비교 쇼핑 사이트들에 대한 효율적인 메타검색 기법

An Efficient Meta-Search Scheme for Comparison Shopping Sites

조 강 의 (Kang-Eui Cho) 단국대학교 자연과학대학 대학원 전산통계학과
조 성 제 (Seong Je Cho) 단국대학교 자연과학대학 정보컴퓨터학부

요 약

인터넷 기반의 전자상거래가 확산되면서, 에이전트 기술을 사용한 비교 쇼핑 사이트들이 등장하여 쇼핑이 점차 편해지고 있다. 하지만, 이들 비교 쇼핑 사이트들이 동일 상품에 대해 서로 다른 가격을 보여줄 때도 있고, 또 어떤 사이트는 특정 상품에 대한 정보를 전혀 보여주지 못할 때가 많기 때문에, 여전히 대부분의 소비자들은 최적의 가격을 검색하기 위해 많은 시간을 소비하고 있다. 또한, 책이나 CD 등의 상품에 대한 가격 검색에서는 온라인 실시간 검색이 요구되므로 시스템 부하를 유발시켜 응답시간이 길어지게 된다. 이러한 문제를 해결하기 위해, 본 논문에서는 지역 데이터베이스와 메모리 캐시를 갖는 메타-비교 쇼핑 시스템을 설계하고 구현하였다. 제안된 시스템은 여러 소프트웨어 에이전트를 이용하여 기존 비교 쇼핑 사이트들에 대해 메타 검색을 수행함으로써, 인기검색 상품들에 대한 가격 정보를 수집하여 유지한다. 실험을 통해 제안 시스템이 효율적인 메타-비교 쇼핑 엔진으로 응답 시간의 지연을 줄여 주며 오버헤드도 적게 유발시킨다는 것을 보였다.

키워드 : 비교 쇼핑 사이트, 메타 검색, 실시간 검색, 에이전트

Ⅰ. 서 론

전자상거래에서 인터넷 쇼핑몰로부터 특정 정보를 알아내기 위한 방법으로 에이전트(agent) 기법이 사용되고 있다. 에이전트란 사용자를 대신하여 사용자가 해야 할 작업을 자동으로 수행하는 프로그램(가상대리인)을 말한다(양재영 등, 2000; 양재영 등, 2001). 최근에는 에이전트 응용기술로 쇼핑 에이전트(shopping agent) 혹은 더 구체적으로 '가격 비교 쇼핑 에이전트'(comparison shopping agent)라는 기법이 전자상거래에 도입되었다. 가격 비교 쇼핑 에이전트는 구매자가 원하는 상품을 최적의 가격으로 구매할 수 있도록 인터넷 쇼핑몰들을 순회하여 가격 등의 정보를 찾아주어 비교구매 할 수 있도록 도우미 역할을 대행해 주는 기술이다(서영우, 1999; 양재영 등, 2000; 양재영 등, 2001; Morris, Maglio, 2001).

국내에서 비교 쇼핑 에이전트 기술을 적용하고 있는 사이트로는 Yavis(2002), Enuri(2002), Bestbuyer(2002), Clickprice(2002), Shopbinder(2002), Omi(2002) 등이 있으며 사용자도 늘어나고 있는 추세이다. 이들 사이트의 검색 구조는 소비자가 검색을 요청하였을 경우, 로봇 에이전트에 의해 미리 구축된 자신의 데이터베이스(DB) 정보를 검색하여 결과를 보여주는 형태가 대부분이다. 그러나 각 사이트는 모든 인터넷 쇼핑몰을 검색하지 않고 자신과 연계된 쇼핑몰로부터만 정보를 수집하여 DB를 관리하기 때문에, 동일 상품이라 하더라도 각 사이트가 제시하는 가격 정보가 다를 때가 많다. 실제 어떤 전자 제품에 대해서는 사이트 A가 최저 가격을, 어떤 귀금속 제품에 대해서는 사이트 B가 최적 가격을 제시해 준다. 또 어떤 사이트는 특

정 상품에 대한 정보를 제공하지 못할 때도 있다. 따라서 상대적 최저 가격을 알기 위해서 소비자는 여전히 모든 비교 쇼핑 사이트들을 방문해 보아야 한다. 사실 인터넷상의 모든 쇼핑몰을 대상으로 검색을 진행한다면 최저 가격을 검색할 수는 있겠지만 과도한 작업부하 및 DB 용량 등의 많은 요소들로 인해 현재의 방식을 채택하고 있다.

한편, Yavis, Clickprice 등의 사이트에서는 도서나 음반 등 가격이 매우 유동적인 품목들에 대해 검색 질의어(query)를 요청하면 DB를 이용한 검색 서비스가 아닌 온라인 실시간 검색을 실시하고 있다(이보길, 2000; 조강의, 2001). 즉 소비자가 도서나 음반 검색 요청을 하면, 비교 사이트는 자신과 연계된 쇼핑몰들에게 다시 실시간 검색 요청을 전달하여 정보를 보여주기 때문에 최근의 가격을 알 수 있지만, 이에 따른 실시간 작업은 과도한 검색시간을 요구하게 되어 시스템의 작업부하를 유발시키며 응답 시간의 지연을 가져오기도 한다.

본 논문에서는 여러 비교 쇼핑 사이트들을 통합 검색하여 상대적인 최저 가격을 한 번에 알 수 있도록 지원하는 메타-비교 쇼핑 에이전트 시스템(meta-comparison shopping agent system)을 설계하고 구현한다. 제안한 시스템에서는 소비자 인기검색 품목들에 대해 에이전트가 기존 비교 쇼핑 사이트들을 주기적으로 조회하여 얻은 가격정보를 지역 DB 및 메모리 캐싱(caching) 공간에 저장할 뿐만 아니라, 별도의 에이전트에 의해 소비자 기호도에 기반한 DB 레코드 교체정책을 적용함으로써 일정한 저장공간에 유용한 최신 정보를 유지할 수 있다. 기존 비교 쇼핑 사이트는 쇼핑몰들을 대상으로 정보를 수집하는데 반해, 제안한 시스템은 비교 쇼핑 사이트들을 대상으로 상품 정보를 수집한다. 따라서, 제안한 시스템은 몇 개의 사이트만 방문하면 되므로 적은 비용으로 최저의 가격을 알아낼 수 있는 한 방법이다. 또한 도서 등의 가격이 유동적인 상품에 대한 정보도 주기적으로 검색하여 미리 캐싱함으로써 서비스 응답 시간을 단축시킬 수 있다.

본 논문은 다음과 같이 구성되어 있다. Ⅱ장에서는 비교 쇼핑 에이전트 시스템의 구조 및 HTML 문서에서 필요한 정보를 추출하는 wrapper 기술에 대하여 살펴보고, 메타검색엔진의 구조에 대해 간단히 기술한다. Ⅲ장에서는 메타-비교 쇼핑 에이전트 시스템을 제안하면서, 정해진 저장 공간 내에서 캐싱 정책을 이용한 효율적인 검색 서비스 기법에 대한 내용도 기술한다. Ⅳ장에서는 제안한 시스템의 구현 및 테스트를 통해 시스템 성능을 비교·분석하고, Ⅴ장에서 결론 및 향후연구 과제를 제시하며 마무리한다.

Ⅱ. 관련 연구

기존 비교 쇼핑 사이트들은 인터넷 쇼핑몰들에 대한 검색엔진으로, 제안한 시스템은 기존 비교 쇼핑 사이트들에 대한 메타 검색엔진으로 볼 수 있다. 2절에서는 메타 비교 쇼핑 에이전트 시스템에 관련된 연구들을 소개한다.

