Title

Artificial Intelligence

Existing and Emerging Ideas ©

by

Godfried B. Williams

CONTENTS

Chapter 1

Overview of Artificial Intelligence

1.1 Background

The history of A.I in most parts of the literature seems to suggest that Experts Systems contributed very much to the growth and popularity of artificial intelligence. One of the early expert system developed was developed by Newell A, Simon H, A and Shaw J.C.

In 1956 a small group of researchers gathered in a conference in Dartmouth to discuss intelligent applications, theory and underlying concepts. According to one school of thought this was the genesis of artificial intelligence. It has become known as the Dartmouth conference.

1.2 What is A.I?

Definitions

1. It is the mimicking of human intelligence by a machine, system or process. Examples of such a machine, system and process are Robots, Auto-pilot navigation systems for aeroplanes, satellite navigation systems and auctioning on the internet. The characteristics of intelligence are mainly, ability to apply judgement, discernment, adaptability, rationality and sensitivity Williams G (2006).

.

2. The branch of computer science that is concerned with the automation of intelligent behaviour Luger and Stubblefield (1993)

3. The study of the computations that make it possible to perceive, reason and act Winston (1992)

4. A field of study that seeks to explain and emulate intelligent behaviour in terms of computational processes Schalkoff (1990).

5. The art of creating machines that performs functions that require intelligence when performed by people Kurzwell (1990).

The definitions of A.I can be classified into 4 areas, namely:

1. Systems that think as humans
2. Systems that think rationally
3. Systems that act like humans
4. Systems that act rationally

Other disciplines that explore intelligence are philosophy, psychology and linguistics. These disciplines attempt to understand intelligence, Whiles A.I attempts to understand and build intelligence systems.

In other for us to build a system that thinks and acts as humans in a rational manner we have to understand subject areas such knowledge representation (logic), learning theory(systems that learn), automatic reasoning and natural language processing.

1.3 Evolution of A.I

Philosophy: 450 B.C to present

Socrates, Plato and Aristotle laid the foundation of western thought and cultural reasoning.

1. Age of Dualism
2. Materialism
3. Empiricist
4. Induction
5. Logic – observation, reasoning and proof (Aristotle, Boole)
6. Means and end analysis

Mathematics (A.D. 800- present)

Philosophy to formal science representation in AI required the study of computation,
logic and probability.

Introduction of Arabic numerals to the western world by Al-Khowarazmi, an Arabic mathematician in the ninth century.

1. Algorithm
2. Incomplete theorem
3. Intractability (Theorem of complexity)
4. Reduction (Theorem of complexity)
5. Decision Theory

Psychology (1878 – present)

German physicist Hermann von Helmholtz (1821-1894), Wilhelm Wundt (1832-1920). The Psychology of human vision.

1. Behaviourism
2. Cognition

Computer programming and systems

1. First operationally programmable computer Z-3 by Konrad Zuse 1941 in Germany
2. 1940-42 the ABC in USA by Atanasoff and Clifford Berry at Iowa State University
3. Mark 1,2,3 at Harvard University
4. ENIAC at University of Pennsylvania first general purpose digital computer
5. IBM 701 in 1952 by Nathaniel Rochester

Linguistics

This is the study of language representation and verbal behaviour. Linguistics contributes enormously to knowledge representation formalisms. Computational linguistics is pivotal to knowledge representation.

1.4 Summary

Chapter 1 provided background of Artificial Intelligence, with insight into a wide range of definitions from Williams G(2006), Luger and Stubblefield(1993), Winston(1992), Schalkoff(1990), Kurzwell(1990) . It also classified A.I into four main areas; Systems that think as humans, Systems that think rationally, Systems that act like humans and Systems that act rationally. Philosophy, psychology and linguistics were areas of study that also explored the foundations of intelligence. A.I has its historical foundations from the age of dualism, materialism, empiricist, induction, logic, algorithm, incomplete theorem, intractability (Theorem of complexity), reduction (theorem of complexity), decision theory, psychology, behaviourism, cognition, computer programming and linguistics.

Chapter 2

Knowledge Representation

2.1 Knowledge representation (KR)

Knowledge is represented through logic.

Rules + Facts = Theory or Rules + Facts = Knowledge

What is a rule?

A natural way of expressing knowledge based on deduction is known as a “rule”. Example of such a rule could be expressed using the following syntax.

If A is true and B is true and C is false then A is true

This is a type of rule of the form “If”: meaning a Logical condition and “Then”: An indicated action. There are many formalisms of expressing logical conditions and actions that form the basis of rules. The natural way of expressing what to do is represented in the form of procedural knowledge. Knowing what to do in any problem situation gives you a sense of direction.

Rules can be used to create a consistent form of knowledge. Consistency is a very important aspect of presenting knowledge related to a specific problem. The concept of consistency leads knowledge standardization. It helps to maintain consistency in the data types used in defining the problem.

2.1 Logic

This is a formal mechanism for representing knowledge. Logic is expressed in a form of language. Logic allows us to reason about the world, rather than acting upon the knowledge we possess as humans.

2.2 Propositional Logic

Propositional logic is a representation language which allows expression and reasoning using Boolean logic, true or false.

Example: “Soccer is a game” “The game is played is by 11 players”

The above statements are propositions, sometimes known as atoms.

Propositional logic has five basic connectives which are used in creating propositions. Below are the connectives and notational representation.

No.	Connective	Notation
1	Not	¬
2	And	^
3	Or	v
4	If then	→
5	If and only if	↔

Example using the connectives

Example 1

A = " Soccer is a game"
B = "Soccer is played by 11 players"

This means the composite proposition A ^ B means that Soccer is a game and the game is played by 11 players.

Example 2

X = "Glaucoma is a disease"
Y = "Glaucoma usually affects ages between 25-85"

The composite proposition = X ^ Y

This means Glaucoma is a disease that affects people within the age range of 25-85 years.

Well formed propositional logic – This is a propositional logic having any of the following forms:

1. A statement is a well formed formula

2. If "A" is a well formed formula, then (¬ A) is a well formed formula

3. If A and B are well formed formulae then

(A ^ B), (A v B), (A→ B) and (A ↔ B) are well formed formula

4. No other formula is well formed

(A ^ (B→ C)) as well as (A v (¬B)) are well formed formulae. However ¬ C is not well formed.
The formula X → Y ^ Z is equal to the formula (X → (Y ^ Z))

2.3 Horn Clause Logic

This is a system for reasoning about real events and describing entities as well as the way such events and entities are presented. Below is a statement representing the relationship between two natural entities?

1 .Kwaku is the father of Kwame

The two entities in this statement are “Kwaku” and “Kwame”
The relationship between these two entities is one to one. One of the entities is the father of the other. In horn clause logic this is represented as:

Father (“Kwame”, Kwaku)

The order of the horn clause statement can be initialised by the knowledge engineer.

The statement in section 2.3 can be represented using variables. Example of such representation is:

R is the great grandfather of Z, IF R is the father of X and X is the Grandfather of Z. Horn clause represents this formula in variable form as:

R (Z, R): - R(X, R), X(X, Z)

A statement like “Kwaku” is the father of “Kwame” is a fact. A statement like Kwame is the grandson of Kofi if Kwaku is the father of Kwame are rules.

2.3.1 Theory representation:

Father (“Bill”, Johannes”)

Father (“Pamela” “Bill”)

Grandfather (Person, Grandfather)

Father (Person, Father)

Father (Father, Grandfather)

The purpose of theory is to provide answers to statements such as:

Is Johannes the father of Bill?
Who is the father of Pamela?
Is Johannes the grandfather of Pamela?

The questions in section 2.1.1 are goal oriented statements.

These could be represented in horn clause logic as:

? – Father (“Bill”, “John”)
? – Father (“Pamela”, Z) // This question demands an answer with respect to the father of Pamela
?- Grandfather (“Pamela” “John”)

Goals can be answered with either “No” or “Yes”

2.4 First Order Predicate logic

This is a formula that satisfies an expression having either True or False. The first order predicate is more expressive than propositional logic. It comprises predicate symbols, variables, two quantifiers, function symbols and connectives. It is a logic representation format that allows formulation of quantified statements. Predicate logic provides a language for expressing assertions (axioms) about a certain “object”. It is very declarative in its linguist representation.

2.5 Knowledge Domain

The representation of the knowledge domain is depicted by a combination of rules and facts. This is the theory underlying knowledge. Knowledge domain is the scope underpinning the study of any subject matter. For example domain that depicts different types of eye diseases.

2.6 Sample Exercises

Exercise 1

1. (a) Translate the following statements to Horn Clause logic

(i) Is Rafael the father of Jacobs?

ANS:
? – Father ("Jacobs",Rafael)

(ii) Who is the father of Clements?

ANS:
? – Father ("Clements", X)

(iii) Is James the grandfather of Lawrence?

ANS:
? – grandfather("Lawrence", James)

(iv) Peter is the father of Bill

ANS:
Father ("Bill", Peter)

(v) Bill is the great grandfather Samuel

ANS:
? – Great grandfather ("Samuel", Bill)

Exercise 2

2. (a) Translate the following statements to Horn Clause logic

(iv) Is Jullian the father of Musa?

ANS:
? – Father ("Musa", Julian)

(v) Who is the father of Ahmed?

ANS:
? – Father ("Pam", X)

(vi) Is Julian the grandfather of Ahmed?

ANS:
? – grandfather ("Ahmed", Julian)

(iv) John is the father of Bill

ANS:
Father ("Bill", John)

(v) Bill is the great grandfather Samuel

ANS:
? – Great grandfather ("Samuel", Bill)

2.7 Summary

Chapter 2 presented fundamentals of knowledge representation with references to rules, logic, propositional logic, first order predicate logic, knowledge domain, horn clause logic, theory representation and sample exercises for the student and reader. "A natural way of expressing knowledge based on deduction is known as a "rule". This is a formal mechanism for representing knowledge, whiles logic expresses a form of language. Logic helps us to reason about the world, rather than acting upon the knowledge we depend upon for reasoning as human beings. The chapter described propositional logic as a representation language which allows us to express and reason using Boolean logic.

Chapter 3

Expert Systems

3. 1 What is an expert system?

An expert system is a system that offers solutions to specific problems in a given domain and also has some capability to give advice at a level comparable to that of experts in the field [1].

An expert system is also a system that handles real-world complex problems requiring an expert's interpretation and solves these problems using a computer model of expert human reasoning reaching the same conclusions that a human expert would reach if faced with a comparable problem.

Below is a general model of an expert system

Expert System = Knowledge + Inference [1]

Knowledge of an expert system comprise a set of rule and fact.

3.2 Characteristics of Expert Systems

1. Lacks learning and adaptability. Performance does not improve through learning

2. Knowledge elicitation is mostly done through expert source, thus a person or a book

3. Takes a long time to extract and develop rules necessary for building an expert system

4. Works within a narrow knowledge domain

5. Performance of system falls when domain knowledge changes

6. Communication to the user is efficient and effective. Can obtain relevant information to system by asking specific questions

7. Cannot deal with noise and uncertainty effectively. A higher knowledge is essential

8. It takes a long to run program if the set of rule is large

9. Maintenance of rule and knowledge can be difficult

3.3 Things required in developing an expert system

1. Knowledge sources
2. Knowledge acquisition
3. Knowledge elicitation
4. Knowledge engineer
5. Knowledge representation

3.4 Rule-based expert systems

Rule based system represent knowledge in terms of rules that tell you what you should do or what your conclusion should be in terms of inference. It comprises of IF-THEN rules, facts and an inference engine or interpreter that controls the application of the rules. There are primarily two basic methods of inference, namely forward chaining and backward chaining systems.

Figure 1 shows components of modern rule-based expert systems development and forms an important part in the investigation of efficiency, effectiveness and design issues of expert systems in applications. [2]

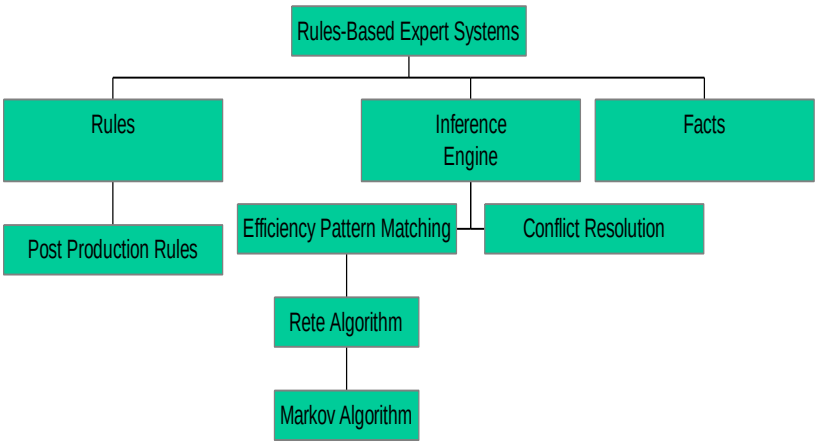


Figure 1 – Modern Rule Base Expert System

Below is an analysis of each part of the rule base expert system model in figure 1. The model depicts both the classical rule-base expert system and extension of the model in recent times. It also reflects the structure of knowledge and reasoning in both traditional and modern expert systems [2].

3.4.1 Facts

Facts are the evidence about the problem in context. The facts of an expert system form part of the knowledge-base of an expert system. Sometimes the facts about a problem can vary due to different views expressed by different experts or may have a low level of certainty of truth about the facts. Anytime different experts disagree on a subject matter, there is some level of inconsistency. This makes the facts expressed about the problem unclear, the non clarity of these facts could be translated as fuzzy. Fuzzy logic could be used in expressing or formulating such a problem. Baye's theorem on the other hand is used in computing the conditional probability of the facts surrounding the problem situation [3].

Fact finding methods such as interviewing, questionnaires, observations and records inspections are ways of eliciting knowledge from potential sources of expert or specialist knowledge. The knowledge that is elicited and gained from the sources is represented using knowledge representation formalisms.

3.4.2 Rules & Production rules

The natural way of expressing knowledge based on inference is in the form of rules. Example of such a rule could be expressed using the following syntax.

If A is true and B is true and C is false then conclusion is X. [4]

This is a type of production rule of the form “If”: meaning a Logical condition and “Then”: An indicated action.

There are several ways of describing these logical conditions and actions. A group of production rules is known as a production system. The objective of the production system is to evaluate the production rules in the correct order, take the required action and resolve conflicts. Production rules make meaning out of observations made in any given scenario that needs expert interpretation and draws conclusions out of the observations [4].

There are many formalisms of expressing logical conditions and actions that form the basis of the production rules. This is managed by a production system. The production system should be able to evaluate the production rules in the correct order, take the required actions and resolve conflicts

The following are a number of advantages that could be associated with production rules:

- There is a natural expression of what to do, thus procedural knowledge.
- Knowing what to do in any problem situation gives you a sense of direction. It also makes any specific action to be taken under the given circumstance structured in an orderly manner.
- The rules can be used to uniformly express any form of knowledge.

- Uniformity is a very important aspect of presenting knowledge related to a specific problem. The concept of uniformity paves the way for consistency. It helps to maintain consistency in the data types used in defining the problem situation [5]. Examples of expert systems that use rules HEPAXPERT-I is an expert system that translates the results of routine serologic test for infection with hepatitis A or B virus [6].

The knowledge-base of the HEPAXPERT-I system has 13(thirteen) IF-THEN rules for hepatitis A and 106(One hundred and six) IF-THEN rules for hepatitis B serology.

3.4.3 Inference Engine

The production system is integrated with the inference engine. The reasoning sub tasks diagram in figure 2 demonstrates the key role of the inference engine of an expert system. In other words we say that the inference engine is the reasoning part of the expert system. Within the inference engine lays all reasoning. The reasoning strategy employed could be that of backward or forward chaining [4].

A forward chaining system begins initially with the facts and uses the rules to derive or draw new conclusions. Forward chaining systems represent the

facts in working memory that is continuously updated. The rules in the system normally represent condition and action rules that contribute to the arriving of any conclusion.

Forward chaining systems usually require the user to encode a set of guidelines to guide a search. The rule-based usually consist of both domain and control knowledge.

Backward chaining begins with a goal or hypothesis and then determines to prove that goal. Sub goals could also be set as a way of proving the main goal. Backward chaining systems are goal driven.

Backward chaining is only very useful if you actually know all the initial facts. Given a goal state we try to prove that state in a backward chaining strategy. If the system is not capable of proving a certain goal, it looks for other rules whose conclusions previously match the goal. We can in principle apply the same set of rules for both forward and backward chaining. Based on the above analysis we can understand the reasoning strategy which expert systems use.

