Mike Minogue & Cameron Church
University of Guelph
Guelph Ontario
N1G 2W1

September 18, 2006

Mr. Shawki Areibi
University of Guelph
Engineering Dept.
Guelph Ontario
N1G 2W1

Dr. Areibi,

Please accept this report as an initial progress report for our research into intelligent agent technology and it's application in the travel industry. The report covers our progress related to literary searches and improvements in our design scope.

Should you have any questions concerning the work contained within this report, please feel free to contact either of us and we would be happy to help clarify it for you.

Sincerely,

Mike Minogue & Cam Church
Engineering Wannabes

Intelligent Agents
An Interim Report

Prepared by Cameron Church & Mike Minogue

Prepared for Dr. Shawki Areibi

October 16, 2000

Executive Summary

After extensive literature research, major design decisions have been made as to where the design is going and how it will get there. Throughout the literature, very little has been found directly related to travel agent technology. A few attempts have been made to bring this technology into a field that could make good use of it, have been futile, but with the information that has been found on intelligent agents in general, good headway has been made into finalising the design scope.

Through the decision matrix, it was decided that a web-based, average person relationship agent would be created to fulfil the design requirements. The learning aspect of the agent will be implemented through the use of statistical analysis, which assigns weights to a variety of possible outputs and thus returns the highest weighted solution back to the user. One major technical hurdle was overcome, by deciding that Visual Basic will be the programming language of choice for the group. Major technical issues still to be decided include the data structure of the data set. This will be an iterative process that will be build simultaneously with the programming of the agent.

The group has found that without an iterative process the final result will not resemble the original goal of the project. There are such a large number of possibilities that the group has already experienced variations from the project goal. Retuming to the original goal will have to be ensured through repeated review meetings in which a tally of where the group is and where the group should be are evaluated and subsequent to these gap analyses corrective actions will be taken to return to the original goal of the project.

The next step that the group will be taking is the implementation of the design decisions that were arrived at as a result of this exercise. This entails the beginning of the implementation of the programming, as well as the construction of the database that the agent will sift through.

Table of Contents

List of Tables

Introduction

Problem Statement

Intelligent agent technology is still in the budding stages of becoming a viable resource to a wide array of people. They can perform searches geared to the person using the agent, but more importantly they can learn on every return to the user how to improve their searching to return only useful data to the user. In this context, an intelligent agent in the travel industry could greatly reduce search time for anybody. The agent could actually replace the travel agents of today. Based on the teaching and past history the agent has collected on the user, it would begin to return more and more useful travel tips for the user.

This design will encompass the agent into route building for road travellers. This is important to clarify as the scope can easily grow beyond the time constraints of the semester. The data set will have to be created to hold all the possible routes and based on the input from the user, the route can be planned by the agent. A web-based interface will be developed using statistical analysis of past uses to determine new routes. Visual Basic will be used to program the agent and an average person relationship will be used for teaching.

Objectives

An intelligent agent that encompasses all the criteria used to evaluate agents from search engines. This includes autonomy, the ability to learn a user's profile and robustness through repeated use. Further to these objectives, the agent will return routes for the user to peruse that will include possible stops, hotels, attractions, etc. Basically everything that would be supplied by a live travel agent. Of course these will be limited to the data held within the database that also must be created. Therefore, the more complete the data base is, the more lifelike the agent will be and the more applicable its development will be toward widespread usage.

Assumptions

The group assumes that access to a workstation, in which appropriate software is loaded, will be available. The data set will be restricted to contain only the information that is necessary (i.e. as listed in section titled Data Set).

Criteria of Evaluation

- The agent should return possible solutions as quickly as possible.
- The Graphical User Interface should be aesthetically pleasing.
- Robustness should be maximised.
- The interface should be intuitively laid out to ease the user interface.
- The agent should be as functional as possible.
- The data set should be as complete as possible to provide a more life like application of the agent.

Background

The intelligent agent that will be produced must at a minimum return a viable set of solutions as defined by the inputs of the user. This entails an adequate amount of training provided by the user, as well as a concise set of inputs for each trial.

Research into intelligent agents has been becoming more and more prevalent in the computing industry. Articles dealing with this problem have been found, but only encompass the theory behind the technology and do not list any important work that has been completed. The closest approximation of this specific aspect of intelligent agents is named Virtual Mattie. VMattie is more clerical in nature but is a good basis for evaluation as to what an agent should be capable of on its own. VMattie exhibits features such as autonomy/intelligence; initiative and the creators are planning on implementing creativity into VMattie's logic. But as of the writing of this report, VMattie is capable of sifting email, scheduling, reminders and

updating mailing lists. The most interesting aspect of the future work that is being looked at is the implementation of guilt for not sending notices on time and anxiety for not understanding a concept. It can be safely said that VMattie is at the forefront of intelligent agent technology and implementation.

