III TECHNOLOGY FOR THE OPTIMIZATION OF THE INTERNAL TASKS OF TAX ADMINISTRATION

Technology for the Optimization of the Internal Tasks of Tax Administration: The Adaptive Agency

Jesús Mena
U.S.A.

Second Prize Winner of the Vº CIAT/IEF Essay Contest

This paper will discuss the infrastructure of a tax agency founded on several biological-based software technologies.

Introduction

Because of the combined factors of an electronic worldwide economy and a shortage in the workforce, the tax agency of the future will depend on the use of software intensive technology. Among all of its information systems the most important ones will be based on those replicating human perception, thought and experience. This "intelligence" agency will be configured around a series of advanced software paradigms, specifically: neural networks, fuzzy logic, genetic algorithms, and case-based reasoning (CBR).

The common attributes of these software technologies is that each has an ability to "learn" from examples, to generate its own rules, and then to continue to adapt from experience to solve new problems or situations. More traditional artificial intelligence (AI) technologies such as expert systems would be used in conjunction with these biological -based systems. However, it will be the use of these software technologies which will be crucial in ensuring the agency's able to meet its mission in the areas of taxpayer compliance, criminal, investigations, administrative management and intelligence surveillance.

Expert Systems

Searching forms a central part of all software development. Search methods are found in primitive forms in virtually every software system that has been developed, for example, as subroutines to search tables. They are also found in very much more advanced forms in artificial intelligence research. Search is manifested in expert system where a program attempts to replicate the problem-solving capabilities of a human expert. Such programs search for some solution to a problem guided by rules which have been extracted from a human expert with the assistance of a knowledge engineer (a programmer).

An expert system, the earliest form of an artificial intelligence system, is a program which interrogates a stored file, known as a knowledge base, that contains the rules which are used by a human expert in order to carry

out some task which requires high-level thinking skills. These rules are extracted from the human expert by the artificial intelligence equivalent of the systems analyst: someone known as a knowledge engineer.

Most expert systems carry out a search process. For example, a tax service representative assistant searches through possible tax regulations guided by the rules in a knowledge base; another example may be an embedded expert system designed to examine real-time transactions records in a bank database for purposes of detecting fraud.

Expert systems are limited however, for once they grow to contain more than around 1000 rules, they become afflicted with a number of problems. They are difficult to build and, equally important, to maintain. It is also difficult to predict what happens when extra rules are added to the knowledge base. It becomes harder to understand the rules in the knowledge base, as they take on increasingly Byzantine structures.

Also, large expert systems tend to become so grossly inefficient as rules are added that, after a certain point, consultation becomes impractical in real time. And it becomes very arduous to tease out the rules in the knowledge base from an expert. Even for small expert systems, with a few hundred rules, this is quite problematic; for knowledge bases with thousands of rules it is impossible.

Researchers have been attempting to look at solutions to the problem of search, where the space of possible solutions is very large and where the criteria for locating a successful solution are fuzzy. The writing of code for large expert systems with hundreds and in some situations thousands of rules is also highly labor intensive: there are not enough programmers in the world to support this kind of endeavor. Because of these and other limitations, expert systems do not have the potential to support real-time large applications. In a reaction to these shortcomings of expert systems the area of software engineering has spawned a group of probabilistic techniques in computer science based on biological models.

Biological Software Systems

Biocomputing refers to biologically inspired approaches to creating software. In recent years a number of these techniques and technologies have emerged. Some are proven and out in the market, others are still being nurtured in research labs. Together they hold great promise in allowing software developers to push the envelope of program complexity and size, and to allow for tractable solutions to thorny programming problems such as pattern recognition.

During the last decade in which real-time networked systems, and distributed applications appeared en masse, with program size and complexity taking a big leap, the limits to the processing of existing software technology was reached. For example, the ten million lines of code in the Space Shuttle software system came to a screeching halt during one launch countdown when an error, caused by a missing comma in one program, was encountered.