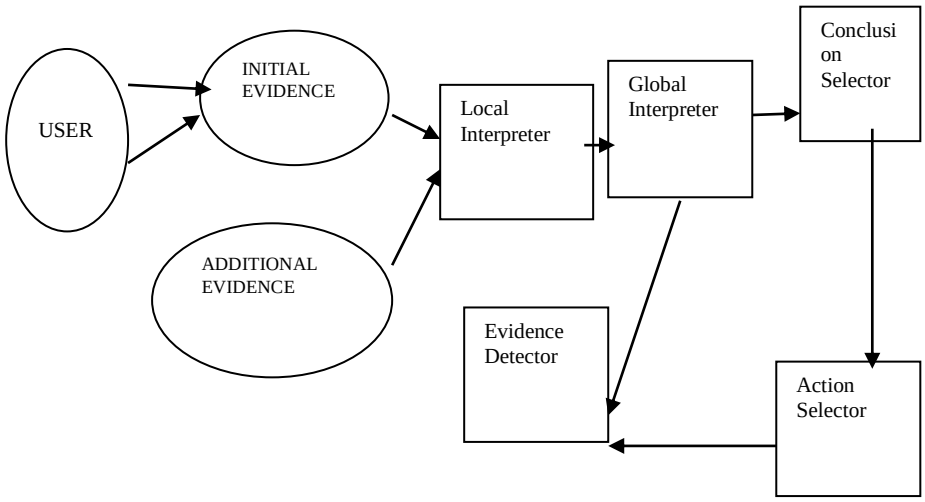


Figure 2 - Information flows for reasoning sub tasks of an expert system.

3.5 Overview of Expert Systems in Medicine

A medical expert system could be defined as an expert system that is capable of providing knowledgeable information to assist in the diagnosis of diseases. The beginning of the 1960's commemorated the genesis of research in artificial intelligence using expert systems in the development for various applications in science including engineering, chemistry, geology and medicine. "Medical expert systems are designed to aid

physician overcome problems, often classification tasks which should otherwise be referred to a specialist in a particular area of medicine".

Below is a summary of expert systems, their domains, knowledge representation formats and methods of inference.

MYCIN, EMYCIN AND NEOMYCIN

The application domain was internal medicine, applied a set of rules and fuzzy logic for knowledge representation and backward chaining as the reasoning strategy.

HEPAXPERT-I

The application domain was Hepatitis, used a set of rules for knowledge representation and a form of efficient pattern algorithm as the reasoning strategy.

ILLIAD/RMD, PACEMAKER AND CASNET

The domain of application was RMS (renal mass diagnosis), Heart and Opthalmology, used a set of rules and causal logic to represent knowledge.

DEVICE (Data-driven and event-driven rule integration using complex events)

Used Rete algorithm as the method of inference. The data and event driven rule integration used the complex events that incorporated production rules into an active object oriented database. The integration was based on rule compilation into a complex event network that emulated a Rete-like discrimination network [14].

DRAWBACKS OF THE CLASSICAL EXPERT SYSTEM MODEL

3.6 Problems associated with rules.

Rules can be difficult sometimes to use when the set of rules governing a particular process is large in number. This may cause the rules not to be as efficient as expected.

Some rules could be highly dependent on other rules. The high inter-dependency of rules on each other can be another form of problem that has to be addressed.

Rules can be inflexible to implement due to its inability to be dynamic in solving a particular problem.

- **Problems associated with facts**.

Facts can be fuzzy or be partially true. That is when several experts have different views concerning any form of knowledge.

The facts in a rule-based expert system can have some level of uncertain accuracy.
Such circumstances demand some form of probability reasoning in order to arrive at a high level of correctness.

- **Problems with forward chaining**

Forward chaining has no reasoning goal in mind, as a result can generate a lot of irrelevant assertions.

- **Disadvantages of backward chaining**

A backward-chaining system asks for information when it is relevant and useful when knowledge is expensive to access.

Backward-chaining systems are quite inflexible because control strategy is built into the system.

3.7 Extensions of Classical Expert System Model

- **Probabilistic reasoning**

Probabilistic reasoning is a form of reasoning that deals with uncertainty using the theory of probability. Bayesian network is a powerful tool for representing the relationship among a set of variables and dealing with uncertainties in expert systems [7]. Bayesian decision theory is the framework for making decisions under uncertainty.

MYCIN was example expert systems that used probability reasoning [1].

The system helped with the diagnosis of infectious diseases such as Meningitis. MYCIN represented its knowledge as a set of IF THEN rules with certainty factors. The following is an example of the representation.

IF the infection is primary-bacteria

AND the site of the culture is one of sterile sites

AND the suggested portal of entry is the gastrointestinal tract

THEN there is suggestive evidence (0.7)

That infection is bacteroid.

The 0.7 is roughly the certainty that the conclusion will be the given evidence. MYCIN was written in LISP and its rules formally represented as LISP expressions. It is important for us to note that MYCIN is a goal directed system that uses backward chaining reasoning strategy. MYCIN became the basis of designing new expert systems. MYCIN though pioneering much expert system research also had a number of problems. This led to new systems such as the EMYCIN and NEOMYCIN [1].

- Why MYCIN used probabilistic reasoning

MYCIN was applied for suspected multiple infections that might be present in a person.

This is reasoning about multiple diagnosis [8]. This has been one of the major difficulties encountered by most expert systems. MYCIN was designed to deal with uncertainty. It is important to note that medical diagnosis itself is innately an uncertain business [8].

The probabilistic reasoning of MYCIN enabled it to provide a high level of estimate in terms of correctness when multiple infections were present in any person.

3.8 Causal representation of knowledge in ES

One of the areas in the application of expert system is the health sector is differential diagnosis. RMD expert system was used in differential diagnosis in renal masses. For the sake of the lay person differential diagnosis is a form of diagnosis where the diagnoses of the disease is not necessarily only dependent on the unique symptoms but also taking into consideration some generic symptoms which may apply to other diseases. The ILIAD expert system was used to develop a pre-operative renal mass diagnosis system [9].

It provided a differential diagnosis before the conduction of an operation by using personal data, medical history, symptoms and signs, laboratory data and specific diagnosis procedures. 123 (one hundred and twenty three) patients were made to undergo the pre-operative diagnosis of the renal

masses differential diagnosis techniques [9]. 18 (eighteen) diagnostic categories were considered. It is not very apparent the techniques used in building the expert system. The method centred on the collection of data for the system.

A second example is "pacemaker " [10] the writers use the theory of model based diagnosis and the theory of abductive diagnosis using Horn formulae to represent causal knowledge of a particular disease. Their work on abductive diagnosis as mentioned in their literature was inspired by "Console and Torasso" The basis of this theory is based from abductive reasoning where observable findings are interpreted with respect to results obtained from diagnostic tests. Their theory of diagnosis stipulates that the abnormal behavior of a system is represented as causal knowledge, relating to abnormal states and subsequently resulting to abnormal findings.

The abnormal states are also denoted as defects, which could be anything from medical disorders or incorrect settings of the pacemaker. Using Horn formulae they able to represent casual knowledge, abnormal states and abnormal findings in a mathematical model. See below the representation.

$d_1 \wedge \cdots\cdots \wedge d_n \rightarrow f$

$d_1 \wedge \cdots\cdots \wedge d_n \rightarrow f$

The above means that d, di, i = 1,....n are positive literal representing defects, and f is a positive literal denoting the observable findings [10]. Console and Torasso contributed to the work on pacemaker reprogramming, they also proposed a simple mechanism designed to weaken the causality relation by means of literal α. The literal is a representation of the incompleteness of knowledge with respect to the underlying causal mechanisms relating causes and effects. The states of the defect *d* and the causality literal α whether true or false could be the determining factors to arriving at a diagnosis. This hypothesis results in a weakened horn formula in the form:

$d_1 \wedge \cdots\cdots \wedge d_n \wedge \alpha_f \rightarrow f$

$d_1 \wedge \cdots\cdots \wedge d_n \wedge \alpha_d \rightarrow f$

The literal denoted as alpha (α) represents the incomplete assumptions that contributes to the diagnosis process. In interpreting the above we will like to highlight the fact that the state of the literal be it true or false determines whether a particular disease's diagnosis will be of a high accuracy or not. It could be seen from the above analysis that the theory of abductive diagnosis is an effective concept and theory used in arriving at an effective diagnosis.

A third example is CASNET (causal association network developed in the seventies. It was used in representing knowledge about diagnosis, prognosis and treatment selection. It represented a generalized representation scheme and method of reasoning. It was applied in ophthalmology in the diagnosis of glaucoma. The causal model provided guiding answers for complex clinical cases. CASNET described a disease in causal terms, relate this description to an associated structure of observations such as symptoms and test results and then described various classifications imposed on the model, thus diagnosis and treatments. In CASNET observations are used to infer the intermediate causal states.

Example: If the applanation tension is 30 mmHg or higher and there is evidence of corneal endema an the patient complains of eye pain

Then the intra-ocular pressure must be highly elevated.
The success rate of CASNET has been recorded as follows:

95% acceptance rate for clinical proficiency and 77% rate of high competence (expert and very competent responses) [4].

3.9 Efficient pattern matching techniques in expert systems.

HEPAXPERT-I is an expert system that translates the results of routine serologic test for infection with hepatitis A or B virus [6].

Knowledge was elicited from serologic findings that contain patterns of serologic test results. 64(sixty-four) for hepatitis A and 4096 for hepatitis B. After the entry of an input pattern, a rule matching algorithm which is indexed was used to provide an interpretive text. A previous evaluation of the expert system was carried on 23368 hepatitis A and 24071 hepatitis B serology request was carried out [6].

Rete Algorithm

Rete algorithm is used as a technique in high performance rule-based systems. It is noted for handling large number of rules. The latest version of the rete algorithm is rete 2. It is faster than the original algorithm when dealing with large amounts of data [11]. Charles Forgy developed the Rete algorithm [12].

It is designed to improved the speed of forward-chained rule systems by limiting the effort needed to recalculate the conflict set after a rule is fired. The major disadvantage is that it demands high memory space requirements during execution. The advantages are derived from two empirical observations:

- Temporal redundancy: The firing of a rule usually changes only a few facts and only a few rules are affected by each of those changes.
- Structural Similarity: The same pattern often appears in the left-hand side of more than one rule [12].

Facts are variable-free tuples, patterns are tuples with some variables, and rules have as left-hand sides lists of patterns. The rete algorithm uses rooted acyclic graph called the rete, where the nodes with the exception of the rules denotes patterns, whiles paths from the root to the leaves represent left-hand sides of rules.

It keeps current information related to the nodes in the graph. Anytime a fact is added or deleted from the working memory, a token denoting the fact and operation is entered at the root of the graph and subsequently send to its leaves to update the information related to the nodes. Rete comprise the root node, one input pattern nodes and two input join nodes [12].

Below is an example of Rete:

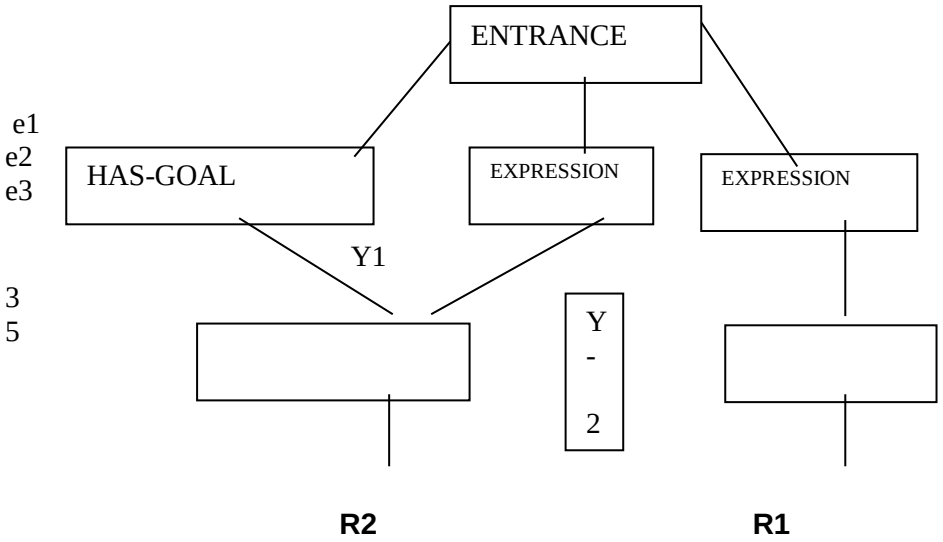
Rules:

(R1 (has –goal ?x simplify)
(expression ?x 0 + ?y)
➔……..)
(R1 (has –goal ?x simplify)
(expression ?x 0 * ?y)
➔……...)

Facts:

(R1 (has –goal e1 simplicity)
(expression e1 0 + 3)
(R1 (has –goal e2 simplicity)
(expression e2 0 + 5)
(R1 (has –goal e2 simplicity)
(expression e3 0 * 2)

Then the Rete is:



x= e3 y=2 x=e1,e2 y=3,5

[12]

Rete algorithm scales to many hundreds of rules and even for dozens of rules. Rete is several times faster than any known alternative algorithm [13].

3.10 – Example implementation of rules

Source of case study: Electronic engineering times Massachusetts; July 1998; Des Young; Marilyn Suey; Jay Canteenwallen; in an article 64-bit CPU ups reliability in WANs (Wide Area Networks).

A five stage pipeline is depicted in the article to demonstrate the simplification of rules that make it easy to flush out the rest of the pipeline and restart the central processing unit, when a sub-system experienced a failure.

Stages in pipeline execution

1. Instruction Fetch (IC)
2. Decode, Register Fetch, Jump/Branch
3. Execution
4. Data Cache Read
5. Write to register file and data cache

The objective of the 5 stages was to simplify and reduce down time, as a result of improving efficiency and productivity in the entire business process.

The established rules used in contributing to the detection of faults were as follows:

1. Fault Tolerant Application Manager
2 Notification
3. Isolation and service restoration for failures in system control

An important rule in implementing expert system is that, there should be (Adequate CPU cycles). CPU cycles less than a certain threshold will not be effective and efficient in resolving system crashes.

3.11 – Summary

Rule-based expert systems have some form of ability to provide expert guidance and response to users who require expert advice from information systems. The literature reviewed indicates that classical expert systems have some limitations. For instance they have difficulties handling problems that need multiple reasoning under uncertainty such as the diagnosis of multiple infections. The facts in the classical model can sometimes be fuzzy. It is difficult to establish what is true and false about the fact declared. The reasoning strategy also employed can not effectively handle problems that need the recognition of patterns. New developments of the traditional expert system model shows that new forms of reasoning strategies and knowledge representation formats are more capable in dealing with such shortfalls. Reasoning strategies such as efficient pattern matching and rete algorithm have been used in the construction of recent expert systems. This investigation indicates that expert systems have some ability in assisting the provision of expert advice for diagnosis and prognosis of diseases in medicine. This also implies that expert systems could assist with provision of health as a way of improving quality of life when there is a deterioration of health. The assertion made here is to imply that they are not an end by themselves, but rather a means to an end.

Chapter 4

Natural Language Processing

4.1 What is Language?

Language has been a medium of communication since the existence of man. We could conclude that language has existed as long as the existence of humankind. Language also defines ancestry.

A Language consists of a set of symbols with syntactic and semantic representation**.** These set of symbols forming the language can be represented in text form.

A Language is meant to be learnt by learner in a learning environment. A language has a lexical structure which is grammatically structured and meaningful.

4.2 What is natural language processing?

A natural language refers to any language spoken and understood by people. Examples of them are, Swahili, English, Japanesse, Udo, french and German.

A natural processing language comprises a learner, learning environment and a language to be learnt.

NLP is a subject which has its foundations from artificial intelligence and Linguistics. This combination of study is generally known as computational linguistics. Computational linguistics comprises theory and application.

4.2.1 Computational Linguistics in theory

This focuses on how humans understand and communicate a language. This relates to meaning of words and the construction of meaningful words.

4.2.2 Computational Linguistics application

This is programming of computers to communicate with people. Although computers communicate in a language the ultimate objective of computer scientist is to train computers to communicate with human in a manner understood by humans.

4.3 Things to know in NLP

Parse tree

A parse tree is a hierarchical tree like structure which shows the structure of a sentence, as well as the parts of speech which form the sentence. For example, how many verbs, adverbs, nouns, pronouns and adjectives are in the sentence?

Parser

A parser is a program that uses a parser tree to assess structure of a language. It uses grammar and lexical structure to find structure in language structure analysis. The lexical structure of the lexicon determines whether a word is a noun, verb or adjective.

4.4 Understanding Language

Things required to understand language are lexicon, grammar, semantics and vagueness.

Scope of NLP:

- *Text summarisation,*
- *Information retrieval,*
- *Semantics (meaning of words)*
- *Graphic to text*
- *Text to speech*
- *Ontology*

4.4.1 Text Summarisation

A text summariser is highlights or extracts text briefs from a text without creating redundancy of the original text.

Automatic text summarisation is the application of software to generate text briefs from a text. Below is a sample of a text summariser in MS-word. Steps 1 and 2 demonstrate how the summariser could be used.