2.1 비교 쇼핑 에이전트 시스템

비교 쇼핑 에이전트(comparison shopping agent)란 소비자가 어떤 상품을 웹을 통하여 구매할 때 기존 쇼핑몰 사이트들을 일일이 수동적으로 방문하지 않고도 가격 정보를 비교할 수 있도록 도와주는 에이전트이다. 널리 알려진 비교 쇼핑 에이전트로는 Bargain Finder와 ShopBot이 있으며, 이러한 기술을 이용한 비교 쇼핑 사이트로는 Amazon이나 Mysimon 등이 있다. 일반 비교 쇼핑 사이트들은 자신과 연계된 쇼핑몰들에 대해 로봇 에이전트가 주기적으로 순회하여 상품 정보들을 DB에 유지하면서, 가격 변경 및 제품 추가 등의 정보가 갱신되면 자신의 DB도 갱신한다. 소비자로부터 검색 요청이 들어오면 에이전트는 DB를 검색하여 원하는 정보를 알맞게 출력하게 된다(양재영 등, 2000; 양재영 등, 2001). 일부 문제점으로는 각 비교 쇼핑 사이트가 유지하는 정보는 자신과 연계된 쇼핑몰의 개수 및 정보에 좌우된다는 것이다. 현실

적으로 검색 시간 및 DB 공간의 제약점으로 인해 한 비교 쇼핑 사이트가 모든 쇼핑몰의 정보를 유지할 수 없으며, 소비자는 특정 상품에 대해 최저 가격을 확인하기 위해서는 과거보다는 덜하지만 여러 비교 쇼핑 사이트를 순회해야만 한다.

Yavis 및 Clickprice 사이트는 도서, 음반 및 DVD 등 가격이 매우 유동적인 상품에 대한 정보는 온라인 실시간 검색을 수행하고 있다(이보길, 2000; 조강의, 2001). 온라인 실시간 검색은 비교 쇼핑 사이트의 웹 로봇이 직접 검색어를 자신과 연계된 각 쇼핑몰에게 실시간으로 보내 출력 결과를 받아오는 형태로 수행된다. 그 결과 온라인 실시간 검색은 개별 쇼핑몰 시스템의 성능 및 네트워크 교통량 상황에 따라 많은 응답 지연을 유발하기도 한다.

2.2 Wrapper 기술

Wrapper는 특정 정보 소스(information source)로부터 그 정보를 이해하고 사용자가 필요로 하는 정보만을 추출해 주는 모듈 또는 프로시주어이다. 비교 쇼핑에 적용되는 wrapper는 각 인터넷 쇼핑몰에서 제공되는 하나 이상의 상품 정보 위치와 이를 추출하기 위한 규칙을 포함한다. 쇼핑몰들의 구조 또는 정보 소스가 각각 다르기 때문에 wrapper는 각 쇼핑몰에 대해 하나씩 만들어져야 한다(Cowie, Lehnert, 1996; Kushmerick 등, 1997). wrapper는 사람에 의해 수동 생성되거나 소프트웨어 에이전트에 의해 자동으로 생성될 수 있다. 수동 생성 방식은 단순하고 시간 소비적인 지루한 작업이며 확장성이 떨어진다. 이에 반해, 에이전트에 의해 자동 생성되는 wrapper는 에이전트가 HTML로 작성된 정보를 분석해야 하는데, 다행히 쇼핑몰들은 준 구조화(semi-structured)되어 있어 자연어 처리에서 사용되는 의미 기반 분석 방법을 사용하지 않고도 필요한 정보를 추출할 수 있다. Wrapper 기술 또한 별도 하나의 연구 주제로 현재 여러 연구진이 wrapper에 대해 연구하고 있다(양재영 등, 2001; Cowie, Lehnert, 1996; Kushmerick 등, 1997). 본 논문의 주요 목표는 메타 가격비교 사이트의 구축이므로 에이전트에 의해 자동 생성되는 일반적인 wrapper를 사용한다.

2.3 메타 검색엔진

메타 검색엔진(meta-search engine)은 여러 기존 검색엔진 각각의 특성을 파악하여 통합 구현함으로써, 정보검색시 한번의 질의어 입력으로 여러 검색엔진의 결과 값을 한번에 볼 수 있게 만든 검색엔진이다(Morris, Maglio, 2001). 현재 기존의 여러 검색엔진들에 대하여 일관된 인터페이스 및 질의를 지원하는 메타 검색엔진이 개발되어 사용되고 있다. 또한 검색 효율을 높이기 위해 다중 쓰레드 구조(multi-thread architecture)를 이용하여 대상 검색엔진들에 질의한 결과를 파싱하여 관련 URL 주소 및 요약 내용 등 사용자 옵션에 따라 정보를 보여주도록 개발되고 있다(이보길, 2000; Morris, Maglio, 2001). 본 논문에서 제안하고자 하는 것은 기존 비교 쇼핑 사이트들에 대한 효율적인 메타 검색엔진 기술이다.

메타 검색엔진의 구조는 크게 세 부분으로 나누어진다. 즉, 사용자 인터페이스 부분, 각 검색엔진들에게 질의를 보내는 쓰레드 관리자 부분, 그리고 각 검색엔진으로부터 얻은 결과를 파싱하여 출력하는 부분으로 구성된다. 사용자는 먼저 브라우저에서 제공되는 폼(form) 형식에 따라, 검색어를 자신의 환경 변수나 기본 입출력 스트림(stream)을 통해 서버에게 전달한다. 검색어를 수신한 서버는 각 검색 엔진의 특성에 맞는 형식으로 질의를 재설정하고, 설정된 질의는 다시 '쓰레드 관리자'로 전달되어 각 검색엔진으로부터 질의에 대한 결과를 수집한다. 마지막으로 각 검색엔진으로부터 얻은 결과를 사용자에 친숙한 형식으로 파싱하게 된다. 즉 파싱 알고리즘은 문서의 처음부터 한 문자씩 읽어 HTML 태그 시작 기호부터 태그 끝 기호까지 버퍼에 저장하고 저장된 태그를 이용하여 URL 및 가중치, 요약문 등을 검출하고 불필요한 태그를 제거한 다음 사용자에게 보여주게 된다.

〈표 1〉 비교 쇼핑 사이트 가격 비교

(2003년 2월 11~12일 현재, 단위: 원)

검색어 \ 비교 쇼핑 사이트	제품군	Yavis	Enuri	Bestbuyer	Clickprice	Shopbinder	Omi
삼성 에어컨 AS-G1710	가 전	1,368,000	1,366,000	1,366,000	1,368,000	1,474,000	1,331,000
린나이 가스오븐 RFO-851Q	가 전	520,000	490,000	495,000	520,000	495,000	510,000
런닝머신 웨슬러 400CS	스포츠	1,790,000	1,751,090	1,751,090	1,790,000	결과 없음	1,790,000
HP 레이저젯 1220	컴퓨터	728,600	669,000	685,000	결과 없음	704,000	670,000
올림푸스 디지털카메라 Camedia C-50Z	전 기/ 전 자	745,000	결과 없음	722,000	723,000	745,000	719,000
삼성 TV CT-20F1S	가 전	154,000	153,000	153,000	153,000	154,000	153,000

Ⅲ. 메타-비교 쇼핑 에이전트 시스템 설계

2000년 8월 Advanced Mart[11] 사이트의 인기검색 상품 108개 목록에 대해 Enuri, Shopbinder, Yavis 등 3개의 비교 쇼핑 사이트들에서 가격을 비교 검색하여 보았더니, 검색율[1]이 각각 18.5%, 50.9%, 51.9%, 최저가격 제시횟수[2]가 각각 12, 21, 21회였다. 즉, Shopbinder의 경우 108개 상품 검색요청 중 55개 상품에 대한 정보만 검색하여 주었으며, 55개 상품 검색결과 중 21개 상품에 대해서는 셋 비교 쇼핑 사이트들 중 최저가격을 출력하였다. 51개의 전기/전자/가전 제품군만을 대상으로 가격을 비교 검색하여 보았더니, 검색율이 각각 35.3%, 49.0%, 52.9%, 최저가격 제시횟수가 각각 11, 9, 13회였다(이보길, 2000). 또한 <표 1>의 최근 결과에서도 하나의 가격 비교 사이트에서 각 상품에 대한 최저 가격을 항상 제시하고 있지 않으며, 어떤 사이트는 특정 제품에 대한 정보를 전혀 유지하고 있지 않다. 이상에서 볼 수 있듯이, 어떤 비교 쇼핑 사이트들에서 검색할 수 없는 인기검색 상품도 존재하며 동일 제품에 대해 최저가격을 제시해 주는 사이트도 고정되어 있지 않다.

