Personal Adaptive Web Agent for Information Filtering

by Imran Khan

A thesis submitted to the Faculty of Graduate Studies in partial fulfilment of the requirements for the degree of

MASTER of SCIENCE

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

February, 1997

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION

PERSONAL ADAPTIVE WEB AGENT FOR INFORMATION FILTERING

BY

IMRAN KHAN

A Thesis/Practicum submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Abstract

This thesis presents one possible design solution for a Web agent which reduces information overload for Web users by autonomously retrieving documents that the user is interested in. The document provides a brief primer on software agents, presents a design solution for an unique model for a Personal Adaptive Web (PAW) Agent, and simulates the individual components of the model.

The agent primer defines what a software agent is, identifies some on-line references to software agents, defines the scope of agent based applications, recommends an agent taxonomy system, and concludes with opinions on the social implications of new technology.

The PAW agent is a personal assistant which learns different categories of Web documents that the user is interested in, then finds and suggests new similar documents to the user. Similar document vector representations and Inverse Document Frequency Weight (IDFW) techniques are employed as in other Information Retrieval Agents. However, this approach is otherwise quite new and produces an agent that is much more autonomous than the others. In fact, the only piece of information that must be supplied to the agent is the number of categories that the user would like to establish.

The PAW agent performs seven subtasks to achieve its goal. It (i) monitors the user while she is browsing the Web, (ii) determines the relevant documents that the user visits, (iii) textually analyses the relevant documents to obtain document vectors using a modified form of the IDFW technique, (iv) classifies the document vectors into categories using unsupervised competitive learning, (v) scans the Web for similar documents, (vi) classifies the new document vectors using the trained neural network, and (vii) decides whether the new documents should be referred to the user.

In accomplishing the above seven subtasks, a real-time database, an automatic text analysis technique, competitive learning network, and a fuzzy inference system are incorporated into the PAW agent. These components are simulated independently in order to verify the design of the individual components.

Acknowledgements

I would like to thank Dr. Howard Card for bringing my attention to this interesting field of research and for many helpful discussions. As well, thanks to Andy Brown and Dean McNeill for insightful discussions on agents. Additional thanks to Professors Jim Peters, Witold Pedrycz, Dave Blight, and Bob McLeod for their involvement.

Financial support from the Natural Sciences and Engineering Research Council and the Canadian Microelectronics Corporation is also gratefully acknowledged.

Contents

List of Figures

List of Tables

Nomenclature

AEA.....adaptive education agent

AI.....artificial intelligence

ANN.....artificial neural network

FIS.....fuzzy inference system

IT.....information technology

PAW.....personal adaptive web

PPM.....period performance measure

WWW.....world wide web

CHAPTER 1

Introduction

"Engineering here is the collision of art & science. It's Imagineering."

-BRAN FERREN

1.1 On-line Explosion

The "creators" of ARPAnet could never have foreseen that their US military research project begun in the late 1960's would have given birth to the current IT phenomena known as the Internet. The text based "network of networks" has quickly procreated as well. The new offspring is known as the World Wide Web (WWW). This is a relatively new Internet application introduced in 1990. A WWW evolutionary jump took place in 1993 with Mosaic which is a graphical Web browsing tool which enhanced the text based browsing. Netscape, Internet Explorer, and several other browsers have come into existence since 1993 which together with network programming languages such as Java, can publish text, graphics, sound, and interactive applets on the WWW.

The users of the Internet and especially of the WWW are growing exponentially. Some estimates report that there are 35 million people on-line and that there will be a billion users by the end of the century. Unlike the users of the radio and television era, the WWW users are not passive entities. They are not only consumers of information but they are also producers. Thus, the WWW is historically the largest and fastest medium for publishing information ever conceived. Given such a large pool of information providers, it is not difficult to imagine that a typical user will be overwhelmed with the vast amount of data on the WWW.

1.2 The Problem

The on-line explosion has created an enormous Information Resource System. But a side effect is that it has given rise to two inherent problems with information systems. The first problem is information overload. This refers to the large amount of information present, and the new information being added constantly on the Web, with the result that a user can not sift through and interpret all the information. For instance, when a query is made and 400,000 enteries are retrieved, the user is expected to traverse through the links and nested links to find the desired material.

The second problem is information recall. This problem addresses the level of accuracy and completeness of the information that the user retrieves from a query on a particular topic. For instance, when a query is made, the user can not be certain that all of the information retrieved is relevant and even if it is, the user can not be certain that the retrieved material is all that is available on the topic.

Presently, there is no complete solution for either problem. However, improvements are being made which help users cope with these problems.

1.3 Solutions

In order to navigate the WWW more efficiently, there are many search engines available. These are reference sites which try to map the contents of the WWW. Given one or more query terms, they recall websites catalogued within their database which match those terms. Most of the search engines use spiders and crawlers to periodically navigate the Web and catalogue their findings. The reader is referred to (Cheong, 1996), for more information on these topics.

Another popular solution to deal with information overload and recall are personal homepages. These are analogous to newspapers because someone else gathers the information in a manageable format. However, unlike newspapers, the user can customize these personal pages. This approach is a complementary solution to the search engine.

Other complementary solutions are personal Web agents. These agents will be explained in detail in Chapter 2. At this time, it is sufficient to say that they are software programs too which the user delegates tasks, in order to reduce her information overload.

In the future, the above three complementary solutions along with other solutions will help ease the burden of information overload and recall for the user.

1.4 Thesis Design Problem

This thesis provides one possible solution for the design of a Personal Adaptive Web (PAW) agent which helps reduce the information overload for its user. The agents' delegated task should be to determine which types of documents its user is interested in then be able to find and suggest new similar documents. The design should result in an agent that is autonomous, personalizable to a single user, and adaptive.

1.5 Overview

The remainder of the thesis is organized as follows. Chapter 2 is a brief background on hierarchical colored Petri nets, fuzzy logic, artificial neural networks, automatic text analysis, and software agents. Chapters 3 and 4 present the PAW agent model and simulation respectively. Chapter 5 presents the conclusions and recommendations. Finally, Appendices D, E, and F present related topics on the Scope for Agent Implementation, Taxonomy Systems for Agents, and Social Implications of New Technologies respectively.

CHAPTER 2 *Background*

"I know what it does but I don't know what it will undo!"

-NIEL POSTMAN

This chapter provides a brief background on the topics of hierarchical colored Petri nets, fuzzy sets, artificial neural networks, automatic text analysis, addresses the question: What is an Agent? and lists on-line references to existing software agents. For more details, the reader may refer to the suggested references and Appendices D, E, and F. Appendix D emphasizes the wide scope of agent-oriented applications, by suggesting how agents be applied to an education system. Appendix E presents some initial thoughts for a taxonomy of agents. Finally, Appendix F comments on the social implications of new technology in general.

2.1 Hierarchical Colored Petri Nets (HCPN)

Petri nets are models that can represent very general discrete event systems. They were developed by C.A. Petri in the early 1960's. The Petri net is a device that manipulates events according to certain rules. The type of Petri nets used in the thesis are the HCPN. The HCPN model is an enhanced version of the original Petri net. This allows very complex schemes to be represented through a hierarchical format and is also able to convey more detailed information.

In this section, a brief informal introduction to HCPN is given. Readers should refer to (Abouaissa et al., 1991; Jensen, 1992) for a more complete and formal explanation. Like any Petri net, the HCPN includes a finite set of places, tokens, transitions, and arcs. The

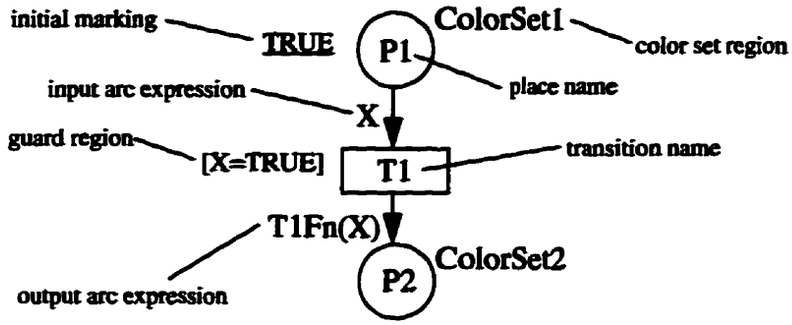


FIGURE 2.1. Colored Petri Net Model.

places represent the possible states of the system which are indicated by circles. The tokens represent the data associated with each place which is indicated by small dots. The transitions are events which are indicated by rectangles. The arcs are indicated by directed arrows which connect the places and transitions in the net. The distribution of the tokens over the places is called a *marking*. The marking changes whenever a transition fires. A transition can fire when each of its input places contains at least one token. After the transition fires, each input place loses one token and each output places gains one token.

In the HCPN there are several enhancements. Each place has three associated regions. The *Name region* is a string which identifies the place. The *Color Set region* indicates the type of tokens allowed in the place. The tokens can be complex data structures. The *Initial marking region* indicates the type and number of tokens at each place at the start of the simulation.

There are also two regions associated with each transition. The *Name region* is a string which identifies the transition. The *Guard region* prevents the transition from firing until the boolean expression (guard) with colors as terms is satisfied. By default, if there is no guard region then the transition fires when the input places have the required number and type of tokens.

The arcs have associated expressions. The input arc expressions represent the tokens that will move from the input place to the transition. The output arc expressions represent the new token that is added into the output place. Figure 2.1 illustrates a simple example of a CPN model.

Hierarchy is introduced in CPNs in order to make large complex models easier to understand. Transitions which are shaded represent the embedded hierarchy. These transitions are decomposed into subnets which are then shown separately.

2.2 Fuzzy Sets

This section introduces fuzzy sets, membership functions, fuzzy measures, and if-then rules. For more detailed studies the reader may refer to (Pedrycz, 1995). Classical sets impose a binary membership whereas fuzzy sets allow analog values of class membership. Consider, for instance, the following question: Is today hot? In classical or crisp set theory, the answer would be either yes (1) or no (0). However, the fuzzy set answer might be 0.5 which means that it is in between hot and not hot.

2.2.1 Membership Functions

A membership function is a curve that must vary between 0 and 1. It is associated with a specific fuzzy set and maps an input value from the input space to its appropriate membership value. In set theory, a set A defined in X is represented completely by $A : X \rightarrow \{0, 1\}$ such that,

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

A fuzzy set A defined in X is characterized by its membership function $A : X \rightarrow [0, 1]$ where A(x) represents the degree of membership of x in the concept conveyed by A.

2.2.2 Fuzzy Measures

Fuzzy measures are operations which are performed on fuzzy sets. In this thesis, only the possibility and necessity measures are used. These concepts are illustrated in Figure 2.2.

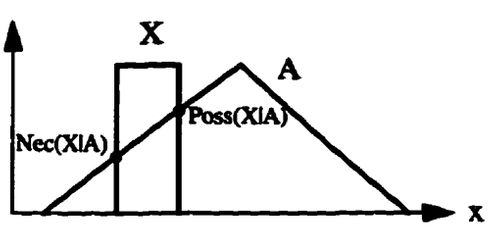


FIGURE 2.2. Possibility & necessity measures.

Let A represent a fuzzy set while X constitutes an input datum. X and A are defined on the same universe of discourse. The possibility measure represents the degree to which X and A overlap and is given by

$$Poss(X|A) = sup_{z \in X}[min(X(z), A(z))]$$

The necessity measure represents the extent to which X is included in A and is given by

$$Nec(X|A) = inf_{z \in X}[max((1 - X(z)), A(z))]$$

2.2.3 IF-THEN Rules

A typical fuzzy if-then rule might be of the following form: if (x is A) and (y is B) then (z is C) where A, B, and C are linguistic terms defined by fuzzy sets on the corresponding input spaces X, Y, and Z respectively. In fuzzy if-then rules, the premise is an interpretation that returns a single real number in the range [0,1] while the conclusion is an assignment that assigns the entire fuzzy set C to the output variable z.

The if-then rule is evaluated in three steps. First, the premise is evaluated which involves fuzzifying the input statements between 0 and 1. Second, the appropriate fuzzy operators are applied, in order to obtain a single real number in the range [0,1]. This is the degree of support for the rule. Finally, the previous result is applied to the conclusion which is known as implication. Thus, if the premise is true to some degree then the conclusion is also true to that same degree. One of the most common implication methods is the min function which modifies the output fuzzy set using truncation.

In general, there are several if-then rules. Thus, the output fuzzy sets for each rule are aggregated into a single output variable usually using the max function. Finally, the resulting fuzzy set is defuzzified usually using the centroid function, in order to obtain a single real number in the range [0,1].

2.3 Artificial Neural Networks

This section briefly introduces artificial neural networks (ANN) and competitive learning. For more detailed studies, the reader may refer to (Haykin, 1994).

An ANN is an information processing machine. It is loosely based on the way the biological nervous system such as the human brain processes information. ANNs are increasingly being used to solve real world problems which do not have algorithmic solutions or where the algorithmic solution is complex. An ANN is composed of a large number of neurons (processing elements) that are interconnected. The neurons and their interconnections are configured to solve specific applications in pattern recognition and data compression using a learning process. The ANN learns by example by repeatedly adjusting the strengths of the interconnections between the neurons.

An ANN viewed as an adaptive machine is a massively parallel distributed processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects. First, knowledge is acquired by the network

through a learning process. Second, interneuron connection strengths known as synaptic weights are used to store the knowledge.

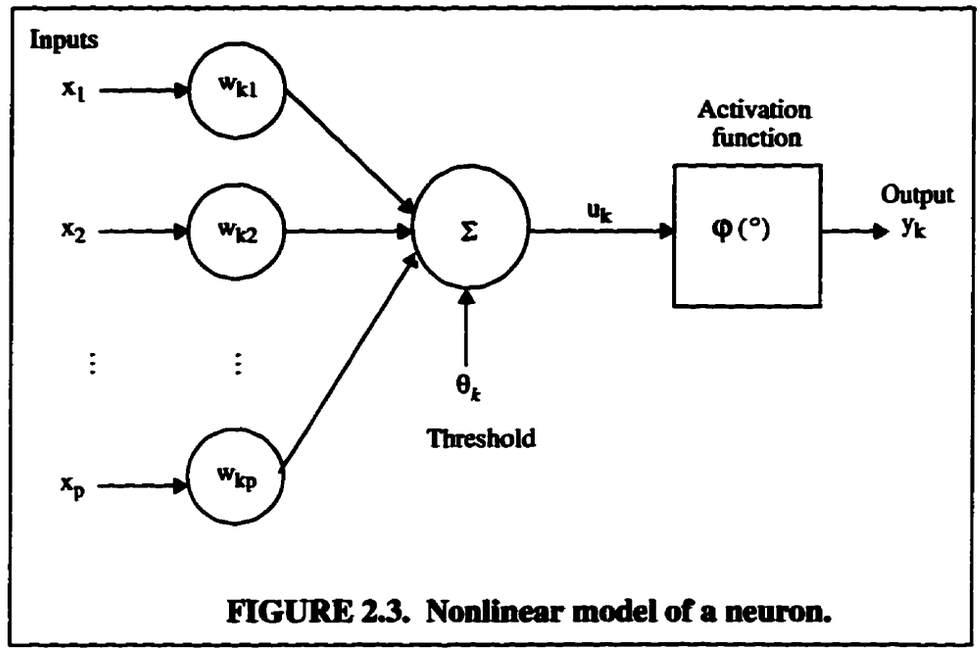


FIGURE 2.3. Nonlinear model of a neuron.

A neuron is an information processing element with three basic elements as shown in Figure 2.3. The first element is a set of synapses. Each synapse is a connection between an input and a neuron, and has an associated synaptic weight. The second element is an adder, which sums the input signals, multiplied by their respective synaptic weights. The last element is an activation function $\varphi(\cdot)$ which is used to limit the output of the neuron. A neuron k can be represented as

$$u_k = \sum_{j=1}^{p} w_{kj} x_j \qquad \& \qquad y_k = \varphi\left(u_k - \theta_k\right)$$

where $x_1, x_2, \ldots, x_p$ are input signals, $w_{k1}, w_{k2}, \ldots, w_{kp}$ are synaptic weights, and θ_k is a threshold value.

2.3.1 Competitive Learning

Competitive learning is employed to cluster a set of input patterns without supervision. The neurons in the competitive layer learn to recognize different regions of the input space where input vectors occur. The ANN has a single layer of output neurons which are fully connected to the input nodes. All the neurons compete with each other to win the competition for a given input vector. A neuron wins if it has the largest internal activity, u_k, for a specific input pattern among all the other neurons. The winning neurons' output is set to 1 while all the neuron outputs are set to 0. The winning neuron learns by moving its weight vector in the direction of the input vector, while the losing neurons do not learn on this input pattern. The standard competitive learning rule, is given by

$$\Delta w_{ji} = \begin{cases} \eta\left(x_i - w_{ji}\right) & \text{if neuron j wins} \\ 0 & \text{if neuron j loses} \end{cases}$$

where η is a learning rate parameter.

