Multi-Agent System (MAS) that Learns to Advise Students

Omankwu, Obinnaya Chinecherem, Nwagu, Chikezie Kenneth

[1]Computer Science Department,
Michael Okpara University of Agriculture, Umudike
Umuahia, Abia State, Nigeria
saintbeloved@yahoo.com

[2] Computer Science Department,
Nnamdi Azikiwe University,Awka
Anambra State, Nigeria,
Nwaguchikeziekenneth@hotmail.com

Abstract

The Internet and the World Wide Web advancement has led to an explosion in the amount of available information. This staggering amount of information has made it extremely difficult for users to locate and retrieve information that is actually relevant to their task at hand. Dealing with this problem of ‘‘information overload’’ will need tools to customize the information space. In this paper we present a multi-agent system that learns to advise students by mining the Web and discuss important problems in relationship to information customization systems and smooth the way for possible solutions. The main idea is to approach information customization using a multi-agent paradigm in combination with a number of aspects from the domains of machine learning, user modeling, and Web mining.

Keywords: Information customization; Multi-agents; E-Learning; Machine learning; User modeling; Web mining;

1. Introduction

The recent proliferation of personal computers and communication networks has a strong scientific, intellectual and social impact on the society. Rapidly evolving network and computer technology, coupled with the exponential growth of the services and information available on the Internet, has already brought us to the point where hundreds of millions of people should have fast, pervasive access to a phenomenal amount of information, through desktop machines at work, school and home, through televisions, phones, pagers, and car dashboards, from anywhere and everywhere. The challenge of complex environments is therefore obvious: software is expected to do more in more situations, there are a variety of users, there are a variety of systems, there are a variety of interactions, and there are a variety of resources and goals.

To cope with such environments, the promise of information customization systems is becoming highly attractive.

The recent popularity of the World Wide Web (Web) has provided a tremendous opportunity to expedite the dispersement of various information creation/diffusion infrastructures. The mass of content available on the Web raises important questions over its effective use. With largely unstructured pages authored by a massive range of people on a diverse range of topics, simple browsing has given way to filtering as the practical way to manage Web-based information. Today’s online resources are therefore mainly accessible via a panoply of primitive but popular information services such as search engines.

Search engines are very effective at filtering pages that match explicit queries. Unfortunately, most people find articulating what they want extremely difficult, especially if forced to use a limited vocabulary such as keywords. The result is large lists of search results that contain a handful of useful pages, defeating the

purpose of filtering in the first place. Search engines also require massive memory resources (to store an index of the Web) and tremendous network bandwidth (to create and continually refresh the index). These systems receive millions of queries per day, and as a result, the CPU cycles devoted to satisfying each individual query are sharply curtailed. There is no time for intelligence. Furthermore, each query is independent of the previous one and no attempt is made to customize the responses to a particular individual.

What is needed are systems that act on the user’s behalf and that can rely on existing information services that do the resource-intensive part of the work. These systems will be sufficiently lightweight to run on an average PC and serve as personal assistants. Since such an assistant has relatively modest resource requirements it can reside on an individual User’s machine, which facilitates customization to that individual. Furthermore, if the assistant resides on the user’s

machine, there is no need to turn down intelligence. The system can have substantial local intelligence and information customization becomes possible.

In this paper, following the same long-term objective of providing a complete E-Learning environment for students and striking for the more general goal of information customization, a multi-agent system that advises students by adopting a machine learning paradigm. Machine learning methods can be used to deal with many different aspects of the problem of advising.

Academic advising, in its simplified version, consists of telling the student which courses he/she should register in based on the profile of the student, on the university laws, and on the courses that are offered in the semester for which advising is needed.

In the following we will discuss some important points in relationship with E-Learning, information customization, user modeling, agent systems, machine learning, and Web mining.

E-Learning

E-Learning is a valuable extension of the distance education paraphernalia, enabled by the new information and communication technologies. Distance education normally occurs in a different place from teaching and as a result requires special techniques of course design, special instructional techniques, special methods of communication, as well as special organizational and administrative arrangements (Moore, 1996). E-Learning is often described as the use of network technology, namely the Internet, to design, deliver, select, administer and extend learning.

One key issue in E-Learning is communication between participants, for which there are two basic types of technological solutions: asynchronous and synchronous. In the asynchronous approach, the interaction between parties does not require them to be engaged at the same point in time. In synchronous communications the interaction between participants requires simultaneous engagement of the participants.

