□특집□

인터페이스 에이전트

윤 충 화[†] 이 형 일[††] 임 윤 택[†††]

◈ 목 차 ◈

최근에 들어 지능형 소프트웨어 에이전트 (Intelligent Software Agents) 라는 용어는 소프트웨어 개발자들이 많이 사용하고 있으며, 또한 에이전트 기법을 연구하는 대학들과 연구소, 그리고 적용한 제품을 개발하는 회사들은 나날이 늘어나고 있는 추세이다. 에이전트 분야는 인공 지능 분야의 하나로 간주되는 분산 인공지능 (Distributed Artificial Intelligence)의 멀티 에이전트 시스템으로부터 발전되었다고 볼 수 있다.

본 연구에서는 소프트웨어 에이전트에 대한 정의와 분류에 대해 살펴보고, 그중 개인적인 비서 또는 교사의 역할을 담당하는 인터페이스 에이전트의 특성과 구현된 다양한 사례를 조사하고, 구현 기법들을 분석하였다.

† 정회원 : 명지대학교 컴퓨터공학과 부교수
†† 정회원 : 김포전문대학교 전자계산과 전임강사
††† 준회원 : 명지대학교 대학원 컴퓨터공학과 석사과정

1. 서 론

최근에 들어 지능형 소프트웨어 에이전트 (Intelligent Software Agents) 라는 용어는 소프트웨어 개발자들이 많이 사용하고 있으며, 또한 학계나 연구소에서도 활발히 연구되고 있는 분야이며, 에이전트 기법을 연구하는 대학들이나, 적용한 제품을 개발하는 회사들은 나날이 늘어나고 있는 추세이다. 에이전트 분야는 크게 분류하여 인공 지능 분야의 하나로 간주되는 분산 인공지능 (Distributed Artificial Intelligence)의 멀티 에이전트 시스템으로부터 발전되었다고 볼 수 있다.

소프트웨어 에이전트와 기존 인공 지능 분야의 차이점은 문제에 대한 접근 방법의 측면에서 고찰해 볼 수 있다. 기존의 전형적인 인공지능 기법은 일반적인 문제 해결기 (General Problem Solver)를 구현하려는 시도로부터 출발하였으며, 기본적으로 지식 표현 기법과 기호와 논리를 이용한 추론 기법에 의존한다. 목표의 설정이 초기부터 너무 비현실적이었으며, 실제 분야에 적용하기에는 실용성이 없다는 결과를 초래하였다. 그러므로 적

용 도메인을 제한하기 시작하였으며, 그 결과로 실제 사용할 수 있는 전문가 시스템 또는 규칙 기반 시스템들이 구현되어 다양한 분야에 많이 사용되고 있다.

이러한 인공지능 기법의 한계로 인하여 새로운 접근 방법들이 시도되었으며, 그중 대표적인 것이 인간 또는 동물의 생체 구조를 이용한 신경망 기법이다. 또한 소프트웨어 에이전트도 이와 같은 맥락으로 간주되고 있다. 문헌에 의하면 이 같은 시도를 바텀업 학파 (bottom-up school)라고 부른다. [1] 이는 지능형 시스템을 개발하기 위하여 전형적으로 사용되던 논리 (rules of logic)를 생물학적인 구조로 대치하려는 시도이며, 단순한 기능을 구현하는, 그리고 비교적 작은 크기를 갖는 프로그램들이 서로 상호 작용을 통하여 시스템 전체의 지능을 구현하려는 것이다.

1.1 지능형 소프트웨어 에이전트의 정의

문헌 조사에 의하면, 여러 학자들이 내린 다양한 소프트웨어 에이전트에 대한 정의를 찾아 볼 수 있으며, 모두가 동의하는 하나의 완벽한 정의를 내리는 것은 인공지능에 대한 정의를 하는 것과 마찬가지로 불가능하다고 사료된다. 그러나 인공지능의 발전과정을 살펴보면, 에이전트의 개념은 1970년대에 Carl Hewitt가 제안한 병행 액터 모델 (Concurr- ent Actor Model)로부터 시작되었다고 볼 수 있다. [2] 그가 정의한 액터라는 객체는 다음과 같이 정의된다.

자체적으로 대화 형식의 병행 수행이 가능한 독립된 객체 액터는 메일 주소와 내부에 정의된 프로시져들을 가진 에이전트로써, 다른 액터 들과 메시지 교환을 통하여 병행적으로 작업을 수행하게 된다.

최근의 문헌을 조사해 보면, 에이전트에 대한 다양한 동의어들을 발견할 수 있다. 이중에 대표적인 것을 들면, knowbots (knowledge-based robots), softbots (software robots), taskbots (task-based robots), userbots, personal agents, autonomous agents, personal agents 등을 나열할 수 있다. 이들은 각각 적용분야의 특성을 고려한 용어들이며, 이러한 특성들을 종합하면 일반적으로 에이전트가 지녀야 하는 속성들을 정의할 수 있게 된다.

① 독립적인 수행능력 (autonomous) : 스스로 행위를 결정하여 사용자의 지시가 없더라도 수행될 수 있어야 한다.

② 목표 지향적 (goal-oriented) : 작업과정에 대한 사용자의 자세한 지시가 없어도, 최종 목표만 주어지면 에이전트가 자체적으로 목표 달성에 필요한 일련의 작업들을 결정하여 수행할 수 있어야 한다.

③ 사전 행동적 (proactive) : 외부 상황의 변화를 감지하고 작업 개시를 자율적으로 결정한다.

④ 계속적인 작동 (temporal continuity) : 일련의 입력에 대해 일련의 출력을 내고 종료하는 단발적인 프로그램이 아니라 계속 메모리에 상주하며 수행될 수 있어야 한다.

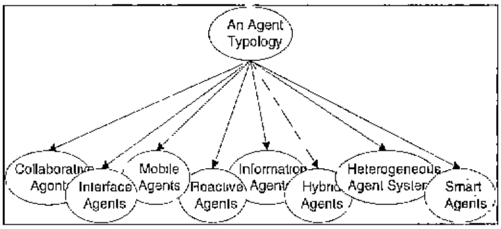
⑤ 통신을 통한 협력 (communicative) : 목표를 달성하기 위해 필요한 정보나 도움을 얻기 위해 다른 에이전트들과의 통신을 통한 협조가 필요하다.