Subsequently research has been going on for software systems that are less brittle, more reliable, robust, and flexible, yet still allow for high levels of functionally. Thus the development of biological-based theories in computer science called biocomputing. The rationale behind biocomputing is not to create big programs. In fact, most current implementations are modest in size and resource consumption. Rather, the rationale comes from the fact that, as the number of lines in conventional programs increase, their complexity approaches that of a small biorganism.

Living systems are intricate structures made out of simpler components (cells), with high degrees of redundancy, fault tolerance, and adaptiveness. Incredibly detailed and complex biostructures arise from very small sets of rules. Similar to the concept and design of neural networks, also known as neurocomputing, neoconnectionism, and parallel distributed processing, other biocomputing technologies include genetic algorithms, fuzzy logic, and CBR.

Genetic Algorithms

Genetic Algorithms mimic the process of natural selection that occurs in nature, with the fittest examples of a species being those who survive. The survival is achieved by the passing on of favorable characteristics to offspring who, in turn, pass on more favorable characteristics to their offspring.

Genetic search works in the same ways as natural selection. In order to understand the main principles of genetic search take the example of designing a telecommunication network for a tax agency, the type of application which is amenable to this form of search approach.

The problem is to design a telecommunication network so that the cost of the hardware and software are minimized, while the routing of the switching through the network is maximized. The first step in a genetic algorithm is to generate randomly a number of first attempts at the solution which satisfy the aim of generating the required amount of communication transactions between two points. A number of solutions are then selected from this set, based on some objective criteria of the goodness of the solution. In the telecommunication network problem, this may be the single cost of a set of packet transmissions via the network. This is the part equivalent to the natural selection process that governs the success of animals and plants - sickly individuals are culled without getting the chance to breed. After this, the designs are mated with each other to produce brand new designs. For example, a telecommunication network design may be constructed which contains sections from both its parents.

Finally, the solutions suffer mutation. This involves random changes to the designs which were formed by mating. There is a lot of confusion about the role of mutation in both genetic algorithms and human genetics. The reason why mutation is employed is because, if just mating and natural selection was used, then there is a potential for some good parts of a solution to be lost.

When a new generation of designs has been formed by natural selection, mating and mutation, the process continues with the development of a further generation using these operations, until a solution is found which matches the criteria for a good solution. In this telecommunication design problem, this would probably be some overall cost figure for the implementation of the network.

In the 1980s researches concentrated on the rule-based strategies used by expert systems. However, now that expert system technology is beginning to falter, there has been a massive explosion of interest in genetic algorithms. In the early 1990s, researches have already applied the techniques to a number of different problems.

Genetic algorithms look highly promising, especially where there is a large amount of non-linearity in the search space. Other promising techniques use the emerging technology of machine learning. Current computer learning technology involves programs based on so-called induction algorithms. These algorithms work in the following way. An induction program is represented with a number of cases involving a series of factors: tax amount, tax type, etc., together with an outcome from each factor tax payed, tax adjusted, or tax not payed.

For example, an induction program might examine a series of past criminal investigations using factors such as taxpayers' reported income, their number and type of assets, and their number of bank accounts. The outcome in this example would be the generation of an examination audit. The induction program examines each case, and attempts to relate the outcome of each case to the variables. For example, it might find that a particular range of bank accounts and number of assets always give indications of criminal activity or underreporting of income. This information is incorporated into a data structure known as a decision tree. The decision tree contains a codification of the reasoning processes which underlie the outcomes.

Each path through the tree contains the combination of factors which correspond to a particular criminal investigation situation. For example, there is a path through the tree which says if a taxpayer has a low number of bank accounts, a low reported source of income and a high

number of personal assets then the taxpayer is potentially underreporting his income or involved in criminal activities. Normally the decision tree will be much more complicated than this and would have a large number of levels.

Once the decision tree has been constructed from the test cases, a simple program accesses this tree, and then provides the advice to an investigative agent or another computer program which complies and compares the various factors from income documents and tax returns. The main advantage of computer learning technology is that it eliminates the need for the manual process to eliciting rules from a human expert.

Several programs have been developed in the financial services industry using this type of induction program to reconstruct the reasoning processes used to evaluate the issuance of loans, or credit cards. Induction algorithms have been used to develop expert guidance systems for banks, and could similarly be used to provide decision-support to a tax agency in determining when to do an examination an adjustment to an assessment, write-off a collection case or open a criminal investigation.