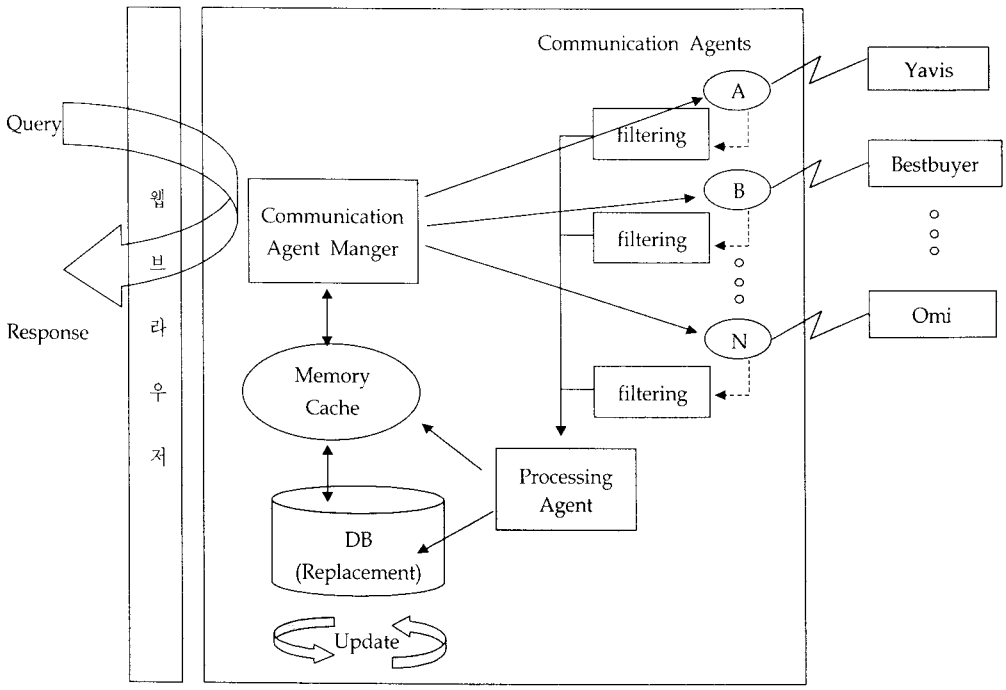
이러한 문제점을 해결하기 위해 기존 비교 쇼핑 사이트들에 대한 '메타 비교 쇼핑 에이전트 시스템'(meta comparison shopping agent system)을 제안한다. 제안된 시스템은 에이전트를 이용하여 소비자 검색빈도가 높은 상품에 대한 정보를 DB에 저장 · 관리하면서, 이들 정보를 기존 비교 쇼핑 사이트들로부터 주기적으로 가져와 DB 정보를 갱신하고 소비자가 요청한 정보를 출력함으로써, 적은 통신부하로 소비자에게 최적의 가격을 제시해 주며 DB에서 적중되었을 경우 응답 시간을 많이 향상시켜 준다. 소비자 검색빈도가 높은 상품에 대한 정보만을 유지하는 첫 번째 이유는 한정된 저장 용량을 활용하여 제안 기법을 구현하기 위해서이며, 두 번째 이유는 정보를 수집하는 작업 부하를 줄이기 위해서이다.

3.1 시스템 개요 및 구조

제안하는 메타 비교 쇼핑 에이전트 시스템은 <그림 1>과 같이, 통신 에이전트 관리자(communication agent manager), 필터링 에이전트(filtering agent), 처리 에이전트(processing agent), DB와 DB 레코드 갱신 에이전트(database record update agent), 메모리 캐시(memory cache) 및 메모리 캐싱 에이전트(memory ca-

1) 108개 상품에 대해 정보검색을 각 비교 쇼핑 사이트에 차례차례 요청하였을 때, 108개 상품 중에서 상품 가격정보가 실제 검색되어 출력된 비율

2) 108개의 상품 검색 중에서, 한 비교 쇼핑 사이트가 셋 비교 쇼핑 사이트들 중 최저 상품가격을 검색하여 출력한 회수



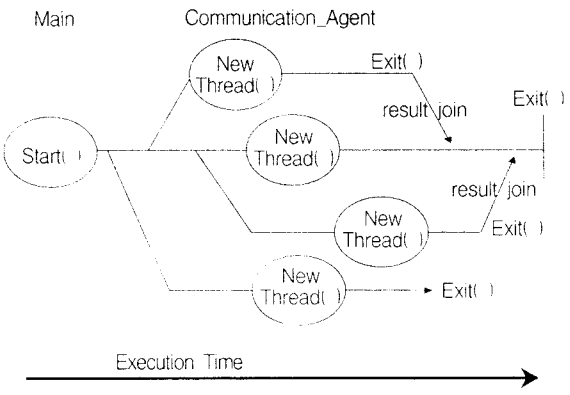
〈그림 1〉 메타 비교 쇼핑 에이전트 시스템의 구조

ching agent) 등으로 구성된다. 최초 DB에 저장될 상품 정보로는 Advanced Mart[11]에서 측정하여 놓은 인기 검색상품 100 순위를 참조하여 구성하였으며, 이 값들은 초기 데이터로 사용된 후 우리의 알고리즘에 따라 차후 자동 갱신된다. DB에 초기 데이터를 구축한 후, 웹을 통해 소비자로부터 상품검색 요청을 받게 되면 '통신 에이전트 관리자'는 '처리 에이전트'와 협력하여 DB와 메모리 캐시에 해당 상품에 대한 정보가 있는가를 조회하게 된다. 정보가 지역 DB에 있으면 그 정보를 즉시 출력하고 다음 검색 요청을 서비스할 준비를 한다. 만약 DB에 없다면 '통신 에이전트'를 통해 비교 쇼핑 사이트들에 대해 온라인 검색을 수행하게 된다. 온라인 검색 결과는 '필터링 에이전트'(filtering agent)에 전달되어 필요한 정보가 추출되며, 추출된 정보들은 '처리 에이전트'에 의해 수집되어 소비자에게 출력되며 동시에 지역 DB 및 메모리 캐시에 저장된다. 이때 필요하면 DB 레코드가 교체된다. DB의 정보는 주기적으로 갱신되어 최근의 가격 정보가 유지된다. 관련된 모듈들에 대한 내용을 좀 더 구체적으로 기술하겠다.