⑥ 학습 기능 (adaptive learning): 과거의 경험에 의거하여 내부 구조 또는 지식베이스를 수정하는 학습 기능이 필요하다.

⑦ 이동성 (mobility): 통신망 또는 인터넷을 이용하여 구조와 플랫폼이 다른 기계로 이동하여 작업을 수행할 수 있어야 한다.

물론 하나의 에이전트가 반드시 이러한 속성을 모두 소유하여야 하는 것은 아니며, 이들 중 어떠한 속성들을 보유하는가에 따라 에이전트를 분류하는 작업이 가능하다.

1.2 지능형 소프트웨어 에이전트의 종류



(그림 1) 소프트웨어 에이전트의 분류

① 정적 에이전트 (static agent)와 이동 에이전트 (mobile agent):

---통신망을 이용하여 에이전트 프로세스가 다른 기계로 이동되어 수행이 가능한지 여부에 따른 분류이다.

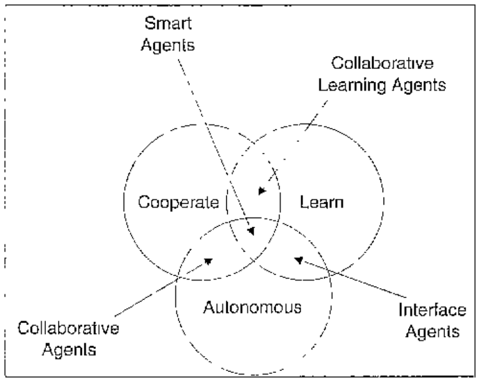
이동 에이전트란, WWW과 같은 광역 통신망에 존재하는 다른 호스트들을 직접 방문하여 정보 검색 또는 교환을 수행하는 소프트웨어 프로세스를 의미한다. 원거리에 위치한 여러 정보 소스를 돌아다니면서 사용자가 원하는 정보를 수집한 후에 사용자에게 돌아와 정보를 제공하는 임무를 수행한다. 이동 에이전트가 필요한 이유는 다음과 같다:

- 통신비용의 절감
- 지역 자원의 부족으로 인한 원거리 호스트 사용의 필요성
- 복잡한 메시지 교환 프로토콜이 필요치 않게 되며, 보다 용이한 태스크 관리
- 비동기적인 컴퓨팅의 가능성
- 유연한 분산 컴퓨팅 구조 또는 환경

현재까지 개발되어 있는 이동 에이전트 프로그래밍 언어를 보면, 대표적으로 General Magic사의 Telescript [3] 를 들 수 있으며, 이외에도 Java, Agent-Tcl, Safe-Tcl, C++등을 이용하여 이동 에이전트를 구현할 수 있다.

② 3 가지 속성에 따른 분류:

- 자율성 (autonomy): 사용자의 중재 없이 독자적으로 작동하는 속성
- 학습 기능 (learning)
- 협동 기능 (cooperation)



(그림 2) 소프트웨어 에이전트의 속성

Collaborative 에이전트는 사용자가 위임한 태스크를 수행하는데 있어서, 자율성과 협동 기능에 중점을 두게 되며, 현재까지 구현된 사례를 검토한 결과 학습 기능에는 소홀한 면을 관찰할 수 있었다. 특히 협동 기능에 의하여, 단일 에이전트로는 해결하기 어려운 작업을 여러 에이전트간의 협동을 통하여 해결한다는 장점을 갖는다.

Collaborative 에이전트 개발의 동기는 다음과 같다:

- 중앙 집중 형태의 단일 에이전트가 해결하기에는 복잡도가 큰 문제를 해결 가능
- 현존하는 기존 시스템들을 연계할 수 있는 가능성
- 근본적으로 분산 특성을 가진 문제에 대한 자연적인 해결 방안 제공
- 특성상 전문 지식 (expertise)이 분산되어 있는 문제의 해결

구현된 대표적인 시스템을 들면, 미국 카네기 멜런 대학에서 개발된 교직원과 특히 외부 방문자들의 일정과 미팅을 관리하는 Pleides 시스템을

꼽을 수 있다. [4]

본 논문의 조사대상인 인터페이스 에이전트에 대해서는 다음 장에서 상세히 설명된다.

③ 정보/인터넷 에이전트:

통신망과 인터넷의 발달로 인하여 폭주하는 방대한 양의 정보를 사용자의 기호 (preference) 에 의거하여 걸러내는 역할을 담당하는 에이전트로써, 그 역할에 따른 분류이다. 이에 반하여 이전의 분류는 에이전트의 기능 또는 속성에 따른 분류이며, 인터넷 에이전트는 인터페이스 에이전트로 간주되기도 한다. 최근에 가장 그 가능성을 인정받는 유형의 에이전트이며, WebCrawlers, Lycos 또는 Spiders와 같은 인터넷 검색 엔진에 부착되어 사용된다.

2. 인터페이스 에이전트

1.2절의 에이전트 분류에 의하면, 인터페이스 에이전트는 사용자를 위하여 임무를 수행하는데 있어서 자율성과 학습 기능에 중점을 둔다. 또한 인터페이스 에이전트는 동일한 작업 환경에서 사용자를 보조하는 개인적인 조수 (personal assistant) 또는 비서의 역할을 담당한다. 여기에서 특기할 사항을 보면, 다른 에이전트와 협동하여 작업하는 경우, KQML과 같은 [5] 명시적인 통신 언어 또는 프로토콜이 필요한 반면에, 인터페이스 에이전트의 경우 사용자와 단순한 질의 응답을 나누게 되므로 (예를 들어 확인, 거절, 보류 등) 복잡한 통신 언어가 필요 없게 된다.