As for intelligent agents in the travel industry, it is hoped that travel agencies will use these agents to obtain optimum packages for its customers, in lieu of the private user having to own the agent. This is of importance as large amounts of travel data are presently set up for use by travel agents, that the general public does not have access to. A very early version of an intelligent travel agent can be found at travelocity.com, but this is more of a large search engine with extensive filters that return the best possible solutions based on extensive inputs by the user, and most importantly, travelocity does not learn the habits of the user unless a registration form is filled out and updated on a regular basis.

The advantage to going to a system based in intelligent agent technology is its speed in problem solving. Add the idea of using multiple agents all sifting for the best solution and conversing during that search, the possibilities are almost endless as to what may be found.

Constraints

- The size of the data set will be restricted to a size that can be manipulated and maintained by the two group members.
- The complexity of the agent will but minimised such that the programming will be within the grasp of the group members.
- The time requirements of the project will be the key to the success of the project.

Method

Design Ideas

The following is brief descriptions of the design alternatives that we have been considering. Although the scope of the solution is in the form of an agent, the blocks that make up the agent are up for consideration. For simplification, however, these blocks can be classified for comparison.

Human-Agent Interaction

Because the agent is an "assistant" to its human operator, the interaction (the inputs into the system) is extremely important. The goal is to minimise ambiguity and confusion that will allow for direct efficient data collection. Ease of development is also important since resource allocation must be maximised in creating the agent's core and not on its interface.

The choices for this block are:

Web Interface
The more the Internet becomes a viable tool in information sharing the more users will become a part of the system. This means that a web browser is becoming a standard operator interface in application development. By utilising the browser we only need to develop the connection between the browsers and the agent. Using existing protocols that connect a browser to a web server can accomplish this task. This can also set up a client-server design philosophy that has it's own benefits such as easy maintenance, platform independence, centralised upgradability etc. The drawback to this option is that a physical connection between the client and the server must be established. This can be overcome by introducing the server on the same machine as the client but that increases the overhead and resource consumption.

Application Interface
This is an interface that is developed along with the program code. It would be equivalent to developing the role the browser plays in the Web Interface option. This interface is dependent upon the programming language that is chosen to develop the core language. That could mean anything from developing the interface using TCL/TK for applications developed in C, C++ or Java, to using the built in functions of Visual Basic. The pros and cons of the different programming languages will be discussed shortly. The

advantages to this approach are that the interface is integrated right into the core of the agent. It results in a stand-alone package. However it contradicts the advantage of centralised upgradability that the Web Interface offers us.

Command Line Interface
The above mentioned options are both graphical user interfaces (GUI). This agent could be done using a command line interface where input and output are based on the computer's I/O stream. This offers the benefit that the interface doesn't need to be developed. Any user information can be inputted via the operating system's command prompt after the request is outputted to the screen by the I/O stream. However, this lacks any aesthetic appeal and requires the programmer to be very specific about the input that needs to be entered. It is not user-friendly.

Data Set

This block is pretty much set. All data will reside in a database. The data will need to be classified. We plan on simplifying the project as much as possible to minimise the data set but retain the solution objective. However at this point we know that a classification scheme must be implemented. To this end we have already begun to brainstorm how we are going to classify the data. To simplify the final set, we have made the assumptions that the user is going on a vacation and will be driving to his/her final destination. We therefore will be looking at places to stay. To classify a specific location we'll look at and choose from the following properties:

- Type of accommodations: Does the traveller prefer a hotel, bed and breakfast or hostel?
- Price of accommodations
- Distance between stopping points: What is an appropriate distance for the traveller to drive in a day. This is based on driving habits.
- Points of interests near the accommodations:
  - Tourists attractions
  - Restaurants
- The travelling time. Does the traveller want to arrive at their destination as soon as possible or do they mind stopovers?

Updating the data set will be done by hand as automation of this task is a project in itself.

Agent Learning Methodology

The agent must be able to "learn" the operator's likes and dislikes. Their interests and desires. There is no point in suggesting a traveller stay in a 5 star hotel if he/she doesn't have the financial mean to afford it. It is acceptable for the few iteration of an agent to suggest these things but as the agent is used more it must be able to hone its skills.