2.4 Automatic Text Analysis

This section briefly introduces automatic text analysis and term weighting functions. For more information on these topics, the reader may refer to (Salton, 1982). The automatic text analysis process involves the assignment of index terms to a collection of documents for the purpose of classifying the documents. The process involves the following steps. The first step is to identify all the individual words within the document collection. The next step is to eliminate all the high frequency words since they are poor discriminators and are not by themselves useful in identifying document content. In English, there are about 250 common words that have this high frequency characteristic and are usually a part of a stop list. After the stop words are eliminated, the suffixes of the remaining words can be removed. This reduces the original words to only their word stems. With the use of the stems as index terms there are potentially more relevant items that can be identified. For more information on suffix removal algorithms, the reader is referred to (Salton, 1982; Salton, 1968). The next step is to assign to each of the remaining words a weight value, reflecting its importance for content identification purposes, as discussed in the next subsection. The index terms with sufficiently high weight values can be assigned to the documents of the collection with a binary index mode. Thus an index term that

occurs in a document is assigned a value of 1 and 0 otherwise. For each document, a document vector is created as $D_i = \{d_{i1}, d_{i2}, ..., d_{it}\}$ where each d_{ij} is the binary value assigned to the jth identifier for document D_i and t represents the number of identifiers.

2.4.1 Term Weighting Functions

There are several different term weighting functions including the inverse document frequency weight (IDFW), the signal-noise ratio, and the term discrimination value. The IDFW method is the only method described in this thesis; for the other methods the reader is referred to (Salton, 1982).

The IDFW assumes that the importance of the term is proportional to the frequency of each term k in each document i (represented as $FREQ_{ik}$) and inversely proportional to the total number of documents to which each term is assigned. The number of documents to which term k is assigned is called the document frequency ($DOCFREQ_k$). A possible measure of the IDFW is:

$$WEIGHT_{ik} = FREQ_{ik} \times \left[\log_2(n) - \log_2\left(DOCFREQ_k\right) + 1\right]$$

where n is the number of documents in the collection. The equation shows that the IDFW value in a given document increases as the frequency of the term increases but decreases as the document frequency increases.

2.5 What is an agent?

The term agent is rapidly becoming the term associated with network software applications and especially with WWW applications. It is interesting to speculate why so many software researchers and developers are using the term agent to refer to their products. The simple answer is that most of them are following a trend to create a mystical and futuristic atmosphere for their products. An example of this *Smart Magic* is reported in (Dibble, 1996). A more analytical opinion which might be more enlightening has been expressed by (Foner, 1993): "Like AI before it, the fairly loose specification of the mean-

ing of agent and the use of a word with a nontechnical meaning in a technical way has blurred the boundaries and made the concept available for appropriation."

Actually, the word agent means different things to different people in different situations. This seems to coincide with (Russell, 1995): "The notion of an agent is meant to be a tool for analysing systems, not an absolute characterization that divides the world into agents and non-agents."

It may be wise for the industry to develop a suitable definition for the term agent. One possible approach is to separate agents into real-world agents (i.e. most animals including humans), software agents (i.e. which exist in computer operating systems, databases, networks, etc.), and artificial life (i.e. artificial biological environments in a computer system) as listed in (Franklin, 1996). According to Franklin et al., characteristics that software programs should have in order to be called agents are:

1. being situated in and part of an environment
2. the ability to sense its environment and act autonomously upon it
3. independence of other entities supplying input or interpreting output
4. acting in pursuit of its own agenda
5. current action affects its later sensing
6. acting continuously over some period of time

Their formal definition is then stated as: "An autonomous agent is a system within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so to effect what it senses in the future."

Franklin's definition yields a general superclass for agents. If more requirements are specified then smaller subgroups of agents can be created. Additional terms which can be used include, but are not limited to, mobile, adaptive, personal, and collaborative which further define how and what the specific agent accomplishes.

In comparison, (Foner, 1993) states that the crucial notions surrounding agents are:

1. Autonomy: agent pursues an agenda independently of the user
2. Personalizability: agent learns about the user without explicit programming
3. Discourse: a contract between the agent and user
4. Risk & trust: agent is delegated a task by the user which introduces a risk that it will do something wrong along with the trust that it will do something right

5. Domain: agent must have a domain which it is familiar with
6. Graceful degradation: it is better for an agent to accomplish some tasks rather then fail to do any tasks
7. Cooperation: user and agent are treated as peers
8. Expectation: user's expectation of agent is important

No matter how an agent is defined, one hopes that they should be delegated tasks in order to assist users. The tasks that they can be delegated are practically unlimited. They can help hide complexity of tasks, perform mundane routine tasks, educate the user, monitor events, collaborate with other users or agents, find information, serve as entertainers, and serve many other functions.

From this introductory material, we can conclude three things. First, an agent does not need to be a program. For instance, a robot can be an autonomous agent. Second, all software agents are programs but not all programs are agents. Finally, it may require some time, for the industry to settle on a robust definition of an agent.

2.6 On-line References for Software Agents

Originally, this section was intended to summarize and critique several software agents already on-line on the WWW. However, since there is no single definition for an agent which is accepted by industry, it would not be appropriate to scrutinize the work of others based on subjective opinions and self-imposed definitions. Instead, several links to agent-related websites are presented. The reader is encouraged to explore these links and decide on whether the programs they encounter are indeed agents.

TABLE 2.1. On-line References for Software Agents

Name / Description	Website
MuBot Description	www.crystaliz.com/logicware/mubot.html
The SodaBot Home Page	www.ai.mit.edu/people/sodabot/sodabot.html
The Software Agents Mailing List	www.cs.umbc.edu/agentslist/
Brightware at a Glance	www.brightware.com/solutions/index.html
Tactical Picture Agent	www.itd.nrl.navy.mil/ONR/aci/agents.html
Welcome to Autonomy Agentware	www.agentware.com/index.htm

TABLE 2.1. On-line References for Software Agents

Name / Description	Website
Autonomous Agents '97	www.dlib.com/events/conferences/agents97/aa97.html
Agents Conferences Database	lcs.www.media.mit.edu/cgi-bin/conference0
MIT Media Lab, Agents Group	agents.www.media.mit.edu/groups/agents/
UMBC AgentWeb	www.cs.umbc.edu/agents/
IBM Agent Building Environment	www.networking.ibm.com/iag/iagsoft.htm
Firefly	www.firefly.com/

CHAPTER 3

PAW Agent Model

"You don't build it for yourself. You know what the people want & you build it for them."

-WALT DISNEY

As stated earlier, the purpose of the Personal Adaptive Web (PAW) agent is to monitor it's Web user in order to learn the different categories of Web documents that the user is interested in, then find and suggest new similar documents for the user.

The agent learns the category of documents that the user is interested in by employing a real-time database, text analyser, competitive learning network, and fuzzy inference system. The high level interaction between the user, the PAW agent, and the WWW is shown in Figure 3.1. First, the user views Web documents using a browser. Second, the agent monitors and records this activity. Third, it learns the category of documents that the user is interested in and finds new similar documents. Finally, it suggests the new relevant documents to the user.

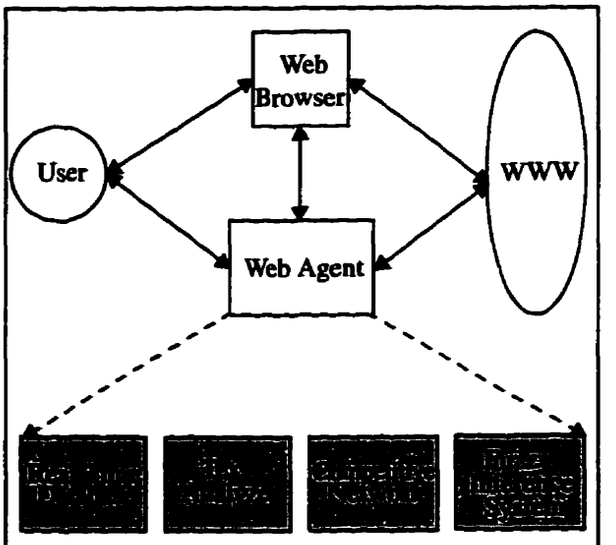


FIGURE 3.1. Block diagram for the interaction between the user, agent, and WWW.

The PAW agent achieves its delegated task by performing the following seven subtasks:

1. Monitor the user while she is browsing the Web
2. Determine which of the visited documents are relevant
3. Textually analyze these relevant documents and reduce each into a document vector
4. Classify the document vectors into categories using unsupervised competitive learning
5. Scan the Web for new similar documents
6. Produce document vectors for the new documents and classify them using the trained neural network
7. Decide whether to suggest the new documents to the user

The above seven subtasks are performed by three different components of the agent which are shown in Figure 3.2.

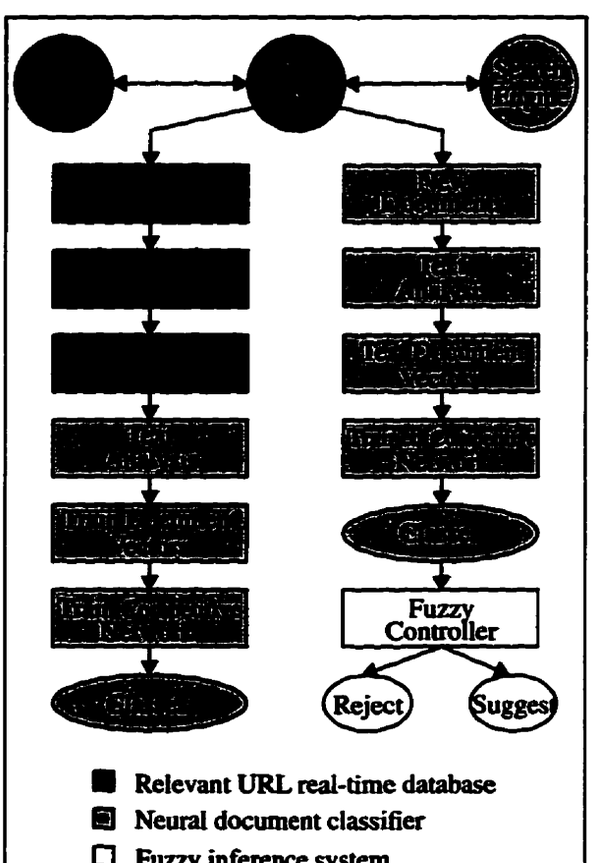


FIGURE 3.2. Components of the Agent.

The first component is the *relevant URL real-time database* which performs subtasks 1 and 2. It uses a novel approach described in the next section to calculate the relevance value of the document and uses this value to determine how long its corresponding record remains in the database.

The second component is the *neural document classifier* which performs subtasks 3 to 6. It uses automatic text analysis and competitive learning techniques which are discussed in the subsequent section.

The third component is the *fuzzy inference system* described in the subsequent section which performs subtask 7. It uses the results from the previous component and two other criteria to decide whether to reject the new document or suggest it to the user.

3.1 Relevant URL Real-Time Database

The purpose of this component is to monitor the user while she is browsing the Web and determine which of the visited documents are relevant. At this point in the development of the Agent, the portion of the software monitoring the user has not been implemented. Instead, a hand crafted browsing activity file for the user is employed. Each record in the file is composed of three fields: a URL (representing the HTML document location), the month, and the date it was accessed. The HCPN model that is developed is shown in Figure A.1 in Appendix A. The database model is implemented using Borland C++ 3.1. The following subsections describe the sub-components of the database model.

3.1.1 Building the Database

The database is partially generated from the user's activity data file. Each unique URL is assigned to a different record in the database. Each record contains the following fields: *URL name*, *month*, *period*, *frequency array*, and *relevance*. Most of the fields are self-explanatory. The *period* field is the length of the interval over which the relevance for the particular URL is calculated. Note that this period is initially set to a default value (7 days) but can change for later periods depending on the number of times the URL is visited over the current period. This will be explained in more detail later. The *frequency array* elements represent the days of the current period. For each day of the period that the URL is visited the appropriate element in the array is assigned a value of 1. Figure 3.3

and 3.4 show a typical user's activity data file (in this example the start of the period is taken to be Dec. 16) and the associated database structure (without the R computed at this point) that is created from it. The decomposed Petri Net for the *Read File* transition in the HCPN is shown in Figure A.2 in Appendix A.

3.1.2 Calculating the Relevance

The relevance is a normalized numerical value which indicates the relevance of the URL in terms of its frequency array characteristics (i.e. how often the document is accessed over the period). In the relevance calculation, the number of visits during the period and the distribution of the visits are both taken into consideration.

http:\\www.ee.umanitoba.ca	Dec 16
http:\\www.media.mit.edu	Dec 16
http:\\www.ee.umanitoba.ca\~imran	Dec 16
http:\\www.yellow.com	Dec 17
http:\\www.mts.com	Dec 18
http:\\www.ee.umanitoba.ca\~imran	Dec 18
http:\\www.mts.com	Dec 19
http:\\www.mts.com	Dec 19
http:\\www.cc.umanitoba.ca	Dec 19
http:\\www.ee.umanitoba.ca\~imran	Dec 19
http:\\www.ee.umanitoba.ca\~imran	Dec 20
http:\\www.yellow.com	Dec 20
http:\\www.mts.com	Dec 20
http:\\www.ee.umanitoba.ca	Dec 20
http:\\www.ee.umanitoba.ca	Dec 21
http:\\www.yellow.com	Dec 21
http:\\www.ee.umanitoba.ca\~imran	Dec 21
http:\\www.cc.umanitoba.ca	Dec 22
http:\\www.media.mit.edu	Dec 22
http:\\www.ee.umanitoba.ca\~imran	Dec 22

FIGURE 3.3. Typical test data file.

URL Frequency Array	Month	Period	R
http:\\www.cc.umanitoba.ca 0 0 0 1 0 0 1	Dec	7	-
http:\\www.ee.umanitoba.ca 1 0 0 0 1 1 0	Dec	7	-
http:\\www.ee.umanitoba.ca\~imran 1 0 1 1 1 1 1	Dec	7	-
http:\\www.media.mit.edu 1 0 0 0 0 0 1	Dec	7	-
http:\\www.mts.com 0 0 1 1 1 0 0	Dec	7	-
http:\\www.yellow.com 0 1 0 0 1 1 0	Dec	7	-

FIGURE 3.4. Associated database generated from test data file.

The algorithm to calculate the relevance for each record (HTML document) is as follows. The relevance is calculated in $\log_2 P$ stages where P is the period over which the relevance is to be computed. In each stage the period is subdivided into regions called bins. In the first stage, the bin size is P and in subsequent stages the bin size is reduced by one half its previous value. Thus in the last stage, the bin size will be either two or three depending on whether the value of P was even or odd respectively. In the above calculations only the integer part of the answers are used.

For the first stage, the partial relevance contributed from this stage is simply the total number of visits in the period. For each remaining stage, the partial relevance contribution is computed as follows. Starting from the leftmost bin until the last bin is reached, adjacent bins are compared. The comparison involves multiplying the number of visits in the left bin interval by those in the right bin interval. This value is then added to the comparison of the next two adjacent bins and so on until the last adjacent pair bins are

reached. Before this accumulated sum is added to the other partial relevance values from the other stages, it is multiplied by a weighting factor. The weighting factor has the effect of placing the most importance on the partial result from the first stage and geometrically reducing the importance for each subsequent stage. This weighting factor is given by

$$\alpha = \frac{1}{2^{stage-1}}$$

After the relevance is aggregated from the weighted sums of the $\log_2 P$ stages, it is normalized by the value of perfect relevance. Perfect relevance is the relevance value which is computed for a hypothetical document that is visited each day of the period, i.e. the maximum relevance value. From this point on, the normalized relevance value will simply be referred to as the relevance. Figure 3.5 shows the sample relevance calculations for the first three documents in Figure 3.4. Note that the first row in Figure 3.5 represents the perfect relevance frequency array characteristics. The decomposed Petri net for the *Calculate R* transition in the HCPN is shown in Figure A.3 in Appendix A.

1	1	1	1	1	1	1
0	0	0	1	0	0	1
1	0	0	0	1	1	0
1	0	1	1	1	1	1

Stage 1 bin

Stage 2 bins

R = {[1][7] + [1/2][(3)(3)+(3)(1)]} / {[1][7] + [1/2][(3)(3)+(3)(1)]} = 13/13 = 1

R = {[1][2] + [1/2][(0)(1)+(1)(1)]} / {[1][7] + [1/2][(3)(3)+(3)(1)]} = 2.5/13

R = {[1][3] + [1/2][(1)(2)+(2)(0)]} / {[1][7] + [1/2][(3)(3)+(3)(1)]} = 4/13

R = {[1][6] + [1/2][(2)(3)+(3)(1)]} / {[1][7] + [1/2][(3)(3)+(3)(1)]} = 10.5/13

FIGURE 3.5. Sample relevance calculations.

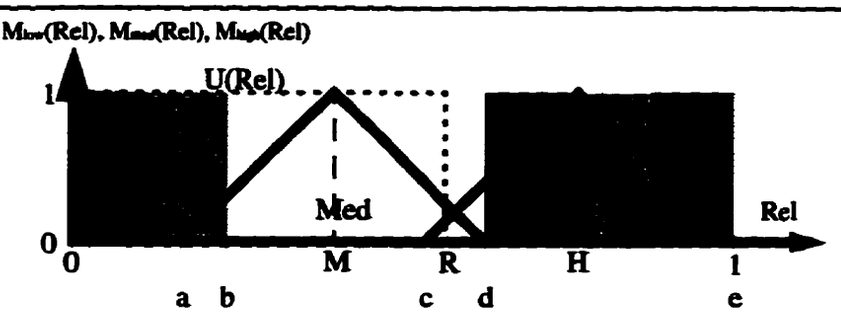


FIGURE 3.6. Fuzzy sets for relevance.