3.2 시스템 주요 모듈

3.2.1 통신 에이전트 관리자

소비자 요청을 웹을 통해 받아들여 해당 정보가 이미 지역 DB에 있으면 즉시 출력해 주고, 없다면 각 기존 비교 쇼핑 사이트에 대응되는 통신 에이전트를 통하여 질의어를 전송하여 결과를 받아오는 기능을 담당한다. Java로 구현된 통신 에이전트 관리자는 <그림 2>에서처럼 start( ) 수행 후 각 통신 에이전트마다 쓰레드를 생성하여 수행하며, 각 비교 쇼핑 사이



〈그림 2〉 통신 에이전트의 생성과 동기화

〈표 2〉 Yavis 및 Shopbinder에 대한 Form action 정보

구 분	Yavis	Shopbinder
Action	www.yavis.com/yavis/searchTotal.jsp	www.shopbinder.com/search.asp
Method	GET	POST
Parameter	goods = ?????	goodname = ?????&dir = similar_word = Y

트와 통신하여 결과 값들을 수집하게 된다. 먼저 수행을 끝낸 쓰레드는 다른 쓰레드가 수행을 완료할 때까지 기다린 후 join하여 수집한 결과 값들을 필터링 에이전트에 전달한다.

● 통신 에이전트

통신 에이전트는 통신 에이전트 관리자가 전달한 질의어를 기존 비교 쇼핑 사이트에 전송하여 결과를 받아오는 기능을 수행한다. 따라서 통신할 비교 쇼핑 사이트들에 대한 입력 정보, 즉 *Form action* 정보를 미리 파악하여야 한다. 예를 들어 Yavis와 Shopbinder에 대한 *Form action* 정보는 <표 2>와 같다.

<표 2>에서 Action 정보는 비교 사이트 검색엔진의 검색 페이지 이름이고, Method는 서비스 요청 방식으로 GET/POST와 같은 방식이 있다. Parameter는 실제 질의어를 전달하는 부분과 기타 검색엔진에 필요한 정보를 전달하는 부분으로 이루어져 있으며, 'name = value & name = value & …' 등과 같이 표현된다. Shopbinder를 예로 들면, 질의어를 전달하는 부분은 goodname = ?????에서 '?????' 부분이다. 그리고, dir = similar_word 등의 정보는 Shopbinder 검색엔진에 필요한 추가 정보들이다. <표 2>의 정보들을 바탕으로 검색 엔진에 HTTP 요청을 보내어 응답을 받으면, 이 응답이 포함된 HTML 문서의 내용을 필터링 에이전트가 파싱하여 처리하게 된다.

3.2.2 wrapper 및 필터링 에이전트

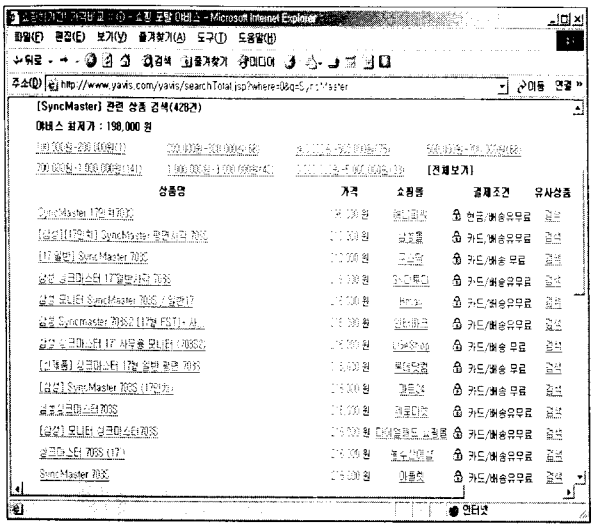
통신 에이전트가 수집한 결과에서 필요한 정보를 추출하기 위해 wrapper 기술을 사용한다. <그림 3>은 소비자가 비교 쇼핑 사이트에 질의를 요청했을 때 응답으로 출력되는 가격정보 테이블의 일반적인 구조이다. L_1에서 L_n은 각 열이 어떤 정보를 가지고 있는지를 나타내는 레이블이고 $P_{m,1}$부터 $P_{m,n}$은 한 상품 m에 대해 소비자에게 제공하는 정보를 나타낸다. 테이블형 정보 구조에서 가격 정보를 추출하기 위한 wrapper 알고리즘은 L_k, $P_{i,j}$에서 각 정보의 표현 및 위치에 대한 내용을 분석한다. 위치정보는 레이블 분석을 통해서 간단히 학습할 수 있으며, 가격정보는 숫자로 표현되므로 표현 구조를 통해 학습 가능하다. 즉, 레이블과 온톨로지(ontology)를 비교하여 그 열이 가지고 있는 정보를 분석하게 된다.

L_1	L_2	…	L_n
$P_{1,1}$	$P_{1,2}$	…	$P_{1,n}$
$P_{2,1}$	$P_{2,2}$	…	$P_{2,n}$
⋮	⋮	⋮	⋮
$P_{m,1}$	$P_{m,2}$	…	$P_{m,n}$

〈그림 3〉 가격 정보 테이블의 구조

<그림 4>는 yavis 사이트에서 "SyncMaster" 검색어로 질의한 결과를 보여 준다. 첫 번째 행의 "상품명", "가격", "쇼핑몰", "결제조건", "유사상품"이 <그림 3>의 L_1, L_2, L_3, L_4, L_5에 각각 해당된다. L_1, L_2, L_3, L_4, L_5로 구성된 다섯 개의 항목 *item_number[1; 5]* 중에서, 필요한 정보, "상품명"(L_1)과 "가격"(L_2) 항목을 추출하기 위해 *item_number[1; 2]*란 규칙을 생성 · 적용할 수 있다. 현재 테이블형 구조에서 레이블명을 이용하여 필요한 위치정보를 인식한다. 모든 가격비교 사이트는 검색 결과를 테이블 형태[3]로 보여주지만 가격 및 상품명을 나타내는 레이블은 서로 다르다. 실제, "상품명" 레이블 대신에 Shopbinder는 "제

3) Bestbuyer의 경우 '텍스트로 보기'(테이블 형)과 '사진으로 보기'가 가능한데, 본 시스템에서는 '텍스트로 보기' 페이지를 파싱함.



〈그림 4〉 www.yavis.com의 결과 테이블 정보 구조

품명"을, Enuri는 "모델명"을 사용한다. "가격" 레이블 대신에 Enuri는 "최저가"를 사용한다. 따라서, 검색 결과에서, 가격 정보를 나타내는 레이블을 찾기 위해 {'가격', '최저가', '제품가격'} 등의 **가격 표현 용어목록**을 사용하며, 상품명 정보를 나타내는 레이블을 찾기 위해 {'상품명', '제품명', '모델명'} 등의 **상품명 표현 용어목록**을 사용한다.

● 필터링 에이전트

필터링 에이전트는 통신 에이전트가 수집한 데이터를 파싱하여 상품명, 하이퍼링크, 가격정보 등을 추출하는 기능을 수행한다. 이를 위해 먼저 가격정보가 포함된 테이블을 찾아야 하는데, 수집된 HTML 문서를 분석하여 가격 정보 테이블을 찾아내는 알고리즘이 <그림 5>에 나타나 있다. 이 알고리즘은 소비자가 입력한 검색어를 가장 많이 포함한 TABLE을 가격목록 테이블이라고 판단한다. <그림 6>은 가격정보가 포함되어 있는 가격목록 테이블에서 가격 및 상품 정보가 어느 열에 위치하고 있는 지를 찾아내는 알고리즘이

Step 1. 검색결과인 HTML 문서에서 <TABLE> 태그를 만날 때까지 한 줄씩 읽어 처리한다.
새로운 테이블 시작을 나타내는 <TABLE> 태그가 없다면 다음 줄을 읽어 처리하고,
<TABLE> 태그를 만나면 Step 2로, 문서의 끝이면 Step 3으로 간다.