Step 1 –

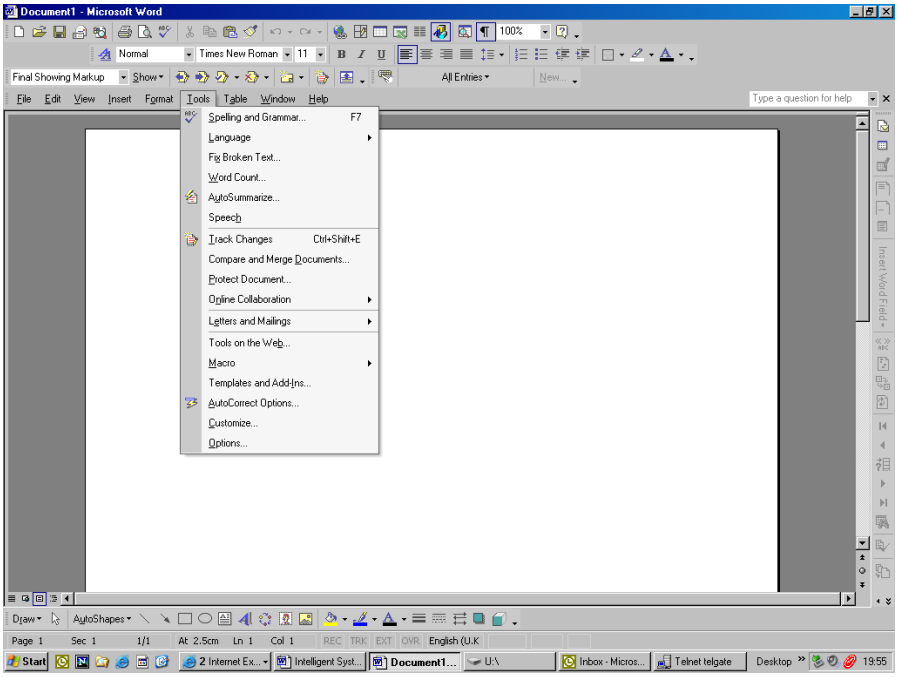


Figure 3 Text summariser in MS-word

Step 2

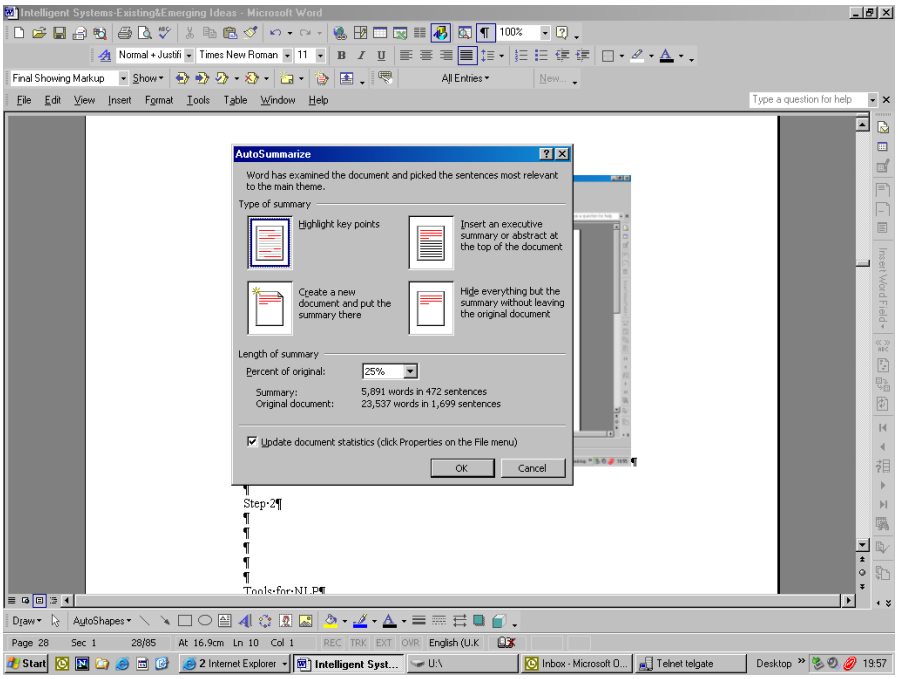


Figure 4- Text summarisation in MS-word

Other summarisation tools for NLP include:

Open NLP, Delph –IN (Integrated technology for deep language processing), Gate and MARF.

4.4.2 Text summarisation and personalisation

(Peppers and Rogers, 1993, 1997) define *personalisation* as a process that allows the relationship with customers on the Internet to migrate from anonymous mass marketing and sales to 'one to one' marketing.

Kobsa also defines personalisation as a form of user modelling Kobsa(2000). The paper does not explicitly state that personalisation is different from user modelling. The interpretation given to his definition is that, *personalisation* is a process that helps user modelling. It is not synonymous to the process of modelling the requirements of users. User modelling seems to encompass other generic user requirement that sometimes transcends the boundaries of the characteristics and attributes of the user.

4.4.3 Personalisation Tools and Information retrieval

This section examines tools that help *personalisation* of the Web in summarisation. The tools are Group Lens (Net perceptions, 2000), Personalisation Server (ATG, 2000). Front mind (Manna), Learn Sesame (Open Sesame, Bowne, 2000).

Personalisation tools are user behaviour oriented bias. They rarely capture the physical attributes of the user. This is a distinct feature of personalisation. Personalisation tools in most cases serve the interest of service providers and product vendors rather than for example persons who use their services or buy their products. The primary design goals of personalisation tools are to support the management information systems of service providers. The tools attempt to capture user features such as interest, knowledge, habits etc. that could help in explaining the behaviour of a particular user. It is also dependent on external factors that are not easy to control. These factors constitute uncontrolled variables when formulating solutions to problems. It also seems to focus on user *preferences* rather than *needs*.

4.4.4 Predictive statistical models for information retrieval

Predictive statistical models use observed samples as the rationale for creating user models. They could also anticipate behaviour through information gathering from like minded people. This notion was introduced by Zukerman and Albrecht (2000) as a *collaborative approach*. This section briefly highlights issues of predictive statistical models as introduced by Zukerman and Albrecht.

Predictive statistical models discussed here are linear models, TFDF (Term frequency inverse document frequency). A weighting scheme commonly used in information retrieval, Markov Models, Neural networks and Bayesian Networks.

User’s interest and behaviour could change over a period of time. Physical attributes and characteristics will not change in the short and mid term.

The models do not focus very much on the present. Most efforts seem to be devoted to addressing problems that occur as a result of changes, rather than focusing on the present needs of the user which is physical and could be modelled. User profiling technique focuses on the primary needs of the user.

Although techniques such as content based filtering is ideal for customising a system's functionality to specific requirements of a particular user, the approach requires each user to provide relatively large amount of data to enable the construction of a statistical model. This is an example of how user modelling over relies on user's input in real time.

The models also seem to be user dependent rather than self dependent. Self dependence means the capability to fulfil a given task on the basis of current information available to any entities or objects which possess functionality.

4.5 Machine learning and challenges

- **The problem of large data sets**

The problem of large data sets was examined in the testing of the Syskill and Webert system (Pazzani and Billsus 1997). Large data sets were used by the system to identify web sites based on the user's interest. A user's profile was built from the interaction between the user and the system. It was designed to aid users in differentiating interesting web pages on a particular topic from uninteresting ones. Browsing was conducted in a restricted manner. The user constructs an index page from a particular topic and then rates the hyperlinks of that page. It works by applying a learning algorithm that goes through a large number of web pages represented as data sets before arriving at a more accurate decision. Large data sets become a problem that usually hinders learning algorithms from achieving accurate timely results. This is because the algorithm has to spend ample time to go through a number of web pages. Although large data sets are problematic in relation to incorporating it in learning algorithms, they provide a significant range of data that serve as a good test case for significance testing in classification problems Clarke and Cook (1998). This allows the learning algorithm to understand the attributes and characteristics that form the pattern of the problem being studied.

The size of the data set being used by the algorithm might also not be of critical importance. This is due to the fact that the size of the data set could be derived from an understanding of the problem being solved. In other words the data sets sampled for the solution should be dependent on the type of problem being addressed. For example persons with particular functional visual abilities might already possess a certain type of profile

that could be used to determine the type of summaries that should be extracted from a web page.

- **Solutions to large data sets**

The specification of profiles for example persons with particular visual abilities for instance could eliminate problems associated with data sets. This is because correct specification of profiles enables the user to cope with the physical interaction when retrieving a web document. Learning algorithms used in providing classification solutions to web documents are usually also dependent on the continuous feedback from the system. Another reason why the problem of large data sets might not be critical is the fact that persons with weaker visual abilities are primarily interested in functionalities such as navigation or the readability of a web document. This task does not require large data set, because the ability to move around effectively without much hindrance, is an important aspect of web usability Dimitrova and Petrie(2002). Identifying a topic of interest is also less important and secondary to such persons when it comes to web document accessibility, during document retrieval. This suggests that the choice of data set should be dependent on the specific problem being solved.

The criterion for measuring accuracy in such classification problems is also based on closeness of a search result to topic of interest. This could be considered as an issue in the sense that going through a large data set has system performance implications. A trade off between the classification process and performance might also affect the level of accuracy of the classifier from identifying topics that are interesting to the user. Reducing the large data set might not necessarily provide a solution to the problem as well. This is because the problem at stake requires large data sets to define more accurately the requirements of the problem being solved. The contrast is that although small data sets could enhance efficiency, it does not deal with the number of variations and possibilities that might be available to model the attributes and interests of a web document reader. This leads us to consider the subject of concept drift, a problem that also affects natural language processing systems.

- **Concept drift**

Concept drift relates to the rationale on which user models are built or the objects for building user models. According to Wibmer and Kumat (1996)

existing methodologies have been based on specific needs without taking into consideration the dynamism of the user's requirements. The idea of concept drift relates to the changes in attributes that occur in a user. The way learning algorithms adapt to these changes that occur becomes the strategy for providing a solution to concept drift. Recent developments in user modelling have also been applied in information retrieval on the web. The main objective in this trend of development is to ensure that the user's information interest are achieved or realised. This could be achieved by content based information filtering. This filtering technique focuses on the areas within the web page that seem to describe relevant or interesting subjects of the topic highlighted by the page. The technique is applied by the automated acquisition of user profiles through a text classification task (Pazzani and Billsus, 1997; Lang,, 1995; Mooney and Roy, 1998). Although a system may try to identify areas of high interest to the user, there is also the possibility that the system could digress from the main topic to aspects of other topics that might not be of interest to the user. This is because interesting topics on the web are not confined to specific web pages. Although concept drift is a problem more related to the identification of interesting topics from non interesting topics in relation to the user's attributes, the same problem does not apply to persons who possess less visual abilities. Profiling the functional abilities of such users addresses the problem of concept drift. The functional abilities of visually impaired persons could be predefined in detail by modelling the physical attributes of such group of persons. For instance such strategies have been used in the retail industry where individual customer profiles are studied and modelled to reflect their buying habits. It could be argued here that buying behaviour of customers does not change rapidly in comparison to the interest of customers. This is because the buying behaviour of customers comprises characteristics such as taste, working habits, style of dressing etc. The interest of a person could change rapidly due to circumstance, environment and nurture. Although not everyone might agree with this claim, the fact is that non physical attributes of a person can not be modelled easily due to its dynamic and fluid nature.

Another challenging aspect of the application of user modelling in real life application in retrieving document is the need for labelled data.

- **labelled data**

This simply means that the user must make a conscious effort to provide the system with a feedback. This will enable the system to build an

understanding of the user and consequently provide the user with the appropriate information desired. The reality is that users are not keen to provide feedback to the systems they use. They find that irritating and boring on most occasions. The ultimate purpose of these systems is to improve them to the level of a natural language. This is one reason why user profiles could be useful in consulting with natural language processing systems. This is because in natural language, there is a high level of intelligence in the communication process. Systems should be able to make intuitive judgements in order to achieve a natural language process during interaction. This could be achieved by modelling the user in a generic manner Kobsa (2001). Generic models could make intuitive judgements by adapting to new requirements based on responses provided by the user, the frequency of response types by the user, and the probability that a user will give a certain type of response. Improving the method of capturing labelled data could enhance user profiles. This is because the labelled data could provide specific directions and pointers to either functional requirements or interest of a user. One way of improving solutions to problems associated with labelled data is by providing shell systems. Kobsa (1990) seem to be the first author to mention user modelling shell systems. This was adopted from the field of expert systems. The system could use interactions between itself and the user to draw inferences that could be used to feedback the system. This feedback could improve the system's knowledgebase that could subsequently increase the intelligence of the system. Next section examines issues related to computational complexity and how solutions to this problem could be improved.

- **Computational complexity**

Computational complexity relates to the computational requirements in achieving practical results in adaptive systems deployed on the Internet and other information processing systems. The negotiation of system requirements of users and computational effectiveness and efficiency of these adaptive systems should be in place to achieve very accurate result. In other words there should a way of balancing resources available with computational requirements of the user. In some occasions there should be a trade off between computational effectiveness and user requirements. This could be achieved through intelligent negotiations between the user and the system. The negotiation should be based on what the user needs in order to make a document readable or accessible and what responses the system is capable of providing in order that the system's functionality does not come to a halt. These are aspects of computational complexity that should be addressed.

The other problem that could be associated with computational complexity relates to the nature of information accessed by the user. Questions relating to the form the information takes should be examined and addressed. For example text responses might need a different negotiation principle as

compared to video or sound responses. The performance and productivity demands in relation to responses from the system on text data might not be the same as that of sound or video data. These are questions, which for instance were not considered by other models including that of (Pazzani and Billsus 1997). Computational complexity, labelled data and concept drift are issues that when not addressed satisfactorily could affect user models for natural language processing systems.

Natural Language Processing and user models

The goal of natural language systems is to enable computer systems to communicate with humans as if humans were being communicated to by other human beings and not by computers (Zukerman and Litman, 2001). Zukerman and Litman stress the importance of natural language systems consulting user models. Taking a user's characteristics into consideration during user modelling is an important activity. The characteristics could range from expertise, abilities and disposition. All these help to determine what a user's profile might look like. For instance, Paris (1989) realises how knowing the expertise of a user could help to determine the level of explanation that should be included within a content page. Paris (1989) uses two types of descriptions known as *basic concepts* for a novice user and *specific artefacts* for an expert user of an information system.

Concepts for novice user and specific artefacts for an expert user

The ideas behind basic concepts according to Paris (1989) was based on the notion that novice users only required information that should be pitched to their level of expertise. The contrast is that expert users require information that is more detail and specific to their level of experience and know how. Paris's model of the user did not take into account the levels and dimensions of expertise that might exist amongst users. The issue of different levels of expertise a user could have in relation to a particular concept was addressed in more detail by Tattersall (1992). The study was driven by three main strategies. These are; Initial strategy and unknown concepts; consolidating strategy and reminding strategy.

Initial strategy and unknown concepts

Tattersall (1992)'s initial strategy has to deal with unknown concepts that users came across. These users according to Tattersall (1992) needed more explanation of the information gathered during the communication process. This provided users with clear directions to relevant information that help them understood the concepts they interacted with. Making a judgement on

what volume of information to be provided to users is not an easy task. This implies passing judgement on information relevance, a characteristic of good and quality information. Guiding the user to direct navigation through the document could be the answer to defining how much and what type of information might be relevant to the user. Guiding users direct the navigation process will allow the user to make the right judgement in relation to relevance. This will place the user in a leading role during the navigation process. This means that the user should be the questioner and facilitator in the navigation process. Most systems designed allow the system to drive the navigation and questioning process. The system in my opinion should only clarify questions from the user. In other words it should be user driven rather than system driven.

Consolidating strategy and previously introduced concepts

This strategy was a way of helping user's that could relate both previous and current concepts in order to build an understanding of the concepts they came across while interacting with a natural language processing system. Concepts in section 7.2.1 have a bearing and relevance to consolidating strategy. This is because users could only apply consolidating strategy when they are experience or experts.

Reminding strategy and known concepts

This was mainly suitable for user's that possessed high level of expertise. Users had to be reminded of the concepts. That was adequate for the expert’s information needs on a particular concept to be met. Both strategies describe aspects of reminding strategy regarding the expertise or experience of users. These strategies are interwoven in a manner that suggests that, determining the level of previous knowledge and experiences of a user is a critical factor to designing such systems.

Issues in user modelling and natural language processing

This section discusses some of the issues that affect user models and natural language processing systems. There are also discussions on the characteristics of user models and their inference patterns. The discussions on the characteristic of user models take into account issues related to

parsing, discourse, content planning and the goals and plans of users in natural language processing systems.

Parsing techniques

Parsing techniques find it difficult to handle incomplete sentences. User's input for instance is not represented properly internally. Natural language processing systems and user models depend on each other for information. The information obtained is used to improve their effectiveness. Examples of such an application could be seen in Caremini et al., 1994; Binsted et al., 1995; Hirst et al., 1997; de Rosis et al., 1999.

Caremini et al (1994) discuss the application of a user model for individual patients. The system adopts a user model that collects information about the patient's symptoms, habits, family history and history of medications taken. The system according to Caremini supported the discourse to the patient's particular circumstance.

Binstel et al (1995)'s system on the contrary mixed general medical information with specific information about the patient. Personalisation was done by health professionals.