3.1.3 Updating the Length of the Period

After the relevance is calculated, we want to determine whether the period over which it is calculated needs to be modified for the next interval. To determine this, a period performance measure is calculated using the relevance value and three fuzzy sets. The triangular membership functions representing low, medium, and high relevance fuzzy sets are shown in Figure 3.6. The relevance value, R, is used to create a step function which is also illustrated in Figure 3.6 and is defined as

$$U(Rel) = \begin{cases} 1, 0 \le Rel \le R \\ 0, elsewhere \end{cases}$$

The fuzzy sets can be grouped into three regions. The relevance value, R, defines the boundaries as follows

$$\text{Region 1, if } (R \le M)$$
$$\text{Region 2, if } (M < R \le H)$$
$$\text{Region 3, if } (R > H)$$

The period performance measure (PPM) is then defined as

$$PPM = PPM1 + PPM2 + PPM3$$

where (note that L, M, and H refer to M_{low}, M_{med}, and M_{high} respectively)

$$PPM1 = max_{Rel \in U \cap [0,M]} [1 - Nec(U(Rel), L(Rel)), Poss(U(Rel), M(Rel))]$$

$$PPM2 = max_{Rel \in U \cap (M,H]} [1 - Nec(U(Rel), M(Rel)), Poss(U(Rel), H(Rel))]$$

$$PPM3 = max_{Rel \in U > H} [1 - Nec(U(Rel), H(Rel))]$$

The period is now modified according to the following guidelines.

- If $0 \leq PPM1 \leq \frac{1}{2}$ then reduce P to the minimum value
- If $\frac{1}{2} < PPM1 \leq 1$ & $M'_{Low}(R) < 0$ then decrease P by $\frac{1}{5}$
- If $PPM2 = 1$ then increase P by $\frac{1}{5}$
- If $0 \leq PPM3 \leq \frac{1}{2}$ then increase P by $\frac{2}{5}$
- If $\frac{1}{2} < PPM3 \leq 1$ then increase P by $\frac{3}{5}$

These guidelines are shown as shaded regions in Figure 3.6. In order to control the size of the database, those documents that fall into the first guideline twice in a row (i.e. consecutive periods), are removed from the database because they are irrelevant. The decomposed Petri nets for the three *Modify Px* transitions in the HCPN are shown in Figure A.4 in Appendix A.

3.1.4 Await the next Period

After any necessary modification to the period has been made for each HTML document record, the model waits until the end of the next period at which time, the entire cycle of

events are repeated. The decomposed Petri net for the *Wait* transition in the HCPN is shown in Figure A.5 in Appendix A. Note that the *Wait* subnet is basically the Richter Chime clock.

3.2 Neural Document Classifier

The purpose of this component is first to textually analyse the relevant documents and reduce them into document vectors. Second, to scan the Web for new documents and produce their document vectors. Finally, to classify the relevant document vectors into categories using unsupervised competitive learning, and then to classify the new document vectors using the trained neural network. The following subsections describe the subcomponents of the neural document classifier.

3.2.1 Generating Relevant Document Vectors

The purpose of this sub-component is to determine a limited set of index terms found from within all the relevant documents which best represent the whole document collection. The Web documents are analysed textually one at a time. For each document, its words are compared to a stoplist. The stoplist is a collection of the most frequent words used in the English language. The words that are shorter then three characters or have a match in the stoplist are discarded since they do not help distinguish among different documents. The remaining words are recorded in a database and monitored to determine how many times they occur in each document and within the whole document collection. After all the words in the document collection have been statistically recorded, each word is assigned a weight according to the modified Inverse Document Frequency Weight (IDFW) method. The formula for the modified IDFW is

$$IDFW\left(term_k\right) = (coef)\left(\log_2(n) - \log_2\left(DOCFREQ_k\right) + 1\right)$$

where

$DOCFREQ_k$ is the number of documents to which term k is assigned

n is the number of documents in the collection

$$coef = \begin{cases} 1 & \text{if} \quad FREQ_{ik} = 1 \\ 1.5 & \text{if} \quad 1 < FREQ_{ik} \leq 5 \\ 2 & \text{if} \quad 5 < FREQ_{ik} \leq 10 \\ 2.5 & \text{if} \quad FREQ_{ik} > 10 \end{cases}$$

The formula for the IDFW is modified from the original equation in order to prevent a single term obtaining a high IDFW value. An example of a situation which can give rise to a high IDFW value is as follows. A word is found in only one of the documents in the entire collection, however it occurs in that document many times, thus resulting in a large IDFW value. In order to avoid this situation, the coefficient is limited to a maximum value of 2.5 as indicated in the modified IDFW equation.

The words with the highest IDFW values are selected as the index terms to represent the collection of documents. The final step is to use only the index terms to extract the document vectors from the database to create the input vectors for the ANN stage.

The C program to generate the relevant document vectors is called *train.c* which is discussed next.

3.2.1.1 Data Structures

There are three main data structures. The first structure is for the stop list which is accessed through an array of strings. Each element in the array is for a word in the stop list. The second structure is for the training database which stores the statistical information about the words in the document collection. The database has a record for each individual word that is not found in the stop list and is larger then two characters. It is implemented using a singly linked list. Each element of the list has a data component and a link to the next element. The data component has five further fields all related to a single term: the name of the term, an array to record the frequency of the term in each document, the total frequency of the term, an array to record the IDFW of the term in each document, and the aggregate IDFW value. The third structure is an array of type *list ele-*

ment which contains only those records from the database which correspond to the index terms. The index terms are the words with the highest IDFW value, and are used to represent the entire document collection.

3.2.1.2 Input / Output Files Accessed

The program accesses one input file called *stop1.dat* which contains the stop list words, and is shown in Figure B.1 in Appendix B. In addition, there are three data files generated by this program. The first file called *names.dat* contains the names of the train data files that are analysed. The second file called *terms.dat* contains the index terms selected by the program. The last file called *nnet.dat* contains the binary document vectors for the ANN training data set.

3.2.1.3 Operation of Program

The flowchart for the operation of the *train.c* program is shown in Figure 3.7. The stop list words are read into their data structure and the output files are initialized. For each training document, all the words are compared to the stop list. If the words are not found in the stop list and are longer than two characters, then the database is updated. The update occurs in one of two ways. If the word is not in the database, a new element is created for it. Otherwise, the word already exists in the database, in which case the frequency counters are incremented.

After all the words in the current document are analysed, the name of the file is appended to the *names.dat* file. Now the next document is analysed as before until all the documents are finished. The next step is to calculate the IDFW for each word and then select the words with the highest IDFW values as the index terms. The index term names are then written to the *terms.dat* file.

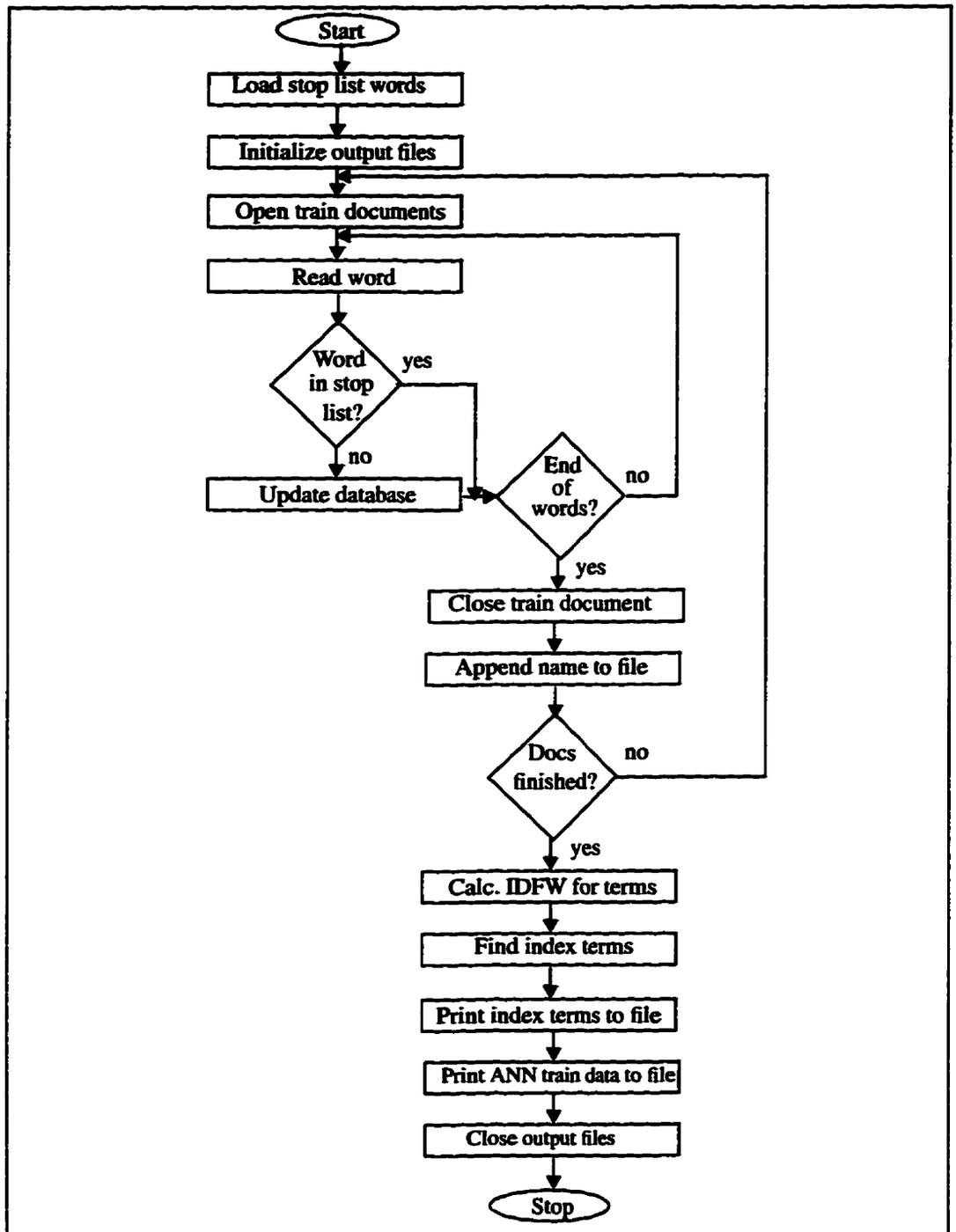


FIGURE 3.7. Flowchart for the operation of the train.c program.

The next step is to write the ANN train data to the *nnet.dat* file. A typical ANN data set containing the document vectors is shown in Figure 3.8. Each row of the training dataset represents an individual document in which the columns have binary values for each of the index terms. A 1 indicates that a particular index terms is present while a 0 indicates that it is missing. Finally, the output files are closed.

```
1 1 1 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0
1 1 1 0 1 1 0 0 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 0 1 0 0 1 0
1 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0
0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 0 0 0 0
1 1 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0
```

FIGURE 3.8. Typical ANN data set.

3.2.2 Generating New Document Vectors

The category names for the documents are used to query search engines to find new documents that might be of interest to the user. The purpose of this sub-component is to use the index terms to generate the document vectors for the newly retrieved documents. The new documents are analysed one at a time. For each document, it is determined whether or not the index terms occur in the document. If the index term exists, then a 1 is recorded in the new database, otherwise a 0 is recorded. The final step is to create the new document vectors from the database which is used to simulate the trained ANN.

The C program to generate the document vectors for the newly retrieved documents is called *test.c* which is discussed next.

3.2.2.1 Data Structures

There are two main data structures that are used. The first structure is for the index terms which is accessed through an array of strings. The second structure is for the binary valued document vectors which is implemented as an array. Each element in the array is a record for a newly retrieved individual document. The record itself is an array which has

one element for each of the index terms. An integer value indicates the number of times the index term occurs in the document.

3.2.2.2 Input / Output Files Accessed

The program accesses one input file called *terms.dat* which contains the index terms generated from the previous program. There are also two data files that are generated by this program. The first file called *names2.dat* contains the names of the newly retrieved document files that are analysed. The second file called *nntest2.dat* contains the binary document vectors for the new document collection.

3.2.2.3 Operation of Program

The flowchart for the operation of the *test.c* program is shown in Figure 3.9. The first action is to initialize the new document collection database. This involves setting all the counters to 0 for each of the new documents. Next, the index terms are read into the index term array and the output files are initialized. For each of the new documents, all of their words are compared to the index terms. If a word is the same as an index term, the corresponding counter of that index term is incremented.

After all the words in the current document are analysed, the name of the file is appended to the *names2.dat* file. Now the next document is analysed as before until all the documents are completed. The next step is to write the document vectors to the *nntest2.dat* file. Finally, the output files are closed.

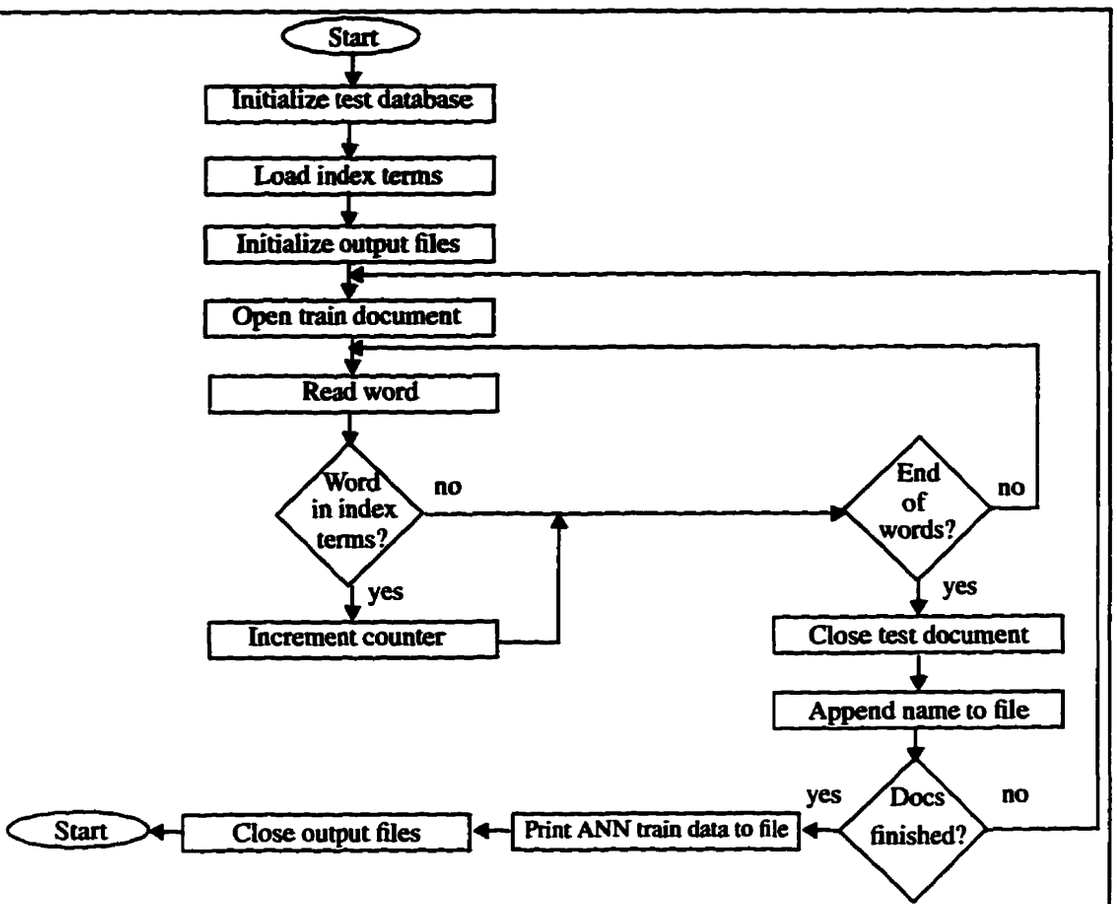


FIGURE 3.9. Flowchart for the operation of the *test.c* program.

3.2.3 Classifying Document Vectors

The purpose of this sub-component is to train an ANN to classify similar document vectors into the same class using a competitive network architecture and competitive learning algorithm. The competitive network is trained using the relevant document vectors generated from the first stage. Thus, the neurons in the layer learn to represent different regions of the input space where input vectors occur. After the network is trained, it is simulated

using the document vectors from the newly retrieved documents. The network can now classify the new document vectors into classes according to its learned categories.

The MATLAB program to simulate the ANN is called *simulate.m* which is described next.

3.2.3.1 Competitive Learning Architecture

The competitive learning architecture is shown in Figure 3.10. The illustration shows that there are R inputs and S neurons. The net input of a competitive layer is computed by finding the negative distance between the input vector and the weight vectors and then adding the biases. The competitive transfer function receives the net input vector from the layer and returns the output vector of the neuron. All neurons have an output of 0 expect for the winner neuron which has a value of 1. The winner neuron is the one with the highest net input. The competitive network is then trained by updating only the weights of the winning neuron, to be even closer to the input vector. This results in the winner being more likely to win the competition the next time a similar input vector is presented, and less likely to win when a significantly different input vector is presented. The ideal number of neurons in the ANN depends on the number of different classes that the data represents, which is information supplied by the user.