Step 2. 이 테이블을 k번째 테이블이라고 하고, 각 행(row)에 대해 다음을 반복
2.1 table_array[k][1]에 현 테이블의 시작 위치 저장
2.2 if(현재 행에서 <TABLE>을 만나면) // 테이블이 중첩된 경우
nested_level을 증가
2.3 if(현재 행에서 검색어가 발견되면) count[k]를 증가시킴
2.3 </TABLE>가 아니면 2.2로 가서 반복
2.4 </TABLE>을 만나면 nested_level을 감소시킨 다음, 아래를 수행
nested_level ≥ 0이면 2.2로 가서 반복
nested_level < 0이면 table_array[k][2]에 현 테이블의 마지막 위치 저장
2.5 k를 증가시킨 후 Step 1로 간다.

Step 3. count[ ]의 값을 비교, 모든 count[ ] 값이 같다면 에러 메시지 출력
3.1 제일 큰 count[ ] 값을 가진 테이블을 가격목록 테이블이라 판단
3.2 table_array[ ][ ]을 이용하여 가격 테이블의 시작위치와 마지막위치를 조회하여
전체 HTML 문서에서 가격목록 테이블만 추출

보조설명	• HTML 문서에서 <TABLE>과 </TABLE> 태그는 테이블의 시작과 끝을 각각 의미함. • table_array[ ][ ]을 이용하여 검색결과 페이지에 있는 각 테이블의 시작과 끝 정보를 저장 • 변수 nested_level의 초기 값은 0으로 테이블(들)의 중첩 단계를 나타냄.

〈그림 5〉 HTML 문서를 파싱하여 가격목록 테이블을 추출하는 과정

다. 즉, 가격표현 용어 목록과 상품명 표현 용어 목록을 참조하여 테이블의 몇 번째 열에 가격 및 상품명 정보가 있는지를 비교하여 알아낸다. 각 비교 쇼핑 사이트마다 가격 정보의 위치가 같지 않기 때문에 이러한 작업 수행은 일반적이다. 기본적으로 한 줄('<TR>'부터 '</TR>')안에 각 상품의 정보가 유지되기 때문에 상품정보와 링크정보를 알아내는 것은 비교적 쉽다.

```
Step 1. 가격목록 테이블의 레이블 행에 있는
        각 레이블에 대해서 다음을 수행
    {'상품명', '제품명', '모델명'} 단어목록을 이용
      하여 상품명을 표현하는 레이블 위치 m을
      파악
    {'가격', '최저가', '제품가격'} 단어목록을 이용
      하여 가격을 표현하는 레이블 위치 n을 파악

Step 2. 가격목록 테이블에서 레이블 행을 제외한
        각 행에 대해 다음을 수행
    m번째 열에 있는 항목을 price_info_table[ ]
      [goods]에 저장
    n번째 열에 있는 항목을 price_info_table[ ]
      [price]에 저장
```

〈그림 6〉 가격정보를 알아내기 위한 패턴 검색

3.2.3 처리 에이전트

필터링 에이전트가 추출한 정보를 수합하여 DB에 입력하고, 그 정보가 소비자가 요청한 질의어에 대한 결과라면 정보를 소비자에 출력한다. DB에 입력되는 정보는 각 비교 쇼핑 사이트(검색엔진)로부터 단순 수합된 것이므로, 이 경우 검색엔진은 다르더라도 쇼핑몰이 같다면 같은 상품이 중복 저장될 수 있다. 중복 저장을 피하기 위해, 본 연구진은 처리 에이전트 내에 재처리(re-processing) 기능을 포함시키는 작업을 진행중이다. 즉, 서로 다른 검색엔진이 같은 상품 정보를 동일 쇼핑몰로부터 검색한 정보인지를 조사하여, 한 쇼핑몰에 대해 상품정보 하나만 DB에 저장하는 기법을 구현중이다. <그림 7>은 통신 에이전트 관리자로부터 요청된 상품정보가 DB에 있으면 바로 출력하여 주고, 그렇지 않은 경우 여러 에이전트와 협력하여 상품 정보들을 가져온 후 DB에 저장하고 필요하면 소비자에 출력시켜 주는 과정을 보여준다. 온라인 실시간 검색 오버헤드를 줄이기 위해, 가격이 유동적인 상품에 대해서도 가격 정보를 비교 쇼핑 사이트로부터 미리 검색하여 지역 DB에 저장할 수 있다.

```
DB에 접속;
if (사용자 질의어에 대해 DB에 해당 상품 정보가
    없다면) {
    에이전트들과 협력하여 비교 쇼핑 사이트로부터
      질의어를 검색하여 결과 수합;
    수합한 정보를 DB에 입력; DB 용량이 초과될
      경우 레코드 교체 정책 적용;
}
질의어에 대한 상품 조회 횟수를 증가시키고, 현재
  조회시간을 입력;
해당 상품에 대한 정보를 출력;
```

〈그림 7〉 처리 에이전트의 주요 작업

지역 DB 용량이 여유가 있을 때는 새로운 상품정보를 계속 DB에 저장할 수 있지만, DB가 꽉 찼을 경우에는 기존 상품정보를 교체하여 공간을 확보한 후 새로운 상품정보를 저장한다. 즉, 소비자가 질의한 상품정보가 DB에 없어 비교 쇼핑 사이트로부터 상품정보를 검색하여 DB에 저장하려 할 때 남은 DB 공간이 없다면, DB 레코드 교체(record replacement, 또는 상품정보 교체) 정책을 적용하여 기존 DB 레코드를 제거하고 그 위치에 새로운 상품정보를 저장한다. 이러한 DB 정보교체를 통해 제한된 DB 공간을 효율적으로 활용할 수 있다. DB에 우선 유지되는 정보는 소비자 검색 빈도가 높은 상품 및 소비자가 최근에 검색한 상품정보이다. DB 용량 초과시 적용되는 교체 정책에서는 일주일 이전에 조회된 상품정보 중에서 소비자 조회 횟수가 낮은 상품정보 순으로 레코드가 교체되고 그 자리에 새로운 상품정보가 저장된다. 만약 교체대상으로 선정된 상품정보들의 조회 횟수가 동일할 경우에는 조회시간이 오래된 상품 정보를 교

체한다. 조회횟수 및 조회시간이 동일할 경우에는 임의 순서로 교체한다. 시스템 구축 초기에는 모든 상품의 조회시간이 일주일 이내이므로 단순히 조회 횟수가 적은 레코드가 교체된다. 나중에는 일주일 내에 참조된 상품은 새로운 상품정보이므로 조회 횟수가 적더라도 DB에 유지된다.

3.2.4 DB 레코드 갱신 에이전트

DB에서 유지되고 있는 상품정보는 항상 변할 수 있는 가격정보를 포함하고 있기 때문에 최신 정보를 유지하기 위해 주기적으로 갱신되어야 한다. 제안된 시스템은 <그림 8>와 같이 주기적으로 매일 한번 DB 상품정보를 갱신한다. DB 전체 갱신은 부하가 많은 작업이므로, 본 시스템에서는 시스템 작업부하가 작은 매일 오전 3시에 갱신이 자동으로 수행되도록 하였다. 또한 정보가 갱신된 시간을 DB에 저장하여 소비자에게 출력할 수 있도록 하였다.

```
for (each RowSet of DB) {
        가격 비교 사이트들로부터 해당
          RowSet에 대한 정보를 수집;
        DB의 해당 RowSet을 갱신; 필요하면
          레코드 교체 정책 적용;
}
```

〈그림 8〉 DB 테이블에서 가격정보의 갱신

3.2.5 메모리 캐싱 에이전트

질의 요청에 대한 응답 시간을 단축시키기 위해, 조회시간이 최근이고 조회 횟수가 많은 상품정보를 메모리 공간 내에 유지하는 에이전트이다. 최근 조회된 정보라면 참조 지역성(locality of reference)에 의해 앞으로도 참조될 가능성이 높다고 보고, 해당 상품정보를 DB에서 가져와 메모리 캐시에 유지한다. 소비자가 요청한 질의가 이 캐시에서 적중되면 즉시 소비자에게 출력되므로 온라인 실시간 검색 및 지역 DB 검색보다 응답이 빠르다. 또한 DB 레코드와 마찬가지로, 갱신 에이전트에 의해 매일 오전 3시에 캐시 정보가 갱신된다.