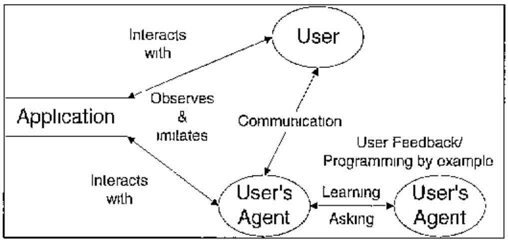
2.1 인터페이스 에이전트의 정의

지난 20년을 돌이켜 보면, 컴퓨팅 환경은 많은 발전을 보여 왔다. 과거의 전형적인 문자 위주의 사용자 환경으로부터 최근 몇 년간의 추세에서 보이듯이 대부분 소프트웨어는 그림 위주의 환경 (graphic user interface)으로 변화되고 있다. 이러한 변화에 Macintosh 운영체제가 이바지한 역할은 상당히 크다고 볼 수 있으나, 여전히 사용자가 직접 입력 도구를 조작해야 한다는 원칙 (direct manipulation principle) 은 변하지 않고 있다. 또한 도움말 기능도 강화되어 사용자들이 새로운 소프트웨어를 익히는데 많은 도움을 주고 있지만, 이 기능 역시 사용자가 요구하였을 경우에만 문맥 인식적 (context-sensitive) 인 도움말을 제공하는 수준에 머무르고 있을 뿐이며, 사전에 미리 자율적인 도움말을 제공하는 수준에는 못 미치고 있는 실정이다.

인터페이스 에이전트는 여기에서 한 걸음 더 진보한 사용자 인터페이스를 제공하려는데 그 목적을 두며, 이는 간접적으로 조작되는 인간-컴퓨터 인터페이스를 구현하려는 Alan Kay의 꿈을 실현하려는 시도로 간주된다. [6] 인터페이스 에이전트의 대한 정의도 다양하지만, 가장 일반적인 정의는 다음과 같다.

동일한 작업 환경에서 사용자와 협력하여 자율적으로 수행되는 개인적인 조수 또는 비서의 역할을 담당하는 프로그램

예전에는 컴퓨터의 사용이 주로 전문가에 국한되어 있었으나, 현대의 컴퓨팅 환경은 날이 갈수록 복잡해지고 있으며, 이에 따라 컴퓨터를 사용하여 처리해야 하는 업무의 종류도 날로 늘어나는 추세인 반면에, 컴퓨터의 대중화로 인하여 점점 더 초보 사용자의 수는 증가하고 있는 실정이다. 그러므로 사용자는 일부 업무를 다른 사람 또는 에이전트에게 의뢰하게 될 것이며, 이에 따라 인터페이스 에이전트의 활용도는 점점 더 커질 것으로 사료된다.



(그림 3) 인터페이스 에이전트의 학습 형태

위의 그림 3에 나타나듯이, 인터페이스 에이전트는 사용자를 효율적으로 보조하기 위하여 다음의 4가지 형태로 기계 학습을 수행한다.

① 사용자의 작업 패턴을 어깨 너머로 관찰하며 모니터링한다.

② 에이전트가 제안한 사례에 대한 사용자의 긍정적 또는 부정적인 피드백을 접수한다.

③ 사용자로부터 명시적인 지시를 받는다.

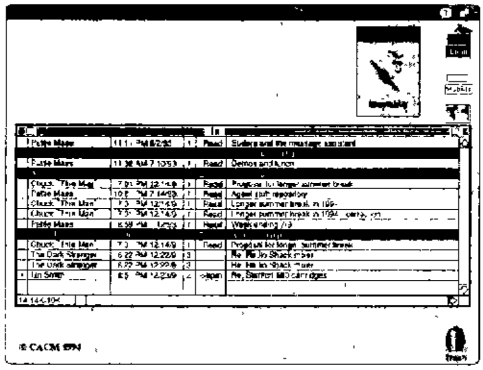
④ 다른 인터페이스 에이전트로부터 해당 문제에 대한 조언을 받는다. 여기에서 이러한 상호작용은 협상하는 관계가 아니며, 단순히 다른 에이전트의 지식 내용을 복사하는 수준에 그친다.

2.2 기존 인터페이스 에이전트에 대한 연구

2.2.1 Maxims

Maxims는 전자우편을 이용하는 사용자를 보조하는 에이전트로써, 미국 MIT 대학의 Lashkari와 Maes에 의해 개발되었다. [7] Maxims 에이전트는 사용자를 대신하여 수신된 전자우편 메세지의 우선 순위를 결정하고, 필요없는 메세지는 삭제하며, 다른 사람에게 전송하고, 정렬하고 보관하는 등의 작업을 학습 기법을 이용하여 수행한다. 그림 4는 Maxims 에이전트의 활동을 보여 주는 화면 예제이다.

Maxims 에이전트는 Macintosh Common Lisp으로 구현되었으며, Apple Events를 이용하여 상용 전자우편 프로그램인 Eudora와 통신한다. Maxims에서는 학습 기법으로 메모리 기반 추론 (memory-based reasoning) 을 사용하였다. [8]

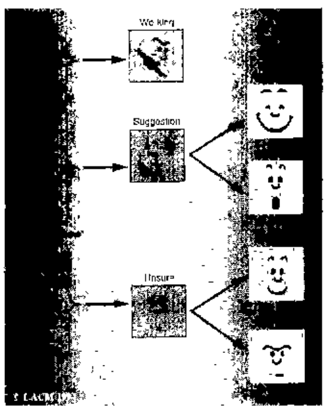


(그림 4) Maxims 에이전트의 인터페이스

Maxims 에이전트는 얼굴 표정을 나타내는 인터페이스를 통하여 사용자에게 에이전트의 작업 상태를 알리고 사용자와 통신한다. 다음의 그림 5는 Maxims 에이전트의 얼굴 표정 인터페이스와 각각의 표정이 의미하는 에이전트의 실행 상태를 보여준다. 그림 5에 나타난 임계치 (threshold) 들은 에이전트가 제안한 내용의 신뢰도 (confidence level)가 속하는 구간을 표시한다.

① <u>"do it" 과 1 사이의 구간</u>: 이 구간에서는 에이전트가 완전히 자율적으로 사용자의 확인 없이 수행되며, 수행한 결과에 대한 보고서를 작성한다. 또한 사용자는 언제라도 에이전트가 수행한 내용에 대한 보고서를 요구할 수 있다.