Online education is today a reality in many sectors of the society, especially in educational centers such as colleges and universities, increasingly high schools, and also professional groups demanding continuous access to education. Several years ago this new educational method was considered as an experimental approach with more disadvantages than advantages. However, today it should be considered not only a complementary educational resource but also a serious alternative that competes to conventional and now classical methods. Both methods will coexist and the logical initial inertia to ignore the new opportunities provided by the new media should be reduced and be faced sooner better than later in the same manner in which many other areas were modified throughout history.
Obviously the adaptation to the new features and services of the E-Learning environment is not immediate and requires experience, time, investment, pedagogical and technical resources, and government or campus administration support.

Information customization

Building software that can interact with the range and diversity of the online resources is a challenge and the promise of information customization (IC) systems is becoming highly attractive. Instead of users investing significant effort to find the right information, the right information should find the users. IC systems attempt to accomplish this by automating many functions of today's information retrieval systems and provide features to optimally use information (Mostafa,2002). IC systems are different from conventional search engines or database systems. Not all information is easy to find. Most people using, for example, the Internet to search for specific information report some frustrating experiences. From the user's perspective, two problems frequently arise when using the search engine. First, the words might not match exactly and hence nothing is returned by the search engine. Second, and much more common, is that too many URLs are returned by the search engine. Furthermore, each query is independent of the previous one. Attempts to customize the responses of the search engine to a particular individual are rare, scarce and feeble. The result is homogenized, least-common-denominator service and no personalization is possible.

Even when the relevant information is easy to find, it will be perhaps boring and time-consuming for the user to perform this task and it would be wonderful if an IC system can identify and present the information with little or no user intervention. The system should also be able to update the presentation as new information becomes available. This will release the user from continually observing the resources. This raises, of course, questions about robustness and persistence of the system. IC systems tend, by their very nature, to be distributed – the idea of a

Centralized IC system is an oxymoron. Distributed systems have long been recognized as one of the most complex classes of computer systems to design and implement. A great deal of research effort has been devoted to understanding this complexity, and to developing formalisms and tools that enable a developer to manage it (Mostafa,2002). Despite this research effort, the problems inherent in developing distributed systems can in no way be regarded as solved. So, in building an IC system, it is vital not to ignore the lessons learned from the distributed systems community. The IC system developer must therefore recognize and plan for problems such as synchronization, mutual exclusion for shared resources, deadlock, and livelock.

IC systems are often expected to be adaptive. Adaptive software is software that adapts, with little or no intervention by a programmer, to changes in the environment in which it runs. Such an adaptive solution will be able to add feedback for performance characteristics and allow the program to make choices autonomously. As a result the system is able to change its behavior-based on its previous experience.

We also expect an IC system to act even if all the details are not specified, or the situation changes. This is the property of autonomy, i.e., the system is able to exercise control over its own actions. Other considerations such as proactiveness (the program does not simply react in response to the environment) and mobility (where does the program run) are also of importance.

User modeling

An IC system is software that acts, in accordance with a user's preferences, in an environment. To realize an IC system acting in accordance with a user's preferences, user modeling is needed. User modeling comes in two varieties, behavioral and knowledge-based (Kobsa, 1990). Knowledge-based user modeling is typically the result of questionnaires and studies of users, hand-crafted into a set of heuristics. Behavioral models are generally the result of monitoring the user during an activity. Stereotypes can be applied to both cases, classifying the users into groups (or stereotypes), with the aim of applying generalizations to people in those groups (Kobsa, 1990).

The typical user profiling approach for IC systems is therefore behavior-based, using a binary or a multi-class behavioral model representing what users find interesting and uninteresting. Machine learning techniques are then used to access potential items of interest in respect to the behavioral Ali and Smith present a method for learning algorithm selection for classification.

Systems based on behavioral models and employing a learning technology are classified according to the type of information required by the learning technique and the way the user model is represented. Algorithms requiring an explicit training set employ supervised learning, while those without a training set use unsupervised learning techniques [12]. There are three general ways to learn about the user: monitor the user, ask for feedback, or allow explicit programming by the user. Monitoring the user's

behavior produces unlabeled data, suitable for unsupervised learning techniques. This is generally the hardest way to learn, but is also the least intrusive. If the monitored behavior is assumed to be an example of what the user wants, a positive example can be inferred. Asking the user for feedback, be it on a case-by-case basis or via an initial training set, produces labeled training data. Supervised learning techniques can thus be employed, which usually outperform unsupervised learning. The disadvantage is that feedback must be provided, requiring an investment of an often significant effort in the system by the user. User programming involves the user changing the system explicitly. Programming can be performed in a variety of ways, from complex programming languages to the specification of simple cause/effect graphs. Explicit programming requires significant effort by the user.