Ⅳ. 구현 및 논의

4.1 구 현

구현 환경으로, 서버 측 응용 엔진 구축을 위해 Java Development Kit 버전 1.3(JDK 1.3)과 Java 2 SDK, Standard Edition 버전 1.3(J2SDK 1.3)을, 웹 사이트의 동적 구성을 위해 JSP를 사용하였다. 웹 서버로는 Apache version 1.3.19를 사용하였고, 웹 서버 내에 자바 서블릿(Java Servlet)과 Beans를 지원하기 위해 Jakarta-Tomcat 3.2.1을 사용하였다. DBMS로는 My SQL 3.23.41 for pc-linux-gnu(i686)를 이용하였고, 업데이트 엔진과 교체정책을 적용하는 에이전트에 관련된 모듈들은 JDK 1.3으로 직접 작성하였다. 개발에 사용된 자바 서블릿은 멀티쓰레드를 지원하기 때문에, 초기화에 따른 오버헤드가 거의 없고 또 사용자 수 증가에 따라 시스템 성능이 저하되지 않는 장점이 있다.

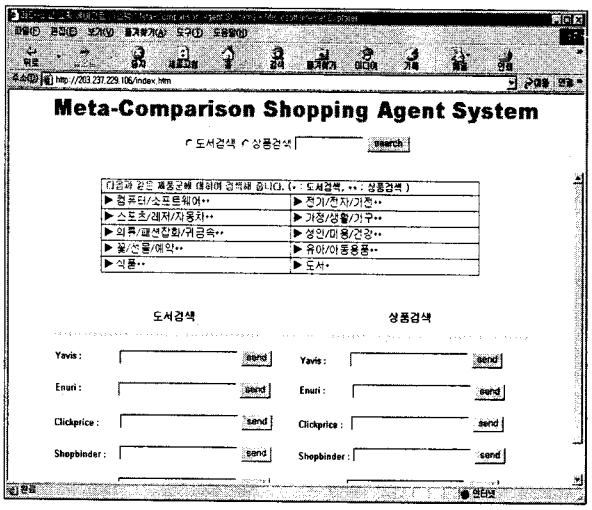
기존 비교 쇼핑 사이트들은 각각 입력 폼을 받아들이는 형태가 다르다. 본 연구에서는 대상 사이트들과 통신하는 통신 에이전트들을 자바로 작성하였다. 본 시스템이 동작할 때의 초기화면이 <그림 9>에 나타나 있으며, 현재는 도서검색과 컴퓨터 등의 상품검색 부분만을 구현하였다. 그림의 상단의 '*search*' 칸에 상품명을 입력하면 여러 비교 쇼핑 사이트를 대상으로 메타 검색이 수행되며, 하단 각 사이트 명 옆의 '*send*'칸에 상품명을 입력하면 해당 비교 쇼핑 사이트만을 대상으로 검색이 이루어지도록 구축하였다. 기존 비교 쇼핑 사이트에서는 도서검색이 일반 상품검색과 달리 온라인 실시간으로 수행되었기 때문에 응답 속도가 매우 느렸으나, 본 사이트에서는 경우에 따라 메모리 캐시 또는 지역 DB에서 적중될 수 있어 평균 응답 시간이 매우 향상되었다. 이외에도 메모리 캐시 및 지역 DB의 주기적인 자동 갱신을 위해 cron table[15]을 활용하였다.

- **데이터베이스 구조**: 지역 DB 테이블은 상품에 대한 가격 정보를 유지하는 itemtable로 구성된다. 이 테이블에 대한 간단한 설명과 속성은 다음과

같다.

- itemtable(itemname, price, site, link, count, atime, mtime): 상품에 대한 제품명, 가격, 조회한 사이트, 추가정보 용 링크, 조회 횟수, 조회시간, 갱신시간 등의 정보가 저장됨.

지역 DB에 해당 정보가 없어 실시간 검색을 하는 경우에는 조회 횟수는 1, 조회 시간은 당일 현재 시간이 되며 화면에는 출력되지 않는다.

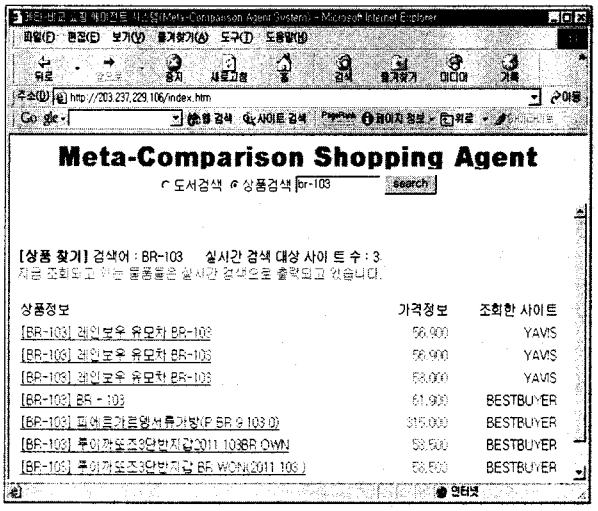


〈그림 9〉 메타-비교 쇼핑 에이전트 시스템 초기화면

4.2 테스트 및 수행 결과

구현된 시스템을 이용하여 상품을 검색한 결과 화면이 <그림 10>과 <그림 11>에 나타나 있다. 검색 요청 시 먼저 지역 메모리 캐시와 DB를 검색하는데 적중되면 상품 정보를 즉시 출력하며, 적중되지 않으면 실시간 검색을 하여 출력한다. 그림에 나타나 있듯이, 현재 실시간 검색시 접속되는 사이트는 Yavis, Enuri, Bestbuyer 등 3개이다. <그림 10>은 소비자가 요청한 상품 정보가 지역 DB에 유지되고 있을 때, 통신 에이전트 관리자와 처리 에이전트가 그 상품에 대한 리스트를 조회하여 화면에 출력한 결과를 보여준다. 출력 정보에는 소비자들이 해당 물품에 대하여 조회한 횟수와, 가격정보가 갱신된 날짜 등이 포함되어 있다. <그림 11>은 소비자가 요청한 상품이 지역 메모리 캐시 및 DB에 등록되어 있지 않은 경우, 온라인 실시간 검색을 수행하여 소비자에 출력시킨 결과를 보여준다.