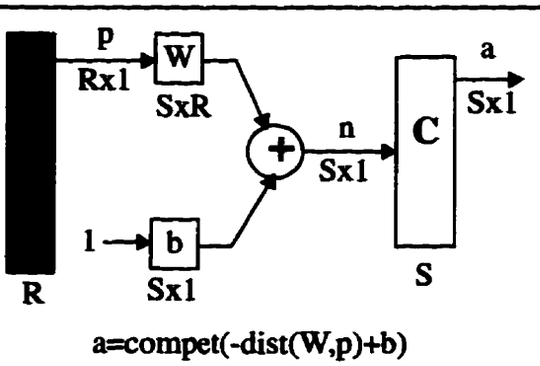


FIGURE 3.10. Architecture of the Competitive network.

3.2.3.2 Data Structures

There are four main variables used. The first is matrix *P* which contains the training vectors. Each column represents an individual training document vector. The second is matrix *W* which contains the weights for the interconnections between the neurons. The

next variable *tp* (training parameters) is a vector which contains the display frequency, maximum epochs, and learning rate parameters. The last set of variables are the testing vectors, p_x where $x \in n$.

3.2.3.3 Operation of Program

The flowchart for the operation of the *simulate.m* program is shown in Figure 3.11. The first step is to load the relevant document vectors into variable *P*. The next step is to initialize the weights for the neurons. This is followed by training the ANN using the competitive algorithm. The trained ANN is then simulated using the new documents. The ANN then classifies the new vectors into the different classes.

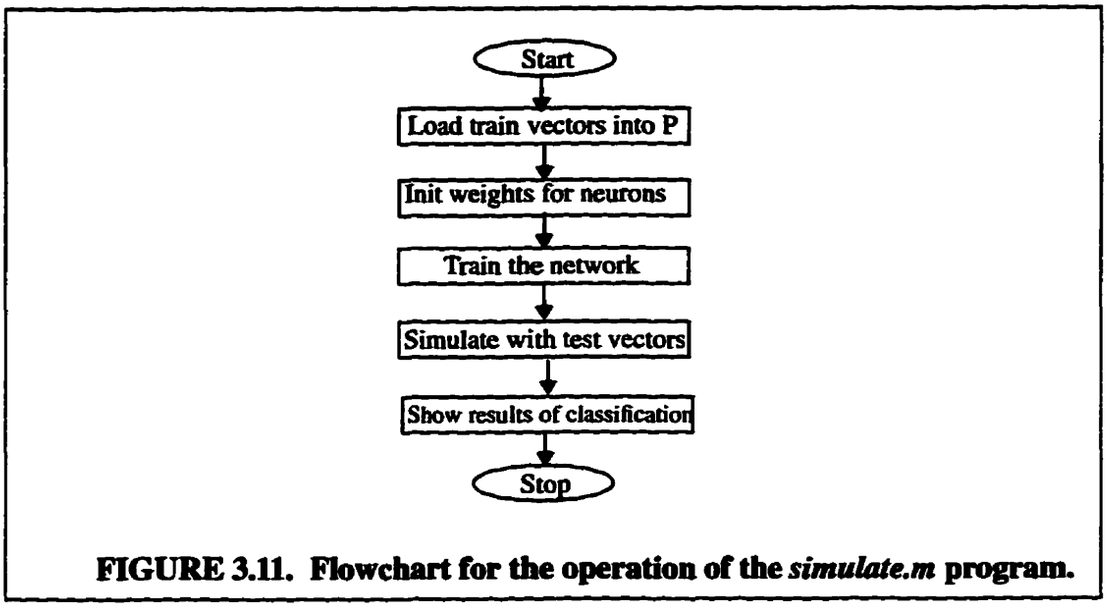


FIGURE 3.11. Flowchart for the operation of the *simulate.m* program.

3.3 Fuzzy Inference System

The purpose of the fuzzy controller is to determine whether the document retrieved by the Web agent should be referred to the user or rejected. The FIS makes a decision using three criteria. The first is *Certainty* which represents the level of belief by the agent that the retrieved information is important. This criteria would be provided by the ANN stage after the new retrieved document is textually analysed and then passed through the trained ANN. However, at this stage of the agent development, the particular measure has not been decided upon. There are three different levels of *Certainty*: Low, Med, and High.

Similarly, there are two different levels of *Query*: Casual and Important. The final criteria is *Responses* which represents the number of responses that the user has requested. There are three different levels of *Responses*: Few, Average, and Many. The last two criteria are supplied by the user when the Retrieval process is initiated.

The Fuzzy controller uses these criteria to determine whether to refer the retrieved document to the user or reject it. Table 3.1 shows how the criteria are used to influence the result of the Fuzzy controller.

TABLE 3.1. Fuzzy controller truth table.[a]

if	CERTAINTY	QUERY	RESPONSES	then	ACTION
	Low	Casual	*		Reject
	Med	Casual	Average		Reject
	Med	Casual	Many		Refer
	Low	Important	Few		Reject
	Med	Important	Average		Refer
	High	*	*		Refer
	Med	Casual	Few		Reject
	Low	Important	Average		Reject
	Med	Important	Many		Refer
	Med	Important	Few		Reject
	Low	Important	Many		Refer

a. The "*" represents don't care condition.

CHAPTER 4

PAW Agent Simulations and Results

"It's kind of fun to do the impossible."

-WALT DISNEY

The PAW agent model described in the previous chapter which includes the relevant URL real-time database, neural document classifier, and the fuzzy inference system are each simulated independently in this chapter.

4.1 Relevant URL Real-Time Database

The program for the database is simulated using a user activity test file similar to that shown in Figure 3.3. Several simulations are carried out for different parameter values as discussed below.

4.1.1 Simulation # 1

The first simulation involves executing the program with the following characteristics for the three relevance fuzzy sets. Refer to Figure 3.6 for the interpretations of the parameters. The parameters are a=0.15, b=0.25, c=0.5, d=0.6, and e=1. Five different trials of the program were executed with these parameters while varying the period from 7 days to 11 days.

The graph in Figure 4.1 shows that as the initial P (period) value is increased, the change in the modification of the new period increases in the negative direction. When the initial

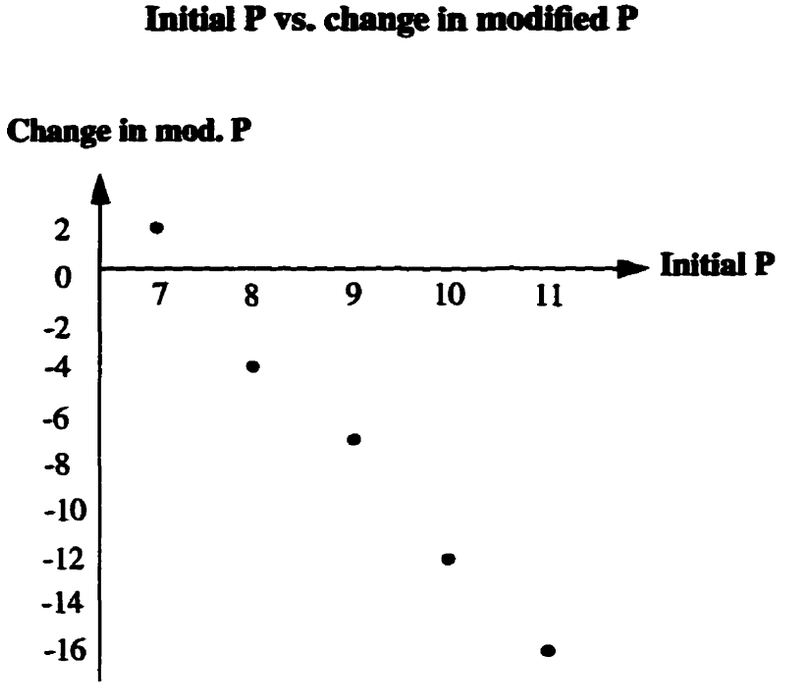


FIGURE 4.1. Results of Simulation # 1.

P increases, it decreases R (relevance) which causes the PPM to fall into the first two PPM guidelines. Thus the next P values are decreased.

4.1.2 Simulation # 2

The second simulation involves executing the program with the following characteristics for the three relevance fuzzy sets. Again refer to Figure 3.6 for the interpretations of the parameters. The parameters are a=0.25, b=0.3, c=0.6, d=0.7, and e=1. This time the relevance fuzzy set have been shifted to the right slightly. Five different trials of the program were executed with these parameters while varying the period from 7 days to 11 days.

The graph in Figure 4.2 again shows that as the initial P value is increased, the change in the modification of P increases in the negative direction. The three fuzzy sets in this sim-

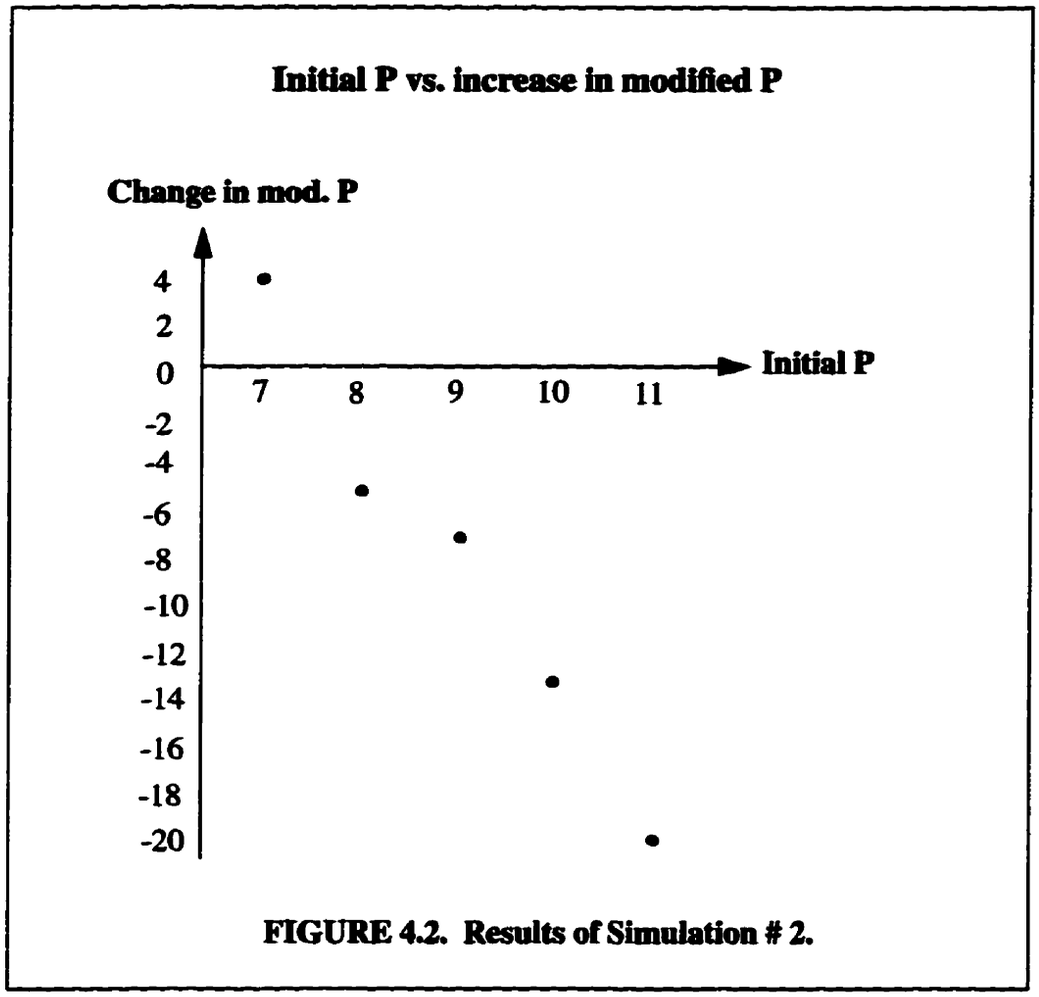


FIGURE 4.2. Results of Simulation # 2.

ulation are shifted to the right, thus there are more chances that guidelines 1 and 2 are met then guidelines 3 and 4 which results in more chances for the next period to decrease then increase.

4.1.3 Simulation # 3

The third simulation involves executing the program with the following characteristics for the three relevance fuzzy sets. Refer to Figure 3.6 for the interpretations of the parameters. The parameters are a=0.35, b=0.4, c=0.5, d=0.55, and e=1. This time the rel-

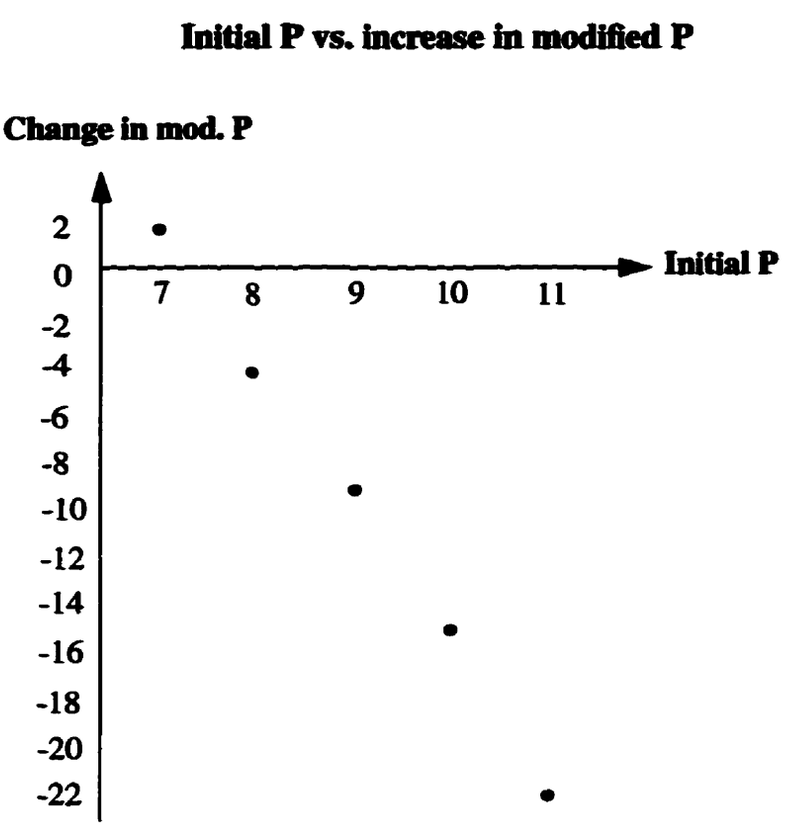


FIGURE 4.3. Results of Simulation # 3.

evance fuzzy sets Low and High have been stretched while fuzzy set Med has been compressed. Five different trials of the program were executed with these parameters while varying the period from 7 days to 11 days.

The graph in Figure 4.3 shows that as the initial P value is increased, the change in the modification of P increases in the negative direction. The *low* and *high* fuzzy sets in this simulation have been expanded while compressing the *med* fuzzy set. This time the range for all five of the guidelines have been increased.

4.1.4 Observations

From the above three experiments, we notice that the degree to which P is modified depends on both the initial value of P and the characteristics of the fuzzy sets. However, the relevance calculations are unaffected by the changes in the characteristics of the fuzzy set as expected. In general, as the initial values of the periods increase, the relevance values decrease. The results of the three simulations are summarized in Table 4.1.

From the results of the experiments, the construction of the fuzzy sets, and the period performance measure (PPM), the following general observation can be made. If the Low fuzzy set is compressed while the Med and High fuzzy sets are expanded then the overall increase in the modification of the period will be maximized. On the other hand, if the Low fuzzy set is expanded while the Med and High fuzzy sets are compressed then the overall increase in the modification of the period will be minimized.

TABLE 4.1. Results of the Simulations

	Simulation # 1	**Simulation # 2**	**Simulation # 3**	
Initial P	**Modified Ps for the 6 Docs**	**Modified Ps for the 6 Docs**	**Modified Ps for the 6 Docs**	**Relevance values for the 6 Docs**
7	7 7 9 7 7 7	7 7 9 7 8 7	7 7 9 7 7 7	.19 .31 .81 .15 .31 .31
8	7 7 9 7 7 7	7 7 8 7 7 7	7 7 9 7 7 7	.13 .21 .66 .13 .24 .21
9	7 8 9 7 8 8	7 8 9 7 8 8	7 7 10 7 7 7	.11 .18 .56 .11 .20 .18
10	7 8 10 7 8 8	7 8 10 7 7 8	7 7 10 7 7 7	.09 .15 .45 .09 .13 .15
11	7 9 11 7 7 9	7 7 11 7 7 7	7 7 9 7 7 7	.08 .13 .39 .08 .11 .13

4.2 Neural Document Classifier

The purpose of the simulation is to train a competitive network with seven neurons to classify documents from seven different categories using eighty training document vectors.

4.2.1 Training Data Simulation

The *train.c* program is executed using a standard stop list obtained from (Rijsbergen, 1979) with only the following additional words inserted: copyright, image, rights, and

reserved. These words are added to the stop list because they are some of the most frequently used words associated with Web documents and do not help differentiate the document content. The program is executed on seventy documents with the maximum number of index terms set to eighty. The training documents that are selected and the results obtained are discussed below.

