Overview of Intelligent Search Agents

Patrick Hoey
University of Massachusetts At Lowell

Abstract

This paper will discuss the brief background of agents and the benefits of using this technology. I will then go more in depth with the specific model of intelligent agents and the basic framework used to deploy agents on a system. Then I will discuss the issues related to intelligent agents, as well as how the different AI models are applied to the different intelligent search agents.

1 Introduction

Intelligent agents in general are meant to be autonomous objects that perform user delegated tasks. From the user perspective, it is tedious to have to explicitly state every detail of the task for the software agent, and by definition of an intelligent agent, the user would expect the agent to be able to infer what the user means when asked.

Originally intelligent agent technology was developed to study the computational models of distributed intelligence, but today there are more practical uses for this technology. The first would be simplifying the complexities of distributed computing and second would be overcoming the limitations of current user interface approaches. Both of these can be seen in a growing trend towards greater abstraction of interfaces to computing services (Bradshaw 1997).

The benefits of using intelligent agent systems can be explained first by looking at the current state of technology. Current distributed systems are operated directly by a user, typically through a graphical user interface (GUI), to perform some task. The problems with this approach is that if the user is looking for information, the search space could be quite large. Another problem is that the actions of some systems are in response to the immediate user interaction. Also, the interface (function driven) is quite rigid on its structure, not being able to adapt to the user.

With intelligent agent technology, the agent system can be scalable, where plugging in more agents to a system (since it is component driven) should be a relatively trivial task. Also, the tasks can be scheduled or even-driven, not dependant on the users immediate input. The agent system is relatively easy to modify, unlike a GUI, just by adding in the agent components to solve the problem. Also, the agent system is task driven, so instead of the user specifying each and every function, this interface can be abstracted out into tasks. The intelligent agent system can also adapt to the users preferences, making task management a trivial exercise.

2 Attributes of an Agent

The minimal set of characteristics that an intelligent agent must possess is as follows (Caglayan 1997):

Delegation: This is a characteristic where the agent performs a set of tasks on behalf of a user or another agent that are explicitly approved by that user.

Communication skills: The agent needs to be able to interact with the user to receive task delegation instructions, and inform task status and completion through an agent-user interface or through an agent communication language.

Autonomy: The agent operates on its own to perform the tasks delegated by the user. This attribute can be anything from running a nightly backup of the code-base to negotiating the best price of a product to the user.

Monitoring: The agent needs to be able to monitor its environment in order to be able to perform tasks autonomously.

Actuation: The agent needs to be able to affect its environment in order to be able to perform tasks autonomously.

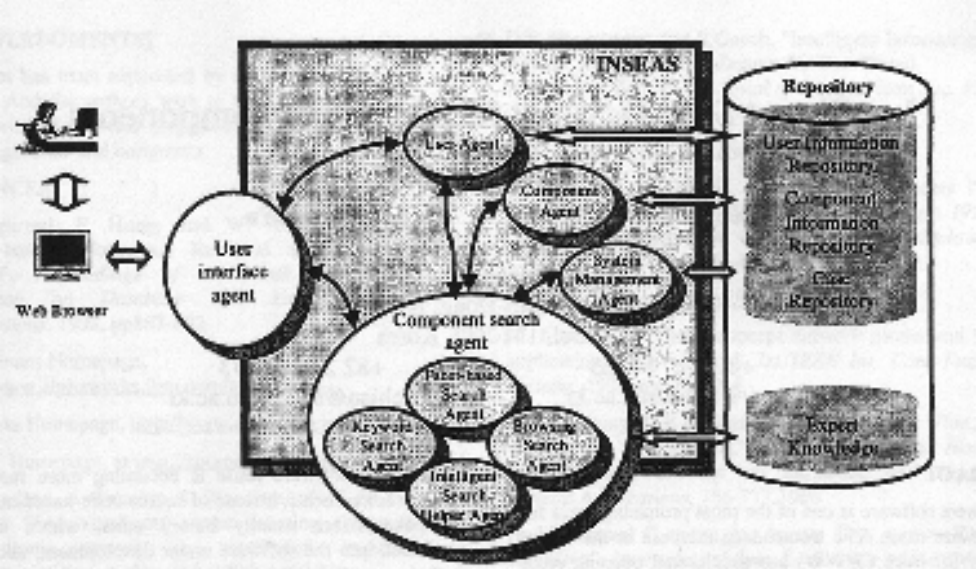

Figure 1. Overall Architecture of INSEAS

Intelligence: The agent needs to be able to interpret the monitored events to make actuation decisions for autonomous operation.