Working: 에이전트는 자율적으로 행동한다.



(그림 5) Maxim의 작업 상황 인터페이스

② "tell me" 와 "do it" 사이의 구간: 이 경우, 에이전트는 자율적으로 수행되는 것이 아니라, 사용자에게 제안을 하게 되며, 이 제안을 수행하기 위한 사용자의 승인을 기다린다.

Suggestion: 에이전트가 문제 해결을 위한 대안을 제시한다.

Gratified: 사용자가 그 제안에 대해 만족했을 경우

Surprised: 사용자가 만족하지 못했을 경우

이때, 사용자는 에이전트를 신뢰하는 정도에 따라 "tell me" 와 "do it" 임계치를 적절히 조정해야 하는 책임을 진다. 만약 "do it"에서 1까지의 구간이 너무 큰 경우, 에이전트가 자동으로 수행하는 내용은 많아 질 것이며, 이러한 상황이 사용자에게 불만족스럽다면, 사용자는 "do it" 임계치를 상향 조정하여 에이전트가 스스로 행동하지 않고 사용자의 의견을 확인한 후에 행동하도록 만들 수 있다.

③ 0 과 "tell me" 사이의 구간: 아직 에이전트가 판단하여 대안을 제시하기에는 충분한 사례가 없는 경우이다. 이 경우 모든 행동은 사용자가 결정하게 되며, 에이전트는 학습하기 위하여 사용자의 행동을 계속 관찰한다.

Maxims 에이전트는 사용자의 행동을 관찰하고, 많은 사례를 학습함으로써 점차적으로 능력이 향상되게 된다. 그러므로 처음 에이전트가 가동되었을 경우, 초기의 성능은 그다지 만족스럽지 않게 된다 ("start-up problem"). 이러한 문제점을 해결할 수 있는 기법에는 다음의 2가지가 있다.

① 사용자가 에이전트에게 사전에 명시적으로 학습 사례를 제시한다.

---사용자는 가상의 상황을 만들고 에이전트에게 그것을 보여줌으로써 에이전트를 학습시킬 수 있다.

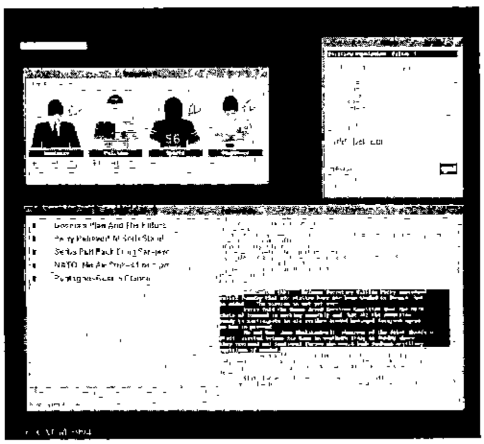
② 다른 에이전트들간의 협력이 가능하다.

---에이전트가 대안을 제시하기에는 확신이 충분치 않을 경우 (계산된 신뢰도가 "tell me" 임계치보다 작을때), 전자 우편을 이용하여 다른 사용자를 돕고있는 Maxims 에이전트에게 도움을 요청할 수 있다.

2.2.2 NewT

현재 가장 광범위하게 사용되는 에이전트중의 하나는, 계속해서 폭주하는 뉴스들로부터 사용자에게 필요한 기사를 찾아내는 정보/인터넷 에이전트 (Information / Internet agent) 이다. World Wide Web등과 같은 원거리 통신망상에서 더욱더 많은 정보가 제공됨에 따라, 사용자들은 방대한 양의 정보 중에서 필요한 정보를 걸러내고 또한 그들이 원하는 기사를 찾아 주는 도구를 더욱 원하게 되었으며, NewT는 사용자가 Usenet Netnews에서 원하는 정보를 찾도록 도와주는 에이전트로, MIT 대학의 Sheth와 Maes에 의해 개발되었다. [9]

NewT는 C++언어로 Unix 시스템에 구현되었으며, 사용자는 각 관심분야에 따라 하나 이상의 뉴스 에이전트를 만들 수 있으며, 사용자가 선택하거나 삭제한 기사의 사례들을 이용하여 에이전트의 학습을 수행한다.



(그림 6) NewT의 인터페이스

그림 6은 사용자가 만든 4 종류의 NewT 에이

전트를 보여준다. 각각은 사업 관련 기사, 정치 관련 기사, 컴퓨터 관련 기사와 스포츠 기사를 위한 에이전트를 나타내며, 이들은 아이콘 형태로 인터페이스에 보여진다. 에이전트들은 각각 관심의 대상인 기사가 도착했을 때, 사용자가 선택하거나 또는 선택하지 않는 사례에 의거하여 학습을 수행한다.

NewT 에이전트는 문서에서 관련된 키워드를 검색하기 위하여 vector-space모델 [10]을 이용한 문서의 전반적인 분석을 수행하며, 또한 저자, 출처, 할당된 인덱스 등과 같은 기사에 관한 구조 정보도 이용한다. 이와 더불어 반드시 선택되어야 하는 유형의 기사들은 사용자가 미리 정의된 템플리트의 내용을 채움으로써 에이전트의 명시적인 프로그래밍도 가능하다. 여기에서 NewT 에이전트의 한계점으로 지적되는 것은, 검색이 키워드에만 국한된다는 사실이다.

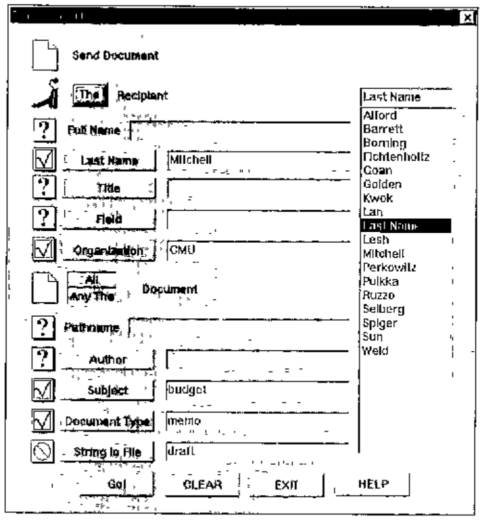
한가지 흥미로운 사실은 MIT 대학의 연구실에서 시도되고 있는 유전자 알고리즘 (genetic algorithms) 기법의 적용이다. 하나의 관심 분야에 대한 다수의 에이전트를 생성한 후, 검색된 정보의 질을 분석하여 성능이 나쁜 에이전트는 도태시키고, 훌륭한 에이전트는 계속 양성하는 유전학 기법을 적용하고 있다는 사례가 문헌에 발표된 바 있다.