4.2.1.1 Relevant Document Collection

Documents from seven different topics each with ten samples are used as the relevant document collection. All the documents were taken from the WWW and saved as text files with no editing performed on them. Winston Cup racing documents were taken from the Winston Cup Online website located at *http://www.winstoncuponline.com/*. Movie review documents were taken from the MOVIEWEB website located at *http://movieweb.com/*. The home family medicine documents were taken from the Canadian Medical Association website located at *http://www.cma.ca/*. The modern aircraft statistic documents were taken from the US Air Force Museum located at *www.wpafb.af.mil/museum*. The 1996 Olympic games documents were taken from the CNN - The Atlanta Olympic Games website located at *http://cnn.com/SPORTS/OLYMPICS/*. Finally, the music group and software agent documents were taken from various other websites.

4.2.1.2 Results of Simulations

The program generates the *names.dat*, *terms.dat*, and *nnet.dat* files. The first file records the seventy files which are processed along with their names. The second file records the top eighty words that are selected as the index terms. The last file records the binary document vectors used for training the ANN. Each row is for an individual training document vector while the columns represent the index terms.

4.2.2 Testing Data Simulation

The *test.c* program is executed using the *terms.dat* and *nnet.dat* files generated from the previous program. The testing documents that are selected and the results obtained are discussed below.

4.2.2.1 New Test Data Document Collection

The new test document collection is gathered using the EXCITE (tm) search engine to collect ten documents for each of the seven topics. The category names are used as queries for the search engine to retrieve the new documents.

4.2.2.2 Results of Simulation

The program generates the *names2.dat* and *nntest2.dat* files. The first file records the seventy files that are processed along with their names. The second file records the binary matrix used for testing the competitive network. Each row is for an individual test document vector while the columns represent the index terms.

4.2.3 ANN Simulation

The ANN program is executed using the train and test document vectors generated from the previous two programs. The independent parameters are set as follows: learning rate (lr) is 0.01, maximum number of epochs (me) is 4000, and the frequency of progress display (df) is 1000.

4.2.3.1 Results of Simulation

The MATLAB screen output for the program indicates the update status of the training after every 1000 epochs. After the network is trained, the original training data and the new test document vectors are presented to it one at a time. The program then displays the class to which the vectors belongs.

Table 4.2 and Figure 4.4 show the results produced by simulating the trained network with the original training data and the new test data. In order to evaluate the trained network in categorizing new documents, an accuracy measure is defined as

$$Accuracy = \frac{1}{p}\sum_{i=1}^{p}\left[\frac{\frac{correct_i}{total_i} + \left(1 - \frac{wrong_i}{p}\right)}{2}\right]$$

where p is the number of categories, $correct_i$ is the number of correctly classified documents in $class_i$, $wrong_i$ is the number of misplaced classes present in $class_i$, and $total_i$ is the total number of documents in $class_i$.

The above equation shows that the accuracy of a particular class depends not only on the number of documents that were correctly placed in the class but is also inversely proportional to the number of misplaced documents. The overall accuracy of the system is the average of the individual class accuracy values.

TABLE 4.2. ANN simulation results.

Categories	*Accuracy Original Training Data*	*Accuracy New Test Data*
1) Winston Cup Racing	80%	100%
2) Movie Reviews	93%	100%
3) Music Groups	74%	78.5%
4) Olympic Events	100%	42%
5) Home Family Medicine	40%	50%
6) Software Agents	89%	100%
7) Modern Aircraft Statistics	100%	87%
Overall Accuracy	82%	80%

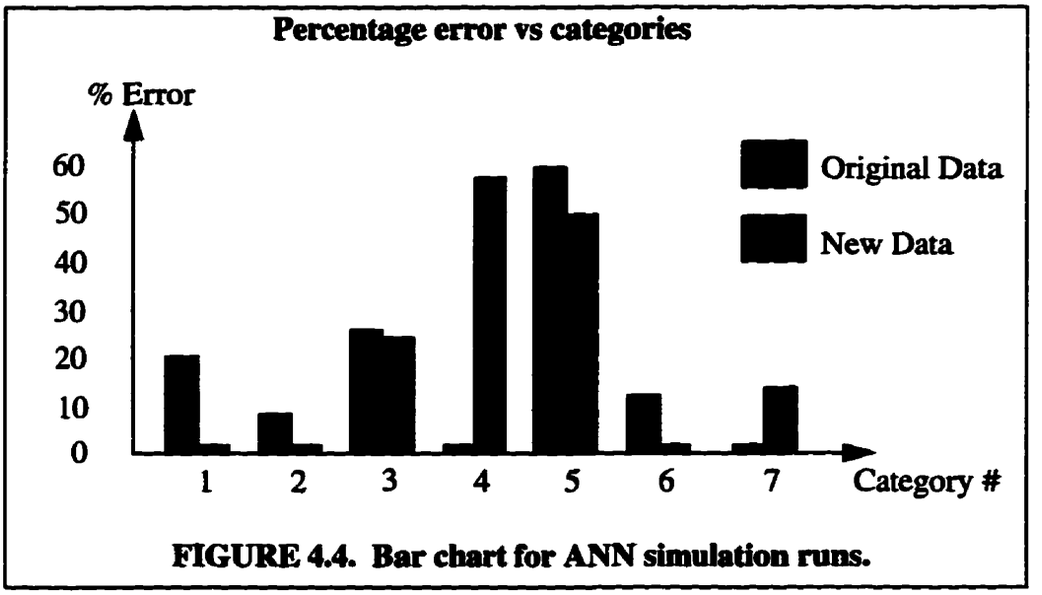


FIGURE 4.4. Bar chart for ANN simulation runs.

Figure 4.5 shows the learning convergence for the new test data. The competitive learning algorithm starts off with a relatively large overall accuracy error (29%) at 1000 epocs and becomes constant (20%) after 3000 epocs.

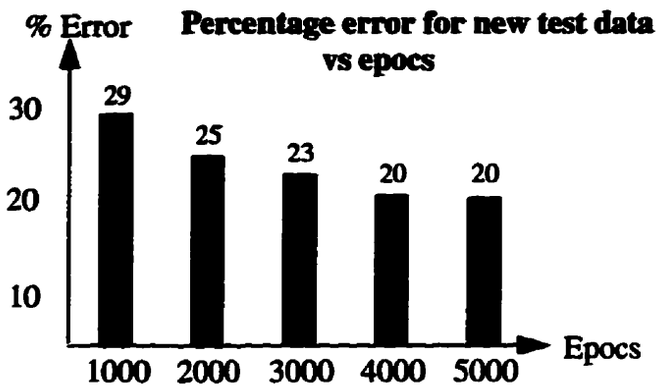


FIGURE 4.5. Bar chart for learning convergence.

4.3 Fuzzy Inference System

The purpose of the FIS simulation is to verify whether the truth table for the FIS presented in Table 3.1 is a reasonable design as the document acceptance/rejection controller.

4.3.1 Results of Simulations

All figures referenced in this section are appended in Appendix C. The FIS for the Web agent controller is shown in Figure C.1. It shows that the FIS has 3 inputs (CERTAINTY, QUERY, and RESPONSES), 1 output (ACTION), and is composed of 11 rules which are represented by the black box. It also shows the Fuzzy operators which are a part of the FIS and the methods used. The only operator that is not used in the Web agent controller is the "OR". The other fields were modified and experimented with to see which combination gave the best results. After conducting, experiments with several combinations, the following choices were adopted. The AND method was "prod", IMPLICATION type was "min", AGGREGATION type was "max", and DEFUZZIFICATION was "centroid".

Each of the 3 inputs have a range in the interval [0,10]. Figure C.2a shows the three membership functions for the CERTAINTY input using bell-shaped membership functions. Figure C.2b shows the 2 membership function for the QUERY input. Figure C.2c shows the 3 trapezoidal membership functions for the RESPONSES input. The output variable

also has a range in the interval [0,10]. Figure C.2d shows the 2 triangular membership functions for the ACTION output.

The 11 IF-THEN rules for the FIS are shown in Figure C.3 and correspond to the rules that were established in Table 3.1. Figures C.4a and b show snapshots of the 11 rules of the FIS along with the final results for *casual* and *important* queries respectively while fixing the CERTAINTY to *med* and RESPONSES to *average*. The results of these two snapshots are consistent with the desired results.

Figures C.4a and b show only one calculation at a time (i.e. a microview of the FIS). In order to see most of the output surface of the FIS, Figures C.5 to 8 have been generated. These plots show the 3D and corresponding 2D behaviour of the FIS for different input combinations. The ACTION output variable indicates whether the retrieved document is to be referred to the user or rejected. If the value of ACTION is greater than or equal to 5 then the document is referred, otherwise, it is rejected. Note that the ACTION output variable also indicates the degree to which the document is referred or rejected.

Only 2 input variables and 1 output variable can be shown at one time. Thus the QUERY input is kept fixed while the CERTAINTY and RESPONSES inputs are varied for each plot. Figures C.5a and b show the 3D and 2D plots respectively for the case when the QUERY input is fixed at a level of 2 (*Casual*). Similarly, Figures C.6a and b show the 3D and 2D plots respectively for the case when the QUERY input is fixed at a level of 4 (less *Casual*). Similarly, Figure C.7a and b show the 3D and 2D plots respectively for the case when the QUERY input is fixed at a level of 6 (*Important*). Finally, Figures C.8a and b show the 3D and 2D plots respectively for the case when the QUERY input is fixed at a level of 8 (more *Important*).

4.3.2 Observations

It is clear from the 3D and 2D plots of the FIS that the greater the level of CERTAINTY and RESPONSES, the greater the chances that the retrieved document is referred to the user. It is also clear that as the level of QUERY increases (i.e. query is important) so does the chance that the document is referred. The results that are obtained are consistent with the design attributes of the individual input and output membership functions and the corresponding rules.

The FIS developed for the Web Agent is suitable for this application because it uses vague information obtained from the user together with the agent to make an appropriate decision. It is worth noting that the form of the membership functions and the rules can be easily adjusted to obtain different characteristics for the FIS.

CHAPTER 5

Conclusions and Recommendations

"Get a good idea and stay with it. Dog it and work at it until it's done, and done right."

-WALT DISNEY

5.1 Discussion

This thesis has presented a design of a Web agent which meets the requirements of being adaptive, personalizable, and able to find new documents which its user is interested. This is achieved by developing three components: (i) a relevant URL real-time Database, (ii) a Neural document classifier, and (iii) a fuzzy inference system decision controller. The PAW agent is made adaptive (i.e. is able to learn by monitoring its user) by using competitive learning. It is made personalizable (i.e. has the ability to tailor itself to its users preferences) by implementing a relevant URL real-time database which monitors the user's activities. The agent accomplishes its delegated task using automatic text analysis techniques and a fuzzy decision controller.

The first component uses concepts and techniques from fuzzy logic to design a real-time database. It keeps track of the URLs that the user has visited in a given time period and decides whether a particular URL should remain in the database (because it is a relevant location for the user) or be discarded (because it has become irrelevant) using a relevancy measure. The simulations for the database indicated that the Period Performance Measure which was developed to determine if and for how long a URL remains relevant produced satisfactory results. This first component was required as a preprocessing stage because it is necessary to separate the useless URLs from those that will be used to train the ANN in the second component. This component could possibly have been implemented without using fuzzy logic, however the fuzzy measure allows meaningful representations to be

associated with its function, which makes it easier to interpret what it accomplishes. As well, the fuzzy measure can be easily modified to give different results by changing the characteristics of the fuzzy sets. The real-time database makes it possible to maintain manageable size for the database, and only relevant URLs are maintained since the other URLs are removed after two consecutive periods of being irrelevant.

The second component uses techniques from automatic text analysis and competitive learning to reduce Web documents into document vectors, and to cluster the document vectors into groups, respectively. The simulation of the automatic text analyser successfully parsed the documents to find the most descriptive words to represent the document collection. The competitive network simulation using Matlab's Neural Network toolbox was also successful in clustering the original and new document vectors with about 80% accuracy. This component allowed the agent to learn the types of documents that the user was interested in. A limitation in the implementation of this component is that the number of categories into which the agent should cluster the documents must be known in advance. The automatic text analysis technique is a good approach because it allows the whole document to be represented by a single vector, rather then using a meta-form technique where only set key words are used. As well, the process of using an unsupervised competitive network for the agent to cluster the documents makes it more autonomous than most other existing information agents.

The third component again uses basic techniques from fuzzy sets to develop a controller which suggests or rejects the new documents that are retrieved. It is based on the agent's level of confidence that the retrieved document is useful for the user (certaintity), the number of responses that the user wants (i.e. low/med/high), and on the type of query (i.e. casual/important). The simulation of the FIS showed that it is very easy to implement a controller using fuzzy logic. The benefit is that the design can be implemented using linguistic terms which are far easier to employ then equivalent numerical methods. As well, this approach gives far more information (i.e it does not simply reveal that the document is accepted or rejected but rather to what extent) and it allows the controller to be easily modified by changing the characteristics of the fuzzy sets.

Is the PAW agent presented in this thesis really an agent? According to Franklin's definition which was stated in Chapter 3, it is indeed an agent and not just a useful software program. The PAW agent (i) is situated in and part of the users machine and the Web environment, (ii) has the ability to sense what its user is doing and to act autonomously on it by finding new documents, (iii) is almost independent of the user or other entities for supplying it input or interpreting its output (the number of categories of documents at this stage of development is provided externally), (iv) acts in pursuit of its own agenda which is to learn the types of documents its user is interested in and to find new ones, (v) dis-

plays current actions which affect its later sensing, which is evident with respect to the periodic document clustering performed by the competitive network, and (vi) acts continuously over some period of time as indicated by the real-time database.

5.2 Recommendations for Future Work

There are several ways in which the PAW agent can be improved or extended. The portion of the software which monitors the user's browsing activity should be implemented and tested. As well, the portion of the software which automatically sends a query to a search engine to retrieve potentially new documents should also be implemented and tested.

For the automatic text analyser component, several different techniques can be further pursued. A larger stop list which is Web-oriented (i.e. contains the most common Web words) should be generated. This will increase the ability of the text analyser to differentiate between Web documents. Perhaps a suffix removal algorithm should also be added to increase the number of stem words that are represented by the document vectors. Other methods that may improve document identification include using continuous valued document vectors instead of binary to allow different weights to be assigned to the terms, other term weighting functions could be implemented to determine how they affect the outcome, and extracting visually significant features from the documents.

For the neural document classifier, a certaintity measure should be defined which indicates to the FIS component the level of belief that the document is relevant (i.e. certainty). The competitive learning algorithm can be generalized, for example to include soft competitive learning and self-organizing maps (Haykins, 1994). This modification allows more than one neuron to participate in learning for each competition. More than one neuron updates its weight after each input pattern is presented. Finally, an accuracy measure should be defined which allows the agent to automatically determine the number of categories (clusters) required without the user supplying this information. This would make the PAW agent even more autonomous.

APPENDIX A

HCPNs for Relevant URL Real-Time Database Component

The figures in this appendix are made reference to in Chapter 3 section 1.

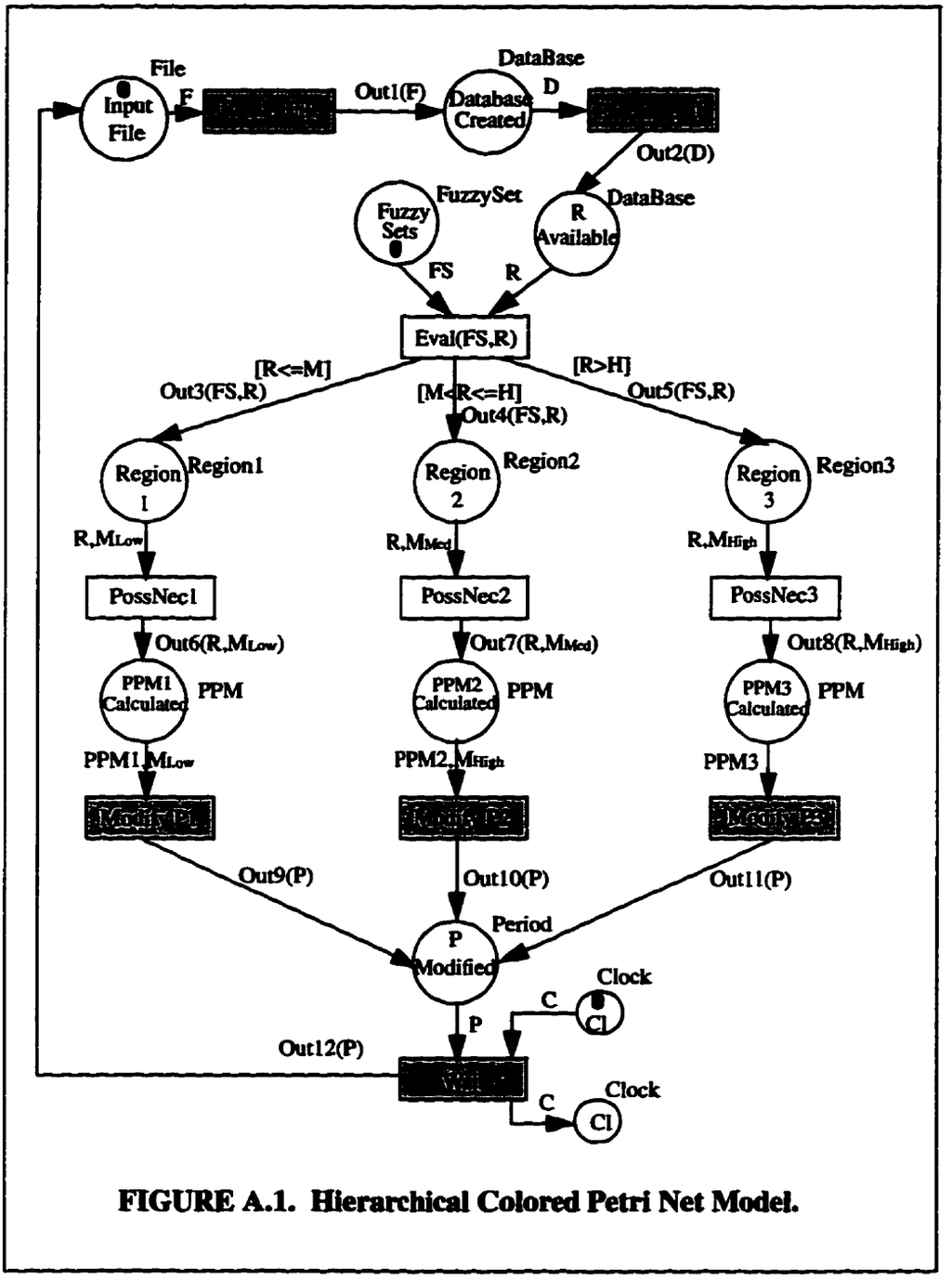


FIGURE A.1. Hierarchical Colored Petri Net Model.