〈그림 10〉 지역 DB에서 적중시 상품검색 화면결과



〈그림 11〉 온라인 실시간 검색시 상품검색 화면결과

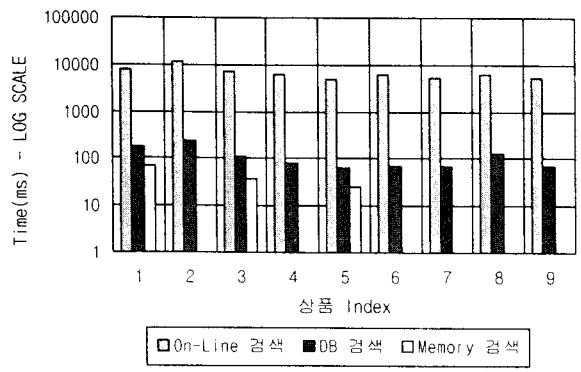
본 시스템을 이용하여 온라인 검색과 지역 DB에서 검색한 상품의 응답속도를 비교하기 위하여 2001년 11월 21일 현재 Auction[12]에서 제공하는 3종류 제품군에 대해 임의로 선별된 인기상품 목록 9개를 대상으로 Enuri, Bestbuyer, Yavis 등 3개의 비교 쇼핑 사이트들에 대하여 같은 상품에 대한 가격을 검색하였

〈표 3〉 구현 시스템에서의 테스트 결과

(단위: millisecond)

Category	상 품 명	온라인 검 색	저장공간에 정보 유지		DB Hit	메모리 Hit
			DB	메모리 캐시		
전 자 제품군	삼성 PAVV SVP-43J7P	7,991	183	69	◎	◎
	삼성 SyncMaster 703S	11,731	242		◎	
	대우 에어컨 DP-304M	7,298	118	39	◎	◎
생 활 용품군	NEO 5001(책상)	6,017	84		◎	
	사무용 듀오백의자 GD-024	5,046	64	25	◎	◎
	알파 맥반석 돌침대 S-4016	6,204	70		◎	
의 류 잡화군	크로노스 항공잠바	5,221	68		◎	
	마시마로/엽기토끼 캐릭터 가방	6,239	128		◎	
	레인보우 유모차 BR-103	5,300	72		◎	

다. 9개 상품에 검색 시간 결과가 <표 3>과 <그림 12>에 나타나 있다. <그림 12>는 <표 3>의 결과 값을 그래프로 전환하여 log scale의 그래프로 나타낸 것이다. 표 및 그림에 나타난 바와 같이 온라인 실시간 검색, 지역 DB 및 메모리 캐시 검색의 시간 차이는 크다. 가격 비교 사이트의 작업부하 상태, 네트워크 환경 등에 따라 검색 시간이 조금씩 차이를 보이고는 있지만, 캐시 검색의 성능이 우수함을 알 수 있다. 검색 요청이 메모리 캐시에서 적중되는 경우 기존 검색과 비교하여(각 비교 쇼핑 사이트로 질의어를 전송하는 통신 시간 + 각 비교 쇼핑 사이트가 DB 접근을 위해 디스크를 접근하는 시간 + 검색된 정보를 전송하는 통신 시간 + 수집된 정보에서 필요한 정보를 추출하는 시간) 등이 줄어들어, 응답시간이 매우 향상되었다. 해당 에이전트가 수집한 정보의 양에 따라서도 검색 시간이 차이가 나는데, 삼성 'PAVV SVP-43J7P'와 '대우에어컨 DP-304M' 결과를 비교하여 보면 정보가 더 많은 첫 번째 상품의 검색 시간이 더 길다.



〈그림 12〉 검색 시간 그래프

본 검색기법과 기존 검색엔진과의 성능 비교를 위해, 상품 검색시 평균 응답시간을 비교한 결과가 <표 4>에 나타나 있다. 표의 결과는 10개 상품정보를 하나씩 차례차례 검색한 후 10개의 응답시간을 구한 다음 평균한 값이다. 이때 각 비교 쇼핑 사이트에서 소비자가 조회한 10개 상품정보를 모두 획득할 수 있다는 상황을 가정하였다. 본 시스템의 경우, 10개 상품정보 중에서 8개 정보는 지역 DB에서 적중되고 2개 정보는 온라인 검색에 의해 얻어지는 상황[4]에서 구한 평균 응답시간이다. 비교 결과를 보면 기존 시스템보다 제안 시스템의 평균 응답시간이 빠르다. 소비자가 검색 요청한 정보가 지역 DB나 특히 캐시에서 더 많이 적중될수록 제안 시스템의 성능은 향상된다. 물론, 소비자가 검색 요청한 정보가 지역 DB에 없어 온라인 검색이 많이 이루어진다면 제안 시스템의 성능은 나빠지게 된다. 최악의 경우 항상 온라인 검색이

4) 실제 본 시스템에서 여러 번 실험한 결과, 소비자 검색요청 중 약 80%는 조회 수가 높은 인기검색 상품에 대한 질의로 지역 DB에서 적중되었으며, 약 20%는 지역 DB에 없는 상품에 대한 질의로 온라인 검색이 수행되었다.

수행된다하더라도 제안 시스템은 메타검색 사이트로서 가치가 있다. 즉, Enuri는 '삼성 SyncMaster 703S', 'NEO 5001(책상)', '유모차 BR-1022'에 대한 정보를 제공하지 못했다. Bestbuyer는 '대우 에어컨 DP-312M' 및 '유모차 BR-1022'에 대한 정보를, Yavis는 '대우 에어컨 DP-312M'에 대한 정보를 제공하지 못했다. Yavis의 경우 매우 많은 상품에 대한 정보를 제공하였지만 다른 비교 쇼핑 사이트보다 상품가격이 높았다. 따라서, 소비자가 기존의 한 비교 쇼핑 사이트만 방문해서는 자신이 원하는 상품 및 최저가격 정보를 얻는다는 보장이 없다. 만약 한 비교 쇼핑 사이트에서 원하는 상품 및 가격 정보를 얻는데 실패하였다면 소비자는 다른 비교 쇼핑 사이트(들)를 방문해야 하는데, 이 경우 기존 비교 쇼핑 사이트의 응답시간은 제안 시스템보다 더 늦어지게 된다. 이러한 면들을 종합적으로 고려할 때 제안한 메타검색 시스템은 가치가 있다.

〈표 4〉 평균 응답시간의 비교

	제 안 시스템	Yavis	Enuri	Bestbuyer
평균응답 시 간	1,437ms	1,952ms	3,402ms	4,519ms

4.3 실제 비즈니스 활용 방안

본 논문의 주요 목적은 소비자들이 기존의 가격 비교 사이트들을 일일이 방문할 필요 없이, 본 사이트만 한번 방문하여 원하는 상품에 대한 정보를 효율적으로 얻을 수 있는 프로토타입 시스템을 구현하고 테스트하여 그 가능성을 확인하는 것이다. 본 논문에서 제안된 기법을 실제 비즈니스에 적용하기 위해서는 DB 등 시스템 용량 등의 대한 고려가 필요하나, 주요 아이디어는 별 수정 없이도 활용 가능하다고 본다. 초기에 Advanced mart 사이트의 인기검색 상품 100여 개 목록을 참조하여 DB를 구축하지만, 이후에는 본 사이트를 방문한 고객들의 각 상품에 대한 실제 조회수를 누적시켜 조회 수가 많은 상품들을 인기 품목으로 유지한다. 초기 DB 구축시에 Advanced mart 이외의 다른 사이트의 인기품목 정보를 같이 고려한다 하더라도, 이는 첫 사이트 오픈 후 초기에 방문한 고객들의 검색요청이 적중될 때 일시적으로 빠른 서비스를 제공하기 위한 것이지 더 이상의 의미는 없다. 본 사이트 오픈 후 어느 정도 시간이 지나 방문하는 고객이 많아지게 되면 각 상품에 대한 조회 수에 근거하여 자체적으로 인기상품이 관리된다. 만약, Advanced mart의 인기검색 상품 분류의 신뢰도가 낮아 최초 DB에서 소비자 검색요청이 적중되지 않는 경우가 있다면 DB에 있는 상품정보는 소비자가 검색 요청한 상품 정보로 교체되어, 최초 DB에 저장된 어떤 상품 정보는 제거된다. 이처럼 Advanced mart 인기검색 상품정보는 최초 DB 구축시에 활용되는 일시적인 것으로, 시스템이 안정되면 본 사이트의 인기품목 관리 정책이 적용된다. 이러한 정책이 고객으로 하여금 본 사이트를 더욱 방문하게 만드는 것이며, 본 사이트의 고객 우대 차원이나 경영 전략 차원에서 타당하다고 생각된다. 또한 Auction 사이트의 인기검색 상품 목록들을 대상으로 본 사이트의 성능을 적중 여부와 응답시간 측면에서 평가하여 본 논문의 인기품목 관리 정책이 타당함을 보였다.