Natural language processing systems and user requirements

The views regarding these strategies in this paper are that, although they seem to indicate the importance of expert levels, the model some how fails to recognise that expertise is relative, subjective and a very dynamic process. An expert in one environment might be judged as an intermediate expert and not advanced in a particular concept. This is not because they have no advanced skills set in that area. It is rather because their advanced skills set may not be highly relevant or appreciated in certain environments. This drives the need to have systems that have the ability to make some level of intuitive judgement. This is because intuitive judgements are based on experience, rather than basic or raw facts. Making systems demonstrate the level of experience possessed in a natural form will be a means of creating systems with intuitive judgement. This could be done through negotiations that take place between the user and the system as mentioned in previous sections.

(Zukerman and Litman, 2001) highlight content planning as the main purpose why natural language processing systems derive information from user models. The assessment of this assertion in this paper is that, although user models provide some form of information regarding the attributes of users in content planning, they appear to be very generic. In other words

they do not meet the specific requirements and the needs of individual users of an information system. For instance programming a computer to be able to interact or communicate with humans in a natural way by understanding their specific intentions could be an extremely difficult task. The reason for this assertion is that techniques in natural language processing have not modelled users in a manner that integrates profiles for persons who use these systems. A typical example is systems built for supporting health services commented by Carenini et al 1994; Binsted et al, Hirst et al 1997., de Rosis et al 1999 .

Carenini et al (1994); Binsted et al., Hirst et al., (1997); de Rosis et al (1999) stipulate that providing more accurate information to users of health services for example patients, reduces the cost of service due to less time spent on interaction between users and hospital staff. In most cases the information needed by specific patients are not included in the content planning process. The way to improve this, I believe is to determine and include information relevant to patients during content planning. Sending information to a patient that has poor relevance to their information needs might increase the rate at which the patient requests the same information. Methods of improving the solution to this problem are to focus on the user's attributes in a more detail fashion.

Zukerman and Litman, (2001) also explain how a user's attributes provide a single dimensional user model while relevance and appropriateness provide multi-dimensional models. This relates to context-based information designed to meet the users' needs, thus making information relevant to the user. A better understanding of the user's beliefs could capture what is relevant and appropriate to the user. This could lead the system to topics that might be of significant importance and interest to the user. This analysis was reaffirmed by Carenini et al (1994); Binsted et al., Hirst et al., (1997); de Rosis et al (1999). Their studies on predictive statistical models for user modelling indicate how relevant information to users could save time in the patient and staff communication process by reducing the cost in communication.

4.6 Summary

Chapter 4 reviewed natural language processing by explaining language as essential component of NLP. The chapter defined a natural language as language spoken and understood by people. It also mentioned that NLP has its foundations from artificial intelligence and linguistics. The scope of

NLP covered text summarisation, information retrieval, semantics and Ontology. There were discussions on machine learning and related challenges. Key challenges were problems with large data sets, concept drift, labelled data and computational complexity.

Chapter 5

Optimisation and Search Techniques

5.1 Introduction

This chapter is an overview of optimisation and search techniques. The chapter focuses on ant colony optimisation, breath and depth search, swarm particle and social adaptation intelligence and genetic algorithm as a stochastic iterative search model.

5.1.1 Definitions of Optimisation

1. Optimisation can be defined as the process or strategy adopted in maximising the throughput or productiveness of a resource in an application or problem environment.

2. It simply means doing better in a situation where productivity and efficiency is a major outcome.

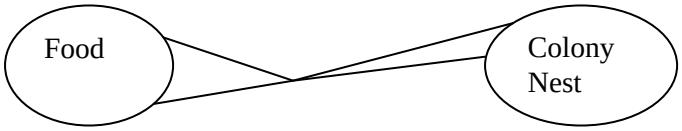
3. It could also mean that the environment might be performing under tight constraints.

It is also a means of choosing the best solution out of a set of available solutions to a problem. For example there are different ways of designing a road traffic system. There are also different methods for organising the production system of a car manufacturing company. There are also methods that a sales person can choose to reach the destinations of potential clients and customers at shortest possible time. There are also the best search strategies that can be adopted to search for an item on the internet. While some solutions are quite straight forward, others are more challenging and difficult. For instance time tabling is a problem which has not been solved successfully in many working environments. This is because one has to choose from a given set of resources, thus lecturers/tutors, rooms, courses and many more. Availability of lecturers and tutors on particular days, times as well as sizes of rooms and facilities could be a bottleneck. Neither lecturers nor students can be at the same places at the same time.

5.2 Ant Colony Optimisation (ACO)

Ant optimisation algorithm is a multi agent system that models the behaviour of each single agent known as “ant”. It is an example of swarm intelligence. The algorithm have been applied in problem areas such as the TSP (Travelling Salesman Problem), routing on communication networks such as the internet. An example of ACO include Meta heuristic model.

Meta – heuristic – This is a colony of argentine ants given a trace route leading to food in an area linked to a colony's nest. The nest and food bridge is made up of two branches of different lengths.



The ants move from the nest to the food location and back. The branches are linked in a way that ants travelling in any of the directions along the routes choose any of the routes when the get to the junction. After a certain period an "ant" decides to use the shortest branch along the routes.
The probability of an ant selecting a shortest route increases with the difference in length between the routes. This is based on the differential path length observed by the ant. The argentine ants while travelling from the nest to the food location leave a chemical behind known as "pheromone" on the path. Anytime an ant arrives at a junction of a branch a probabilistic decision is made based on the amount of pheromone left on the path and can be smelled. This process has an effect known as autocatalytic.

The main goal of the ant is to be able to find a shortest route along paths between food and nest. Supposing we represent the entire network route as a graph of vertices, where the vertices are paths, then we could formulate this mathematically as;

N = (V, A) is a connected graph with v = |V| vertices. The algorithm can be used to find a solution to the shortest path problem defined on the network, where the solution is a path on the network forming the entire route.

The path is linked with each vector of the network. This path is known as the pheromone trail. Each pheromone trail is read and stored. This is propositional to the utility of the vectors that form the path. This forms the basis for ants to initiate better solutions when problems are encountered enroute to and fro food and colony nest. The decision of an ant (x) located on a vector (v) uses the trail of chemicals on the path to calculate the

probability which it should choose to vector position along the routes. This suggests that a search is influenced by the amount of pheromone on a path.

5.2.1 Characteristics of ant colony optimisation

1. Although each ant is complex enough to locate a possible solution to a problem being considered, an effective solution can only emerge as a result collective interaction among a team of ants

2. An ant makes use of only information local to itself and the destination it is visiting to collect food.

3. Ants communicate with other ants through the mediation of information on chemicals deposited on the paths.

4. There is weakness in adaptability of the ants.

5. Ants search for the most optimised cost involved in providing a solution

6. Ants have memories that can store information on paths travelled. This memory can be used to retrace the path

7. An ant can be in different states as they travel to vectors on the network of the paths and routes

8. A routing network can be derived from the paths built from the pheromone trail.

9. Different neighbourhood can be created from the chemical trails on the network

10. An ant can update the pheromone trails identified on the paths

5.3 Particle Swarm and Social adaptation intelligence

Definition – It is a phenomenon of social behaviour among living organisms. It models a population of individuals by assessing the

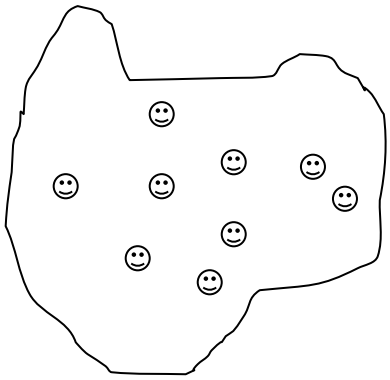
collaborative efforts required in solving a problem. It can be used for showing symbiotic, parasitic and dependency relations among the particles that constitute the neighbourhood of the particles.

The technique is usually classified under evolutionary algorithms. It attempts to search for the best neighbour and characteristics in the neighbourhood which is worthy of mimicking and imitation in providing an optimised solution. It can be used for optimising different intelligent applications, such as artificial neural networks.

5.3.1 Characteristics of particle swarm

Particle swarm shares features with evolutionary algorithms. These features include; adaptation, altering and population based phenomena. Whiles evolutionary algorithms do not keep past information in memory, particle swarm uses keep past information through learning in identifying best candidate or feature in a community.

Conceptual Overview – Particles in a neighbourhood



The particles in the neighbourhood can be drawn to different communities within the conceptual region for a number of reasons. This can be the case as a result conditions such as conduciveness, interest, common culture and language, food and entertainment. There can also be conditions related to harshness such as persecution, starvation and deprivation.

The next section tackles the subject of genetic algorithm as a popular search algorithm for addressing constraint satisfaction problem. Both harsh and conducive conditions can optimise a particle's behaviour and performance. Both conditions can also lead to negative optima. This is common to social communities and organisations.

5.3.2 Learning paradigm – Whiles most intelligent systems improve learning through reinforcement and back propagation, particle swarm learn by imitating the behaviour or particles in the neighbourhood. It is irrational and epiphenomenal.

5.3.3 Systems parameters

5.3.3.1 Parameters necessary in building swarm particle intelligent systems

The main parameters necessary for building swarm particle intelligent system are mainly; problem dimension, individuals, neighbourhood size, weight, velocity, upper limit and constraints.

Dimension – Dimension relates to the multi-facet nature of the problem that has to be tackled in order to solve the problem. The dimension determines the simplicity and complexity of the problem.

Individuals – This defines the members in a neighbourhood. It can range from 2 to 50. This can increase depending on the problem. It must be noted such an increase can have a negative effect on the iteration process and performance of the neighbourhood.

Neighbourhood – This is the individuals that form the population of the neighbourhood. Neighbourhood can be classified into small and large.
It can either be an entire population or just a section of the neighbourhood. A common problem to a neighbourhood is the problem of local optima. This happens when particles draw themselves to a single particular particle that identifies a solution to a problem.

Upper limit and Constraint – This change in vector position and direction of a particle due to a specified threshold for particle movement is known as upper limit. Each element in a vector set is assigned a weight with a positive random number. Adjusting the weight of a network can determine the extent which a particle moves towards or from optima.

Maximum velocity – Maximum velocity is a parameter for putting a cap on the speed of individual particles during iteration of the population. The value assigned to velocity can be based on the dimension of the problem. It should be such that a particle can switch from local to remote optima.

The next section presents genetic algorithm and programming as a common technique employed for optimisation and search base problems.

5.4 What is genetic algorithm?

A genetic algorithm is a randomised search method based on the biological model of evolution through mating and mutation. This randomised search method breeds effective solutions to problems [6]. These problem solutions are encoded into bit strings that are tested for fitness; the best strings are combined to form new solutions using methods similar to the Darwinian process of survival of the fittest and exchange of DNA which occurs during mating in biological systems.

The History of genetic algorithm is usually traced to John Holland. In his publication [7] Holland describes the ability of simple representations (bit strings) to encode complicated structures and simple transformations which have enough power to improve such structures. He also showed that with the proper control structure, rapid improvements of bit strings could occur under certain transformations, so that a population of bit strings could be made to evolve as a population of animals would [8]. One important result was that even in large and complicated search spaces, genetic algorithms would tend to converge on solutions that were globally optimal or nearly so.

5.5 Case studies in Genetic Algorithm

Genetic algorithms have been applied to problems as diverse as graph partitioning and the automatic creation of programs to match mathematical functions. In engineering and biomedical domains, genetic algorithms have been used for various optimisation problems. Industrial settings like telephone networks, planning construction lines, trucking companies, circuit boards and connecting components can all make use of genetic algorithms to assist in finding the optimal solution. Whereby, an optimal solution is the most efficient thereby reducing costs for the industries.

5.5.1 Genetic algorithm (GA) and biological immune system

The application described below is Genetic Algorithm based modelling of a biological process, the *immune system*. A major advantage of the immune system simulation is in its seamless integration into a GA based search for an optimal design. In the biological immune system, foreign cells and molecules are termed antigens while antibodies are molecules for eliminating the antigens.

Type-specific antibodies recognise antigens. This task of recognition is not easy due to the very large number of possible antigens. This recognition problem in biological systems translates into a complex geometry matching

process. The antibody can have a geometry that is specific to a particular antigen. Antibodies that have specific geometry are termed specialists while those that can capture a broader group of antigens by partial matching are termed generalists [12].

The GA based immune system modelling represents antigen and antibody molecules as binary string structures. Even though this modelling is not the biological model where genes have a four-letter DNA alphabet, it still fits into a GA based design optimisation strategy. We believe this is because a measure of the degree of match between binary bit string representations of the antigen and antibody can be defined. This representation is also good because levels of complexity can still be introduced into the matching function. E-g the number of matches on a bit-by-bit basis can be the basis of the fitness function and/or the length of contiguously matching sub-strings.

Now to get the optimal result (of antibody population) for antigens, the principles involved in building genetic algorithm were used.

- All antibodies had their fitness initialised to zero
- The fitness function of the antibody was based on the antibody similarity to the antigens (or complimentarily)
- An antigen was selected at random
- A sample of antibodies of size μ was selected from the antibody pool without replacement
- The score of each antibody was computed by comparing against the selected antigen. The antibody with the highest score had its score added to its fitness value. But the fitness of all other antibodies remains unchanged.
- The antibodies were then returned to the antibody population with the process repeated a number of times.
- Based on the fitness computed, a GA simulation was carried out with crossover and mutation to evolve the antibody population through one generation of evolution.
- The process was then repeated from selection of antigen till convergence in the antibody population was got.

Now in determining whether the optimal solution will be made up of specialist antibodies or generalist antibodies depends on the sample size μ (which is the control parameter). This is because the smaller the sample size, the greater the probability that number of specialist antibodies will increase and ultimately take over the population with substantial increments in the sample size.

Our view is that the GA simulation of this biological process gives a good representation of genetic algorithm principles because convergence that is

normally avoided in GA by enlarging and diversifying the initial population or sample size can be used to generate specialist antibodies.

This will serve a good purpose in biological system if there's the need for specific antibodies for an antigen invasion. It is reasonable to say that this will be of immense benefit to the biomedical industry for counter attacking antigenic molecules.

In a situation where the problem keeps changing, that is, different antigens keep attacking; the belief is that evolution of the generalist antibodies will be ideal since this generalist antibody population will maximally cover all the antigens.

5.6 What is genetic programming?

There are basically two main reasons why human beings have been captivated by the idea of building intelligent machines; first is the ability to mimic animal-like intelligent behaviour and secondly to provide functional solutions for difficult or complex tasks. The traditional computer approach has served us in diverse ways by providing solutions to problems with defined problem domains. However there are some areas such as speech recognition, vision, etc where the traditional computer approach to solving problems cannot be applied. Advanced techniques from artificial intelligence which can cope with limited knowledge of problem domains to develop solutions are required. Genetic programming evolved in the light of the above challenge, that is, how can computers learn to solve problems without being explicitly programmed? Genetic programming addresses this challenge by providing a method that automatically creates a working computer program from a high-level problem statement of the problem. This is achieved by:

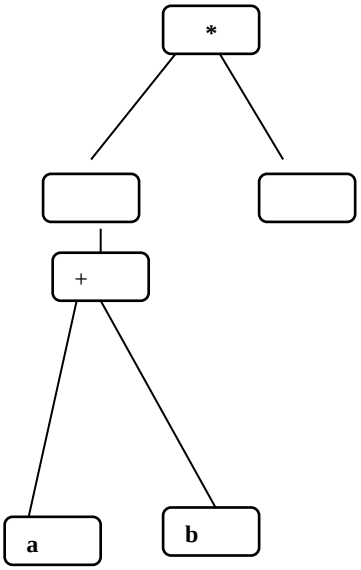
- Randomly creating computer programs leading to a progressive evolution of a population of programs over a series of generations.

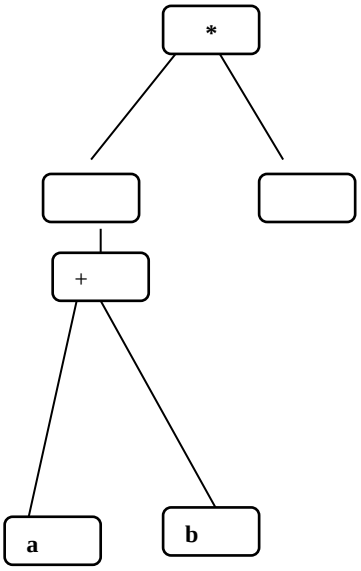
- The evolutionary search making use of the Darwinian principle of natural selection/survival of the fittest and patterning it after biologically inspired operations.

.

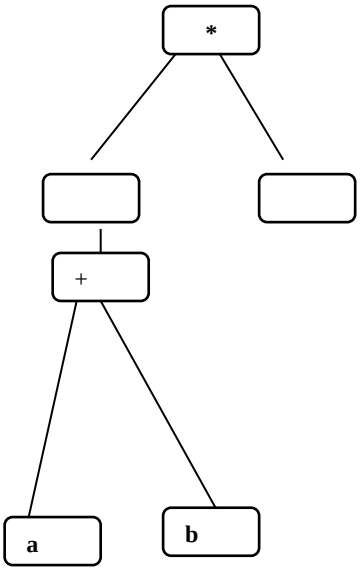
These operations include reproduction, crossover (sexual recombination), mutation and architecture altering operations patterned after gene duplication and gene deletion in nature [9].