User profiles are of great importance for information extraction and information customization since they are essential for deciding what kind of information is needed, where this information can be found, how this information can be retrieved, and how this information should be presented to the user. User profiles will therefore have a great influence on the solution to be adopted for implementing an IC system. In our case they will have a strong impact on the multi-agent system to be created.

Agent approach

A convenient metaphor for building software to interact with the range and diversity of online resources is that of an agent. An agent is a person that performs some task on your behalf. We would like to have a program that navigates the online resources to find the specific information that is strongly suspected to be there. You care about the result, and are happy to delegate the process to an assistant. You expect an agent to act even if all the details are not specified, or the situation changes. You expect an agent to communicate effectively with other agents. Agents can be viewed as a new model for developing software to interact over a network. This view has emerged because of the predominance of networks in the world. Information, knowledge, and electronic resources in general, are distributed across a network and programs and methods are needed to access them and present them in a customized manner. Using agents adds a layer of abstraction that localizes decisions about dealing with local peculiarities of format, knowledge conventions, etc. and thus helps to understand and manage complexity. Agents should therefore be seen as an abstraction that appears to provide a powerful way of conceptualizing, designing, and implementing a particularly complex class of software systems.

Multi-agent systems are systems composed of multiple interacting agents, where each agent is a coarse-grained computational system in its own right. The hypothesis/goal of multi-agent systems is creating a system that interconnects separately developed agents, thus enabling the ensemble to function beyond the capabilities of any singular agent in the setup. To arrive at a multi-agent solution, concepts such as those found in object-oriented computing, distributed computing, expert systems, etc. are necessary but do not suffice because distributed computing modules are usually passive and dumb. Also, their communications are usually low-level while multi-agent systems require high-level messages. Lastly, and importantly, multi-agent systems applications require a cooperation-knowledge level while these systems (object-oriented computing, expert systems, etc.) typically operate at the symbol and knowledge levels [13].

The approach of multi-agent systems seems to be a suitable framework for developing IC systems since many of the properties of IC systems or requirements on these systems such as being autonomous in that they are able to exercise control over their actions and act without user intervention, being adaptive (learning) in that they are able to change their behavior-based on their previous experience, and being proactive (goal-oriented) in that they are able to take actions that involve resource identification, query formulation and refinement, retrieval, and information organization for their users, coincide with those required on multi-agent systems and on agent-based systems in general (see, for example [14–16] for a thorough discussion of agent-based systems and their properties). The IC system proposed in this article for dealing with the problem of academic advising adopts the multi-agent paradigm.

Machine learning

Human expertise, needed for solving problems, should be transferred and transformed from some source of knowledge to a program. This transfer is usually accomplished by a series of lengthy and intensive interviews between a knowledge engineer, who is normally a computer specialist, and a domain expert who is able to articulate his expertise to some degree.

Unfortunately, the productivity of the interviewing process is typically so poor for many reasons ([17]): first of all, specialist fields have their own jargon, and it is often difficult for experts to communicate their knowledge in everyday language. Secondly, the facts and principles underlying many domains of interest cannot be characterized precisely in terms of a mathematical theory or a deterministic model whose properties are well understood. Thirdly, experts need to know more than the mere facts and principles of a domain in order to solve problems. For example, they usually know which kinds of information are relevant to which kinds of judgment, how reliable different information sources are, and how to make hard problems easier by splitting them into sub-problems which can be solved more or less independently. Eliciting this kind of knowledge is much more difficult than eliciting particular facts or general principles. Fourthly, human expertise, even in a relatively narrow domain, is often set in a broader context that involves a good deal of commonsense knowledge about the everyday world and it is difficult to delineate the amount and nature of general knowledge needed to deal with an arbitrary case, etc.

The rather low output of the knowledge acquisition phase has led researchers to look upon it as ‘the bottleneck problem’ of expert systems applications [18]. This dissatisfaction with the interview method has encouraged some researchers to try to

automate the process of knowledge acquisition by looking at the sub-field of Artificial Intelligence known as machine learning for a solution to the bottleneck problem. The idea is that a computing system could perhaps learn to solve problems in much the same way that humans do, that is to say, by example. A program is needed which learns the concepts of a domain under varying degrees of supervision from a human teacher.

In one approach, the teacher presents the program with a set of examples of a concept, and the program's task is to identify what collection of attributes and values defines the concept.

The field of machine learning has enjoyed a period of continuous growth and progress in recent years. Precise definitions of learning are hard to find, but most researchers would agree that it is a characteristic of adaptive systems which are capable of improving their performance on a problem as a function of previous experience, for example, in solving similar problems [19]. Thus learning is both a capability and an activity. Any learning program must have the ability to represent and reason about problem solving experience, as well as the ability to apply such representations and inferences to the solution of the current problem.