2.2.3 Internet Softbot

인터넷 소프트봇 (Internet Softbot)은 Washington 대학의 Etzioni에 의해 개발된 정보 에이전트이며, [11] 다양한 인터넷 자원들과의 상호작용을 위해 Word-Wide Web과 Unix Shell을 사용한다. 소프트봇의 Effector는 ftp, telnet, mail과 다양한 파일 조작 명령어를 포함하며, archie, gopher, netfind 등의 인터넷 도구들이 소프트봇의 센서 (sensor) 역할을 담당한다.

소프트봇 에이전트는 사용자가 고급 (high-level) 요구사항 또는 목표를 시스템에 제시하면, 소프트봇 에이전트는 그 요구사항을 어떻게 만족시킬 것인가를 결정하기 위하여, 탐색, 추론, 그리고 지식을 이용한다. 또한 소프트봇은 사용자의 요구사항이 애매하고, 불충분하고, 또한 오류가 있다 하더라도 이를 극복할 수 있는 능력을 갖는다.

기본적으로 소프트봇은 first-order logic으로 표현된 사용자의 요구사항을 처리한다. 또한 초보자를 위하여 사용자가 빈칸을 채우기만 하면 되는 템플리트를 제공한다.



(그림 7) Softbot의 입력 템플리트

그림 7의 템플리트가 채워지면, 그 내용은 X-windows GUI나 mosaic으로부터 전자우편을 경유하여 소프트봇에게 보내지게 되며, 이는 다음과 같은 소프트봇 내부 표현으로 자동 번역된다.

```
(forall (?d :in files)
   (if (file.type ?d memo.document)
      (subject of.document ?d "budget")
      (not (string.in.file "draft" ?d)))
   (delivered.to ?d ?obj341)))
```

소프트봇 인터페이스 설계의 주안점을 보면, 인터페이스의 "look and feel" 기능보다는 인터페이스의 표현력과 융통성을 증대시키기 위하여 소프트봇의 인공지능 능력을 어떻게 향상시킬 것인가에 중점을 두었다. 특히, 소프트봇 인터페이스는 다음의 개념을 구현하였다:

① 목표 지향적 (Goal oriented): 요구사항은 사용자가 무엇을 원하는지를 나타내며, 소프트봇은 언제, 어떻게 요구사항을 만족시킬 것인가를 결정할 책임을 진다.

② 관용성 (Charitable): 대부분 요구사항은 사용자가 원하는 바를 완전하고 올바르게 명세하지 못한다. 그러므로 소프트봇은 요구사항에 포함된 실마리나 힌트만을 가지고 사용자가 원하는 바를 처리하여야 한다.

③ 균형성 (Balanced): 소프트봇은, 추가 정보없이 정보를 찾아내는데 드는 비용과 사용자에게 질문을 던짐으로써 절감되는 비용을 고려하여, 사용자를 추가적인 질의응답으로 성가시게 하는 정도를 적절히 조절하여야 한다.

④ 통합성 (Integrated): 소프트봇은 다양한 인터넷 서비스와 도구들에 대해 일관되고, 표현력 있는 단일 형태의 인터페이스를 제공한다.

2.2.4 CAP

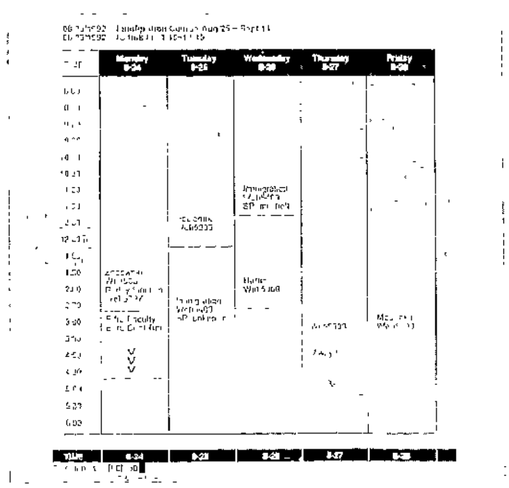
CAP(Calendar APprentice)는 기계 학습 기법을 이용하여 사용자의 일정을 관리하는 에이전트로써, 미국 카네기 멜런 대학의 Mitchell과 McDermott에 의해 개발된 규칙 기반 시스템이며, [12] 5년간 시스템을 운영한 결과, 지식베이스가 수천개에 이르는 규칙으로 구현되었다.

CAP를 설계하기 위하여 고려한 두 가지 귀납적 학습 기법은 결정 트리 기법 [13] 과 신경회로망을 이용한 역전도 알고리즘 [14] 이었으며, 다음의 이유를 근거로 결정 트리 기법을 채택하였다.

① 결정 트리가 만들어내는 규칙은 사람이 쉽게 이해할 수 있는 형태로 표현되지만, 신경회로망의 학습 결과인 가중치는 그 의미를 파악하는 작업이 용이하지 않다.

② 결정 트리의 규칙들은 학습된 정보들 하나 하나를 단계적으로 표현하지만, 신경회로망의 경우에는 전체 학습된 정보들이 하나의 묶음으로 표현된다. 그러므로 결정 트리 기법으로 만든 시스템은 사용자가 각각의 정보가 학습될 때마다 학습된 규칙들을 모니터링하며 튜닝할 수 있지만 신경회로망에서는 그 작업이 불가능하다.

CAP 에이전트의 사용자 인터페이스는 그림 8과 같이 사용자가 일정을 확인하거나 수정할 수 있는 온라인 달력과 명령을 직접 입력할 수 있는 명령어 라인으로 구성되어 있다.