All of these characteristics are important for an intelligent search agent. The user would delegate to the search agent (who is monitoring the system for requests) a task of finding information through a query interface. The search agent should already have a list of resources which reflect the users personal interests. The search agent would then autonomously retrieve documents, filter out the irrelevant information (actuation) and return the results. The user would communicate with the agent through an interface, about what material the user wants and doesn't want. This will improve the search results as the intelligence aspect is that the search agent will modify its search and filter approaches based on the users choices.

3 Agent System Framework

3.1 Overview of Framework

Intelligent search agents are composed of many important attributes for them to function properly. Now we must go from a bottom-up approach to a top-down approach and look at the entire architecture from a component system view. The framework for which an intelligent search agent system should be modeled after is represented in Figure 1 (Park 1999).

The architecture is called the intelligent search agent system. In this multi-agent system, several autonomous agents interact with each other to perform some set of complex and multi-domain tasks, which would be inefficient if delegated to one agent. Search agents should use expert knowledge, rule-base reasoning and case-based reasoning for the intelligent search and user interface.

3.2 Object View of Architecture

The user interface agent should provide an efficient search environment and user-friendly representation of the search results. For the agent to adapt to the users preferences, it should allow the user a feedback mechanism where the user can give priorities to similar words for the query. The role of the user interface agent is to provide users with a convenient and efficient search environment.

The component search agent should be able to execute at least the four basic types of search methods: keyword-based search, facet-based search (searches based on preferences), browsing-based search and interactive search with a helper agent (feedback method).

The keyword search agent would receive keywords from user and retrieve similar words based on relevance and relationships between concepts (interacts with the expert knowledge repository).

The facet-based search agent receives the preferences from the user interface agent, calculates the weights between words and relevance between concepts (interacts with the expert knowledge repository).

The browsing search agent builds a category tree where the user browsers through the different classification factors (interacts with the expert knowledge repository).

The intelligent search helper agent will assist the user or other agents when necessary. It uses rule-based reasoning and feedback interaction with the user to determine what information the requester needs (interacts with the expert knowledge repository).

We have three agents that act as brokers between the other agents. The user agent handles all the preferences and requests and hands them off to either the component search agent or the user information repository for storage of user preferences. The component agent interacts directly with the component search agent and gathers and stores the environment type information in the component information repository. The system management agent handles system search performance and therefore stores and retrieves data from the case repository, which stores the user queries and other specifics about the searches.

3.3 Profile-Article Comparison

The comparison of a user profile with a document involves matching the structural similarity between a profile graph (where G = (V,E)) and the retrieved documents, using profile weights to influence this comparison. The type of algorithms to use with this approach are ones related to neighborhoods, where a neighborhood of a node is defined as the set of nodes accessible from it, constrained by a specific path length (O'Riordan 1995).

The neighborhood algorithm that our search agents would use is:

$$NH(A,B) = 1/(n^2 - n) \sum \alpha_A(v, v') \Delta \alpha_B(v, v')$$

where $1/(n^2 - n)$ is the number of edges that match, divided by the total possible number of edges, and iterates for each unique node surrounding the particular node.

The Δ acts like a symmetric difference operator, familiar to programmers as an "exclusive or" (A ^ B), which helps compare all nodes surrounding a particular node which occur in only one of the two graphs.

This comparison algorithm for graphs captures the fact that a phrase in a retrieved document that also occurs in the profile is an important signifier of the relevance of the articles compared to the profile.

3.4 Relevance Feedback

Via the user agent, when dealing with searches, the user may provide relevance feedback based on those articles retrieved. A tag is attached to the article based on whether or not it is relevant. Based on this tag, the weights (numerical relevance on edges of undirected graph) are modified using a vector space model, where a document is represented as a vector of attributes. These weights are calculated dynamically from weights associated with the terms in the profile:

$P_{j+1} = \alpha P_j + \beta D$, if D is tagged as relevant
$P_{j+1} = \alpha P_j - \gamma D$, if D is tagged as not relevant

P_j is the vector of query term weights (calculated from the word's frequency in the profile and the dataset of articles being examined).

D is the article given as feedback.

D shifts the vector P_j (by incrementing the position or decrementing the position), so that P_j's new position is a better representation of the users interests and needs.

After each iteration of this algorithm, we need to calculate the new profile edge weights to use in this algorithm, to basically get more specific dataset that represents the user's preferences:

$$w(k, l) = (P_{j,k} + P_{j,l}) / 2$$

w(k , l) is the new edge weight for an edge joining these two terms if an edge exists.

4 Issues

The main issues when dealing with agents fall into seven distinct categories (Wooldridge 1998), varying in granularity from the top of the organization to the development team.