1. Reproducing - This is a replication or copying of individual programs to form a new generation’s population.

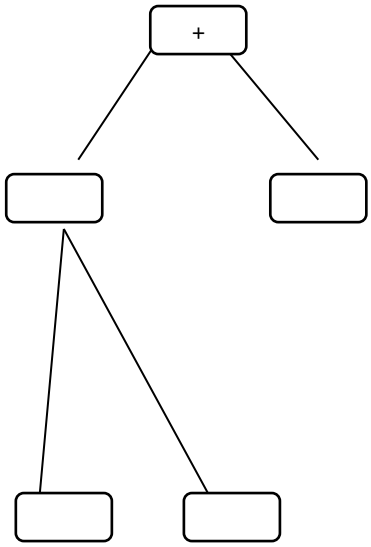




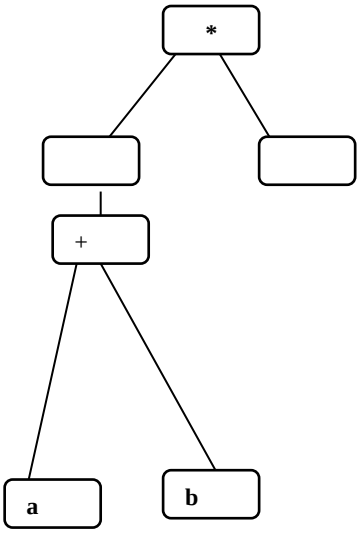
2. Crossing over – This is a binary operator which generates children after a joining operation. It is the effect of two parents to reproduce. Random nodes are selected from the parents to form new set of children. It creates new offspring of programs to form a new generation's population.



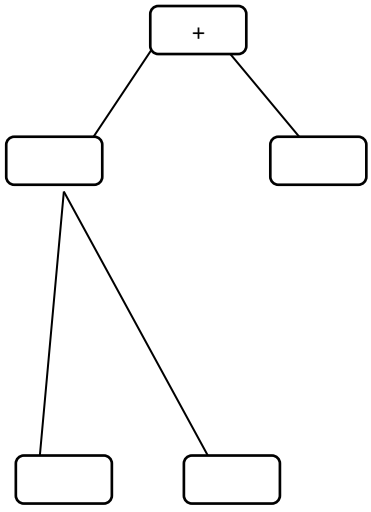
Parent 1



Parent 2



Child 1



Child 2

3. Mutating – This is the process of creating a single offspring for a new population by randomly multiplying part of an existing set of programs in a population to transform it to a new program or set of programs .

4. Architecture altering – This is the process of choosing an architecture altering operator in order to change certain aspects of a program in the population.

5.7 General steps in creating a genetic program

Step 1- Create an initial population as set of programs also known as creatures manually or automatically

Step 2 - Stop automatic creation of program population

Step 3 - Apply fitness function to individual program members of the population.

Step 4 – Choose surviving individual programs of population using a probability.

Step 5 – Apply genetic operators such as reproduction, cross over, mutation and architecture altering.

Step 6 – Repeat steps 3 to 5

5.7.1 Fittest function

Fittest function is derived from the criteria specified for fitness. For example in the natural world of sports, sportsmen and women have to pass a fitness test in other to be selected for a tournament. In this same regard a program is considered fit if it meets a certain criteria designed to pass fitness and be selected. Such a criterion for program fitness could include, loosely coupling and highly cohesive of the individual modules, procedures that form the program. It can also include memory consumption of the program during execution, cabbage collection features of the program for performance optimisation and fault tolerance levels of the program. Fitness therefore can be represented using different program inputs. Searching for the fittest program is mainly based on probability of the fittest function in the population of a particular generation. Programs can either be selected or passed over after this process.

5.8 Public knowledge of genetic programming

The field of genetic programming became publicised in the early 90's with the publication of Koza's book on genetic programming [9]. Genetic programming has its foundation in the established field of genetic algorithm. An earlier paper actually [11] refers to genetic programming as hierarchical genetic algorithms because the principles that underlie them are very similar. John R. Koza [11] effectively described the essence of genetic programming by saying the best computer program that appeared in any generation, given the best solution so far, is by virtue of genetic programming.

Genetic programming works best for several types of problems. One of the benefits of genetic programming is in applying it to problems where there is no one (ideal) solution, for example, there is no one solution to driving a car because driving a car consists of making compromises of speed versus safety as well as many other variables. In such a case genetic programming will find a solution that attempts to compromise and being the most efficient solution from a large list of variables.

5.9 Optimising GA to GP

Genetic programming also works best in finding solutions where the variables are constantly changing. For example in a car routing problem the program will find one solution for a smooth concrete highway and find a totally different solution for a rough unpaved road. Nature has provided the evidence of evolved systems that possess high levels of complexity. Genetic programming makes use of this process in a computer evolving programs rather than creatures. In theory, this means computer evolution could similarly evolve very complex computer programs. However, in practice, certain limitations affect finding optimal solutions to various tasks. One of such limitations is time constraint. The maintenance of the population and the simulation of the evolutionary process are usually a very small proportion of the total run time. It is in the evaluation of fitness of the individuals in the population where most of the execution time is spent.

This execution time can be very long when a complex simulation is required and this will of course be carried out for every individual in the population. Whilst the ever increasing performance of modern CPU's will make more problems amenable to genetic programming in the immediate term, parallelisation of the technique offers the most effective means by which genetic programming can be made to produce more timely results[10].

The above review of genetic programming concepts have triggered the importance to further look into the usefulness, effectiveness and efficiency

of coding genetic programs in Java within the spectrum of other programming languages.

5.10 Case studies in genetic programming

5.10.1 Why JAVA is suitable for genetic programming?

Java is a pure object oriented programming language that originated as part of a research work to develop advanced software for a wide variety of network devices and for writing programs for small computers embedded in consumer electronics appliances like microwave ovens and television sets. The letters 'J', 'A', 'V', 'A' does not mean anything, the name was chosen purely on the value of its connotation. The creator of the language, James Gosling, firstly named the language **oak [13].** This name was used from 1991 to 1994 but it did not survive the trademark search and so dropped in favour of the name **Java**. Even though James did not start with the intention to create a new language, he was looking for a way to run programs unchanged on a variety of computers. Thus, he set out to create a compiler and environment that would enable C++ programs to run on incompatible platforms without having to be recompiled. As the research progressed he found out that he would have to modify some C++ conventions. This eventually led to the birth of Java.

The Java programming language is designed to meet the challenges of application development especially in heterogeneous, network-wide distributed environments. [14]
Among these challenges, most importantly, secure delivery of applications that consume the minimum of system resources, applications that can run on any hardware and software platforms and applications that can be extended dynamically. Since it is quite obvious that Java attempts achieve optimisation of resources we will be confident to think that it will be a very suitable language for genetic programming. This is because genetic programming as a programming technique is used to optimised solutions for problems that need resources to be optimised.

Java presents a new viewpoint in the evolution of programming languages which is the creation of a small and simple language that is still comprehensible enough to address a wide variety of software application development. A major design goal of Java is to make it familiar to a majority of programmers where a large fraction of these programmers are familiar with C and C++. This will enable them to learn Java within a short period of time. Although the Java programming language follows C++ to some degree, Java gained its simplicity from the removal of features (like pointers) from its predecessors.

5.10.2 Genetic Programming and JAVA

This section focuses on the desirable features of Java in relation to the characteristics of a genetic program. It is an analysis of the desirable features in relation to the characteristics that make up a genetic program.

Java has certain desirable features such as simplicity, object orientation, architectural neutrality, security, robustness, portability, dynamic, threaded, high performance and interpreted. One of the primary characteristics of the Java programming language is a simple language that can be programmed without extensive programmer training. The fundamental concepts of Java are grasped quickly and programmers can be productive quickly. The Java programming language is designed to be object oriented from the beginning. The object orientation features of Java that could be very useful to a genetic program are encapsulation, inheritance and polymorphism. The creation of a creature in genetics could be equated to the creation of an object class in Java. In other words we could stipulate that a creature in a genetic program is an object class. The attributes of an object class could be passed onto other object classes via the application of inheritance. We can from this basic analogy perceive that Java's objects could be mapped onto a genotype (creature). Object orientation has been seen as a more appropriate technique for simplifying complex problem. It has the ability to function in increasingly complex, network-based environments. The object feature of Java can also permit reliability in a genetic program in the sense that objects could evolve into new objects. The ability of objects to evolve via polymorphism can not be over ruled nor under valued.

Robustness and security are other important features. The Java programming language is designed for creating highly reliable software. This is achieved by an extensive compile-time checking followed by a second level of runtime checking. Also, Java operates a simple memory management model that eliminates entire classes of programming errors that is common with C and C++. This memory management model operates by creating new objects with a new operator. This gives programmers confidence because they know the system will find many errors on time. Security is of paramount importance since Java operates in a distributed environment. The memory management features of Java such as the garbage collector could be very useful in a genetic program. This is because unfit creatures in a genetic program could be discarded or made redundant to enable the program to make a more efficient use of the resources available on the system.

The run-time system and the language have security features designed into them so that this makes applications written in Java secured from invasion from outside. Also, in the network environment, applications developed in the Java programming language are secured from unauthorised code that tries to create viruses or invade the file systems. The encapsulation features of the language also contribute to security and robustness. Security

and robustness can not be underestimated in terms of it's importance and significance to genetic programs. It is quite obvious and transparent that the applications reviewed in the previous paragraphs highlight the importance of security.

Java technology has been designed to support applications that will be used in different network environments. In such environments there exist different hardware architectures, operating systems and language interfaces on which theses applications will execute. To make this possible, the Java compiler generates *byte code*s (An architectural neutral intermediate format designed to transport code to multiple hardware and software platforms). Java technology is strict in its definition of the basic language. Java specifies the sizes of its basic data types and the behaviour of its arithmetic operators. Programs are the same on every platform, there are no data type incompatibilities across hardware and software architectures. Programs translated to byte code seem to be written in the instruction set of a typical computer. But, byte code is neutral because it does not use the instruction set of any particular computer, instead, byte code is executed by a program that pretends it is a computer, based on the byte code instruction set. This program is known as a *byte code interpreter*.[15] A byte code interpreter that intends to execute the byte code produced by the Java compiler is called a *virtual machine*. This enables any Java application to run on any machine that a Java virtual machine has been implemented.

Java is portable because its byte-code format is architecturally neutral and as such the Java byte-code can be interpreted in any environment provided the Java Virtual Machine is installed. Java also explicitly specifies the size of each of its primitive data type as well as its arithmetic behaviour.
This makes sure there are no implementation dependent aspects of the language specification [16]

Genetic programs should be designed to meet a high level of dynamism. This is the case because genetic programs are designed to solve performance related problems. Multi-processing systems can make effective use of this characteristic. A genetic program has some level of dynamism embedded that the versatility of Java as a language could enhance in any application. Interactively and spontaneously transporting a program to another part of any system seem to be a peripheral function of a genetic program. This is normally made possible by extending an evolutionary call from one part of a system to another. This is certainly a useful feature. Java is a dynamic language and any Java class can be loaded into a running Java interpreter at any time. The dynamically loaded classes can then be dynamically instantiated. Information can be obtained dynamically from a class at runtime. Java's dynamic nature works well with its distributed nature. Both features make it possible for Java interpreter to download and run code from across the Internet. A thread is an individual direction of execution. It enables long pauses between what a user does and in a way control the functionality of the program. In Java, the most

common use of a thread is allowing your applet to go off and do something while the browser continues to do its job. [17] In a GUI based network application like the Web browser, it's easy to imagine multiple things going on at the same time. A user could be scrolling a page, listening to an audio-clip while the browser is downloading a file containing text or image.

Java provides support for multiple threads of execution (also known as lightweight processes) that can handle different tasks. Working with threads in C or C++ is quite difficult but Java makes programming with threads easier by providing built-in language support for threads. The java.lang package provides a thread class that provides methods to start and stop threads and set thread priorities. Java's language syntax also supports threads directly with the *synchronised* keyword. This keyword makes it very easy to mark sections of code or a whole method that should only be run by a single thread at a time. An important benefit of multithreading is that it improves the interactive performance of graphical applications for the user [17]. Using threads in a genetic program could amount to enormous results. This is true in the sense that threads are used in Java to obtain very high performance. That fact that you could split a program into several aspects of functionality is an advantage to any genetic program. We will to a reasonable extent agree to this proposition since it is apparent that a genotype or creature of any genetic program is designed to optimise the use of available resources. Performance is the benchmark of any genetic program. Java as a technology and a programming environment to a high extent considering it’s threaded attribute will enhance the efficiency of any genetic program. High performance is a major quality of the Java platform. It achieves superior performance by using a scheme in which the interpreter can run at full speed without needing to check the run-time environment. The automatic garbage collector also runs as a low priority background thread, this leaves a high probability of availability of memory when required and thus leads to better performance. Java is an interpreted language, so it'll never be as fast as a compiled language like [17]. Compiled C code runs almost ten times as fast as interpreted Java byte-codes, but the Java speed is adequate enough to run interactive, GUI and network-based applications. This is because the application is often idle, waiting for the user to do something or waiting for data from the network. In the spectrum of available programming languages, Java falls in between. At one end of the spectrum are high level, fully interpreted scripting languages such as TCL and the UNIX shells. Even though these languages are great for prototyping and also highly portable, they are very slow. At the other end of the spectrum are low level compiled languages like C and C++, they have very high performance but they are not portable. Java lies in the middle of this spectrum and the performance of Java's interpreted byte-codes is much better than the high level scripting languages. Java still offers the simplicity and portability of these high level languages.

5.10.3 Genetic Algorithm and the fractal landscape System

The construction of the fractal landscape was done using Genetic Algorithm and implemented in Java. [18] The landscape is formed placing big pyramids at randomly selected places and also adding smaller pyramids in an increasing number. The heights of the overlapping pyramids are added to form the surface of the landscape. If the first pyramid with a height of 100 is put on the plane, a point P at the lower part of its incline can be raised to a point 30. Later on, if a smaller pyramid with height 15 happens to be near point P, such that point P sits in the middle of its incline, point P can then be raised to a height of 37.5. This procedure of adding pyramids continues and it stops when the pyramids reach a given lower limit in their height. The heights of all points define the shape of the landscape. The distinguishing feature of the fractal landscape is that it looks the same from any distance. The Java applet visualises the height of a point by its colour, the higher the height the brighter the colour, thus it goes from black to blue, green and then white. To produce landscapes, it starts by putting two 128 height pyramids on an area of 400 by 300 points, for each consecutive step, the pyramids height is reduced by one half and its number is multiplied by four. When it makes the second step, 8 pyramids with height 64 will come up. It continues this until the pyramids have a height of 1 in the last step.

5.10.4 The problem and constraint

If the landscape looks more or less like a natural fissured landscape, finding the absolute maximum is not going to be simple because there's no efficient straight- forward algorithm. For example, if one starts at random point and go up, it'll lead to a locally maximum summit (height) and higher summits could be around. The only way to find the absolute maximum is to measure all points and keep the last found maximum (co-ordinates and height) in memory. But with many points even super computers may need years of calculation to evaluate all the points. Since the height of the landscape is its **fitness** it is therefore often sufficient to find a point higher than the given height. There could be many equal high maximums distributed over the landscape (as seen in this case) and it is sufficient to get one of them. Genetic algorithm is well suited for this scope. Our opinion is that the GA analogy for this model and evolution depicts a good representation of the natural genetic evolution. This is because the co-ordinate of each bug (I-e pyramid on the 400 by 300 point) corresponds to its genetic code. This consists of two genes that range from 0-299 and 0-399. The genetic code of the offspring is also a random combination of those of its parents and the evolution of this model was based on the recombination of a randomly generated initial population. This recombination also ensures sufficient diversity in the gene combination because it is based on ranking in a list of ascending order and not on fitness. But, when it comes to selection the fitness value of the bugs is the determining factor. The use of Genetic algorithm in the problem situation is

justified this is because genetic algorithms are good for problems where constraints have to be satisfied and optimisation of different solutions is needed. The constraint being heights of bugs are scattered all over the plane, thus, there is no one point where the absolute maximum is. Due to this constraint the GA mechanism does not use the fitness value to process its recombination. After the recombination, offspring are born between parents with a high fitness, these areas around these fit bugs are now selected. The algorithm stops when all the bugs are at the same point, that is, when they have the same genetic code. In order to come up with an optimal solution (the best solution under the given conditions) should be considered in relation to the starting size of the initial population. If too small an initial population is used the bugs will converge prematurely before having found the incline of the highest summit. The genetic algorithm toolkit used was developed in Java.

Genetic programming and electrical hardware evolution

Another interesting area of genetic programming application is in the evolution of electrical hardware. Genetic programming has been used to evolve analogue electronic circuits by Koza et al [10]. The technique used component creating functions and connection modifying functions to create the final circuit from operating on a simple embryonic initial circuit. These evolved circuits are evaluated on a patched version of the freely available SPICE (Simulated Program with Integrated Circuit Emphasis) simulator. It is a very computationally expensive application and it uses parallel computing. The most common applications involve training against a simulation where a single fitness case is presented or against some collected data where many fitness cases are presented to the evolved program.