Learning programs are often classified in terms of the underlying strategy employed (see for example [20]). Roughly speaking, the strategy used depends upon the amount of inference that the program has to perform on the information available to it.

At one extreme, programs which learn by the direct implanting of new knowledge (for example by being reprogrammed, or being supplied with new data), are performing no inference at all. This is usually referred to as rote learning: a reasonable human analog would be the learning of multiplication tables. Programs using this approach to achieving problem-specific expertise in a computer are also called hand-built classifiers. Hand-built classifiers correspond to teaching by giving a person a domain theory without an extensive set of examples; one could call this learning by being told. Hand-built classifiers are non-learning systems (except insofar as they are later altered by hand). They simply do what they are told; they do not learn at the knowledge level [21].

At the other extreme, there is unsupervised learning: a generic term which covers tasks such as theory formation, which more closely resemble human efforts at scientific discovery.

Supervised learning is a kind of learning which can be regarded as having a strategy which is halfway between the two extremes mentioned above. In supervised learning, a program is typically presented with examples which help it to identify the relevant concept. These examples have known properties, which are normally represented as attribute-value pairs. The learning involved is supervised, because the examples provide the program with clues as to what it is looking for, as well as providing a space of attributes for its consideration.

The most common form of supervised learning is called inductive learning. An inductive learning program is one which is capable of learning from examples by a process of generalization. This kind of learning is sometimes also called empirical learning [22,23]. Empirical learning corresponds to giving a person lots of examples without any explanation of why the examples are members of a particular class. Empirical learning systems inductively generalize specific examples. Thus, they require little theoretical knowledge about the problem domain; instead they require a large library of examples.

Artificial neural networks (ANNs) are a particular method for empirical learning. ANNs have proven to be equal, or superior, to other empirical learning systems over a wide range of domains, when evaluated in terms of their generalization ability [24,25].

Although the almost complete ignorance of problem-specific theory by empirical learning systems may mean that they do not address important aspects of induction, it is interesting to see in the following study, how domain-specific knowledge about academic advising of students can be employed by a domain free neural network learning algorithm. A back-propagation neural network is used to automate the process of knowledge acquisition, i.e., acquire the expertise of the human academic adviser.

Web mining

Data Mining is a multidisciplinary field which supports knowledge workers who try to extract information in our ‘‘data rich, information poor’’ environment. Its name stems from the idea of mining knowledge from large amounts of data. The tools it provides assist us in the discovery of relevant information through a wide range of data analysis techniques. Any method used to extract patterns from a given data source is considered to be a data mining technique. When the data resides on the Web it is analyzed by means of Web mining techniques and the process is that of Web mining.

Given the vast and ever growing amount of information available in the Web and the fact that search engines do not seem to help much, how does the average user quickly find what he or she is looking for?

As mentioned earlier, IC systems seem to be the appropriate solution. The approach is to personalize the Web space – create a system which responds to user queries by potentially aggregating information from several sources in a manner that is dependent on the user's identity.

Existing commercial systems seek to do some minimal personalization based on declarative information directly provided by the user, such as their zip code, or keywords describing their interests, or specific URLs, or even particular pieces of information they are interested in (e.g. price for a particular stock). More elaborated solutions are eagerly awaited from applying new information customization

techniques, i.e., developing specific IC systems specialized on the Web – Web mining systems.

Current Web mining research aims at creating systems that (semi) automatically tailor the content delivered to the user from a Web site. This is usually done by mining the Web – both the contents, as well as the user's interaction [26]. Web mining, when looked upon in data mining terms, can be said to have three operations of interest – clustering (finding natural groupings of users, pages, etc.), associations (which URLs tend to be requested together), and sequential analysis (the order in which URLs tend to be accessed). As in most real world problems, the clusters and associations in Web mining do not have crisp boundaries and often overlap considerably. In addition, bad exemplars (outliers) and incomplete data can easily occur in the data set, due to a wide variety of reasons inherent to Web browsing and logging.

Thus, Web mining and personalization requires modeling of an unknown number of overlapping sets in the presence of significant noise and outliers (i.e., bad exemplars). Moreover, the data sets in Web mining are extremely large. The Web contains a mix of many different data types, and so in a sense subsumes text data mining, database data mining, image mining, and so on. The Web contains additional data types not available in large scale before, including hyperlinks and massive amounts of (indirect) user usage information. Spanning across all these data types there is the dimension of time, since data on the Web changes over time. Finally, there is data that is generated dynamically, in response to user input and programmatic scripts.