Neural Networks (Nets)

Neural networks are essentially memories. They do not compute answers so much as memorize answers. Humans for example do not commute the answer to "what is two plus two" instead by extensive learning in elementary school, have memorized the answer to this problem. It is stored in the equivalent of a look-up table. Hence if a predictive model created with a neural network is organized with the correct memory (look-up table), it does most of the computational required.

Neural nets can be either software or hardware structures which are taught to recognize the patterns that are presented to them. Researchers in this area often claim that their techniques mirror the processes which occur in the brain when patterns are recognized: that in the same way that the neuron structures in the brain configure themselves to recognize scenes, faces and structures, the software and hardware structures that are used to implement neural nets are changed by altering linkages and configurations by informing them of a correct response or an incorrect response.

The application areas of neural nets are broad and surprisingly widespread, although the principal task is pattern recognition. One of the earliest neural net models, Bernard Widrow's adaptive linear element (Adaline) has been in use since 1959 as adaptive hardware filters to eliminate echoes on phone lines. In recent years artificial neural nets have been used for pattern recognition, image processing, compression, speech synthesis, natural language processing, noise filtering, robotic control, and financial modeling. Naturally their capacity for the construction of predictive models capable of assisting an agency in making decisions on its management of information is crucial due to their structural nature: they are data driven.

A key advantage of neural nets over conventionally written programs is the way in which nets imitate the brain's ability to make decisions and draw conclusions when presented with complex, noisy, irrelevant, and/or partial information. Another advantage is that neural net application is not a handcrafted program, but rather the result of feeding training data to a net model, which then learns to output the desired results. Once a net has been trained, it will also be able to deal with input that is different from what it has been trained with, as long as the input is not too different. This is a big advantage over conventional software, which must be specifically programmed to handle every anticipated input. Presumably, this evade the problem of the "missing comma" of the aborted Shuttle launch.

Neural nets are currently the highest power code-efficient computer language and are basically "pattern processors" capable of distributed intelligence control. One of the more promising attributes of neural nets is that they can be made "adaptive" that is, they can

contain a mechanism for "learning" as they process information (e.g., tax returns). This capability arises from the nature of the pattern-processing which neural nets perform.

The learning of a net occurs when an "instructor" presents the adaptive system with a pairs of data elements: an input and an ouput. This pair consists of an example, an input "tax case characteristics" with which the neural net associate a separate "tax classification condition" output. These case characteristics might be variables from a tax return, while the classification condition may be full payment, delinquent status, or fraud.

After the neural net system is trained, it is presented with only "tax case characteristics" in order to solicit "tax classification condition" responses. Such a system thus can be readily built to capture the needed associations to anticipate and predict the outcome of cases as they make their way through the tax agency's processing stream. Neural nets can provide the adaptability and learning needed to solve compliance problems, do case diagnoses and predict revenue management. The use of neural nets capable of learning can thus optimize production and efficiency yield on a dynamic learning basis. Based on historical tax case data they can be trained to predict how future cases will evolve and what their outcome will be: probability mapping of cases for more efficient and intelligence processing.

Fuzzy Logic

Fuzzy logic, or the theory of fuzzy sets, is a technique that is inspired by nature without intending to be a realistic model of any physical objects. Invented by Lofti Zadeh in 1965, fuzzy logic is an extension to mathematical logic that allows for "soft" values that are in between the hard values of true and false (0 and 1). The intend is to be able to deal in a meaningful way with imprecise notions or concepts that do not have exact boundaries. This does not mimic how the physical human brain works, but it does follow how the human mind seems to carry out its reasoning. The gradient classification and clustering of taxpayer cases as "soft" can be achieved by the use of fuzzy logic systems.

Fuzzy logic has found many practical adherents in Japanese companies. This technology is now a key element in products such as Canon autofocus cameras, Hitachi washing machines, and Nissan and Subaru transmissions, as well as in the handwriting input recognition found in the Sony Palmtop computer.

