special Issue in Annals of Information Systems*, Volum 8, Springer Science + Business media. ISBN: 978-1-4419-1279-4, 2009, pp. 1-18, 53-74,112.

122. Stephanou H. E. and Sage A. P., Perspectives on imperfect information processing. *IEEE Transactions on Systems, Man and Cybernetics, Volume SMC-17, Number 5,* 1987, pp. 780-798.

123. Stergiou Christos, and Siganos Dimitrios, *Neural Networks*, http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html, 1996, [Accessed 2008].

124. Stuckless R., Developments in Real-Time Speech-To-Text Communication For People With Impaired Hearing. http://www.cs.uvm.edu/~icdm/10Problems/10Problems-06.pdf. Communication access for people with hearing loss, Baltimore, York Press., 1994, pp.197-226. [Accessed 1994].

125. Tajchman G., Fosler E., and Jurafsky Danial, Building Multiple Pronunciation Models for Novel Words Using Exploratory Computational Phonology. *Eurospeech-95*, 1995, pp. 2247-*2250.*

126. *The American Heritage® Science Dictionary*, http://dictionary.reference.com/browse/proposition, Houghton Mifflin Company, 2005, [Accessed 2006].

127. *Training and Test Data and Speech Header Software-NIST Speech Disc*, CD1-1.1, The Darpa Timit Acoustic-Phonetic Continuous Speech Corpus, 1990.

128. Wang Xue, *Incorporating Knowledge on Segmental Duration in Hmm-Based Continuous Speech Recognition*. PhD thesis, Institute of Phonetic Sciences, Herengracht – Amsterdam/ Netherland, 1997, pp. 1-8, 11-12.

129. Wasserman P.D., *Advanced Methods in Neural Networks*. Van Nostrand Reinhold, New York, ISBN: 0-442-00461-3, 1993, pp. 35-55,147-205.

130. Weisstein Eric, *Wolfram Math World*. http://mathworld.wolfram.com/VectorFunction.html. web of Wolfram Research, Inc., 1999 [Accessed 1999].

131. White G. and Neely R. B., Speech recognition experiments with linear prediction, bandpass fltering, and dynamic programming. *IEEE Transaction on Acoustic Speech Signal Processing,* Volume 24, Issue 2, 1976, pp. 183 – 188.

132. Wicker Z. and Terhadt E., Automatic Speech Recognition Using Psychoacoustic Models. *Journal Acoustic Society*, Vol. 65, 1979, pp. 487-498.

133. Witten H. Ian, *Principles of Computer Speech*. Academic Press, ISBN: 0-12-760670-9, 1982.

134. Wong Hau-San, Hierarchical Multi-Classifier System Design Based On Evolutionary Computation Technique. *Multimedia Tools and Applications journal, Springer Netherlands*, 2007, pp. 91-108.

135. Wooters C. and Stolcke A., Multiple-Pronunciation Lexical Modelling in a Speaker Independent Speech Understanding System. *Proceeding International Conference on Spoken Language Processing (ICSLP-94), , Yokohama, Japan, September 1994*, pp. 1363-1366.

136. Wu Hongwei and Mendel J. M., Uncertainty Bounds and Their Use in the Design of Interval Type-2 Fuzzy Logic Systems. *IEEE transactions on fuzzy systems*, Vol. 10, 2002, pp. 622-639.

137. Yang Qiang, 10 Challenging Problems in Data Mining. *International Journal of Information Technology & Decision Making,* Vol. 5, No. 4, 2006, pp. 597-604.

138. Yong Shi , Ding Chris, *ICDM 2009 Workshop on Optimization Based Methods for Emerging Data Mining Problems (OEDM'09),* Held in conjunction with The 2009 IEEE International Conference on Data Mining (ICDM2009), Florida, USA, 2009.

139. Zadeh L. A., Fuzzy Sets. *Journal of Information and Control*, 12, 1965, pp. 338-353.

140. Zadeh L. A., The Concept of a Linguistic Variable and its Application to Approximate Reasoning. *Information Science,* Vol. 8, 1975, pp. 199-249.

141. Zadeh L.A., Possibility Theory and Soft Data Analysis, *Mathematical Frontiers of Social and Policy Sciences*, Editor: Cobb L. and Thrall R.M. , West view Press, Boulder CO, 1981, pp. 69-129.

142. Zhang Byoung-Tak, Artificial Neural Nets, Adaptive Genetic Programming, and Intelligent Learning Machines. *VeKNI Nachrichten*, volume 41, 1995, pp. 17-31.

143. Zhang Shichao, Zhang Chengqi, and Wu Xindong, *Knowledge discovery in multiple databases*. Springer, ISBN: 978-1-85233-703-2, 2004, pp. 27-63.

144. Zhang Wu, Chen Zhangxin, and Douglas Craig C., *High Performance Computing and Applications*. 2nd international conference, HPCA 2009, Shanghai, China, 2009, pp. 529.