Political Pitfalls: One problem would be where the organization oversells the agents. This is important because agent technology may make it easier to solve certain problems, but they do not solve everything. Another problem us when the organization gets religious or dogmatic about using agents, which is naive since software agents are not the universal solution to every problem.

Management Pitfalls: One problem would be management that doesn't know why they want software agents, but just care about using the buzzword to develop the next product. Another problem is when management doesn't know what agents are good for, which represents the lack of understanding about the degree of its applicability to a problem.

Conceptual Pitfalls: This problem arises when the development team believes that agents are a silver bullet, but there is no proof as of now that agents provide any type of advance in software development. Another problem is where the developers confuse buzzwords with concepts, believing falsely that knowing some technical jargon from software agents gives them an in depth understanding of the complex nature of the software agents.

Analysis and Design Pitfalls: One common problem with agent technology is that the development team does not exploit related technology, but instead reinvents the wheel for each project. Another aspect of agent development is not exploiting concurrency, which is one of the benefits of an agent system, that multiple agents can be running concurrently on multiple threads, all executing different tasks.

Micro- Agent Level Pitfalls: A common problem with this aspect of the development cycle is where the team wants to develop a generic architecture for the agents, so that they can reuse it over multiple problem domains. This is also false, because each problem when dealing with agents needs a unique solution for the specific problem domain. Another problem is that the agents themselves when developed do not have any intelligence (or any artificial intelligence), which defeats the purpose of using agents.

Macro- Society Level Pitfalls: These problems arise when systems have either too many agents or too few agents for their specific problem, which either leads to unnecessarily complex systems or a system where a few agents do all the work, which violates the coherence property in object oriented design.

Implementation Pitfalls: The first problem is that the team wishes to implement the agent technology from a clean slate, but forgets that there is mission critical legacy code that must be supported with the new technology. It is essential to work with these components, since they can not be ignored or replaced. Another problem is ignoring the standards that are already in place when developing software agent components of an application, which can save valuable development time.

5 Artificial Intelligence

To conceptualize the artificial intelligence (AI) model for any intelligent agent architecture, a person must contrast traditional AI concepts with what is expected in the agent system (Maes 1994) .

5.1 Traditional AI Concepts

In traditional artificial intelligence, an intelligent system is typically composed of functional components, such as perception, execution, natural language communication, a learner, planner and inference engine. The central representation includes things such as beliefs, which are updated by the perception component and processed by the inference engine and natural language component. Desires and intentions are produced by the planner component.

The functional components of the system are modeled as general and as domain-independent as possible.

The artificial intelligence system has a complete, correct internal representation of its environment.

The AI modules are a sequential organization within the system, where one set of modules

performs functions, and after it is finished, another set of modules perform their functions.

The activities of the AI modules are modeled as a result of "deliberative thinking" process, where all the modules work to attain a common system goal.

Finally, the AI modules typically consist of compilation or reformulation of what a system already knows.

5.2 Intelligent Agent AI Concepts

When dealing with intelligent agents, the system is composed of competence modules, or behaviors, where each module is task oriented.

Each module is responsible for all functionality needed to perform its given tasks, which creates highly specialized modules which are domain dependant.

There is little emphasis for an agent system to model its environment, since the modules have their own tasks to complete.

Intelligent agent systems are highly distributed and decentralized by nature, making for a highly efficient concurrent processing system, which can do many tasks at the same time.

The agents activity us not modeled as a result of deliberate thinking. There is no common system goal to complete other then each individual module completing their own tasks.

Finally, in an intelligent agent system, learning and development are crucial aspects of an adaptive autonomous agent.

6 Summary

In summary, we can see the benefits as well as the issues regarding implementing this technology. The basic framework is a good start when implementing an intelligent search agent system, while keeping the development time and maintenance to a minimum.

Building an object model of the tasks you wish to accomplish within your system is a start. The key concepts with this is the communication between the agent modules. There should be a low coupling, where each module has a low dependence on other agents, as they should be able to perform their tasks with minimal overhead. There should also be a high cohesion, where the modules all share a strong relationship towards achieving their tasks related to the system. It is also important to understand the fundamental differences between the traditional AI model and the agent system model, as they accomplish two separate agendas. By following some of the guidelines herein, the intelligent search agent system should run into few implementation or deployment problems.

References

[1] Piszcz, Alan. "A Brief Overview of Software Agent Technology". July 1998. The Mitre Corporation. http://www.soc.staffs.ac.uk/~cmryl/refpapers/abosatrel003.pdf.

[2] Bradshaw, Jeffrey M. "An Introduction to Software Agents". 1997. Software Agents, The MIT Press. http://www.soc.staffs.ac.uk/~cmryl/refpapers/01-bradshaw.pdf.