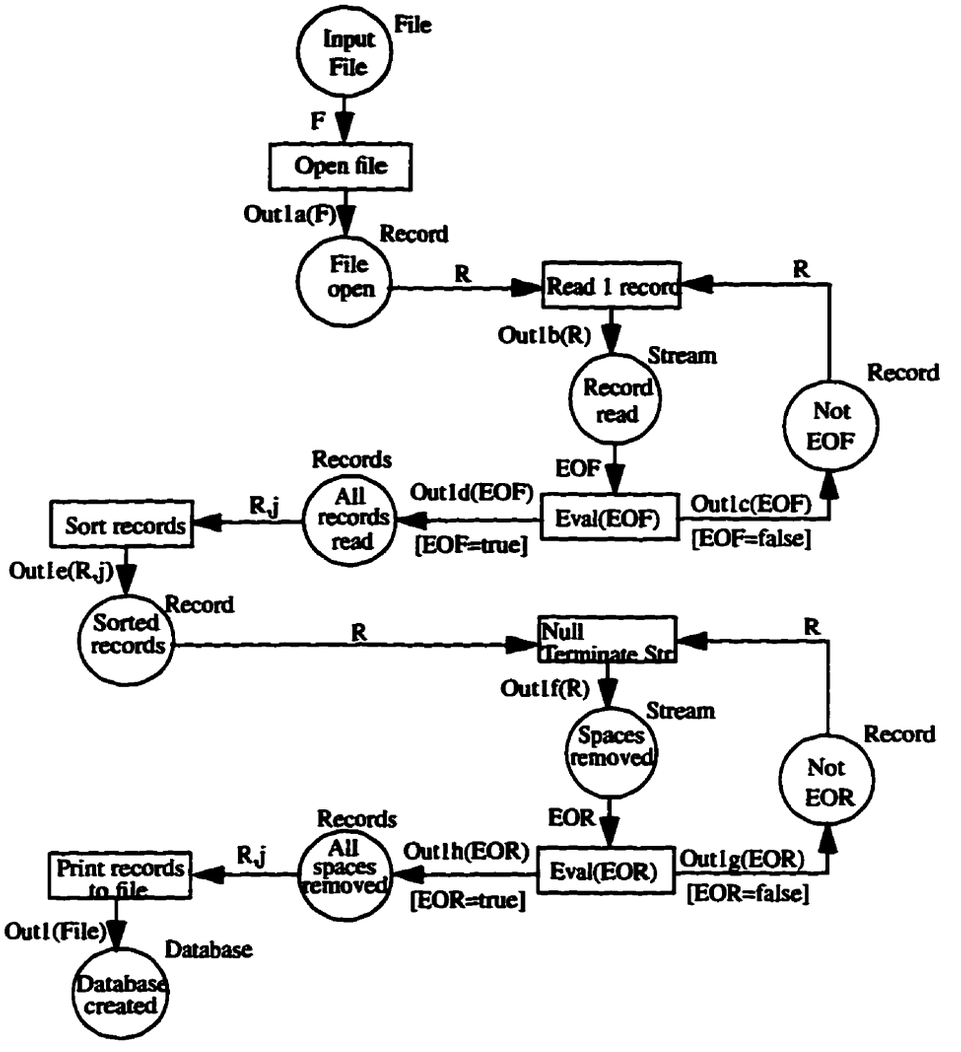


FIGURE A.2. Decomposed Petri net for *Read File*.

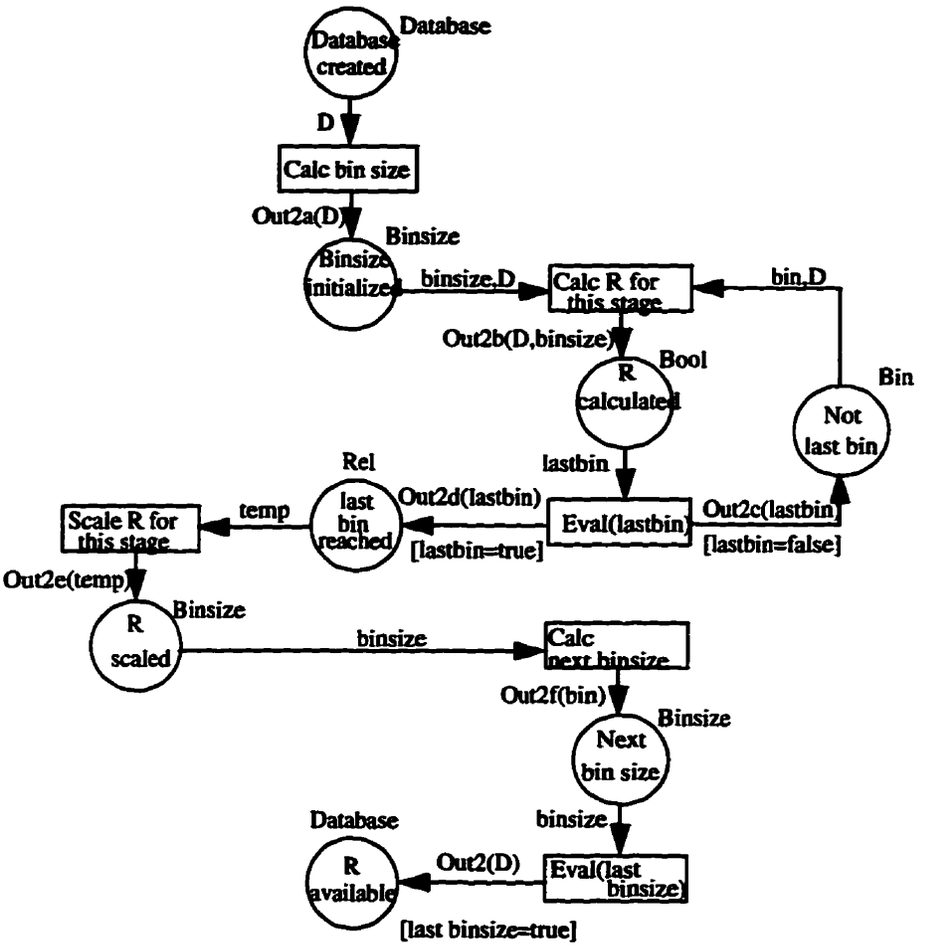


FIGURE A.3. Decomposed Petri net for *Calculate R*.

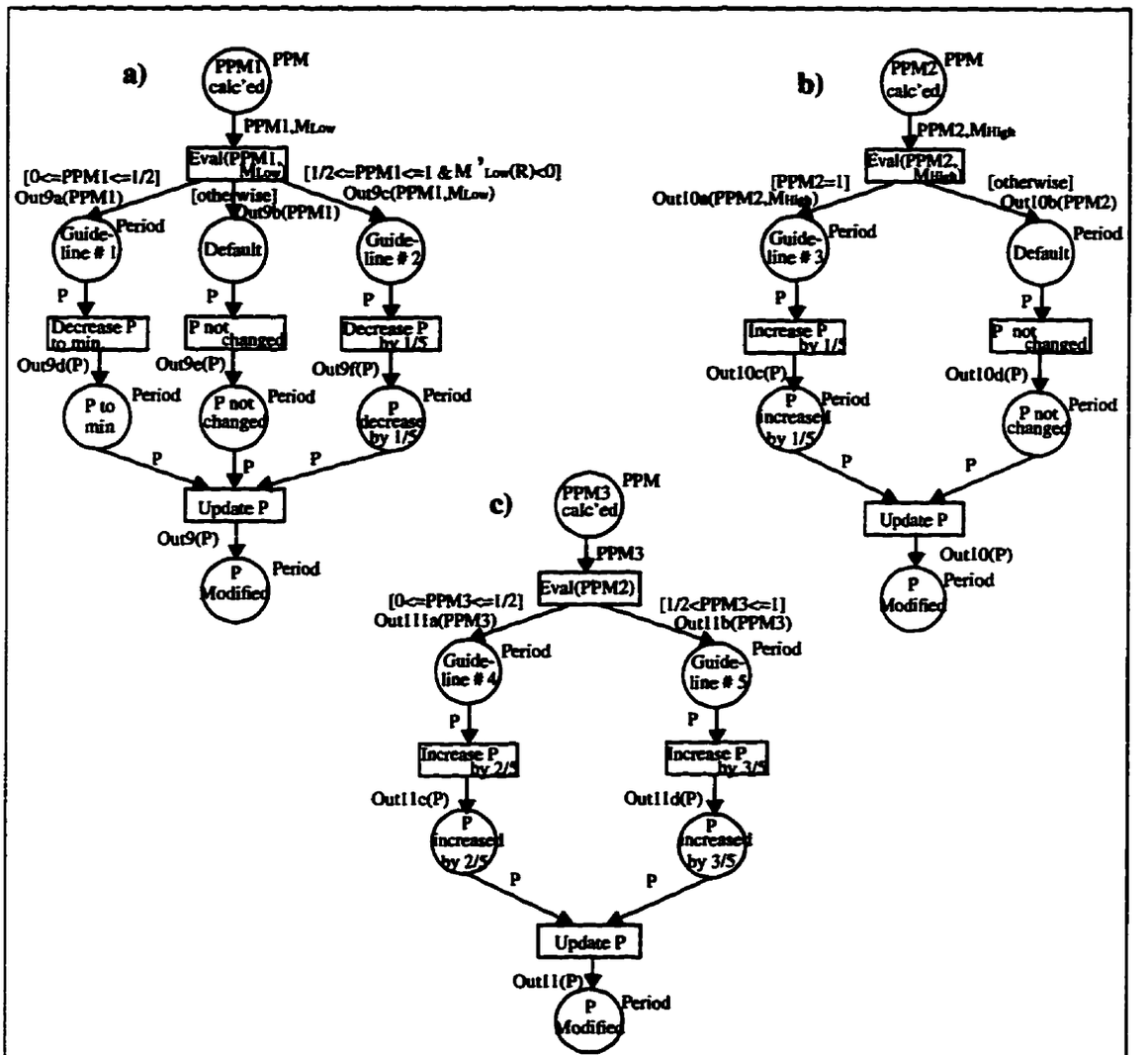


FIGURE A.4. Decomposed Petri nets for a) *Modify P1*, b) *Modify P2*, & c) *Modify P3*.

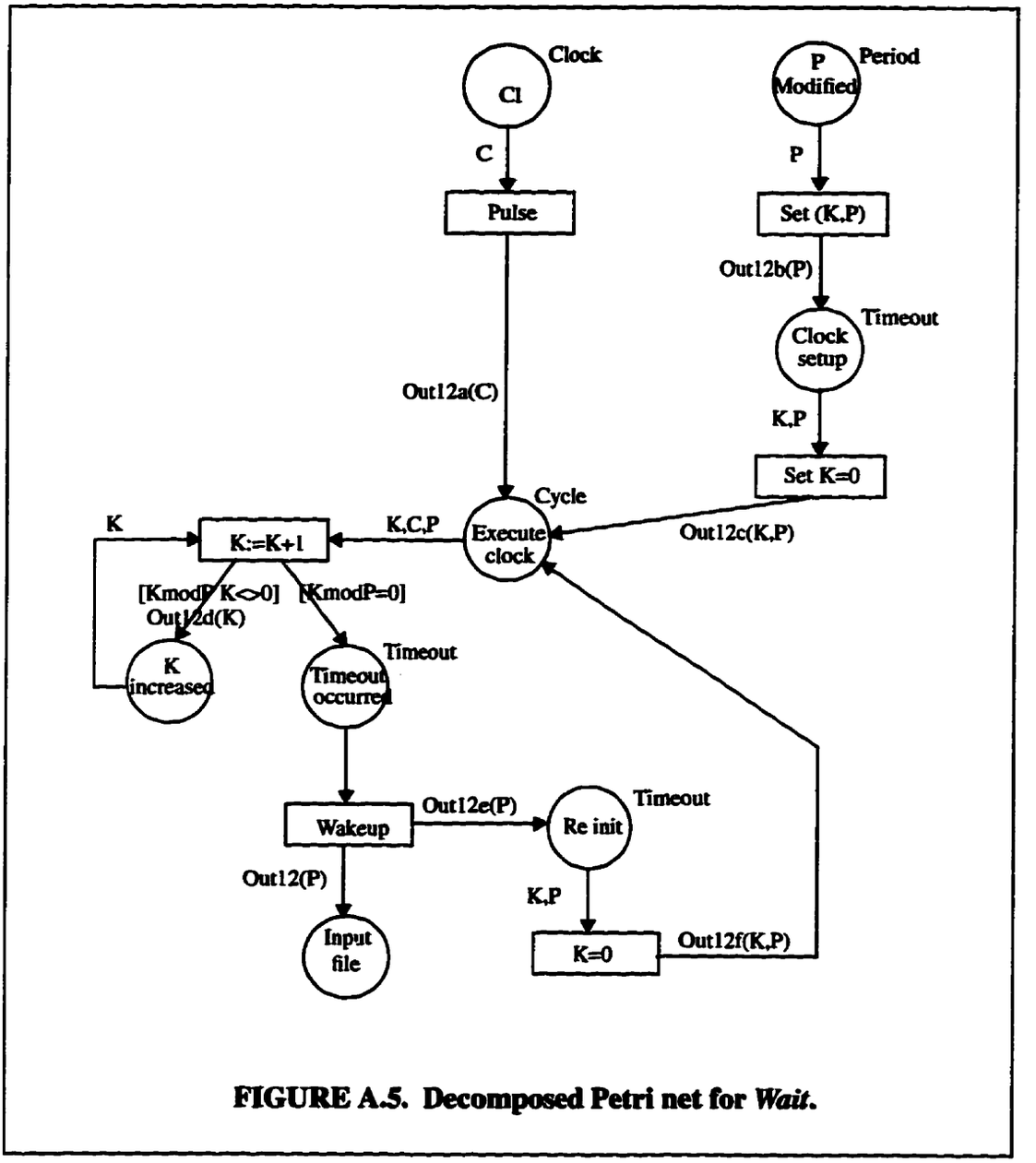


FIGURE A.5. Decomposed Petri net for *Wait*.

APPENDIX B

Neural Document Classifier Component

The figure in this appendix is made reference to in Chapter 3 section 2.

about	being	have	mostly
above	below	hence	much
across	beside	her	must
after	besides	here	myself
afterwards	between	hereafter	namely
again	beyond	hereby	navigation
against	both	herein	neither
all	but	hereupon	never
almost	can	hers	nevertheless
alone	cannot	herself	next
along	could	him	no
already	copyright	himself	nobody
also	down	his	none
although	during	home	noone
always	each	how	nor
among	either	however	not
amongst	else	image	nothing
and	elsewhere	inc	now
another	enough	indeed	nowhere
any	etc	into	off
anyhow	even	its	often
anyone	ever	itself	once
anything	every	last	one
anywhere	everyone	latter	only
are	everything	latterly	onto
around	everywhere	least	other
bar	except	less	others
became	few	link	otherwise
because	first	ltd	our
become	for	many	ours
becomes	former	may	ourselves
becoming	formerly	meanwhile	out
been	from	might	over
before	further	more	own
beforehand	had	moreover	page
behind	has	most	per

FIGURE B.1. Stop list words.

perhaps	there	whence
rather	thereafter	whenever
reserved	thereby	where
rights	therefore	whereafter
same	therein	whereas
seem	thereupon	whereby
seemed	these	wherein
seeming	they	whereupon
seems	this	wherever
several	those	whether
she	though	whither
should	through	which
since	throughout	while
some	thru	who
somehow	thus	whoever
someone	together	whole
something	too	whom
sometime	toward	whose
sometimes	towards	why
somewhere	under	will
still	until	with
such	upon	within
than	very	without
that	via	would
the	was	yet
their	well	you
them	were	your
themselves	what	yours
then	whatever	yourself
thence	when	yourselves

FIGURE B.1. (cont'd) Stop list words.