To mine data from the Web is therefore different from mining data from other sources of information. Interesting results are expected from novel mixings of these different data types to achieve novel goals. Also, any discussion of data mining from the Web requires a discussion of issues of scale [27]. In addition, scalable robust techniques to model noisy data sets containing an unknown number of overlapping categories should be developed [28].

The problem domain: academic advising

Having decided on a research direction, the following question emerges. What constitutes a good domain and problem? The key characteristic of an interesting domain is that there is a variety of resources in differing formats but there is some common overall structure. Too much structure reduces the problem to known methods. Too little structure makes the problem very difficult. Having structure is useful to guide the search and identification of relevant information.

We will illustrate our ideas using MASACAD, an example consisting of an E-Learning application. In this application the focus is on academic advising for students.

Academic advising

Advising Services are committed to supporting students throughout the development and achievement of their educational goals. The foundation of this support is the establishment of an educational partnership between the student and his or her academic advisor. Advisors often are thought of as the people who help students identify appropriate courses for their degree program. However, the course registration process is only a piece of the partnership that helps to guide students along other paths of their educational development. Advising Services adhere to a developmental approach to academic advising, where the advisor is a facilitator, and partner in the learning process. It is understood that the advisor, while a valuable resource and educator, is not the holder of all truths, but rather, will assist the student to discover and develop his/her educational plan. By working collaboratively with students, advisors help to ensure that they have a successful college experience.

An effective advising relationship is a partnership between the student and the advisor. Both the advisor and the student have responsibilities in this relationship. All students are therefore assigned an academic advisor whose interests are compatible with those of the major. The advisor is a very helpful companion who provides insight during the college experience. The advisor also provides some assistance with:

Course selection, major requirements, and scheduling.
Applications to graduate or professional programs.
Student activities, including student/faculty research. Employment opportunities.

But the student, as the advisee, must take the initiative and request to meet with his/her advisor. For most majors, regular meetings with the advisor are of benefit to the student during his/her college experience.

8.2. Why a software assistant is needed for academic advising?

Academic advising is designed to provide the student with the information and the support that he/she needs to make informed plans and decisions about his/her program and to achieve his/her educational goals. The student is always in charge of his/her education – he/she makes the decisions. Academic advisors can help the student learn what he/she needs to know, sort out his/her questions, think through his/her options, identify shortcuts, avoid pitfalls, and make his/her program right for him/her. Advisors help the student find opportunities, avoid problems, and work through any difficulties he/she might encounter. Just as one might look to a lawyer for advice on a legal problem, or to a financial advisor for help with investment plans, one may think about an academic advisor as the consultant of the student on his/her educational planning.

Academic advising is an important key to academic success. The relationship between quality advising and student success is significant. That is why many people on campus are involved in the process. Faculty, staff, students and professional advisors contribute to the advising relationship by:

Encouraging ongoing, supportive, and informed contact with students.

Explaining policies, procedures, and academic requirements. Exploring skill levels, such as writing, mathematics, and study skills.

Making referrals to campus resources.

Assisting with degree planning as well as career exploration and preparation.

Ultimately, however, the responsibility for seeking adequate academic advising belongs to the student. Students must: Know and meet degree requirements.

Ensure timely progress toward a degree through appropriate course selection.

Be aware of current academic and departmental information.

In order to help the student, improve the advising process and make it easier, and overcome the many problems that may occur such as:

The limited number of advisors that are available in contrast to the huge number of students.

The advisors are not available all the time.

Some advisors are new to the university and do not have enough knowledge/experience with advising.

Serious consequences may occur if mistakes are made, for example, during course selection.

An intelligent assistant in form of a computer program will be of great interest. Such an intelligent assistant will automate the advising process in the university, and hence, help the university in its efforts towards introducing new technologies. It will also simplify the task of faculty, staff, students, and professional advisors and make it possible to save time and effort and prevent mistakes. All these advantages are added to the many advantages of any assistant software that is used to solve problems that ordinarily require human expertise such as:

Permanence: expertise does not leave with personnel.

Multiple experts: knowledge from multiple experts can be combined.

Increased reliability: increased confidence that a decision is correct.

Fast response: may respond faster than a human.

Steady and unemotional: stress and fatigue are known to affect human performance.

Increased process quality: provide consistent advice, therefore reducing the size and rate of errors.

Restriction of the general goal of academic advising

The general goal of academic advising is to assist students in developing educational plans which are consistent with academic, career, and life goals and to provide students with information and skills needed to pursue those goals. More specifically, advisors will assist students in the following ways:

Guide students through the university's educational require-ments.

Assist in scheduling the most appropriate courses. Introduce them to pertinent resources.

Promote leadership and campus involvement. Assist in career development.

Assist students with the timely completion of their degree. Help students find ways to make their educational experience personally relevant.