[3] Maes, P. "Modeling Adaptive Autonomous Agents". 1994. Artificial Life Journal, MIT Press. vol 1, no. 2. pp. 135-162. http://www.soc.staffs.ac.uk/~cmryl/refpapers/alife-journal.ps

[4] Seoyoung Park and Chisu Wu. "Intelligent Search Agent for Software Components". 1999. Software Engineering Conference. pp. 154 -161. http://www.soc.staffs.ac.uk/~cmryl/refpapers/00809596.pdf

[5] Gerd Wagner. "Artificial Agents and Logic Programming". 1997. ICLP'97 Post-Conference Workshop on Logic Programming and Multiagent Systems. http://www.soc.staffs.ac.uk/~cmryl/refpapers/lpmas_ps.ps

[6] Hyacinth Nwana and Divine Ndumu. "A Perspective on Software Agents Research". 1999. The Knowledge Engineering Review. http://www.soc.staffs.ac.uk/~cmryl/refpapers/hn-dn-ker99.ps

[7] Björn Hermans. "Intelligent Software Agents on the Internet". 1997. First Monday, vol. 2, no. 3. http://www.firstmonday.dk/issues/issue2_3/ch_123/

[8] Kathryn Heilmann , Dan Kihanya , Alastair Light , Paul Musembwa "Intelligent Agents: A Technology and Business Application Analysis". 1995. University of California at Berkeley. http://www.mines.u-nancy.fr/~gueniffe/CoursEMN/I31/heilmann/heilmann.html

[9] James Jansen. "Using an Intelligent Agent to Enhance Search Engine Performance". 1997. First Monday, vol. 2, no. 3. http://www.firstmonday.dk/issues/issue2_3/jansen/

[10] G. Michael Youngblood. "Web Hunting: Design of a Simple Intelligent Web Search Agent". 1999. ACM Crossroads, vol. 5, no. 4. http://www.acm.org/crossroads/xrds5-4/webhunting.html

[11] Donna Haverkamp and Susan Gauch. "Intelligent Information Agents: Review and Challenges for Distributed Information Sources". 1997. Journal of the American Society for Information Science. http://www.ittc.ukans.edu/~sgauch/papers/JASIS97.ps

[12] Chen, L. and Sycara, K. "WebMate: A Personal Agent for Browsing and Searching". 1998. Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems (ACM, Agents '98), pp. 132-139. http://www.ri.cmu.edu/pub_files/pub1/chen_liren_1998_1/chen_liren_1998_1.pdf

[13] Bogonikolos, Nikos. and Fragoudis, Dimitris. "Archimides: An intelligent agent for adaptive - Personalized Navigation within a Web Server". 1999. Proceedings of the 32nd Hawaii International Conference on System Sciences. http://www.computer.org/proceedings/hicss/0001/00015/00015020.PDF

[14] MAJ Bernard J. Jansen. "A Software Agent for Performance Improvement of Existing Information Retrieval Systems". 1999. http://www.iuiconf.org/99pdf/1999-003-0037.pdf

[15] Steiner, Thomas. "Software Agents in Tourism Industry-Prototypes and Prospects". 2000. Intelligent Agents, vol. 1, no. 1. http://www.svifsi.ch/revue/pages/issues/n001/a001Steiner.pdf

[16] Eliassi-Rad, Tina. "Building Intelligent Agents That Learn to Retrieve and Extract Information". 2001. ftp://ftp.cs.wisc.edu/machine-learning/shavlik-group/eliassi-rad.thesis.pdf

[17] Mathias Bauer , Dietmar Dengler , Gabriele Paul, Markus Meyer. "Programming by example: programming by demonstration for information agents". 2000. Communications of the ACM, vol. 43, no. 3, pp. 98-103. ACM Press. ISBN/ISSN number: 0001-0782.

[18] Michael Wooldridge and Nicholas R. Jennings. "Pitfalls of agent-oriented development". 1998. International Conference on Autonomous Agents, pp. 385 - 391. ACM Press. ISBN/ISSN number: 0-89791-983-1

[19] Adrian O'Riordan and Humphrey Sorensen. "An intelligent agent for high-precision text filtering". 1995. Conference on Information and Knowledge Management, pp. 205 - 211. ACM Press. ISBN/ISSN number: 0-89791-812-6

[20] Ralph Depke , Reiko Heckel and Jochen M. Kuster. "Improving the agent-oriented modeling process by roles". 2001. International Conference on Autonomous Agents, pp. 640 - 647. ACM Press. ISBN/ISSN number: 1-58113-326-X.

[21] Caglayan, A., Harrison, C. "Agent Sourcebook". 1997. Wiley. ISBN/ISSN number: 0-471-15327-3