APPENDIX C

Fuzzy Inference System Component

The figures in this appendix are made reference to in Chapter 3 section 3.

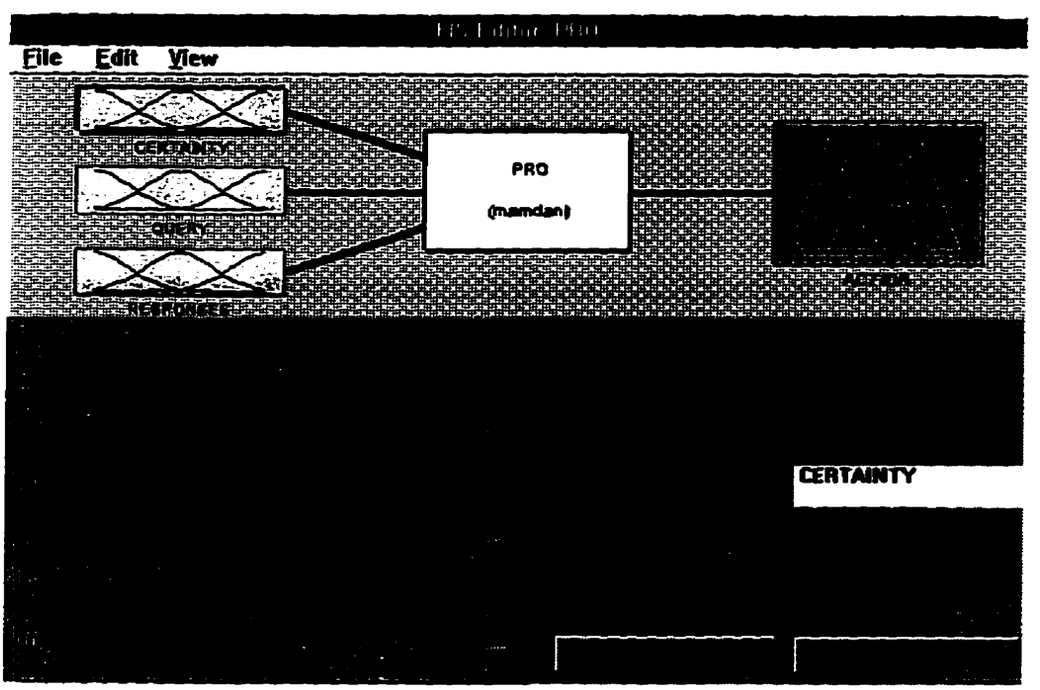


FIGURE C.1. Block diagram for FIS.

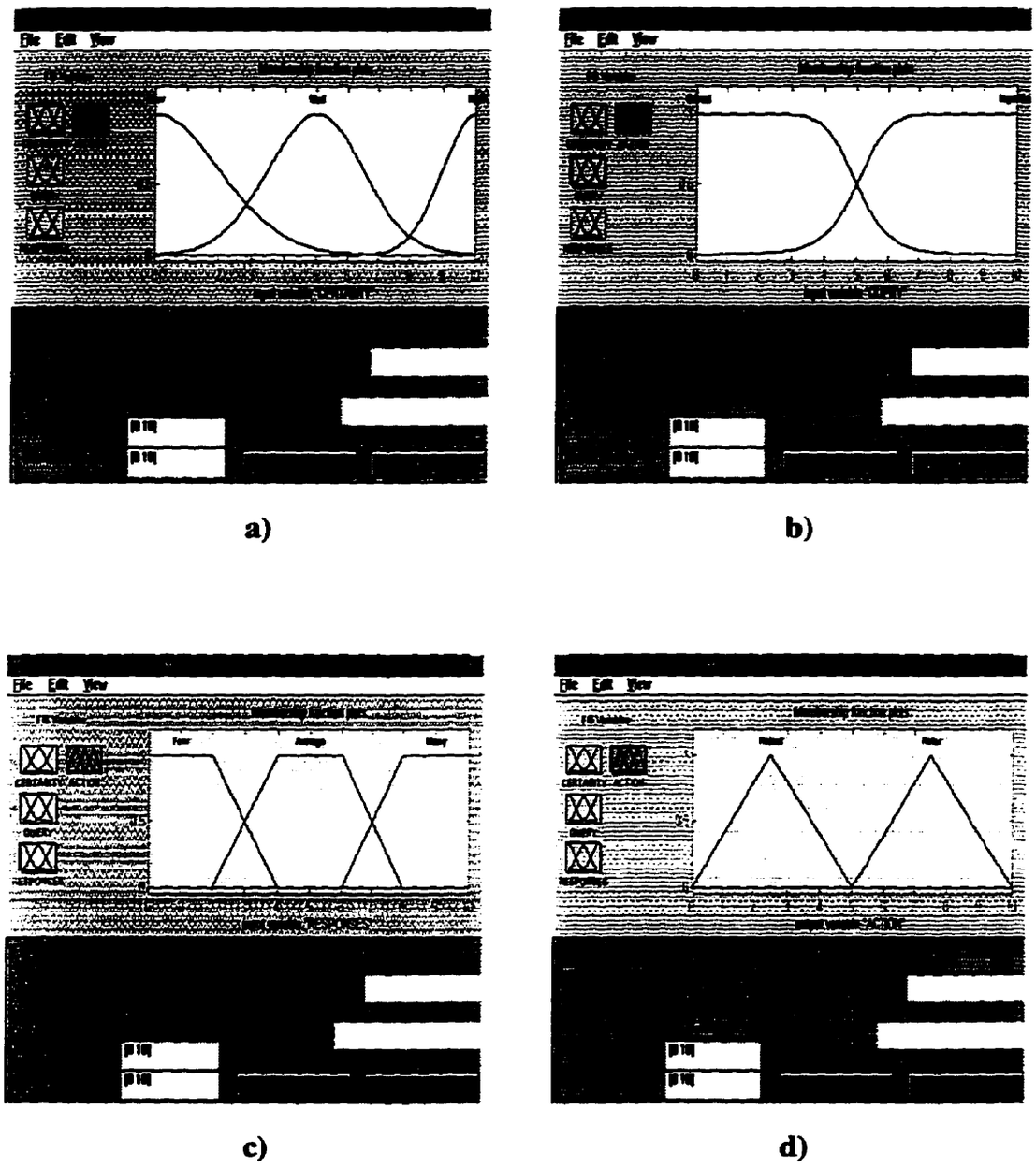


FIGURE C.2. Membership functions for inputs a) *certainty*, b) *query*, c) *response*, & ouput d) *action*.

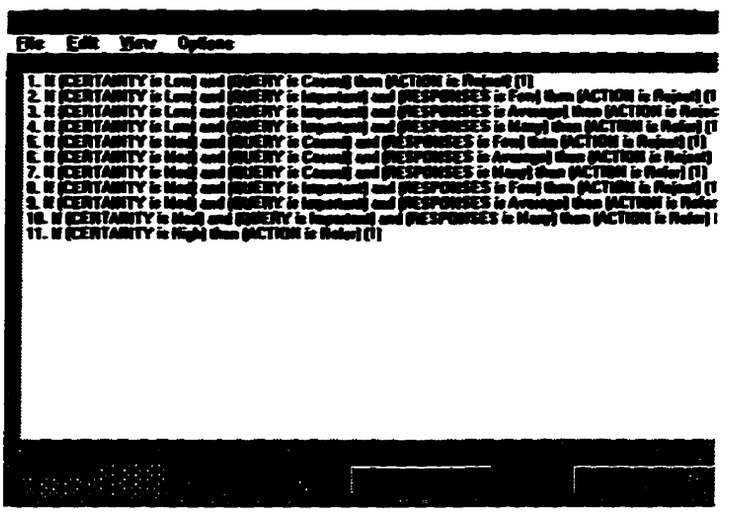
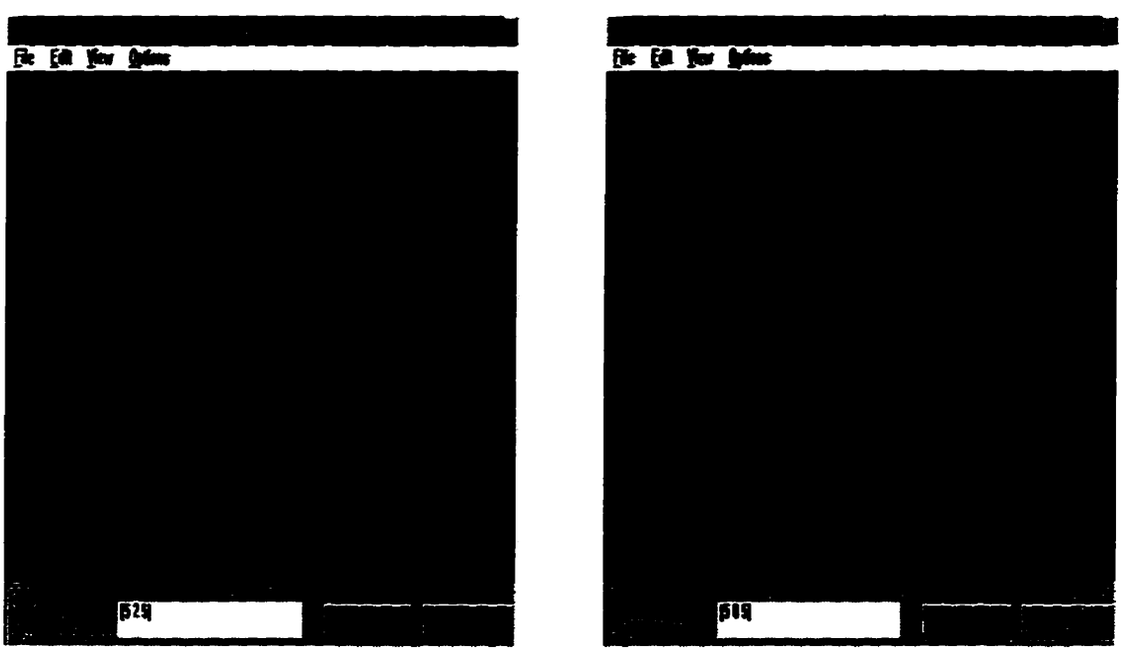
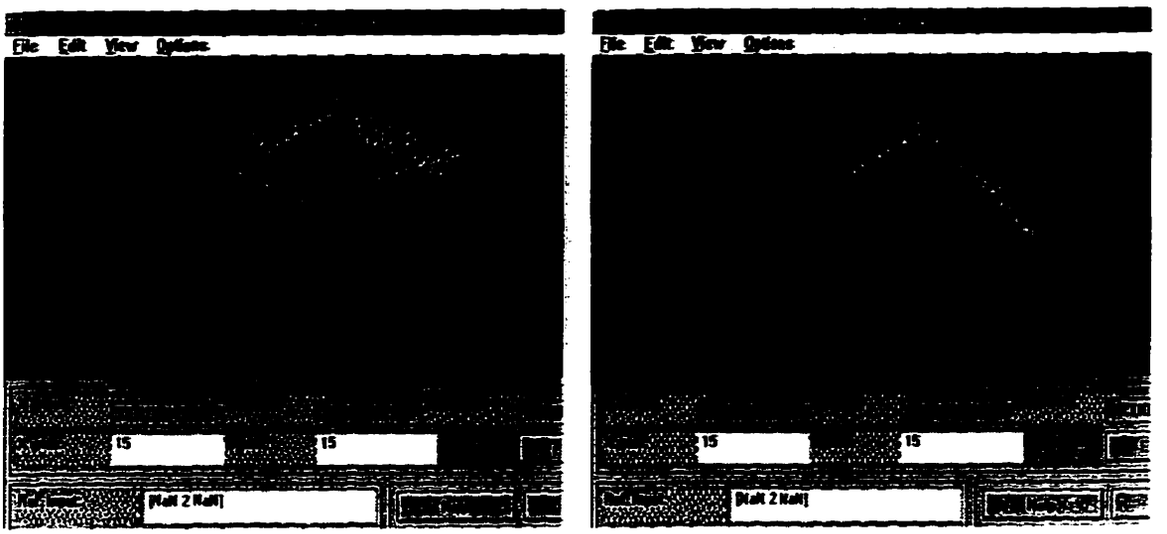


FIGURE C.3. IF-THEN rules for the FIS.



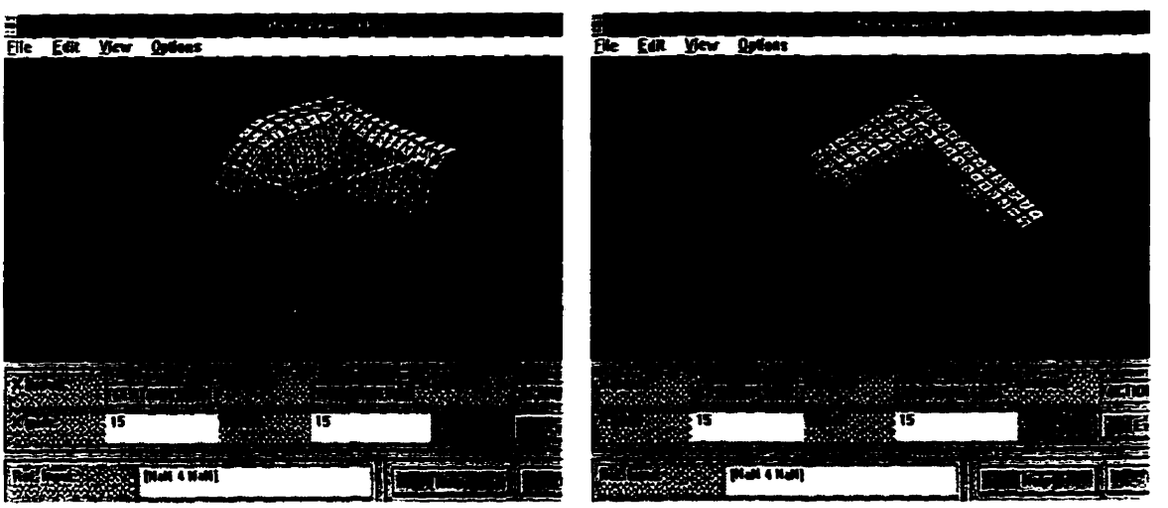
a) *b)*

FIGURE C.4. Snapshot of the IF-THEN rules for a) *casual* & b) *important* queries.



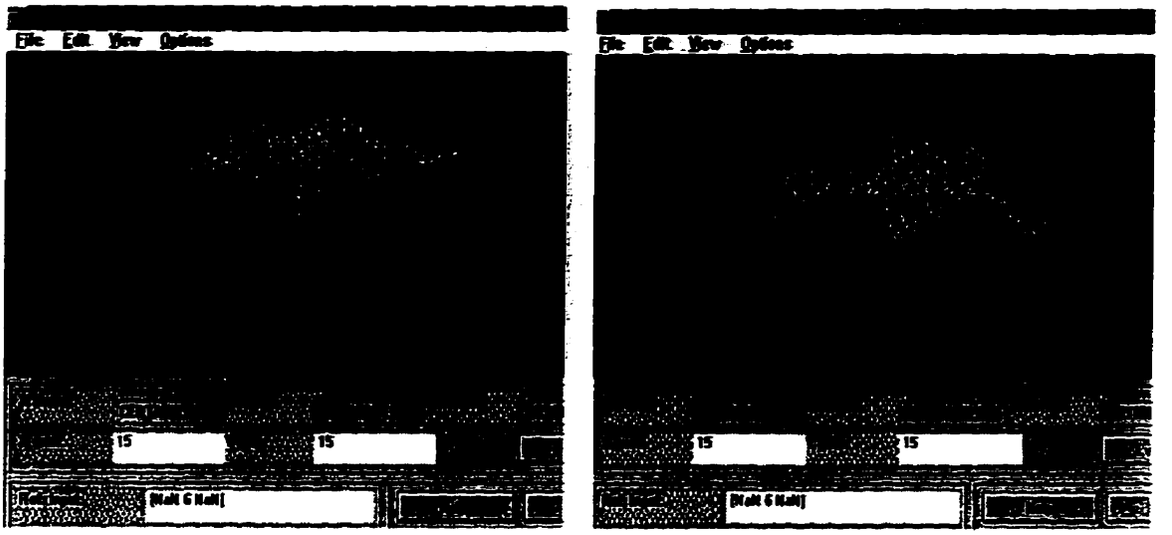
a) b)

FIGURE C.5. a) 3D and b) 2D FIS response for QUERY input level of 2.