The goal of academic advising, as stated above, is too general because many experts are involved and because a huge amount of expertise is needed. Hence, realizing an intelligent software assistant that is able to deal with all the details shown above will be too difficult, if not impossible. In the following implementation we will therefore restrict academic advising and understand it as just being intended to provide the student with an opportunity to plan programs of study, select appropriate required and elective classes, and schedule classes in a way that provides the greatest potential for academic success.

The task is still interesting and of moderate size since when planning a program of study and selecting classes, there are quite a lot of things to consider such as:

Prerequisites.
Course availability.
Effective course sequencing.
Work load.
Instructor–student match-ups.

Later, when the major problems are understood, improvements and extensions can be tempted, and attempts can be made to tackle the advising problem in a more general framework.

Resources needed for academic advising

There is a lot of diverse resources that are required to deal with the problem of academic advising:

First of all, one needs the student profile that includes the courses already attended, the corresponding grades, the desires of the student concerning the courses to be attended, and perhaps many other information. The part of the profile consisting of the courses already attended, the corresponding grades, etc., is maintained by the university administration in appropriate databases to which the access is restricted to some administrators. The part of the profile consisting of the desires of the student concerning the courses to be attended exists actually only in the head of the student and should therefore be asked for from the student before advising is performed. However, attempts may be made to let the system learn the desires, for example, by monitoring the student or/and looking at his/her profile.

The second resource needed for solving the problem of advising are the courses that are offered in the semester for which advising is needed. This information is as well maintained by the university administration in appropriate Web sites and is accessible for everyone.

The third resource needed for solving the problem of academic advising is expertise. Expertise is the extensive, task-specific knowledge acquired from training, reading, and experience. It is the knowledge that allows experts to make better and faster decisions than non-experts when solving complex problems. It consists of facts, theories, as well as rules and procedures about a problem area. For the problem of academic advising this type of expertise may be referred to as the university laws concerning academic advising.

Why a multi-agent system?

First of all, academic advising is intended to be a good domain and problem to test the adequacy of the multi-agent paradigm for dealing with information customization. It provides a complex and dynamic environment and constitutes a wonderful experimental test-bed for investigating the issue. Conversely, dealing effectively with the problem of academic advising will require a multi-agent system. The multi-agent paradigm seems to be appropriate and even superior to other approaches such as a traditional distributed database approach enhanced with an intelligent user interface for the reasons elaborated in the following.

The resources needed for academic advising are physically distributed and dynamic:

Their content may change: it is for example frequent that, at the beginning of a semester, some of the offered courses are canceled, and some other ones are added. The profile of the student also changes frequently, for example when the grades of the exams taken are entered. Also the interests of the student concerning the courses to be attended may change. As an example, students usually want to enroll themselves in courses that are attended by some of their friends. Changes to the university regulations, in contrast, especially those concerning the advising process like the curriculums for the different majors, are comparatively less frequent.

Their form (structure) may change: they may be available via a Web page, an intelligent agent, a database, a legacy system, etc.

Their location may change: existing ones may be moved and new ones may be incorporated.

Hence, dedicating a separate intelligent agent to each individual resource for coping with all of its peculiarities will have many advantages such as:

Reduction of the scope of changes: there is no need to change the whole system when changes concern only a specific resource. In this case only the concerned agent is changed.

Easy incorporation of new resources: only one new agent is needed for each new resource.

Easy extension and improvement of the system: we may think, for example, of a self-adjusting system, i.e., the system itself searches for appropriate resources and each time a resource is identified, it is wrapped with an appropriate agent.

Details of the agent-based solution

Because, as mentioned earlier, all the resources needed for academic advising can be physically distributed and are also dynamic, the customized presentation of information for the student should be updated continuously as new information becomes available. This happens with no user intervention using an autonomous and learning multi-agent system. The

system monitors the changes and alerts the user automatically when something changes.

The Bee-gent system

As a solution to the problems of network communication, we use Bee-gent (Bonding and Encapsulation Enhancement Agent) [29], a communication framework based on the multi-agent model. The Bee-gent framework is comprised of two types of agent. ‘‘Agent wrappers’’ are used to agentify (i.e. providing an agent interface) existing applications, while ‘‘Mediation Agents’’ support inter-application co-ordination by handling all communications. The mediation agents move from the site of an application to another where they interact with the agent wrappers. The agent wrappers themselves manage the states of the applications they are wrapped around, invoking them when necessary.

The Bee-gent system has many desirable features. The mediation agent manages the coordinating interactions of the various applications in a unified manner. It is easy to maintain consistency because it is not necessary to divide and distribute the coordinating interactions on the basis of each individual application, and therefore development is simplified. It is also easy to modify the system configuration and the coordinating interactions because these interactions are encapsulated.