Teacher-Student Relationship
This method is akin to an intimate teacher-student relationship found in a school. At the beginning of the relationship it can be assumed that the student knows nothing about the course material and the teacher knows all. As their time together passes the student begins to learn from the teacher by both direct instructions and indirect instructions. For the direct instructions the teacher will tell the student how to develop a suitable solution to a problem given a set of inputs. The teacher supplies the inputs. The indirect instructions will come from feedback from a solution based on direct instructions. By presenting the teacher with a solution set the student can then analyse the solution that is picked from the set and begin developing an internal instruction set so that the teacher needs to supply less and less inputs for the next problem. They can be approximated by the student's "intuition" which is based around its internal instruction set. This revolves around the fact that the specifics of a problem will differ from problem to problem, the context will remain the same. This relationship will continue until the student and teacher will part ways. It is important to note that the student will never surpass the teacher. This solution may take more time since the student starts with no knowledge of the problem. But the benefits are a customisation from the ground up. The instruction set that is developed will be developed for that teacher alone.

Average Person Relationship

This method is the same as the teacher-student relationship (TSR) with the exception that the agent is preloaded with an average user's profile. That means an internal instruction set is present before the initial interaction between operator and agent. This will be refined in the same way as the TSR method with the same goal of an intimate relationship. This relieves the agent of the task of actually writing the instruction set. In this method the agent needs only refine the instruction set. In the case of a fuzzy logic decision process (to be discussed later) this means changing the weighting of the member functions instead of writing the member functions all together. The writing of the initial member functions would be left to the programmer who would have to do the research and classification. This method may offer a shorter learning curve thus reaching an efficient internal instruction set sooner. The disadvantage is that the initial instruction set must be sufficiently general to allow customisation but needs to remain in context of the problem the agent will tackle. It will be a time consuming task.

Decision Process

How the agent develops its internal rule set is an important aspect that differentiates an agent from a software application. In "real life" decision making is not black and white. It is full of grey areas. The human brain is an extremely abstract dynamic system. How one individual arrives at a decision is unique from all others. It is this system an agent needs to mimic if it is to succeed in producing a custom efficient solution to the operator. Three of these solutions are as follows:

Simple Statistical Analysis
Under this scope this method means that the agent scans choices picked by the traveller on previous trips. It then assigns weighting to certain parameters depending on how often they appear in previous solutions. These weightings are then used in a decision process for prioritising one solution in a set over the others. This method is simple to implement however each parameter is treated independently of all the others. This can lead to erratic solutions since mutually exclusive parameters are not taken into account.

Fuzzy Logic
Fuzzy logic is perfect for an agent that needs to reason with imprecise or incomplete data. It uses the method assigning a degree of truthfulness or falsehood to outputs. By passing an input value through a set of rules then a decision can be made that is not completely true. By updating and refining the member functions a better decision can be made on an input of values.

Neural Networks
Neural network algorithms are designed to mimic the neural networks that make up a human brain. It is a collection of nodes that are joined by a neural net. These nets are trained by experience. They are reconfigured around the right answers that are produced by inputs fed through the nets. It can be thought of as "mapping a set of input vectors to a set of output vectors". The problem with neural networks is that they need to be trained with a large amount of data in order to develop the right patterns. Which in turn is time and resource consuming.

Programming Languages

The mortar that ties all these blocks together is the programming language. Which language is most suited for developing this agent?

C
C is an old programming style. Tried and true. It is a sequential data oriented language. It is however the language that the two developers know the best. However it lacks any intuitive graphical user interface development. It requires Tcl/Tk that the developers don't know. It is geared towards stand-alone applications.

C++/Java
For this project C++ and Java can be thought of the same language with slight differences. They are both modular, object-oriented languages. This makes programming easier on many levels. Although C++ is a faster language Java is platform independent. The developers however have a greater grasp on C++. These

can fit in with the web based user interface. Java however comes with Interface library to help develop a friendly interaction environment.

Visual Basic
VB is becoming a powerful language that incorporates an easy to develop front end with a power backend that can include database work. It's a modular programming language where it can interact easily with other applications. The major draw back is that it is specific to the Windows platform only. It can be ported to other Operating Systems with considerable difficulty. The BASIC language is quite easy to learn and very straightforward.

Alternative Evaluation

A procedure was set up to choose between alternatives. A simple decision matrix was used to differentiate between choices of building blocks. The most appropriate choices were then added together to form the final solution.

In order to differentiate between different alternatives, a set of global categories was set up to compare between them. These are:

Relevance towards problem (Weight = 0.35)
Agent design is a broad spectrum. It was important to focus down on a specific area of design to toss out technologies that benefit other areas of agent design but not this one.

Knowledge Base (Weight = 0.25)
The more we know about a certain technology the less we have to teach ourselves. It is important to note that the best technology may not be implemented because of time constraints, learning new skills has to be minimised.

Available Resources (Weight = 0.20)
Why re-invent the wheel? For fuzzy logic there exists software packages that take as input, member functions and input variables and output the results. The code is already in place. Also programming languages such as Java have a wide resource of objects and libraries developed for download, and Visual Studio offers a great development environment for Visual Basic.