Fuzzy theory holds that all things are a matter of degree. Broadly, fuzzy logic attempts to mimic imprecise, human thinking by operating on IF-THEN statements that describe conditions or actions. Input values are compared to a predefined set of values and are determined to either belong to the predefined set to some degree or to not belong to the set. If the input value belongs to the set, appropriate action is taken by the system. The degree to which an input value belongs to the predefined set can also determine the intensity of the resulting system action. This is motivated by the recognition that constructing robotic systems capable of autonomous, flexible, intelligent behavior is an inherently interdisciplinary task. Another role of knowledge based reasoning exploits the fact that programs are data, allowing them to dynamically modify their own structure in order to improve their performance or extend their competence. As with the other technologies discussed previously fuzzy logic is adaptive and able for example, to refine a gradient system of scoring tax cases or variables on tax returns for subsequent and further processing in an agency's service centers or call site plants.

CBR

CBR is another new paradigm for building knowledge-based systems. Unlike rule-based reasoning, which relies on chains of deductive reasoning, CBR relies on the retrieval of relevant experience (case histories) from previous similar situations. The basic approach of a CBR system can be stated simply: to build a "case base" of previously solved problems and to solve a new problem, search for similar past cases and adapt previous solutions to the current situation.

In a well-designed CBR system, tax administrative

situations that are not properly resolved from existing cases can be captured automatically, referred for proper description and proper resolution and then updated into the case base. In this normal pattern of usage, CBR applications can get better as they are used, as more and more cases are encountered. This provides the program with the ability to "reason from experience" and "learn from experience".

The fundamental concepts of CBR originated in artificial intelligence (AI) research conducted at Yale University around 1980. Researchers were trying to build computer programs that could understand textual input. The goal was to be able to type a news story into a computer program, have the program produce a summary of the important points and answer users' questions about the story. One of the fundamental problems of the early versions of these primitive story-under-standing programs was that they had no capability to remember what they read. In contrast, with a CBR system, end-users can simply capture cases from actual events as they occur.

The ability to retrieve and manipulate past problem-solving examples accurately is important for many domains: diagnosis, classification, prediction, planning, design, law, process control and monitoring, advanced manufacturing, and configuration are good examples. Despite the value of remembering past cases for problem solving, humans can forget what has worked in the past, or why, for that matter.

This can lead to the "reinventing the wheel" syndrome so prevalent in government and industry. However computers do not forget. If we can get them to remember the right thing at the right time, we can reap the benefits of the past success and failures of the agency and its employees.

A rule-based expert system solves problems by taking an input specification (or developing one through a question-and-answer dialog with the user) and then "chaining" together the appropriate set of rules from the rule base to arrive at a solution. Given the same exact problem situation, the system will go through exactly the same amount of work to come up with the solution. (In other words, rule-based systems don't inherently learn). In addition, given a problem that is outside the rule-based system's original scope the system often can't render any assistance. Finally, as previously discussed rule-based systems are very time-consuming to build and maintain because rule extraction from experts in labor-intensive and rules are inherently dependent on other rules, making the addition of new knowledge to the system a complex debugging task.

Case-based systems operate in a very different way and quite similar to neural nets. Given an input specification, a case-based system will search its case memory for an existing case that matches the input specification. If successful this rate increases as new cases are added to the system, so that exact matches the input problem and goes directly to a solution, making it possible to provide solutions to potentially complex problems quickly. If, on the other hand, their is no match, the system will retrieve a case that is similar to the input situation but not entirely appropriate to the provide completed solution.

The case-based system (or the human user) must then find and modify small portions of the retrieved case that do not meet the input specifications. This is called case adaptation. The result of case adaptation is a completed solution, but it also generates a new case that can be automatically added to the system's case memory for future use by the agency.

Learning is a basic part of a CBR system's architecture. So, if the same problem is given at some future point, the system will bypass the effort required the first time it solved the problem since the institutional memory will recall it. In general, case-based systems are easier to build and maintain because the knowledge engineering task is reduced to the simpler problem of defining terms and collecting preclassified cases from the expert, and adding new knowledge is as easy as adding a new case to the agency's case memory.