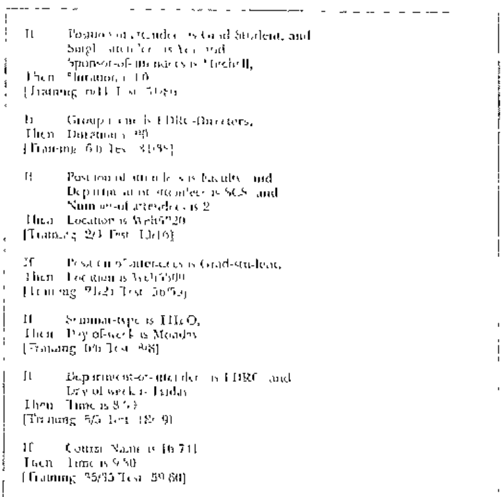


(그림 8) CAP의 인터페이스

일반적으로 사용자가 새로운 미팅 일정을 추가하였을 때, CAP 에이전트는 미팅의 형태, 참석할 인원들, 일시, 시간과 장소등을 사용자가 차례로 입력하게 한다. 그리고 해당 미팅이 잠정적인가 또는 확실히 정해진 일정인가도 확인한다.

그림 8의 하단 부에 보이는 명령어 프롬트는 미팅 시간을 입력하는 상황을 보여준다. CAP 에이전트가 제안한 미팅 시간은 60분이지만, 사용자는 이를 무시하고 30분을 입력하였다. 즉 CAP 에이전트는 그간 학습된 자료들을 이용하여 사용자의 일정 관리에 대한 제안을 하게 되며, 또한 사용자는 임의로 에이전트의 제안을 무시하고 새로운 일정을 입력할 수도 있다. 이때 사용자가 입력한 정보는 CAP 에이전트의 새로운 학습자료로 사용된다.

다음은 결정 트리 기법을 사용하여 학습된 규칙의 예를 보여준다.

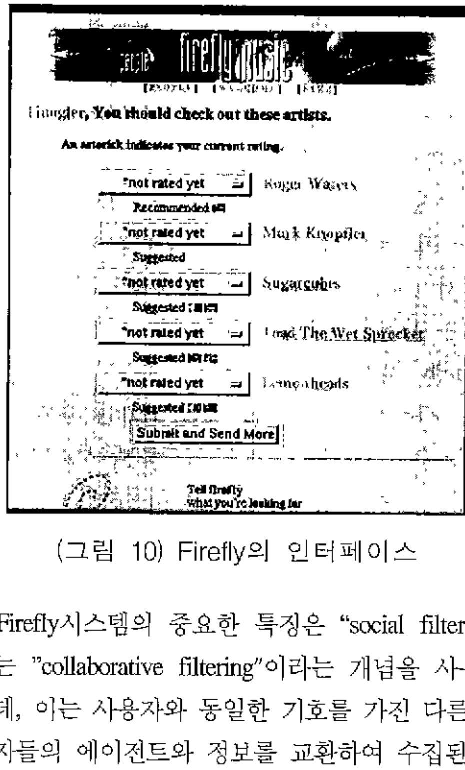


(그림 9) 생성된 CAP의 규칙

앞에서 설명된 바와 같이, 이 규칙들은 사용자가 이해하기 쉬운 형태로 나타나게 되며, 특기할 사실은 각 규칙의 성능 (즉 에이전트의 제안을 사용자가 수락한 비율) 에 의해 우선 순위가 매겨진다는 것이다. 규칙의 하단 부에 나타난 [Training: 6/11 Test: 52/86] 은 해당 규칙이 도출된 학습 당시 자료에 대한 성능과 규칙이 만들어진 이후 시스템 운영중의 성능을 표시한다.

2.2.5 Firefly

1996년에 Pattie Maes가 개발한 인터페이스 에이전트로써, [15] 사용자의 기호 또는 취향에 의거하여 음악 앨범이나 영화를 추천하는 역할을 한다. 이미 상용화되어 많은 사용자를 확보하고 있는 시스템이며, 그림 10은 Firefly 시스템의 인터페이스를 보여 준다.



(그림 10) Firefly의 인터페이스

Firefly시스템의 중요한 특징은 “social filtering” 또는 ”collaborative filtering”이라는 개념을 사용하는데, 이는 사용자와 동일한 기호를 가진 다른 사용자들의 에이전트와 정보를 교환하여 수집된 정보를 사용자에게 추천하는 기법이다. 이에 반하여, 2.2.2절에서 살펴본 NewT의 경우에는 다소 약한 형태의 social filtering을 사용하는데, 그 이유는 다른 NewT 에이전트의 지식 또는 노우하우를 그대로 복사하는 형태를 취하기 때문이다.

여기에서 주목할 만한 사실은 Firefly시스템은 음악이나 영화에 대한 전문지식이 전무하다는 것이며, 단지 관련된 사용자들에 대한 정보만 갖고 있다는 것이다. 각 사용자들의 음악적인 지능을

원하는 사용자에게 전달함으로써 시스템 전체적인 지능을 구현한다는 단순한 개념이지만, 시스템 외부에서 관찰하는 사용자들은 Firefly시스템이 보유한 전문지식이 방대한 것으로 착각한다는 사실이다.

Firefly 시스템은 서론에서 설명되었던 바텀업 학파의 이론을 대표적으로 구현한 에이전트이며, 이는 과거의 경험 사례를 바탕으로 학습을 시도하는 시스템이 아니라, 많은 사용자를 학습으로 대체한 결과를 만든다. 특히 각 사용자가 Firefly 시스템에 기여하는정도는 그저 몇 장정도의 앨범을 평가해 주는 것에 그치지만, 그것들이 모여서 이루어진 전체의 지능은 상당한 것으로 보이게 된다. 이러한 모든 이점은 WWW이 공유 가능한 매개체 (sharing medium)로 등장함으로써 가능해 졌다는 사실을 주목할 수 있다.