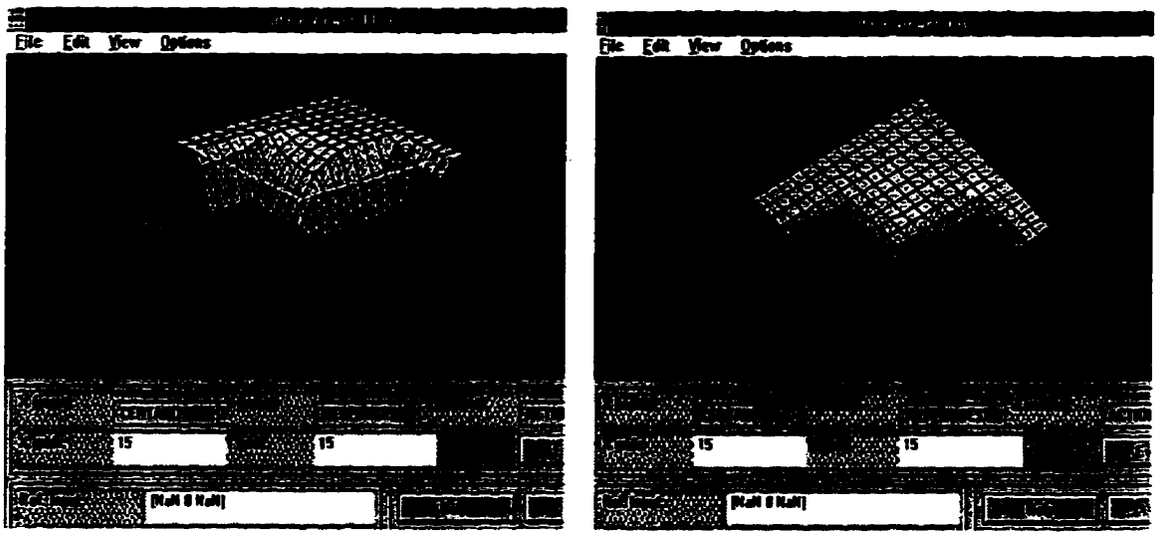
a) b)

FIGURE C.6. a) 3D and b) 2D FIS response for QUERY input level of 4.



a) b)

FIGURE C.7. a) 3D and b) 2D FIS response for QUERY input level of 6.



a) b)

FIGURE C.8. a) 3D and b) 2D FIS response for QUERY input level of 8.

APPENDIX D

Scope for Agent Implementations

This appendix emphasize the scope of agent-oriented applications, by suggesting how agents be applied to an education system. Agents can be implemented in most systems to enhance performance or reduce information overload. To show that agents need not only be associated with robots or the Internet, a brief explanation on how agents could benefit an education system is proposed in this appendix. This agent system represents students, professors, and others involved in a typical education environment with personal student or educator agents respectively. The personal student agents learn the best approaches and conditions to teach their clients. This allows the students to learn the prescribed material more effectively. The personal educator agents are used to assist the educators in teaching the material more effectively.

D.1 The Education Problem

An education system is meant to prepare or enhance the skills of students for the real world. One of the motives for education is to prepare for a career. People desire high salary jobs with a future. As society moves towards a more technical way of life, the need for core knowledge in a specific field along with applied information technology skills are becoming essential.

It seems that the need for a good education and the ability to learn and communicate reasonably are prerequisites for a successful career in most fields. If this demands a proper

education, is the current education system able to provide us with it? This is a simple question to ask but rather difficult to accurately measure.

In my opinion, the education system is inherently too large and static to be able to fulfil the needs of all the students at a given time. We should not expect it to meet everyone's needs completely. Instead, the education system should be designed so that the majority of students benefit from it. Thus, one might attempt to identify how well the education system works for different types of students. Identifying these subgroups and their respective quality of education might help to answer the above question.

However, even without referring to the quality of education, it is safe to assume that for most education systems, there is at least some room for improvement. These improvements might be required in both the curriculum and in the methods by which the material is being taught.

The present discussion addresses the second type of improvement within the student-educator relationship. The main issues in improving how the material is taught are the ability of the educator to teach, the ability of the student to learn, and the student to educator ratio. It seems intuitive that a smaller student to educator ratio would be more desirable because the educator could better understand the student and provide a more personal teaching approach. The ideal situation would be a personal tutor for each student. This however, is not feasible.

Figure D.1 shows the components of a proposed adaptive education agent system which

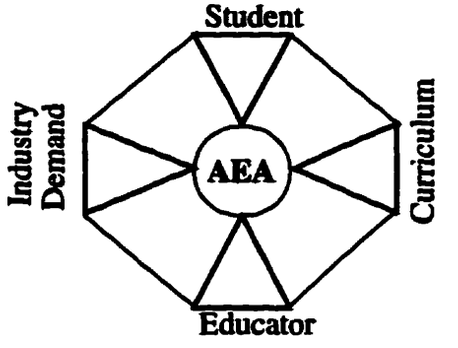


FIGURE D.1. Block diagram for the adaptive education agent system.

may help to enhance the current education system. It combines the curriculum with input from students, educators, and industry to improve the learning and teaching skills.

D.2 Adaptive Education Agent System

One method by which to redesign the education system in order for it to perform better (i.e. provide better education to students, be more responsive to current industry needs, allow informative feedback, and allow personalization) involves the development of an Adaptive Education Agent (AEA) system. This agent system is complementary to traditional teaching methods. The framework for the AEA system is briefly explained in the following paragraphs. It is not intended to address all of the design issues, but rather to motivate an agent-user oriented approach.

For this discussion, the term agent is defined to be a software program that assists a user by monitoring events and procedures in order to perform a particular task that has been delegated to it. The associated term adaptive implies that the agent is capable of learning from the user through observation.

Figure D.2 shows the structure of the adaptive education agent system. There are both student and educator agents in this system, which act as personal assistants for their clients. They continuously monitor their clients and make suggestions to the education

system which help improve the learning or teaching performance. The student agent learns the habits and capabilities of its client (through testing) and suggests to the education system how the prescribed material should be presented to the individual student so that it is most effectively comprehended. The educator agent monitors the performance of its client (through feedback) and suggests to the education system how the teaching methods can be improved to more effectively transfer the material to the students.

The core education system consists of six components: the course material, test material, marking procedures, credit assignment procedures, feedback, and a student classifier, as shown in Figure D.2.

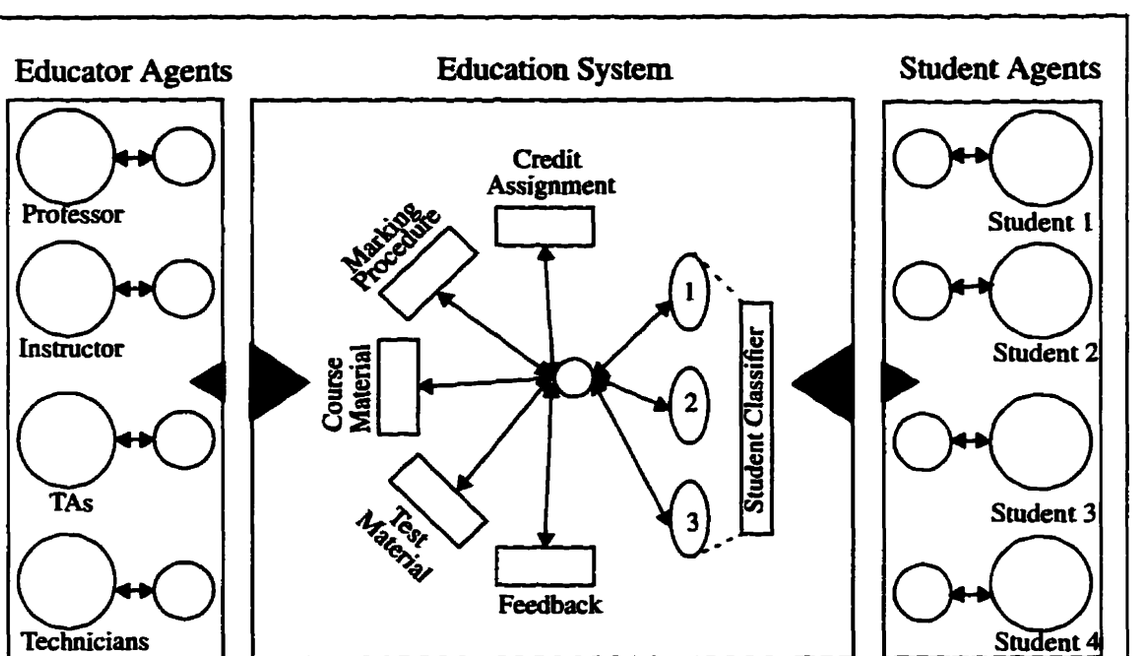


FIGURE D.2. Adaptive Education Agent System.

The student classifier initially looks at the students' profile and categorizes the students into three groups using characteristics such as past performance, level of education, running performance in the course, time spent on the course, individuals expectations of the course, students self-evaluation, and a general questionnaire to gauge their interests. They are separated so that the individual needs of the students can be more easily

implemented. The three categories represent below average, average, and above average students. The course material is divided into levels which represent review, mandatory, and advanced topics. Depending on which category the student is in, the course material is presented to her in a manner which helps her to learn the material most effectively. Similarly, the test material is also divided and different groups are given different levels of tests. In order to be fair to all students, the marking scheme takes into account the different levels of tests. The credit assigned to the students depends on their category and their performance. In order for the education system to be adaptive, both the educators and students are required to give explicit and implicit feedback.

The education agent system recognizes that not all students think and learn alike. It separates the students and conveys the prescribed course material according to their abilities and needs. However, students can change the group that they belong to if they show significant improvement or if their performance drops. As well, the system recognizes that students who excel or learn more from the course should be properly recognized which is reflected in the credit assignment scheme. While separating the students into groups is necessary, it is advisable that this be done privately so that it does not adversely affect the students' self confidence.

APPENDIX E

Taxonomy System for Agents

This appendix presents some initial thoughts for a taxonomy of agents. New software organisms or agents are continuously being created. One of the fascinating and attractive aspects of the agent world is its extraordinary diversity. It seems that almost every possible implementation in task, architecture, physiology (composition), and life style (behaviour) is being created. In order to make sense of these diverse agents, it is necessary to group similar organisms together and to organize these groups in a non-overlapping hierarchical structure.

The discussion in this section does not provide a rigid protocol for an agent taxonomy, nor does it discuss whether a particular software program is an agent or not. Instead, it suggests a biological-oriented approach to systematically identify and classify agents.

E.1 What is an Agent Taxonomy?

Agent taxonomy is the science of agent classification. Synonymous to agent taxonomy is *agent systematics* which is the scientific study of software organisms with the ultimate objective of characterizing and arranging them into an orderly manner. Agent taxonomy can be regarded as the following three processes.

1. **Classification**: arrangement of software organisms into groups based on mutual similarity/evolutionary relatedness.

2. **Nomenclature**: assignment of names to taxonomic groups in agreement with published rules.
3. **Identification**: practical side of taxonomy, the process of determining that a particular isolate belongs to a recognized taxonomy.

E.2 Model for the Agent Taxonomy

Scientists familiar with microbiology might agree that the foundation for an agent taxonomy could be based on a microbial taxonomy. Readers unfamiliar with this topic may refer to (Holt, 1984). In microbial taxonomy (or biology in general), the most general taxonomic group is the kingdom and the most basic taxonomic group is the species. Classification is based on analysis of possible evolutionary relationships (phylogenetic characteristics) or on overall similarity (phenetic classification). Characteristics useful in taxonomy (because they reflect the organization and activity of the genome) are: morphological, physiological, metabolic, genetic, and ecological.

Initially, most of the above terms and concepts may appear to be unrelated to software agents. However, many researchers and software developers are currently attempting to make software agents behave more like biological entities. In which case, a reasonable approach to accomplish it, may involve keeping biological aspects and terminology in mind while developing them.

For instance, characteristics useful in microbial taxonomy might be related to agent taxonomy as follows:

- morphological: structural features of agents such as interface type and size
- physiological: the agent architecture
- metabolic: what type of information the agent consumes/produces and how it processes it
- genetic: the blueprints for the agent which allow them to collaborate with other agents and to produce new improved generations of agents
- ecological: the agent environment which influences its life cycle pattern and symbiotic relationships

In preparing a classification scheme, one places the software microorganism within a small, homogeneous group that is itself a member of a larger group in a non-overlapping hierarchical arrangement. A category in any rank unites groups in the level below it on

the basis of shared properties. As in microbial taxonomy, the following ranks may be employed: species, genus, family, order, class, division, and kingdom.

An appropriate naming convention might follow the binomial system presented by the Swedish botanist Corl von Linne. This assigns to each entity a two part name. The first part is capitalized which is the generic name (associated with the rank it belongs to). The second part is lower case which is the specific name (represents the individual entity). For instance, suppose two software agents were identified to belong to the *mobile* rank then they might be named *Mobilus crawler* and *Mobilus webber.* If later on, the *Mobilus crawler* agent was re-implemented as a *personal* agent then its name might be changed to *Personus crawler* instead. Thus the specific name is stable while the generic name may change.

APPENDIX F

Social Implications of New Technology

This appendix comments on the social implications of new technology in general. It is not sufficient for engineers (including software developers) to concentrate merely on the performance of a product. They also have an obligation to see how that product affects the society at large. In this appendix, the issue of computers in the classroom will be discussed. This is just one example where engineers/developers should contemplate the effect of new technologies on society.

The discussion is motivated by the documentary *Computers in the Classroom: Why some people are worried* which was aired by The National on January 15th, 1997. A brief analysis of it is given which reflects why engineers, developers, policy makers, and society should be aware of the social implications of new technologies.

F.1 Documentary Overview

There are basically two types of supporters for computers in the classroom. Both sides recognise that computers are a valuable tool for students. So where is the conflict? The first group is urging for a fast paced approach where their goal is to have a computer on each desk. The second group is endorsing a more cautious approach.

F.2 Fast Paced Supporters

These supporters are encouraging education officials to purchase billions of dollars worth of computers and to provide on-line access for students that are four years and older. Some of their paraphrased comments are listed below.

- computers serve the curriculum needs; they teach students to read, write, and think
- students (especially young ones) do not find using computers to be a struggle, in fact they enjoy it very much
- technology should not control students, they should use it to create new technology
- the Internet is a good tool because it allows the student to become a producer of information, not merely a consumer
- some students can not afford their own computers, so there must be computers in the classroom (w.r.t. high school students)
- an increasing number of jobs in the workforce demand computer literacy
- learning to use computers is as important as reading
- the Internet gives students access to a plethora of information: in some cases this includes the latest material on that subject
- the current education curriculum must be redesigned to incorporate computers and on-line interaction/learning

F.3 Cautious Supporters

These supporters are for a more conservative approach which introduces computers at all grades but does not require a computer on each desk. They feel that increasing the number of computers at the expense of teachers is not the correct policy. Some of their paraphrased comments are listed below.

- it is costing 4 billion dollars to buy computers for each student in Canada while teachers are being cut
- we might prefer better qualified teachers who get paid more
- there are currently (in the US) 40% of teachers who are not qualified to teach the subject that they are in

- educators think new technologies are automatic solutions to educate students better
- unless there is money to throw away, we should not be spending so much money in acquiring computers
- computers are not necessarily good teachers
- people who spend money to hard-wire classroom with computers treat access to information as though students don't have enough access to information already
- we have to be able to distinguish between useful and useless information on the Web
- teachers should be concentrating on developing fundamental skills such as reading, writing, and comprehension which don't always require a computer

F.4 Analysis

There is no doubt that computers are here to stay. Some of the comments by the fast paced supporters do indeed indicate that computers for each student are essential. Other comments, however, may or may not lead to a helpful environment for students. Even though in some cases students are better educated about computers than their teachers, it seems that educators are the ones that are pushing for the computers and on-line access in the classroom. This lack of education on some of the educators part is creating a stumbling block and a frenzy for teachers to jump on the IT bandwagon whether they understand why they are doing it or not.

On the other hand, most of the comments by the cautious supporters do substantiate that there is no severe urgency to get computers on each desk at the expense of the quality of education. Besides, there is no shortage of computers in the schools since there is already an 11 to 1 computer to student ratio in Canada.

Educators, at the time when radio and then TV became widespread thought that these new media would help students become better students. However, there is no clear evidence that this ever materialized. Computers (including the Internet) may, in the same way, not drastically effect the quality of education for students. Computers like any other tools have to be properly administered to maximize the benefit that they hold. Thus the issue goes back to those who will have to teach the students how to use these new powerful tools, the teachers.

In conclusion, perhaps the important questions to ask are: What is the problem to which the new technology is a solution? If we solve it, what new problems will be created?

References

Abouaissa, H., Rabenasolo, B., Ferney, M., and Vaudrey, P. (1991). *General Models for Representing Reactive and Complex Systems Using Hierarchical Coloured Petri Nets.*

Cheong, F. (1996). *Internet Agents.* New Riders Publishing.

Dibbel, J. (1996). Intelligent agents are changing cyberspace for good. The key ingredients: new technology, clever programming and a bit of Smart Magic. In *TIME Digital Magazine.*

Foner, L. N. (1993). What's An Agent, Anyway? A Sociological Case Study. MIT Media Labs. URL: *http://ftp.media.mit.edu/pub/Foner/Papers/Julia/Agents__Julia.ps*

Franklin, S. and Graesser, A. (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages.* Springer-Verlag.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation.* MacMillan.

Holt, J.G. (1984). *Bergey's manual of systematic bacteriology.* Volume 1. Baltimore: Williams & Wilkins.

Jensen, K. (1992). Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. In *EATCS*, Volume 1. Springer-Verlag.

Pedrycz, W. (1995). *Fuzzy Sets Engineering*. CRC Press, Inc.

Rijsbergen, C. J. (1979). *Information Retrieval*. Second Edition. Butterworth & Company Ltd.

Russell, J. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NY: Prentice Hall.

Salton, G. (1982). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Salton, G. (1968). *Automatic Information Organization and Retrieval*. McGraw-Hill.