In its simplest form, a case is

a list of features that lead to a particular outcome. Some examples include the information on a tax form and whether or not the tax was payed, and a taxpayer's compliance history and the associated results of past examinations or collection actions. In its most complex form, a case is a connected set of subcases that form the problem-solving task's structure; for example, the design of a telecommunication network. The designs of a network are made up of subdesigns of the components that comprise the whole, each of which could be considered a case unto itself.

CBR approaches work well in domains that are not well understood, because the system doesn't need to know why something worked in the past. It needs only to recognize a future situation in which the plan could be used. CBR systems can often finesse some of the difficult representational and behavioral modeling needs of a domain that make using rule-based approaches impossible because the rules are not known. In fact, the inductive techniques inherent in CBR systems can, in time, facilitate the process of "understanding" the domain by performing analysis that can show previously unknown relationships between the domain features.

Conclusion

The future agency will have no centralized data center, instead a web of information will network a multitude of sub-systems and users. The agency will be a "virtual" organization: ubiquitous, adaptive, and responsive -- capable of rapid dissemination of information to agency users and the taxpayer public. Real-time access to data will be provided for responsive adjustments to legislative, compliance, and business plans needs. The core of these information systems will revolve around decision support biological based software systems.

In a global economy in which decentralized electronic fund transactions are commonplace, a new method for monitoring and taxation of these transactions will require a new infrastructure able to capture and evaluate large volumes of data. This will require the use of the software paradigms based on pattern recognition for targeting criminal activity.

At the core of this decentralized organization will be the use of technology enabling the agency to maintain a fluid state: responsive to change as the needs arise.

The goal of this organizational design will be the capability to quickly adapt to changes in a rapid time frame. The key for doing this will be the use of biological based software technologies enabling the agency to self-adjust and to "learn" from the information it processes. The agency will remember everything and forget nothing.

A series of inference "engines" and predictive "models" will be used in the processing stream of this agency to maximize efficiency. Expert systems will provide multiple decision support applications to ensure the experience of domain experts, such as agency technicians and seasoned employees, is codified and institutionalized for re-use by new or less experienced employees. As data is processed by the agency probabilistic technologies such neural networks and case-based reasoning (CBR) will analyze it for diagnostic pattern recognition. Data processing will become data learning enabling the agency to construct hundreds of predictive models enabling it to anticipate conditions before they occur. The decisions of forecasts the models make would be result of their ability to learn and adapt.

The cost benefits for the use of these biological software systems are several, they don't sleep, they improve their skill over time, they remember everything enabling them to become the agency's institutional memory, they can serve to provide management answers to questions on the characteristics and trends of the agency's inventories, these systems can reduce cost by improving the accuracy in the selection of the type of cases with the potential for high yield and collectability, they can reduce inefficiency by quickly alerting management on the characteristics of unproductive work.

In the future, a tax agency will be dependent on a selected and small number of elite highly skilled workforce of professional specialist able to use information systems for the processing of large number of cases efficiently and in a more productive manner. These information systems will required the use of adaptive self-learning software to ensure the agency of the future is able to meet its mission.

Bibliography

Barletta, Ralph "*An Introduction To Case-Based Reasoning*. (Artificial Intelligence-Based Systems That Use Past Experiences, Or Cases, To Solve Problems)". Al Expert VOL.: v6 ISSUE: n8, p. 42 (7), August 1991.

Ince, Darrel "Answer in the genes?" Computer ASAP VOL.: v5 ISSUE: n9 p.40(3), March, 1991.

Martin, S. Louis "Design Support Grows For Fuzzy Logic" Computer ASAP VOL.: v36 ISSUE: n13A p.1 (2), July 25, 1991.

Morley, Richard E. Editor, "Neural Networks At A Glance: A Compilation of Synapse Connection Articles", Graeme Publishing, 1989.

Valdes, Ray "What Is Bio Computing? Biollogically-Inspired Approaches To Creating Software". Dr. Dobbs Journal VOL.: v16 ISSUE: n4 p.46 (3), April, 1991.

Vrooman, Gary "Case Based Reasoning" Computing Canada VOL.: v17 ISSUE: n15 p. 28 (1), July 18, 1991.