When the mediation agent migrates it carries its own program, data and current state. Frequency of communication is reduced compared to a purely message-based system and network loads are decreased largely because communication links can be disconnected after launch of the mediation agent. Processing efficiency is improved because the mediation agent communicates with the applications locally.

Conclusions

In this paper we were able to demonstrate on the E-Learning example of MASACAD how the multi-agent paradigm, combined with ideas from machine learning, user modeling, and Web mining, can be used to approach a solution for the problem of information customization.

MASACAD is a decision support system and at the same time an information customization system. Without MASACAD, the student, in order to take the decision about course enrolment, has to invest significant effort to find the right information which is distributed, dynamic, and available in different formats.

Many of my students have had the opportunity to run the system and test it. It was interesting to observe that the most fascinating point for them was to see the system presenting the information found in a somehow customized manner for each individual student. The results are encouraging and future work will be focused on improving the system and studying how such simple examples built with insight should lead to identification of key difficulties, useful abstractions and a general method for solving the problem and revelation of the issues.

References

[1] M.S. Hamdi, An agent-based approach for intelligent user interfaces, in: Proceedings of the Fourth International Conference on Intelligent Pro-cessing and Manufacturing of Materials (IPMM-03), Sendai, Japan, 2003.

[2] M.S. Hamdi, Information extraction using multi-agents, in: Proceedings of the International Conference on Internet Computing (IC ’03), Las Vegas, Nevada, USA, 2003.

[3] M.S. Hamdi, Automation of information extraction, in: Proceedings of the Seventh International Conference on Automation Technology (Automation 2003), National Chung Cheng University, Chia-yi, Taiwan, 2003.

[4] M.S. Hamdi, A multi-agent based approach for retrieving information, in: Proceedings of the Second IASTED International Conference on Communications, Internet and Information Technology (CIIT 2003), Scottsdale, Arizona, USA, 2003.

[5] M.S. Hamdi, Extracting and customizing information using multi-agents, in: A. Scime (Ed.), Web Mining: Applications and Techniques, Idea Group Inc., 2005, ISBN: 1-59140-414-2, pp. 228–252.

[6] M. Moore, Distance Education: A Systems View, Wadsworth Publishing Company, 1996. http://www.cde.psu.edu/de/what_is_de.html#definition.

[7] J. Mostafa, Guest editor’s introduction: information customization, IEEE Intell. Syst. 17 (6) (2002) 8–11.

[8] G.R. Andrews, Foundations of Multithreaded, Parallel and Distributed Programming, Addison-Wesley, 2000.

[9] A. Kobsa, User modeling in dialog systems: potentials and hazards, AI Soc.: J. Human Machine Intell. 4 (3) (1990) 214–231.

[10] E. Rich, User modeling via stereotypes, Cognitive Sci. 3 (1979) 329–354.

[11] F. Sebastiani, Machine learning in automated text categorization, ACM Comput. Surv. (CSUR) 34 (1) (2002) 1–47.

[12] T.M. Mitchell, Machine Learning, McGraw-Hill, 1997.

[13] A. Newell, The knowledge level, Artif. Intell. 18 (1982) 87–127.

[14] J.M. Bradshaw, An introduction to software agents, in: J.M. Bradshaw (Ed.), Software Agents, AAAI Press, Menlo Park, California, 1997, pp. 3–46.

[15] S. Franklin, A. Graesser, 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, published as Intelli-gent Agents III, Springer-Verlag, 1997, pp. 21–35.

[16] E.M. Stuart, Interface agents: a review of the field, Technical Report Number: ECSTR-IAM01-001, University of Southampton, August 2000, ISBN 0854327320.

[17] P. Jackson, Introduction to Expert Systems, 3rd ed., Addison-Wesley, Harlow, England, 1999.

[18] E.A. Feigenbaum, The art of artificial intelligence: themes and case studies of knowledge engineering, in: Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 1977, pp. 1014–1029.

[19] H.A. Simon, Why should machines learn? in: R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning, Tioga, Palo Alto, CA, 1983 (Chapter 2).

[20] J.G. Carbonell, R. Michalski, T. Mitchell, An overview of machine learning, in: R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning, Tioga, Palo Alto, CA, 1983 (Chapter 1).

[21] T.G. Dietterich, Learning at the knowledge level, Mach. Learn. 1 (1986) 287–316.

[22] J.R. Quinlan, C4.5: Programs for Empirical Learning, Morgan Kaufmann, San Francisco, CA, 1993.