5.11 Summary

It is quite evident from the fractal landscape problem reviewed in the previous section and the analysis of the embedded characteristics of Java as a whole make the language receptive to genetic programming. However the studies showed a hypothetical analysis of the extent to which it has been applied in genetic programming applications. Most of the applications reviewed were developed in C programming language. Although C might seem superior to Java in terms of performance, it will be difficult and an unfair proposition to rule out the potential and immense benefits that were revealed from the findings. The strongest view is that, modern application requirements seem to attract a programming solution that embeds Java's rich characteristics , such as object orientation ,encapsulation, polymorphism and inheritance. The investigations revealed

that a creature in genetic program could be represented as an object. The creature will have all characteristic encapsulated. It was also found that polymorphism could be used in Java to evolve new creatures or genotypes that could inherit attributes relevant to the survival and the effectiveness of new generation creatures. The encapsulated features of the language also enable security and memory management in the application of genetic programs. Notwithstanding these strengths it will also be premature to assert Java's suitability in genetic programming.

5.12 Sample exercise

1. (a) Explain the concepts underlying "ant colony optimisation meta-heuristics**".**

Model answer

A type of multi-agent systems in which the behaviour of each single agent called artificial ant is inspired by the behaviour of real ants. It is an effective form of swarm intelligence. It has been applied in problem areas such as the travelling salesman problem. and routing on telecom networks. 3 marks awarded for definition and example.

A laboratory colony of argentine ants is given access to a food source in an arena linked to the colony's nest by a bridge with two branches of different length. The ants move from the nest to the food source and back. Branches are arranged such that, ants going in either direction from the nest to the food back choose between one branch or the other. The idea is that after a transitory phase the ants choose the shortest branch. The emergence of the shortest path selection is influenced by a behaviour known as autocatalysis. The main aim of an ant is to find a shortest path between a pair of nodes on a graph on which the problem representation is suitably mapped. 7 marks awarded for description and clarification of concept.

(b) Briefly explain any five characteristics of individual ants of a colony**.**

Model answer

An ant searches for a minimum cost feasible solution

1. An ant has a memory that could be used to store information on the path it followed. Memory could be used to build feasible solution, evaluate the solution found and retrace the path backward
2. An ant in state can move to any node in its feasible neighbourhood.
3. An ant can be assigned a state and one or more termination conditions. Usually the state is expressed as a unit length sequence that is a single component.
4. An ant located on a node can move to a node chosen. The move is selected by applying a probalistic decision rule.
5. When moving from node to neighbour node the ant can update the pheromone trail on the arc. This is called on-line step by step pheromone update.

Chapter 6

Artificial Neural Network

6.1 Introduction

This chapter introduces the reader to basic concepts in artificial neural networks and applications of concepts and theory.

Definitions

It is a model based on the biological human brain system capable of making sense out of highly complex data set, which when analysed using traditional methods might not be meaningful (Williams G, 2005).

A neural network is collection of neurons representing the tiny cells of the human brain (Williams, G, 2004). A single neural network of the biological human brain could consist of billions of neurons interconnected.

A neural network is a collection of neurons that resembles the tiny cells of the human brain. A network can consist of a few to billion neurons connected in an array of different methods. ANNs attempt to model these biological structures both in architecture and operation. (Matthews J, 2000).

A neural network is an information processing paradigm that represents the way biological nervous systems such as the brain process information. It shows a large number of interconnected processing elements called (neurons) working in unison to solve a specific problem (DARPA, 1998)

The first artificial neural net was implemented over 50 years ago. Most subsequent research has been concerned with learning techniques and algorithms. One major milestone in the development of neural net technology has been the creation of the so-called error back propagation algorithm ten years ago. This learning approach proved to be very powerful and most neural net applications today use it. First, it computes the output values of the neural net for the input values of the current training example. Then, it compares these output values with the desired output value given by the training example. The difference, called error, is now used to determine how a neuron in the net shall be modified. The mathematical map of the error back into the net is called error back propagation. (Altrock, Constantine1994).

According to the *DARPA Neural Network Study* (1988, AFCEA International Press, p. 60):

> ... A neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

(Haykin, S. 1994) defines it as a massively parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: The network through a learning process acquires knowledge via the strengths of its interneuron connection known as synaptic weights are used to store the knowledge.

(Nigrin. A, 1993) also defines a neural network as a circuit composed of a very large number of simple processing elements that are neural. Each element operates only on local information. Furthermore each element operates asynchronously; thus there is no overall system clock.

(Zurada, J.M. 1992), *Introduces Neural Systems as* physical cellular systems that can acquire, store, and utilize experiential knowledge. The above definitions do not differ so much, they all have the same idea behind the concept and with the variety, and readers can choose the one that is easier to understand.

6.2 Architecture and Configuration of ANN

The architecture of neural network is the topology of physical structure that depicts interconnected nodes and links forming a network. The node of a network can be connected to itself or other networks. Nodes are processing points which represent objects in a conceptual context. Node can be classified as inputs, output and hidden. Input node serve as receiving points from external data sources, while output nodes discharge data out of the system. Hidden nodes serve as black box to the network architecture. The nodes have no link with external data sources or environment.

6.2.1 Weights

Weights are connection strengths of the network. The weight enable the network to adjust the logic underpinning the topology during training. I usually represented in the value range [1.0]. A positive value means excited connection, while a negative value is an inhibited connection. In such cases there is no connection. Let us represent the connection strength from j-th unit to i-th unit at the k-th excitement in an environment by $W^k ij$ or activation function of the k-th environment of i-th unit by $f(i^k)$. The common function here is the summation of input signals that are output of units where the links are represented as:

Summation i-th and k-th = Summation (j) $W^k ij$

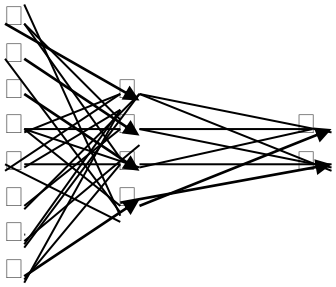
This denotes a signal by environment k of the i-th unit.

6.2.2 Perceptron

Frank Rosenblatt introduced the term to represent a single neuron associated with weighted inputs and its preprocessing characteristics. Although the perceptron was capable of recognizing and learning some form of data it also lacked the ability to learn non linearly separable data. Further research conducted by Minsky and Papert in 1969 revealed the weaknesses of the perceptron Eg. The exclusive nor (XOR) problem. Their contributions backed by mathematical concepts described the limitations of a single layer perceptron by not able to do some basic pattern recognition operations like determining the parity of a shape or determining whether a shape is connected or not. What they did not realized was that given the right training multi-level perceptrons could do those things mentioned.

6.2.3 Feed-Forward Network

This architecture and training method allows signals to travel in one direction only, thus from input to output. There are no loops. It means that the output of any layer does not affect the same layer which the output came from. It is an association of input to output. It is mostly used in pattern recognition problems. They are usually organized in top down or bottom up. Below is an example of a feed forward network.[1]



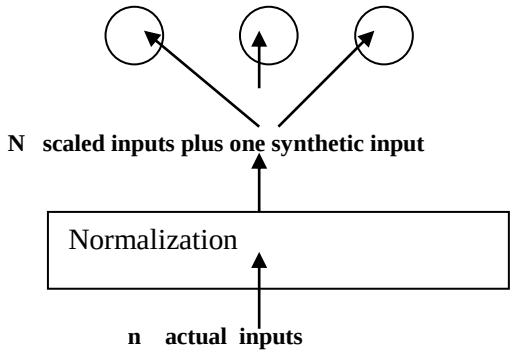
Inputs Hidden layer Outputs - Figure 4

Automatic association of inputs to outputs is a typical feature of a feed forward network. This means that the output of the network in many cases is hypothetical. Feed forward networks could either be single or multiple layered. This training method applied by this network in most cases is of a supervised nature. Feed forward networks are sometimes described as hierarchical networks. The data flow of the network is a bottom or top down approach. With respect to bottom up inputs are at the bottom layer whiles a top down has inputs at top layer. Hidden layers fit the middle part of the architecture. Perceptron and MLP are samples of feed forward networks. Feed forward networks are multi-layered networks comprising linear units, threshold units or non-linear units. There are instances where it can use a non supervised approach for training.

[1] www-dse.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report

6.2.4 Non Supervised Training - Kohonen Network

This is a feed forward network architecture that uses competitive learning or non supervise training mechanism. The neurons compete for the privilege of learning. It is basically a two layer architecture, although some schools of thought present it as a three layer network due to the normalization of the inputs of the network. Kohonen network is very famous for recognizing patterns which do not already exist and needs to be detected. Alternatively it is a network which its output neuron values can not be predetermined. It is highly suitable for classification problems that evolve from patterns that could mostly be predicted.



Kohonen network [2] **figure 3**

There are "n" number of inputs, including another input produced by the normalization of the inputs. This means that the network has n + 1 neurons. It has m outputs each one called a Kohonen neuron. The output of a Kohonen output is defined as the sum of the weighted inputs. There is no activation function applied neither any bias term. This is mathematically represented as:

$$\mathbf{Out} = \sum_{I=0}^{n-1} x_i w_i + s w_n$$

[2] Timothy Masters, Practical Neural Network Recipes in C++, Academic Press

Kohonen network is mostly used as a classifier. Computing the weights via training using the above equation, the network receives an unknown case. The output activation neuron found to have the highest activation becomes the winner. It uses a winner takes all algorithms. This work applies Kohonen SOM, a form of unsupervised network for examining translation of images in digital forensic investigation for counter-terrorism.

6.3 Self Organizing Map (SOM)

The Self-Organizing Map (Kohonen 1995) is a neural network model for analyzing and visualizing high dimension data. It relies on competitive learning; the output neurons of the network compete among themselves to be activated. The learning process in SOM is *unsupervised*, meaning that no teacher is needed to define the desired output for an input. In a self-organizing map, the neurons are placed at the nodes of a lattice that is usually one or two dimensional. The Self-Organizing Map defines a neighbourhood relation between synaptic vectors. This neighbourhood relation can be a rectangular or hexagonal lattice of map units.

The initialization of the map is implemented by assigning small values picked from a random number generator and no prior order is imposed. Once the map is initialized, there are three essential processes involved in the formation of the Self-Organizing Map:

Competitive process: For each input vector, the winner (Best Matching Unit) is searched, which minimizes the Euclidean distance measured between input data sample x and the synaptic vector w_j.

$$i(x) = \arg\min_j \| x - w_j \|, \; j = 1,2,...,l \qquad (1)$$

Cooperative process: The winning neuron locates the centre of a topological neighbourhood of cooperating neuron. The topological neighbourhood assumes a time-varying form of its own, as shown by $h_{j,i(x)}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right)$. The parameter σ is the "effective width" of the topological neighbourhood described by $\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right)$, where σ_0 is the value of σ at the initiation of the SOM algorithm, and τ_1 is the *time constant*. The lateral distance $d_{j,i}$ between winning neuron i and excited neuron j is defined as $d_{j,i}^2 = \| r_j - r_i \|^2$. The unique feature of the SOM algorithm is that the size of the topological neighbourhood shrinks with time. The neighbourhood function applied above is the Gaussian function.

Adaptive Process: For the network to be self-organizing, the synaptic weight vector w_j of neuron *j* in the network is required to change in relation to the input vector *x*. All the neuron in the lattice that lie inside the topological neighbourhood of winning neuron *i* will be adjusted by

$$w_j(n+1) = w_j(n) + \eta(n)h_{j,i(x)}(n)(x - w_j(x)) \qquad (2)$$

Equation (2) has the effect of moving the synaptic weight vector w_i of winning neuron *i* toward the input vector *x*. The learning-rate parameter $\eta(n)$ should be time varying.

Map Training

Map training contains two phases. The ordering phase takes 1000 iterations or possibly more, the learning parameter $\eta(n)$ should begin with a value close to 0.1 and decrease gradually. The neighbourhood function $h_{j,i}(n)$ should initially include almost all neurons in the network centred on the winning neuron *i*, and then shrink slowly with time.

The convergence phase is to fine tune the map and provide an accurate statistical quantification of the input space. The number of iterations constituting the convergence phase must be at least 500 times the number of neurons in the network, and may have to go on for possibly tens of thousands of iterations. The learning parameter $\eta(n)$ should be maintained during the convergence phase at a small value, on the order of 0.01. The neighbourhood function $h_{j,i(x)}$ should contain only the nearest neighbours of a winning neuron.

After trained with the two phases, the map is ready to be used in the real time service.

6.4 Feedback Neural Networks

These are networks that have loops alongside the main nodes that form the network connections. Figure 5, 6 and 7 are examples of feedback networks.

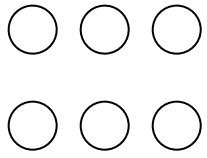


Figure 5 – Bi-layer network

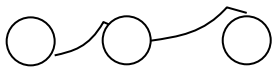


Figure 6 – Single layer network

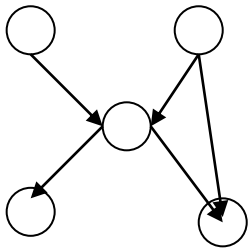


Figure 7 – Complete Feedback

6.5 Case study (Digital forensic investigation & ANN)

Imaging techniques have been applied to a number of applications, such as translation and classification problems in medicine and defence. This paper examines the application of imaging techniques in digital forensics investigation using neural networks. A review of applications of digital image processing is presented, whiles a Pedagogical analysis of computer forensics is also highlighted. A data set describing selected images in different forms are used in the simulation and experimentation.

6.4.1 Puzzle of digital evidence and computer forensics

Evidence Collection requires some form of methodology in order to establish consistency among different types of investigations Rude (2000). Rude suggests (5) steps for such a methodology; Preparation, Snapshot, Transport, Preparation and Examination. The steps assert that, the evidence collected should comprise both external and internal parts of the computer system.

Rodney Mckemmish (1999) defines forensic computing as the "process of identifying, preserving, analysing and presenting digital evidence in a manner that is legally acceptable". According to the author it comprises four (4) key elements. These are the identification of digital evidence, the preservation of digital evidence, the analysis of digital evidence and the presentation of such evidence in a court of law.

The process of building evidence is accompanied by a number of challenges. There are questions related to the location of evidence? Traceability and how the evidence is collected? It is logical to state that human experts have the responsibility of tracing and collecting such evidence.

The preservation of evidence could also be tampered and damaged not only by human error, but also virus and worm attacks and mal-ware. In such a situation one will need a robust technique capable of recovering incomplete data and information. This is to make recovery effective such that original images of hard disks and file structures are meaningful. This problem has caused the need to carryout research using non traditional methods such as artificial neural networks for evidence collection. This case analyses the application of SOM in image evidence collection and forensic analysis.

6.5 Methodology

Objective of experimentation –

To assess the effectiveness of neural network for the translation of incomplete or hidden images for forensics analysis.

Tool used for implementation – The tool used for the implementation is neural connection 2.1 update (SPSS).

- **Design Requirements and Topology Selected** –
    - The Kohonen Self Organising Map was selected as the architecture for implementation.
    - Number of total **epochs** is 10 since the data selected or the investigation is not high.
    - Number of **neighbourhood set** to 2.
    - **Error response** - Euclidean Dist.
    - **Weight distribution** is from the data
- **Data Collection and Preparation**

Data set - Refer to appendix 1 and Tables 1 and 2.

Type	Weight		Height		Size	Resolution
	Pixel	Inches	Pixel	Inches	Bytes	
True colour (24 bit)	798	2.66	1098	3.66	2628612	300
256 colour (8 bit)	798	2.66	1098	3.66	876204	300
Monochrome (1 bit)	798	2.66	1098	3.66	108702	300
Gray Scale (8 bit)	798	2.66	1098	3.66	876204	300
True colour (24 bit)	798	3.99	1098	5.49	2628612	200
256 colour (8 bit)	798	3.99	1098	5.49	876204	200
Monochrome (1 bit)	798	3.99	1098	5.49	108702	200
Gray Scale (8 bit)	798	3.99	1098	5.49	876204	200
True colour (24 bit)	798	7.98	1098	10.98	2628612	100
256 colour (8	798	7.98	1098	10.98	876204	100

bit)						
Monochrome (1 bit)	798	7.98	1098	10.98	108702	100
Gray Scale (8 bit)	798	7.98	1098	10.98	876204	100

Table 1

Input 1	Input 2	OR	NOR	XOR	AND	NAND	XNOR
0	0	0	1	0	0	1	1
0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0
1	1	1	0	0	1	0	1
0	0	0	1	0	0	1	1
0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0
1	1	1	0	0	1	0	1
0	0	0	1	0	0	1	1
0	1	1	0	1	0	1	0
1	0	1	0	1	0	1	0
1	1	1	0	0	1	0	1

Table 2

Table 1 is a numerical representation of the images highlighted in appendix 1. Appendix 1 has the following image and pixel resolutions; True colour 24 bit, 256 colours 8bit, Gray scale 8bit, Monochrome 1bit, a smudge image and an embossed image. The pixel resolution values were not captured from Microsoft photo shop/editor for the smudge and embossed images. The binary data for these two images were hypothetically derived. The two images formed the advanced features of the images that had to be tested.