Ease of Use (Weight = 0.20)
This is an important factor that relates to our overall knowledge of a skill. How easy will it be to implement the technology into the solution? Does it offer a friendly development environment and can it work with other applications and technologies?

With that said the following table shows how the weighting works out. 1 is the lowest mark and 3 is highest mark available.

	Relevance (0.35)	Knowledge (0.25)	Resources (0.20)	Ease (0.20)	Total
Human-Agent Interaction					
• Web Based	3	3	3	3	3
• Command Line	3	3	1	1	2.2
• Application	3	1	1.5	1	1.9
Agent Learning Methodology					
• TSR	3	1	1	2.5	2
• APR	3	1	1	3	2.1

Decision Process					
• Simple Statistical Analysis	3	3	1	3	2.6
• Fuzzy Logic	3	2.5	2	2	2.475
• Neural Networks	2	1	2	2	1.75
Programming Language					
• C	2	3	1	1	1.85
• C++/Java	3	1.5	3	1.5	2.325
• Visual Basic	3	1.5	3	2	2.425

[Table #.#]

Selecting the top blocks from the table we find that optimum solution path given the problem statement and technologies available to us is a Web-Based front end backed up with an APR agent using a Simple Statistical Analysis decision process. All brought together using Visual Basic. Again it may seem to the reader that because fuzzy logic and neural networks are better solutions for interpreting and dealing with incomplete inputs we would select one of them as our decision process. However the time restrictions and learning curve of the Simple Statistical Analysis method won out in the end.

Design Methodology

Now that the design pieces have been specified, a methodology will be put into place that will tackle the "putting together" of the final solution.

A project plan was developed and submitted already but for convenience it is included in the appendix of this report. The methodology we will be following is a divide and conquer methodology. We'll start creating the individual building blocks. With them completed, we will bring them together and start to tie it all together. Each block will be treated as a mini design phase. Our philosophy is one such that we will keep it as simple as possible. All extra values will be left till the end. At which point, should there be time remaining, we will prioritise their importance and add them in accordingly.

Needless to say, testing is a critical part of any computer design project. To accomplish a successful testing plan we'll incorporate it from the beginning. Instead of a sequential process where testing is done at the end of the project, we'll include it in the iterative design process. This way we won't be left, should we fall behind time wise, with little or no testing time.

Conclusions and Recommendations

The most important task of this design team is to remain focused on the problem at hand. Agent technology encompasses a large paradigm. Each problem is radically different from its brethren and seldom do they have parts of solutions in common. On a global basis each agent must follow a certain criteria to be called an agent. However as customisation occurs each agent must be treated individually. Just as the agent will treat each operator individually.

Development time is the largest constraint in this project. This was the main reason why one technology was picked over the others. For example, we know that a fuzzy logic decision core would be more suited to handle the decision making portion of an agent. However, the design team is less familiar with implementing fuzzy logic and using statistical analysis is easier to implement and maintain. It will be up to the design team to make sure that all necessary information is collected to maximise the truthfulness of the results.

At this point, the team is passing through the transition from research and literature gathering into the design phase. The next step is to take the above mentioned design solution and begin to implement the ideas into reality.

Should you, Prof. Areibi, have any concerns or suggestions about the design progress or this report, we would be more then happy to hear from you. We will meet with you on our scheduled time on Wednesday October 18, 2000 to discuss this interim report.

References

Feldman, Susan and Yu, Edmund (1999). Intelligent Agents: A Primer. http://www.infotoday.com/searcher/oct99/feldman+yu.htm, *Searcher Vol 7, No.9*

Wooldridge, Michael and Jennings, Nicholas R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, October 1994 (Revised January 1995).

Generesreth, Michael and Ketchpel, Steven (1996). Software Agents. Standford University Computer Science Department.

Elliot, Clark and Brzezinski, Jacek (1998). Autonomous Agents as Synthetic Characters. *AI: Magazine.* 13-29, Summer.

Lieberman, Henry. *Autonomous Interface Agents*. http://lieber.www.media.mit.edu/people/lieber/Liebrary/Letizia/AIA/AIA.html. Visted 9/20/00.

Eaton, Peggy S. and Freuder, Eugene C., and Wallace, Richard J. (1998). Constraints and Agents: Confronting Ignorance. *AI: Magazine*. 51-63, Summer.

Lewis, Michael (1998). Designing for Human-Agent Interaction. *AI: Magazine.* 67-78, Summer.

Franklin, Stan (1997). Autonomous Agents as Embodied AI. *C ybernetics and Systems*. 28:6 499-520.

Ackerman, Mark, et al. Learning Probabilistic User Profiles, American Association for Artificial Intelligence, Summer 1997.

Michaelides, Alexis, Intelligent Agent Technology in Travel Reservation Systems, Imperial College of Science Technology and Medicine, May 9 1997.