[23] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland

(Eds.), Parallel Distributed Processing, vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.

[5] J.W. Shavlik, R.J. Mooney, G.G. Towell, Symbolic and neural net learning algorithms: an empirical comparison, Mach. Learn. 6 (1991) 111–143.

[6] L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, J. Connor, D. Park, M. El-Sharkawi, R.J. Marks, A peerformance comparison of trained multi-layer perceptrons and trained classification trees, Proceedings of IEEE 78 (1990) 1614–1619.

[7] R. Cooley, B. Mobasher, J. Srivastava, Web mining: information and pattern discovery on the World Wide Web, in: Proceedings of the Ninth IEEE International Conference on Tools with Artificial Intelligence (ICTAI '97), Newport Beach, CA, November, 1997.

[8] M. Hearst, Distinguishing between web data mining and information access, in: Proceedings of the Knowledge Discovery (KDD-97), Newport Beach, CA, 1997.

[9] R. Krishnapuram, A. Joshi, O. Nasraoui, L. Yi, Low complexity fuzzy relational clustering algorithms for web mining, IEEE Trans. Fuzzy Syst. 9 (4) (2001) 596–607.

[10] T. Kawamura, T. Hasegawa, A. Ohsuga, S. Honiden, Bee-gent: bonding and encapsulation enhancement agent framework for development of distributed systems, Syst. Comput. Jpn. 31 (13) (2000) 42–56.

[11] T.G. Dietterich, Machine learning research: four current directions, AI Mag. 18 (4) (1997) 97–136.

[12] D.H. Wolpert, The lack of a priori distinctions between learning algo-rithms, Neural Comput. 8 (7) (1996) 1341–1390.

[13] C. Schaffer, A conservation law for generalization performance, in: W. Cohen, H. Hirsh (Eds.), Proceedings of the 11th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1994, pp. 259–265.

[14] J.L. McClelland, D.E. Rumelhart, G.E. Hinton, The appeal of parallel distributed processing, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, MIT Press, 1986, pp. 3–44.

[15] J.V. Haxby, E.A. Hoffman, M.I. Gobbini, The distributed human neural system for face perception, Trends Cogn. Sci. 4 (6) (2000) 223–233.

[16] R.R. Poznanski (Ed.), Biophysical Neural Networks: Foundations of Integrative Neuroscience, Mary Ann Liebert Inc. Publishers, New York, 2001, ISBN: 0-913113-90-5.

[17] D.O. Hebb, The Organization of Behaviour, John Wiley, New York, 1949.

[18] P. Lewicki, T. Hill, M. Czyzewska, Nonconscious acquisition of infor-mation, Am. Psychol. 47 (6) (1992) 796–801.

[19] D. Patterson, Artificial Neural Networks, Prentice Hall, Singapore, 1996.

[20] L. Fausett, Fundamentals of Neural Networks, Prentice Hall, New York, 1994.

[21] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan Publishing, New York, 1994.

[22] P.J. Werbos, Beyond regression: new tools for prediction and analysis in the behavioral sciences, Ph.D. Dissertation, Committee on Mathematics, Harvard University, Cambridge, MA, 1974.

[23] C. Bishop, Neural Networks for Pattern Recognition, University Press, Oxford, 1995.

[24] M. Watson, Practical Artificial Intelligence Programming in Java, 2002, http://www.markwatson.com/opencontent/.

[25] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, D.B. Rosen, Fuzzy ARTMAP: a neural network architecture for incremental super-vised learning of analog multidimensional maps, IEEE Trans. Neural Networks 3 (1992) 698–713.

[26] A.H. Tan, Adaptive resonance associative map, Neural Networks 8 (3) (1995) 437–446.

[27] C. Brown, L. Gasser, D.E. O'Leary, A. Sangster, AI on the WWW supply and demand agents, IEEE Expert 4 (1995) 50–55.

[28] O. Etzioni, D.S. Weld, Intelligent agents on the internet – fact, fiction, and forecast, IEEE Expert 4 (1995) 44–49.

[29] B. Hermans, Intelligent Software Agents on the Internet: an inventory of currently offered functionality in the information society and a prediction of (near-)future developments, Ph.D. Dissertation, Tilbury University, Tilbury, The Netherlands, 1996.

[30] J.R. Oliver, On artificial agents for negotiation in electronic commerce, Ph.D. Dissertation, University of Pennsylvania, 1996.

[31] Y. Arens, S. Feiner, J. Foley, E. Hovy, B. John, R. Neches, R. Pausch, H. Schorr, W. Swartout, Intelligent User Interfaces, Report ISI/RR-91-288, USC/Information Sciences Institute, Marina del Rey, California, 1991.