상품 가격 정보는 끊임없이 변하므로 관련 DB 정보를 주기적으로 갱신해야 한다. 본 메타검색 시스템의 DB 정보는 시스템 및 통신 부하가 적은 매일 오전 3시에 갱신에이전트가 비교 쇼핑 사이트의 정보를 검색하여 갱신하도록 하였다. 만약 고객 수가 증가하여 더 많은 상품정보를 저장할 필요가 있을 경우에는, 별도 대용량 DB 서버 및 고성능 사용자 요청 처리 서버를 각각 도입해야 한다. 이들 서버를 이용하여 더 많은 상품정보 및 고객 정보를 체계적으로 관리해야 하며 또한 고객 요청을 신속히 처리해야 한다. 대용량 DBMS로는 Oracle이 적합하며, 고성능 서버 구입 비용이 많이 든다면 고성능인 아닌 일반 서버들을 서로 고속 통신 선으로 연결하여 클러스터링 시스템을 구축함으로써 비용을 절감할 수 있다. 마지막으로 실제 상용 시스템이 구축되어 비즈니스를 하게 된다면 최선의 DB 정보 갱신 주기 및 교체 정책 등은 주기적으로 DB 접근 형태를 분석하여 결정하는 것이 바람직

하다.

V. 결론 및 향후 연구

현재 비교 쇼핑 사이트가 구축되어 있지만, 각 사이트가 모든 인터넷 쇼핑몰로부터 정보를 수집하는 것이 아니기 때문에 최저 가격을 얻기 위해서 소비자는 각 비교 쇼핑 사이트를 여전히 순회해야 한다. 또한 서적이나 음반 등의 상품은 가격이 유동적이므로 매번 온라인 실시간 검색을 수행하고 있는데 이는 응답 시간의 지연을 초래한다. 본 논문에서는 각 비교 쇼핑 사이트들만을 메타검색 함으로써 적은 노력으로 주어진 상품에 대한 상대적인 최저가격을 제공할 뿐만 아니라, 기호 상품 정보를 지역 시스템의 메모리 캐시와 데이터베이스 공간에 유지시킴으로써 평균 응답 시간을 대폭 줄여 주는 에이전트 시스템을 제안하였다. 제안한 시스템을 실제 구축한 후 테스트를 통한 실험에서 좋은 성능 향상을 보였다.

앞으로 필요한 정보만을 효율적으로 추출할 수 있는 wrapper 기술에 대해서 먼저 연구하고자 한다. 또한, 선호도가 높은 상품에 대하여 소비자들이 그 물품을 얼마나 선호하는지를 측정하고, 월별 또는 계절별로 선호되는 상품군·품목 검색 기능 등을 두어 소비자들이 보다 쉽게 정보를 얻을 수 있는 시스템 등에 대한 추가 연구가 필요하다.

† 이 연구는 2003학년도 단국대학교 대학연구비의 지원으로 연구되었음.

참 고 문 헌

서영우, "전자상거래의 총아, 에이전트", 프로그램 세계, Jun. 1999, pp.155-166.

서희경, 양재영, 최중민, "준구조화 정보소스에 대한 지식기반 Wrapper 학습 에이전트", *정보과학회 논문지: 소프트웨어 및 응용*, 제29권, 제1-2호, 2002, pp.42-52.

신봉기, 김영환, "웹 에이전트", *정보과학회지*, 제15권, 제3호, 1997, pp.61-67.

양재영, 최중민, 김중배, "비교 쇼핑 에이전트 시스템", HCI 2000 학술대회 발표논문집, 2000, pp. 851-856.

양재영, 김태형, 최중민, "MORPHEUS: 확장성이 있는 비교 쇼핑 에이전트", *정보과학회 논문지: 소프트웨어 및 응용*, 제28권, 제2호, 2001, pp.179-191.

이보길, "비교 쇼핑 사이트들에 대한 메타검색 기법", 단국대학교 대학원 전산통계학과 석사학위 논문, 2000.

조강의, "캐싱 기법을 이용한 비교 쇼핑 사이트들에 대한 메타검색의 성능 향상", 단국대학교 대학원 전산통계학과 석사학위 논문, 2001.

Cowie, J., and Lehnert, W., "Information Extraction," *Communication of the ACM*, Vol. 39, No. 1, 1996, pp.80-101.

Kushmerick, N., Weld, D. S., and Doorenbos, R., "Wrapper Induction for Information Extraction," In *International Joint Conference on Artificial Intelligent(IJCAI)*, Japan, 1997.

Morris, J., and Maglio, P., "When Buying On-line, Does Price Really Matter?," Proceedings of the Conference on Human Factors in Computing Systems (CHI 2001), Seattle, WA, April 2001.

Advanced Mart, 2002, http://www.am.co.kr.

Auction, 2002, http://www.auction.co.kr.

Bestbuyer, 2002, http://www.bestbuyer.co.kr.

Clickprice, 2002, http://www.clickprice.co.kr.

Cron Table, 2002, http://kldp.org/Translations/html/LAME/using-cron.html.

Enuri, 2002, http://www.enuri.com.

Omi, 2002, http://www.omi.co.kr.

Shopbinder, 2002, http://www.shopbinder.com.

Webnara, 2002, http://www.webnara.com→http://www.commero.com.

Yavis, 2002, http://www.yavis.com.

Information System Review
Volume 5 Number 1
June 2003

An Efficient Meta-Search Scheme for Comparison Shopping Sites

Kang-Eui Cho* · Seong Je Cho*

Abstract

With the spread of electronic commerce on the Internet, comparison shopping sites with agent technique are getting popular for the best shopping. However, most consumers are still spending much time to search for the best price through the sites because each of them may show a different price even for the same goods or a site does not show any information about specific goods. Additionally, the search for the best price of the goods like books and CDs may cause the system to be overloaded and the response time to be long due to an on-line real-time search. In this paper, we have designed and implemented a meta-search system for comparison shopping sites with a local database and memory cache to resolve the above problems. The proposed system collects and maintains the price information of popular goods among the comparison shopping sites using several software agents. The experimental results show that our system is an efficient meta-comparison shopping engine and reduces the latency of the response time with little overhead.

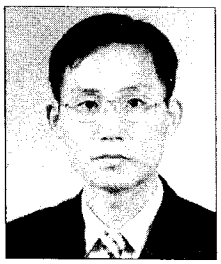
Keywords: Comparison shopping sites, Meta-search, Real-time search, Agent

* Division of Information and Computer Science, Dankook University

◐ 저 자 소 개 ◑

조 강 의 (choke93@daum.net)

단국대학교 식품공학과에서 학사 학위, 단국대학교 대학원 전산통계학과(현 컴퓨터과학 및 통계학과) 석사 학위를 취득하였다. 대민전자와 Natual Approach에서 펌웨어 및 자연어 처리 검색 시스템 분야에 연구개발 경험이 있다. 주요 관심분야는 에이전트 시스템, 전자상거래, 인터넷 응용 등이다.



조 성 제 (sjcho@dankook.ac.kr)

서울대학교 컴퓨터공학과에서 학사, 석사, 박사 학위를 취득하였다. 미국 University of California, Irvine에서 객원 연구원이었으며, 현재 단국대학교 정보컴퓨터학부 컴퓨터과학전공 부교수로 재직 중이다. 주요 관심분야는 전자상거래 및 인터넷 보안, 에이전트 시스템, 시스템 소프트웨어, 실시간 시스템, 분산 시스템 등이다.