Firefly 시스템의 문제점으로 지적된 하나의 사실은, 사용자들의 시각을 좁게 만드는 결과를 가져 올 수도 있다는 것이다. 왜냐하면, 시스템이 제공하는 정보만 접하게 되므로 사용자가 시스템에 전적으로 의존하게 되며, 그 외의 다른 정보는 검색하려 하지 않는 결과를 초래할 수도 있게 된다.

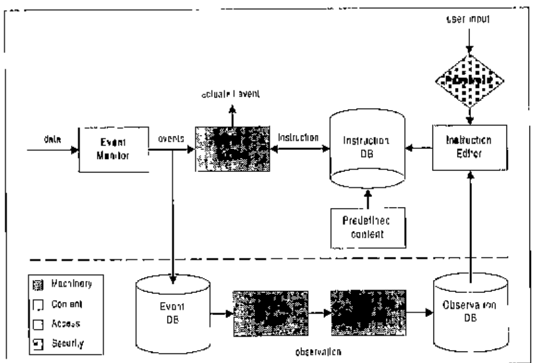
또한 Firefly 시스템도 2.2.1절에서 설명된 “Start-up Problem”의 예외일 수는 없었으나, Firefly 시스템의 경우는 가동된 후 즉시 수 천명의 사용자가 시스템을 이용하기 시작하였으므로 별다른 문제가 되지 않았다고 하며, 이러한 “Start-up Problem”은 모든 에이전트에 공통적으로 적용되는 문제점이다.

2.2.6 Open Sesame!

1993년에 발표된 Open Sesame 1.0은 미국의 Caglayan등이 개발한 최초의 상용 인터페이스 에이전트이며 [16], Mac OS에서 작동된다. 데스크탑 또는 운영체제 에이전트라고도 하며, 기계 학습 기법을 이용하여 사용자로부터 지식을 획득한다. 과거 사용자의 작업 패턴들을 기반으로 에이전트가 제시한 제안에 대하여 다음 4가지 옵션이 주어진다.

① 제안을 수락하고, 에이전트로 하여금 그 작업을 자동화하는 규칙을 생성하게 만든다.
② 제안을 무시한다.
③ 제안의 내용을 수정할수 있다.
④ 제안을 완전히 무시하지는 않고 다음 기회로 결정을 연기한다.

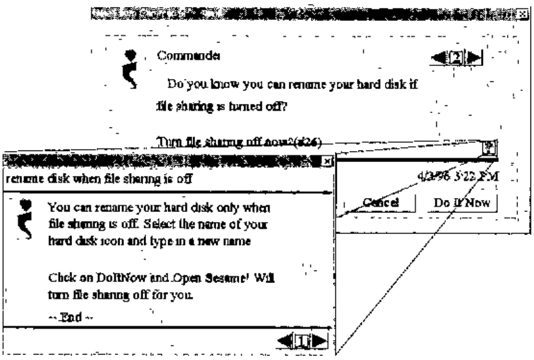
Open Sesame 시스템의 전체적인 구조는 그림 11과 같으며, Version 1.0에서 사용된 학습 기법은 신경회로망의 ART (Adaptive Resonance Theory)에 기반을 둔 GEN-ART기법이며, 2.0에서는 신경망의 단점을 보완한 Smarts Engine을 사용하였다.



(그림 11) Open Sesame의 세부적인 구조

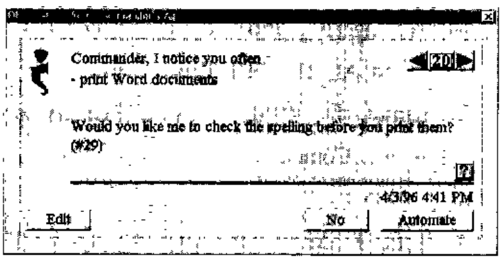
그림 11의 사실 번역기(Fact Interpreter)가 처리한 사실들은 Observation DB에 저장되어 시스템이 다양한 제안을 하는데 이용된다. 그중 몇 가지를 예로 들면 다음과 같다.

● 자율적인 조언 (Proactive assistance): 다음은 사용자가 공유 디스크의 이름을 변경하는 작업을 제대로 해내지 못할 때, 에이전트가 조언을 하는 예제이다.



(그림 12) Open Sesame의 Proactive Help 기능

● 반복되는 사용자의 작업을 자동화: 다음은 사용자가 Word 문서를 프린팅하기 전에 반드시 철자 체크 작업을 실행하는 작업 패턴을 에이전트가 관찰한 다음, 이를 자동화하려는 제안을 보여준다.



(그림 13) Open Sesame의 Automation 기능

3. 구현상의 문제점

지능형 소프트웨어 에이전트가 완벽하게 구현된다면, 컴퓨터 사용자들의 작업이 상당한 부분 수월해 지는 것은 사실이지만, 이에 따라 부수적으로 생기는 문제점 역시 다양해지는 면을 관찰할 수 있다. 그중 대표적인 것을 나열하면 다음과 같다.

● 소프트웨어 에이전트의 이론적 취약성: --- 에이전트 분야는 새로운 분야이므로 아직 이론적인 배경이 부족하며, 시스템의 아키텍춰, 구현 환경등이 취약한 형편이다. 또한 Collaborative 에이전트의 경우, 각 에이전트들의 관리 또는 조정(coordination) 에 관한 분명한 이론이 없다면 무질서나 병목 현상이 쉽게 발생하게 된다.

● 제반 학습기법의 성능:

---기존의 기계 학습기법은 다양하게 제안되어 있지만, 에이전트 기법이 적용되는 도메인에 적절한 기법을 찾아내려면 철저한 실험이 수행되어야 한다. 또한 학습 기능을 가진 에이전트의 성능을 사용자가 신뢰할 수준까지 높이는 작업도 오랜 실험과 튜닝 기간을 요한다.

● 인터페이스 에이전트 구현의 문제점

---인터페이스 에이전트에 대한 연구는 새로운 분야이므로, 아직까지 성능을 객관적으로 평가할 벤치마크 자료가 없다는 사실이 연구와 구현의 어려움으로 대두되고 있다. 또한 인터페이스 에이전트의 성능을 입증하자면 실제 응용 프로그램에 접목시켜서 일정 기간 운용한 결과를 남들이 납득할 수 있도록, 효용도에 대한 객관적인 성능 평가가 반드시 뒤따라야 한다는 사실이다.