Table 2 is a binary derivative of the pixels values in table 1. These binary values in conjunction with the pixel values were used as data sets for training the SOM Kohonen network.

6.6 Summary of results

This application explored the use of a SOM in assessing image translation and recovery using the images outlined in appendix 1. The program showed that ANN could be effective in reconstructing and making sense out of incomplete data. Appendix 1 formed the test data for deriving training sets for the neural networks application. Tables 1 and 2 were data sets describing images presented in different forms input to the Kohonen SOM-Self Organising Map model for simulation and experimentation. The experimentation showed that ANN had the potential of addressing image translation and recovery problems which infrared and ultra-violet imaging techniques did not in recovering digital evidence on digital media as a counter measure to terrorism and cyber crime. The future work of this project is to test the application with a wide range of image data set and assess the performance implications.

6.7 Sample exercises

2(a) Compare and contrast any 2 characteristics and features of an expert system and a neural network.

(b) Explain the following algorithms

(i) Steepest descent
(ii) Conjugate gradient

Model answer

Steepest descent

It is the learning gradient of a backpropagation learning /Training algorithm. It is the change of the adjusted inputs in relation to the output of the network. This results in the change of the network error. The errors are then propagated back to the network for further adjustment. Minimising the steepest descent, thus the gradient will result in a corresponding error generalization capabilities.

Conjugate gradient

This is a much more robust learning algorithm for a multivariate function. This is due to the fact that it displays superiority as a result of the way it attempts to minimize the error function. Multiple minimum points along the gradient are created in order to determine the most minimal point along a parabolic curve. A guess search is applied in determining the most minimal point along the slope. A significant advantage is that, the time used in the training and learning of the network could be significantly short. The main disadvantage is short learning periods could serve as trade off for effective learning and a well fitted model due to an off target minimal point**.**

(c) Outline the key differences between supervised and unsupervised neural networks.

Model answer

Unsupervised networks have the capability to make sense and meaning out of mal-formed, incomplete or information which is not immediately known. Eg. is the Kohonen self organising map. Valuable in exploratory work.

A network that could be trained to reproduce its input at its output. This has many applications such as guided image enhancement and filtering noise from signals. Also very useful for problems that need prediction.

3. (a) Explain how the following activation functions applies input data in training neural network. Sigmoid, Radial Basis Function (RBF) and Gaussian function.

Model answer:

The activation function transfers input signals/stimuli to output signals by moderating the relationship between input and output. Let us denote the total input, activation and output of the i-th unit by Si total, Ai and Oi respectively. I can be denoted as:

i receives either inhibitory or excitatory stimuli from other units. The total stimuli to unit i Si total is a function of the output of the other units having links improving upon the unit i. This should be the basis for explaining sigmoid, RBF and Gaussian function**.**

(b) Compare and contrast features and functions of fuzzy and probabilistic neural networks**.**

Model answer:

Fuzzy neural networks are characterised by their architecture, activation functions and connections strengths, and learning paradigm. Based on fuzzy sets concepts, Probabilistic Neural Networks are useful in probabilistic neural networks. When the flow of operations are assigned with primitives to individual neurons in each layer the algorithm could be transform to look like a four layered network.

Appendix 1
Pixel: Resolution:

Smudge Image, Embossed image

Source: http://ndevilla.free.fr/lena/

6.8 Sample Neural Network Task

Table 1 is a modified market share of Google as at from 24th December to 24th march 2006. Using data in columns 2 and 4 construct a Multiple Layer Feed forward Network using back propagation algorithm for predicting Google's shares on the market for the month of April, May June and July.

Last Trade:	**365.80**	Day's Range:	362.51 - 370.09
Trade Time:	Mar 24	52wk Range:	177.64 - 475.11
Change:	**23.91 (6.99%)**	Volume:	15,182,162
Prev Close:	341.89	Avg Vol (3m):	13,859,900
Open:	368.52	Market Cap:	108.73B
Bid:	365.40 x 100	P/E (ttm):	72.85
Ask:	365.80 x 200	EPS (ttm):	5.02
1y Target Est:	481.21	Div & Yield:	N/A (N/A)

After Hours (RT-ECN): **361.81** **0.01 (0.00%)**

STOCK DATA

Last Trade:	**360.79**	Day's Range:	352.51 - 370.09
Trade Time:	February 24	52wk Range:	167.64 - 475.11
Change:	**21.91 (6.99%)**	Volume:	13,182,162
Prev Close:	311.89	Avg Vol (3m):	12,859,900
Open:	358.52	Market Cap:	106.73B
Bid:	325.40 x 100	P/E (ttm):	62.85
Ask:	325.70 x 200	EPS (ttm):	4.02
1y Target Est:	381.21	Div & Yield:	N/A (N/A)

After Hours (RT-ECN): **362..81 ↑0.01 (0.00%)**

STOCK DATA

Last Trade:	**361.50**		Day's Range:	362.51 - 370.09
Trade Time:	January 24		52wk Range:	176.64 - 475.11
Change:	↑21.31 (6.99%)		Volume:	14,182,162
Prev Close:	331.89		Avg Vol (3m):	13,859,900
Open:	358.52		Market Cap:	109.73B
Bid:	355.30 x 100		P/E (ttm):	71.85
Ask:	365.50 x 200		EPS (ttm):	4.02
1y Target Est:	441.21		Div & Yield:	N/A (N/A)

Last Trade:	**375.80**		Day's Range:	332.51 - 350.09
Trade Time:	December 24		52wk Range:	187.64 - 375.11
Change:	↑20.91 (6.99%)		Volume:	14,182,162
Prev Close:	331.89		Avg Vol (3m):	13,819,700
Open:	358.52		Market Cap:	107.63B
Bid:	365.40 x 100		P/E (ttm):	70.85
Ask:	361.80 x 200		EPS (ttm):	4.02
1y Target Est:	431.11		Div & Yield:	N/A (N/A)

6.9 Summary

Chapter 6 explained the concepts underlying artificial neural networks by highlighting common types of networks. The common types of artificial neural networks covered supervised and unsupervised networks. There was a presentation of case studies extracted from a paper presented at a conference organised by international society for optical engineering that applied a self organising map unsupervised network for recovering incomplete images for forensic purposes.

Chapter 7

Software Agents

7.1 What is a software agent?

A software agent is a suite of computer programs that has social, adaptive, mobilization, autonomy and reasoning capabilities. A software agent running on a computer should be able to sense the system's environment and perform intelligent task through inherent functions and procedures. The adaptive nature of a software agent is an attribute derived from existing artificial intelligence techniques such as artificial neural networks. Autonomy is a desirable feature of an agent this means that it should not be subjected to the control of another agent, being it software or human. Supposing an agent exclusively relies on inherent knowledge without being able to learn from its environment then, it is logical to conclude that it lacks autonomy. Whether an agent lacks autonomy or not depends on how much manipulation it can be subjected to.

7.2 History of Agents

Software agents partly originated from multi-agent systems (MAS). MAS are concepts that underpin the way distributed systems and computing work. Software agents also borrow certain concepts from social sciences, computational linguistics and traditional neural networks along side of adaptive intelligence. The general concepts derived from distributed computing include modularity, parallelism, reliability, portability, security, remote code execution, concurrency and synchronization.

According to (Nwana 1996) general concepts could be traced back to early days of research in Distributed Intelligence in 1970s, citing the work of (Hewitt,1977). Hewitt proposed the concept depicting a contained, interactive and concurrently-executing object which he called "ëactorí". This object applied the concept of inter process communication, enabling message passing to and fro different objects.

(Wooldridge, 1995) Wooldridge & Jennings (1995a), Wooldridge & Jennings (1995b) and Wooldridge *et al.* (1996) presents symbolic reasoning as the mechanism which influence the way decisions are made in real world. Other traditional work include systems such as the actor model (Hewitt, 1977), MACE (Gasser *et al*., 1987), DVMT (Lesser & Corkill, 1981), MICE (Durfee & ontgomery, 1989), MCS (Doran *et al*., 1990), the contract network coordination approach (Smith, 1980; Davis & Smith, 1983), MAS/DAI planning and game theories (Rosenschein, 1985; Zlotkin & Rosenschein, 1989; Rosenschein & Zlotkin, 1994). Gasser (1991)

emphasises the society of agents over individual agents, while micro issues relate specifically to the latter. In any case, such issues are well summarised in Chaib-draa et al (1992), Bond & Gasser (1988) and Gasser & Huhns (1989). Last 10 to 15 years work has evolved, covering areas such as TAEMS (Decker & Lesser, 1993; Decker, 1995) DRESUN (Carver et al., 1991; Carver & Lesser, 1995), VDT (Levitt et al., 1994), ARCHON (Wittig, 1992; Jennings et al., 1995)" These are sources extracted from (Nwana 1996).

7.3 Activities of agents

A sensible activity of a software agent is dependent on factors such as performance, perception of environment, knowledge, responses to requests made by users of the software.

7.4 Structure of software agent

The structure of software agent comprises architecture and a procedural logic coded as a program. The architecture is the design on which the program is developed and implemented. The architecture usually comprises percepts, actions, goals and environment. The percept is mapped onto the actions that need to be performed in order to achieve goals in the environment which it is implemented. Software agents show perception, transformation and associated actions in the application environment which is meant to execute.

7.5 Common benefits of software agents

- Can adapt to new patterns

The learning characteristic of agents could enable them to adapt to new communication patterns on the networks by learning new activities.

- Independent and Autonomous nature make them neutral.
  No loyal affiliations

Autonomy help to alleviate bias and loyal affiliations that could influence the performance of the agent with regards to the integrity of the information collected.

- Serve as black box analogous to the black box installed in airplanes

Continuous activity monitoring on the networks by agents could be useful in a post e-crime scenario Williams, G (2004).

7.6 Trust issues with software agents

- They could serve as double agents. An example is the Babington plot of 1586
- Information (Evidence) collected could be diluted as a result of bogus messages through vulnerable communication channels and lines.

- Agents could be made to compromise their stand (integrity).
- An agent might not be able to preserve information gathered
- Ad-hoc network behaviour could elude the agent
- They might not conform to existing legal framework and law
- Rogue access points could be used in spoofing bogus information
- Insecure delegation and lack of accountability by agents to the law. Agencies might not be accountable to the law
- No established and cooperation amongst existing agencies. Poor collaboration and cooperation amongst heterogeneous platforms.

Chapter 8

Games

8.1 Game Theory

Game theory forms the basis of game modelling. It is one of the approaches to formal modelling strategies to problem solving. The basic principles underlying games are;

8.1.1 Specification of requirements using formal methods

Creating a data repository

Identifying players of games

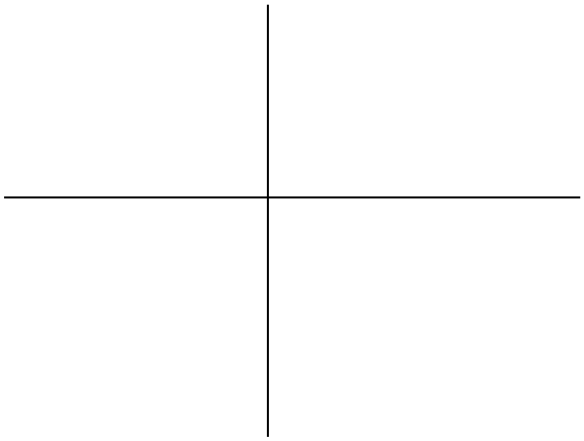
Deriving strategies of games leading to cause and effect

Selecting players or key figures that influence the game

Identifying possible moves in the game

8.2 General Concepts and Terms:

Equilibrium – This is to ensure that all players have equal chances and opportunities prior to the start of the game. It is also a method of bringing a game to a balance. Equilibrium shows the principle of “Pareto” which means there is no such a thing like a fee lunch. Every free lunch is paid for. There is a loose and gain principle in equilibrium.
The equilibrium can be represented on the Cartesian plane, where the vertical line is the Y axis and the horizontal is the X axis.



General equilibrium – Infinite agents communicating among themselves, however acting independently

Nash equilibrium – This is when a player adopts the best strategic move to another strategy. Moves that are simultaneous have Nash equilibrium.

Belief – The probability that a player will choose a certain strategy.

Dominant Strategy - A strategy is said to be dominant when a strong strategy confronts a weak strategy.

Mixed Strategy – A strategy is considered mixed when a player makes moves in a random manner from possible set of strategies.

8.3 Principles of games

1. A game can take a normal and extensive form

2. A game has a start and termination

3. A game is defined using information sets

4. A game has a backup strategy and set of moves

8.4 Formal Specification of games:

Goal – This is the research problem which the game is required to solve

Story – This is an architecture or design of the game

Synopsis – An overview of the game

Control – This is the manipulation and handling of the game set by boundaries outlined in the game

Rules – Sets the context and framework of the game

Translation and Interpretation of rules – Checking for consistency in representations made by players

Solution – Normality and balance of moves and strategies introduced in the game
Examples:

Goal state, Goal (Motivation and Objective)

Story (Forming soccer or table tennis team)

Setting: 5 aside or 2 aside each member in the team has the skill to make team to win. Each of them is thinks rationally and has interests in the game strategy. This can sometimes affect the optimisation of the team's performance.

Goal formalisation - Identification of states (a,b,c)

(a,b,c) defines skills and capabilities

If b helps a , the probability that a scores a goal is $(a+b)/(a+b+c) = P_{ba}$

If b helps c, probability that c wins is $(b+c)/(a+b+c) = P_{bc}$

Probability that “c” helps “a” and “a” does not help “b”.

B values the point scored by a is U_{ba}

Cost of b helping a is K_{ba}

Cost of neutrality is 0

B moves then either a and c scores

8.5 Alliances in games

A move by b in the game affects a in positively or negatively. This can be represented in probabilistic form as:

1. $P_{ba}\ (U_{ba}) + (1- P_{ba})\,(U_{bc}) - K_{ba}$

A move made as b’s alliance with “c” can be represented as:

2. $P_{bc}\ (U_{bc}) + (1- P_{bc})\,(U_{ba}) - K_{bc}$

The point of b’s neutrality can be represented as:

3. $P_{ba}\ (U_{ba}) + (1- P_{ba})\,(U_{bc}) - K_{ba} = 1.\ P_{bc}\ (U_{bc}) + (1- P_{bc})\,(U_{ba}) - K_{bc}$

This can be simplified as:

$(P_{ba} + P_{bc} - 1)\,(U_{ba} - U_{bc}) = (K_{ba} - K_{bc})$

Further simplification can be presented as:

$[b/(a+b+c)]\ (U_{ba} - U_{bc}) = (K_{ba} - K_{bc})$

8.6 General Matrix Games

Matrix Games

This is a generic representation of any type of game for two players denoted as *A* and *B*. Player *A* or *B* chose between numbers of possibilities in the game. This could be denoted as M. For each combination of choices, there is a corresponding pay cost associated with the choice made. This is an amount of money or something of measurable value.

This is represented as move *M x L* a pay cost matrix
M = (aij) and *L* is a possible strategy.
Where *aij* is the pay cost when A makes a move i and B chooses strategy *j*.

IF *M* has an element *aαγ* which forms the least element in a row of a matrix and the largest in its column, we can conclude that the game is strictly deterministic.

Most optimised game plan

The most optimised method of play is for A to play an independent strategy *α* and *B* to play independent strategy γ. These are strictly deterministic moves. If any does not play an independent strategy the game is not strictly deterministic.

Supposing that A plays *M=(aij)* strategies with *probabilities x1, x2,……. xm* and *B* plays his *L* possible strategies with probabilities *y1, y2…….yn*. The pay cost aij is determined with probability xiyi hence A's expectation will be represented as:

Summation of aijxiyi =)x1, x2,……...xm)

Where *x,y* are probabilities associated with the moves in vector columns.

Suppose player A chooses a strategy available in a vector position (x'1, x'2, x'm). And B anticipated the strategy, B will make a move from (y1, y2, ……yn) to counteract A's strategy.

8.7 Sample exercise:

(1) In the children's game of "stone-scissors-paper" the two players simultaneously thrust forth (a) clenched fist (stone), (b) two fingers (scissors), or (c) a flat palm (paper). If both players present the same object, the play is drawn; otherwise the winner is determined on the basis that "stone blunts scissor, scissors cut paper and paper wraps stone." Construct a payoff matrix for this game, assuming that the loser pays the winner £1. On purely common sense grounds how would you rate the merits of the three possible moves available to each player.

(2) Two players each think of one of the numbers 2, 3,4,5, 6, 7. If both choose the same number, player A pays player B that number of pounds sterling. If they choose different numbers, then B pays A the number of pounds sterling that A chose. Construct the payoff matrix for this game.

Model answer:

The answer should be based on an assumption. The assumptions made should be fully explained. Eg. Assuming that A and B choose the number 2. A pays B 2 pounds. If different numbers are chosen and supposing A chose 3 B pays A 3 pounds. The payoff matrix could be represented as:

Player B

2	-3
2	**3**

Player A

(3) Two players A and B simultaneously place a dime on a table. If the coins match, i.e., both show heads or both tails, then player A takes both coins; if they do not match, i.e., one shows heads the other tails, then player B takes both. Construct the payoff matrix for this game.

Model answer

Player B

H	T
1	-1
-1	**1**

4. (a) Briefly explain the following concepts underlying game theory:

(i) Mixed strategy

Model answer and hint:

Choosing moves in a random fashion from complete strategies. Examples should be given to support answers

(ii) Dominant strategy

Hint: This should be based on the ff:

Best strategy vrs weakest strategy

(iii) Nash equilibrium

Hint: This should be based on the ff:

Nash equilibrium is when each player is playing a best response to other strategies.

(iv) General equilibrium

A way of normalising and stabilizing game's welfare. It is when communicating is among (infinite) agents however each agent's action is independent of the other.

(v) Pareto optimality

Hint: This should be based on the ff:

Pareto optimality means that every decision made a player in the game to compensate the moves made by another player has an effect on the player that made the move. "Every free lunch is paid for"

References

Alberico R, MARY Mico, Expert systems for reference and information retrieval, Mecker 1990

(Anderson J, 1995) - An introduction to Neural Networks - MIT Press 1995

Andre and Rangayyan (2006) Classification of breast masses in mammograms using neural networks with shape edge sharpness and texture features. Working paper

Andreas Raggl, Mazen Younes, Markus Bonner, Wolfgang Slany: DocuFLIP++-OptiFLIP++/GenFLIP++ library description, 1996 http://www.dbai.tuwien.ac.at/proj/StarFLIP/docuflip/OptiFLIP/genflib.htm

Askers R James (2003) Aviation week & space Technology, New York Vol. 158, Iss 13;pg. 21, White House Budget Office Questions Effectiveness of Passenger Profiling

Bassiliades N, I. Vlahavas, DEVICE: Compiling production rules into event driven rules using complex events, 11/11/96, Elseiver

Bond, A. H. & Gasser, L. (1988), Readings in Distributed Artificial Intelligence, San Mateo, CA: Morgan Kaufmann.

(Bill McCarty 2002); Red Hat Linux Firewalls 1st Edition; Red Hat

(Bonsor K, 2002) How the Airline works.

Castillo, E., Gutierrez, J.M., and Hadi, A.S (1997), Expert Systems and Probabilistic Network Models, Springer-Verlag, New York

Carver, N. & Lesser, V. (1995), "The DRESUN Testbed for Researcg in FA/C Distribution Situation Assessment: Extensions to the Model of External Evidence", In Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, June, 33-40.

Carver, N., Cvetanovic, Z. & Lesser, V. (1991), "Sophisticated Cooperation in Distributed Problem Solving", in Proceedings of the 9th National Conference on Artificial Intelligence 1, Anaheim, 191-198.

Chang, Hanng, Wang, Hsiely 96, Effects of medical expert system on differential diagnosis of renal masses: A prospective study

Chaib-draa, B., Moulin, B., Mandiau, R. & Millot, P. (1992), "Trends in Distributed Artificial Intelligence", Artificial Intelligence Review 6, 35-66.

(Carlson C, 2003) Federals look at data mining, The E-week Enterprise News & Reviews.

(Cox., R.J), An Airline security Experts speaks out Against CAPP 2.

(Crews C, W, 2000) 'Partial' Information Awareness, Cato Institute

(Come D, 1999) Marco Dorigo and Fred Glover. New ideas in Optimisation.

(DARPA, 1988) Neural Network Study (1988, AFCEA International Press).DARPA Neural Network Study (1988, AFCEA International Press)

DARPA Neural Network Study (1988, AFCEA International Press, p. 60):

Decker, K. S. (1995), "Distributed Artificial Intelligence Testbeds", In OíHare, G. & Jennings, N. (eds.), Foundations of Distributed Artificial Intelligence, Chapter 3, London: Wiley, forthcoming.

Decker, K. S. & Lesser, V. R. (1993), "Designing a Family of Coordination Algorithms", in Proceedings of the 11th National Conference on Artificial Intelligence, Washington, 217-224.

Dent, L., Boticario, J., McDermott, J., Mitchell, T. & Zabowski, D. A. (1992), "A Personall Learning Apprentice", In Proceedings of the 10th National Conference on Artificial Intelligence, San Jose, California, AAAI Press, 96-103.
Doran, J., Carvajal, H., Choo, Y. & Li, Y. (1991), "The MCS Multi-agent Testbed: Developments and Experiments", in Deen, S. (ed.), Cooperating Knowledge based Systems, Heidelberg: Springer-Verlag, 240-251.
Durfee, E. H. & Montogomery, T. A. (1989), "MICE: A Flexible Testbed for Intelligent Coordination Experiments", In Proceedings of the 1989 Distributed Artificial Intelligence Workshop, 25-40.
Durfee, E. H., Lesser, V. R. & Corkill, D. (1987), "Coherent Cooperation among Communicating Problem Solvers", IEEE Transactions on Computers C-36(11), 1275-1291.
(Escamilla T, 1998); Intrusion Detection: Network Security Beyond the Firewall 1st Edition; Wiley
(Enete. N, 1997) Java Jumpstart: A Beginner's Guide to Internet Programming. Prentice Hall PTR Prentice-Hall, Inc
Flanagan D. Java in a Nutshell. Second Edition 1997

Forgy, C.L. Rete: A fast algorithm for many pattern/many object match problem. Artificial Intelligence 19, 1(1982) 17-37. Copyright 2002 The HALEY Enterprise, Inc. All rights reserved

Fogel, l. J., Owens, A.J. and Walsh, M.J.: Artificial Intelligence through Simulated Evolution. New York: John Wiley,1966
(Fayyad. U, Grinstein., G.G, Wierse., A, 2002), Information In Data mining And Knowledge Discovery, Morgan Kaufmann publishers, San Francisco, ISBN 1-55860-689-0.
Genetic Algorithms: An Overview http://www-pub.cise.ufl.edu/~ddd/cap6635/Fall-97/Short -papers/12.htm
gpanimatedtutorial : http://www.genetic-programming.com/gpanimatedtutorial.html
(Ghosh A & Schwartzbard A 1999) - A study using Neural Networks for anomaly detection and misuse detection - Reliable Software Technologies. URL : http://www.docshow.net/ids/usenix_sec99.zip
Gao, Venkateswarlu, Quddus (2005) Multi scale corner detection of contour images using wavelet transform.
Gonzalez Rafael C, Woods Richard E. 2002 Digital Image Processing
Published by Prentice-Hall, Inc. ISBN 0130946508

Gasser, L. (1991), "Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems", Artificial Intelligence 47, 107-138.

Gasser, L. & Huhns, M. (1989), Distributed Artificial Intelligence 2, San Mateo, CA: Morgan Kaufmann.

Gasser, L., Braganza, C. & Herman, N. (1987), "MACE: A Flexible Testbed fo Distributed AI Research", In Huhns, M. (ed.), Distributed Artificial Intelligence, Research Notes in Artificial Intelligence, London: Pitman, Chapter 5, 119-152.

Gasser, L. , Rosenschein, J. S. & Ephrati, E. (1995), "Introduction to Multi-Agent Systems", Tutorial A Presented at the 1st International Conference on Multi-Agent Systems, San Francisco, CA, June.

Pascal Glauser. Genetic Algorithms-an Intuitive Introduction July 7, 1999 http://home.sunrise.ch/pglaus/gentore.htm

Hartmut Pohlheim: Evolutionary Algorithms: Principles, Methods and Algorithms. Http://www.geatbx.com/docu/algintro.htm

Holland J, "Adaptation in Natural and Artificial Systems"; 1975

Howe, Balagarasamy, Expert Systems for management and engineering pg. 187

Horak, K., Adlassing, K.P., Development and retrospective evaluation of HEPAXPERT-I, Elseiver, 09/1994.

Hastings G.(2003). 5 common mistakes in Computer forensics

Haykin, S. (1994), Neural Networks: A Comprehensive Foundation, NY: Macmillan, p. 2

Holland, J, H,: Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press, 1975

(Hurley H., 1999), Telephony, Forecasting fraud Vol. 236,Iss.7;pg.50

(T.Honkela, S.Kaski, K.Lagus, and T.Kehonen 1996), Newsgroup Exploration with WEBSOM Method and Browsing Interface. Technical report A32. Otaniemi, Helsinki University of Technology.

(Simon Haykin 1998); Neural Networks: A Comprehensive Foundation 2nd Edition; Prentice Hall

Hewitt, C. (1977), "Viewing Control Structures as Patterns of Passing Messages", Artificial Intelligence 8(3), 323-364.

(InSeon and Ulrich Ultes-Nitsche, 2002) An Integrated Network Security

(Jean-Philippe 2001); Application of Neural Networks to Intrusion Detection; SANS Institute.
(Mark Roy, 2003) Digital Rights Group Takes Swipe at CAPPS II, Business News.

(Koza. R. John, 1992) Genetic Programming: on the Programming of Computers by means of Natural Selection. MIT Press, Cambridge, MA, USA, 1992.
(Koza. R. John, 1989) Hierarchical genetic algorithms operating on populations of computer programs. In N. S. Sridharan, editor, Proceedings of the Eleventh, International Joint Conference on Artificial Intelligence IJCAI-89, volume 1, pages 768-774, San Maeto, CA, USA,, Morgan Kaufman

(KSEAUK,2002). Surrey University,Guildford, Surrey, UK.

(Kung S. Y 1997); Digital Neural Networks 1st Edition; Pearson Education POD

(Samuel Kaski 1997): Data Exploration Using Self-Organizing Maps. Thesis for the degree of Doctor of Technology . Espoo, Helsinki University of Technology.

Lesser, V. & Corkill, D. (1981), "Functionally Accurate, Cooperative Distributed Systems, IEEE Transactions on Systems, Man, and Cybernetics C-11(1), 81-96.
Levitt, R., Cohen, P., Kunz, J., Nass, C., Christiansen, T. & Jin, Y. (1994), "The Virtual Design Team: Simulating how Organisational Structure and Communication Tools affect Team Performance", In

Masters T(1995), Practical Neural Network Recipes in C++, Academic Press

(Mark Roy, 2003), TSA May Order Airlines to Share Data, The leading event for the wired and wireless ISPs

Mckemmish R.(1999) What is forensic computing? Australian Institute of Criminology (Trends & Issues) in crime and criminal justice. ISSN – 0817-8542, ISBN 0—642-241023. No 118
The Haley Enterprise Inc. Business Rule Processing. 1997 CLIPS Conference NASA Johnson Space Center , Houston Texas
(Nash, K.S, 1998) Computer world. Framingham: Vol..32, Iss.6;pg. 1,Electronic profiling
Nigrin, A. (1993), Neural Networks for Pattern Recognition, Cambridge,
MA: The MIT Press, p. 11:
Netfilter and IPTables. URL: http://www.netfilter.org/
Nordic Workshop on Secure IT, Systems .On-line Reference/Websites
Nwana(1996), Cambridge University Press
Oort, Tholen, Lucas, 1999, An intelligent system for pacemaker reprogramming, Elseiver, 17 (1999). Copyright 1998. Production Systems Technologies, Inc.
Pasi Eronen, Jukka Zitting 2001); Proc. Conference on Korean Science and
Engineering Association in UK
Rude T (2000). Evidence Seizure Methodology for Computer Forensics, CISSP. SearchDatabase.com News Writer.
Rosenschein, J. S. (1985), Rational Interaction: Cooperation Among Intelligent Agents, PhD Thesis, Stanford University.
Rosenschein, J. S. & Zlotkin, G. (1994), Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, Cambridge: MIT Press.
Rechenberg, I.: Evolutionsstrategie-Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Stuttgart: Frommann-Holzboog, 1973.
Schwefel, H. -P.: Numerical optimization of computer models. Chichester: Wiley & Sons, 1981.
Smith, R. G. (1980), "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", IEEE Transactions on Computers C29 (12).
Smith, R. (1996a), "Software Agent Technology", Proceedings ofThe First International Conference on the Practical Applications of Intelligent Agents and Multi-Agent Technology, London, UK, 557-571.
Smith, R. (1996b), Personal Communication.
Szolovits, Pauker, Categorical and Probabilistic reasoning in medicine revised
(Swartz., N, 2004) Information Management Journal. Lemexa:Vol. 38,Iss 2;pg.18,1pgs, U.S. to Start Airline Background Checks

SNORT The Open Source Intrusion Detection System. URL:

SOM_PAK. URL : http://www.cis.hut.fi/research/som_lvq_pak.shtml

Sommer P(1997). Computer forensics, an introduction. Virtual City Associates

SOMTOOLBOX. URL http://www.cis.hut.fi/projects/somtoolbox/ South Africa, July 2002

Stonier R and Anwer (2002). Multi-layered fuzzy image filter for removing impulse noise. pp.221, Elmagharby A.S and Dees R., Proceedings of the 11th International conference on intelligent systems of the International Society for Computers & their Applications (ISCA). July 18-20, 2002. Boston, Massachusetts, USA.

Sun Microsystems, Inc.: The Java Language Environment, A White Paper. http://java.sun/docs/white/langenv/Intro.doc 1.html

Temizel and Vlachos (2005) Image resolution up scaling in the wavelet domain using directional cycle spinning (CS).

(Teuvo, Kohonen, 2000); Self-Organizing Maps 3rd Edition; Springer-Verlag

(Timo Honkela 1997), Self-Organizing Maps in Natural Language Processing. Thesis for the degree of Doctor of Philosophy. Espoo, Helsinki University of Technology.

(Turley J, Ron R, Corrigan K, 2002) Panel 1, Subcommittee on Aviation Hearing on Aviation security with a focus on passenger profiling.

(Turley J, Ron R, Corrigan K, 2002) Panel 1, Subcommittee on Aviation Hearing on Aviation security and the future of the airline industry

(Verton D, 2002) .Feds mulling new airline surveillance system

Van Der Gaag L (1991). Principles of Expert Systems, Addison Wesley

Weiss S, Casmir Kulikowski, A practical guide to designing expert systems, Chapman and Hall

Winston P H, Sundar Narasimhan. On To Java 1996 Addison Wesley

Weber J, Special Edition Using Java 1.1, Third Edition, 1997

(Williams., G. 2004). Synchronizing E-Security, Kluwer

Approach Pairing Detecting Malicious Patterns with Anomaly Detection.
WEBSOM project site, 2000. URL http://websom.hut.fi/websom/
Williams G. Aber J (2004) Neural network and airline security. 8th world multi-conference on systemics, cybernetic and informatics USA. July 18-21, 2004 – Orlando, Florida, Invited Session Organiser.
Williams, G.(2005) Translating Statistical images to text summaries for partially sighted persons on mobile devices. Iconic image maps approach. IS & T/SPIE's 17th Annual Symposium Electronic Imaging Science & Technology , Multimedia on Mobile devices 16 - 20 January 2005, San Jose, California, USA. www.catsa-acsta.gc.ca (22/02/04)
Williams G (2004), Conference on CSI 2004, Orlando Florida
Wittig, T. (1992) (ed.) ARCHON: An Architecture for Multi-Agent Systems, London: Ellis Horwood.
Wooldridge, M. (1995), "Conceptualising and Developing Agents", In Proceedings of the UNICOM Seminar on Agent Software, 25-26 April, London, 40-54.
Wooldridge, M. & Jennings, N. (1995a), "Intelligent Agents: Theory and Practice", The Knowledge Engineering Review 10 (2), 115-152.
Wooldridge, M. & Jennings, N. (eds.) (1995b), Intelligent Agents, Lecture Notes in Artificial Intelligence 890, Heidelberg: Springer Verlag.
Wooldridge, M., Mueller, J. P. & Tambe, M. (1996), Intelligent Agents II, Lecture Notes in Artificial Intelligence 1037, Heidelberg: Springer Verlag.
Zlotkin, G. & Rosenschein, J. S. (1989), "Negotiation and Task Sharing among Autonomous Agents in Cooperative Domains", Proceedings of the 11th IJCAI, Detroit, Michigan, 912-917. [an error occurred while processing this directive
Zurada, J.M. (1992), Introduction To Artificial Neural Systems, Boston: PWS Publishing Company, p. xv:
http://davis.wpi.edu/~matt/courses/soms/#Introduction
http://investor.ncr.com/news/20040114-126604.cfm (28/03/04)
http://www.dbmsmag.com.9807m01.html (13/02/04)
http://www.fairisaac.com/Fairisaac/Solutions/innovations_neural_networks4.htm (27/04/04
http://www.house.gov/transporation/aviation/09-21-01/09-21-01memo.html(25/02/04)
http://www.snort.org/
http://www.statsoftinc.com/datamine.html#mining (01/04/04)

Bibliography

Corne D, Dorigo M and Glover, New ideas in Optimisation, 1999