● 과학적인 성능 분석기법의 부재

---지금까지의 인터페이스 에이전트는 정성적인 분석만 수행되었으나, 잠시 동안의 유행이 아닌 컴퓨터 과학의 한 분야로 인정받으려면, 반드시 과학적인 성능 분석이 부수되어야 한다는 것이다. User Interface 분야의 경우, 20 여 년 동안 연구되면서 과학적인 성능 분석 기법에 대한 연구가 많이 진행되어 왔으며, 이제는 하나의 interdisciplinary 분야로 인정을 받고 있는 실정이다. 인터페이스 에이전트 분야가 이러한 전철을 따르자면, 사용자의 에이전트에 대한 신뢰도, 학습 수행결과의 정량적 분석, 학습 기법들의 비교 분석을 위한 벤치마크 자료의 개발등 과학적인 접근 기법의 개발이 시급한 실정이다.

4. 결 론

본 연구에서는 소프트웨어 에이전트에 대한 정의와 분류에 대해 살펴보고, 그중 개인적인 비서 또는 교사의 역할을 담당하는 인터페이스 에이전트의 특성을 조사하고, 구현된 다양한 사례를 조사하고, 구현 기법들에 대해 알아보았다.

조사된 인터페이스 에이전트의 구현 기법과 기계 학습 기법들에 대한 정보는 인터페이스 에이전트 구현에 긴요하게 사용될 수 있으며, 인터페이스 에이전트의 프로토타입이 구현된다면, 그로부터 얻어질 수 있는 장점은 다음의 3가지로 크게 요약될 수 있다.

첫째, 사용자나 소프트웨어 개발자의 수고를 덜어주게 된다. 사용자는 인터페이스 에이전트의 활용으로 인하여 새로운 소프트웨어 사용방법의 습득이 용이하게 되며, 소프트웨어 개발자의 측면에서 보면, 기존의 개발 방식은 모든 가능한 예외상황을 모두 미리 예측하여 대처할 방안을 미리 프로그래밍하여야 하나, 인터페이스 에이전트의 학습 능력을 이용하면, 이러한 예외 상황을 미리 처리할 필요가 없어지게 된다.

둘째, 시간이 흐름에 따라 소프트웨어는 사용자의 기호나 사용 습관에 길들어져 유연하게 적응할 수 있게 된다.

마지막으로, 여러 사용자들의 사용 경험이나 사용 비결 (know-how)을 공유할 수 있게 된다. 학습기능을 통해 지능 획득이 가능한 인터페이스 에이전트의 개발은 사용자의 편의성을 증대시킴으로써 소프트웨어의 부가가치를 높이는데 크게 기여할 것으로 사료된다.

참고문헌

[1] Interview with Pattie Maes: "Humanizing the Global Computer", IEEE Internet Computing, Vol. 1, No. 4, pp. 10-19, July 1997.

[2] C. Hewitt, "Viewing Control Structures as Patterns of Passing Messages", Artificial Intelligence 8 (3), pp. 323-364, 1977.

[3] J. White, Mobile Agents White Paper, 1996.

[4] K. Sycara, "Intelligent Agents and the Information Revolution", UNICOM Seminar on Intelligent Agents and their Business Applications, Nov. London, pp. 143-159, 1995.

[5] T. Finin, Y. Labrou, J. Mayfield, "KQML as an Agent Communication Language, Software Agents, The MIT Press, pp. 291-316, 1997.

[6] A. Kay, "Computer Software", Sci. Amer. 251, 3. pp. 191-207, 1984.

[7] Y. Lashkari, M. Metral, P. Maes, "Collaborative interface agents", Proc. of the National Conference on AI, The MIT Press, 1994.

[8] C. Stanfill, D. Waltz, "Toward Memory-based reasoning", Comm. ACM, 29, 12, pp. 1213-1228, 1986.

[9] B. Sheth, P. Maes, "Evolving Agents for Personalized Information Filtering", Proc. of 9th IEEE Conference on AI for Applications, 1993.

[10] G. Salton, M. McGill, Introduction to Modern Information Retrieval, McGraw- Hill, N.Y., 1983.

[11] O. Etzioni, D. Weld, "A Softbot-Based Interface to the Internet", CACM, Vol. 37, No. 7, pp. 72-76, July 1994.

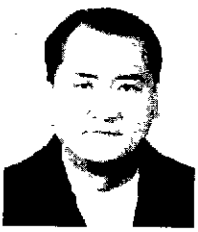
[12] T. Mitchell, R. Garuana, D. Freitag, J. McDermott, D. Zabowski, "Experience with a Learning Personal Assistant", CACM, Vol. 37, No. 7, pp. 81-91, July 1994.

[13] S. R. Safavian, D. Landgrebe, "A survey of

decision tree classifier methodology", IEEE Trans. on Syst., Man, and Cybern., Vol. 21, No. 3, pp. 660-674, 1991.

이 형 일

1985년 명지대학교 전자계산학과 (학사)
1994년 명지대학교 대학원 전자계산학과 (석사)
1997년 명지대학교 대학원 컴퓨터공학과 박사과정 수료
1985년-1989년 (주) 쌍용컴퓨터 근무
1990년-1995년 CHNO System Con-sulting Co. 근무
1997년-현재 김포전문대 전자계산과 전임강사



윤 충 화

1979년 서울대학교 자연과학대학 수학과 (학사)
1984년 미국 텍사스 주립대 전자계산학과 (석사)
1989년 미국 루이지아나 주립대 전자계산학과 (박사)
1990년-현재 명지대학교 컴퓨터공학과 부교수
관심분야 : 신경회로망, 전문가시스템, 지능형 소프트웨어 에이전트, 프로그래밍 언어

임 윤 택

1997년 명지대학교 컴퓨터공학과 (학사)
1997년-현재 명지대학교 대학원 컴퓨터공학과 석사과정 재학중
관심분야 : 인터페이스 에이전트, 기계학습