준구조화된 정보소스에 대한 지식기반의 Wrapper 학습 에이전트

(A Knowledge-based Wrapper Learning Agent for Semi-Structured Information Sources)

서 희 경 † 양 재 영 †† 최 중 민 †††
(Heekyoung Seo) (Jaeyoung Yang) (Joongmin Choi)

요 약 정보추출은 한 문서에서 그 문서의 중심적 의미를 나타내는 특정 구성요소를 인식하여 추출하는 작업이다. 기존의 정보추출 시스템은 대부분 정보추출 규칙인 wrapper를 수동으로 구성하여 적용하였기 때문에 추출의 정확성은 높지만 유연성, 확장성, 효율성의 측면에서 문제점이 발생하였다. Wrapper를 자동으로 생성하는 일부 연구에서도 도메인 지식의 획득과 표현의 어려움, 그리고 여러 정보소스 사이에 나타나는 문서형태의 구조적 이질성 때문에 정확한 정보추출이 이루어지지 못했다.

본 논문에서는 이러한 이질적이고 복잡한 형태의 실세계 정보소스로부터의 정확한 정보추출을 추구하는 정보추출 에이전트인 XTROS를 제안한다. XTROS는 도메인 지식을 이용하여 준구조화된 형태의 정보소스에서 제공하는 문서를 분석하고 학습하여 wrapper들을 자동으로 생성하고, 이 wrapper들을 이용하여 정보추출과 정보통합을 수행한다. 본 논문에서는 특별히 도메인 지식과 wrapper를 모두 XML 문서의 형태로 구성하는 새로운 표현기법을 제시함으로써 도메인 지식표현의 용이성과 wrapper 해석기 구현의 간결함, XML이 지닌 이식성 등을 최대한 활용하고자 하였다. Wrapper의 정보추출 규칙은 도메인 지식과 샘플 문서를 이용하여 자동으로 생성된다. 정보추출 규칙을 자동으로 생성하는 알고리즘의 핵심은 도메인 지식을 바탕으로 샘플 문서의 각 논리 라인에 의미를 부여하고 이 논리 라인 의미의 나열로부터 반복되는 패턴을 찾아내는 것이다. 이 패턴의 위치와 구조를 XML 문서로 표현한 것이 wrapper가 된다. XTROS 시스템을 부동산 매물정보를 제공하는 다수의 실제 웹 정보소스에 대해서 테스트한 결과 이질성과 복잡성을 가진 대부분의 정보소스로부터 정확한 wrapper 생성과 정보추출이 가능하였다.

키워드 : 정보추출, 랩퍼인덕션, 지식기반 학습

Abstract Information extraction(IE) is a process of recognizing and fetching particular information fragments from a document. In previous work, most IE systems generate the extraction rules called the wrappers manually, and although this manual wrapper generation may achieve more correct extraction, it reveals some problems in flexibility, extensibility, and efficiency. Some other researches that employ automatic ways of generating wrappers are also experiencing difficulties in acquiring and representing useful domain knowledge and in coping with the structural heterogeneity among different information sources, and as a result, the real-world information sources with complex document structures could not be correctly analyzed.

In order to resolve these problems, this paper presents an agent-based information extraction system named XTROS that exploits the domain knowledge to learn from documents in a semi-structured information source. This system generates a wrapper for each information source automatically and performs information extraction and information integration by applying this wrapper to the corresponding source. In XTROS, both the domain knowledge and the wrapper are

· 이 논문은 2000년 한양대학교 교내연구비 지원으로 연구 되었음.
† 정 회 원 : 삼성종합기술원 HCI Lab 연구원
hkseo@sait.samsung.co.kr
†† 학생회원 : 한양대학교 컴퓨터공학과
jyyang@csc.hanyang.ac.kr
††† 종신회원 : 한양대학교 컴퓨터공학과 교수
jmchoi@csc.hanyang.ac.kr
논문접수 : 2001년 1월 11일
심사완료 : 2001년 10월 15일

represented as XML-type documents. The wrapper generation algorithm first recognizes the meaning of each logical line of a sample document by using the domain knowledge, and then finds the most frequent pattern from the sequence of semantic representations of the logical lines. Eventually, the location and the structure of this pattern represented by an XML document becomes the wrapper. By testing XTROS on several real-estate information sites, we claim that it creates the correct wrappers for most Web sources and consequently facilitates effective information extraction and integration for heterogeneous and complex information sources.

Key words : Information Extraction, Wrapper Induction, Knowledge-based Learning

1. 서 론

정보추출은 한 문서에서 그 문서의 중심적 의미를 나타내는 특정 구성요소를 인식하여 추출하는 작업을 가리킨다[1,2]. 정보추출의 예로는 날씨정보를 제공하는 웹 문서로부터 지역, 날짜, 최고온도, 최저온도, 습도 등의 정보를 골라내거나, 또는 아파트 정보 문서로부터 방의 개수, 매매가, 전세가, 전화번호 등을 추출하는 것을 들 수 있다.

정보추출의 대상이 되는 정보소스에 존재하는 문서들은 크게 비구조화(unstructured) 문서, 구조화(structured) 문서, 그리고 준구조화(semi-structured) 문서의 세 가지 형태로 나누어진다. 비구조화 문서는 신문기사와 같이 일반 자연어 문장으로만 구성되어 있고 어떤 일정한 형식이 없는 텍스트 문서를 말한다. 비구조화 문서로부터의 정보추출은 구문분석과 같은 자연어처리를 필요로 하기 때문에 매우 어렵다. 구조화 문서는 실제 데이타와 그 데이타가 나타내는 의미를 메타 데이타를 통해 명시적으로 표현하는 문서 형태이다. 테이블이 하나의 예가 될 수 있으며, 이때 테이블 헤더는 그 열이 나타내는 의미를 기술하므로 메타 데이타라고 할 수 있다.

준구조화 문서는 구조화되어 있는 부분과 구조화되어 있지 않은 부분이 한 문서 내에 혼합되어 있는 형태를 말하는데, 예를 들면 이메일 문서(제목,수신자, 날짜 등은 구조화, 메일 내용은 비구조화), 검색엔진의 검색결과 페이지, 온라인 쇼핑몰에서 상품검색의 결과 문서, 그리고 본 논문에서 초점을 맞춘 부동산 매물 검색 결과를 출력한 문서(<그림 1>)들이 준구조화 문서 구조를 가지고 있다. 웹에서의 대부분의 준구조화 문서는 질의를 만족시키는 데이타베이스의 일부 내용을 일정한 출력형식으로 표현한 형태를 지닌다. <그림 1>에서 보면 주택 매물에 대한 기술은 "Beds:1, Baths:1"과 같이 1이라는 데이타와 그것의 의미(이 예제의 경우 Beds는 침실의 수, Baths는 욕실의 수를 가리킴)가 같이 나타나므로 구조화된 형태이고, 기타 문서의 첫머리에 나타나는 헤더 정보나 오른쪽의 광고와 같은 정보는 비구조화 형태이다. 여기서 Beds나 Baths와 같이 데이타의 의미를 기술하는 메타 데이타를 라벨(label)이라고 하며, <그림 1>의 문서와 같이 라벨과 데이타 값이 쌍으로 구성되어 있는 문서를 "라벨 문서"(labelled document)라고 한다[3]. 본 논문에서는 이러한 웹 정보소스의 준구조화 문서, 특히 라벨 문서를 대상으로 한다. 즉 준구조화 문서에서 해더 정보나 광고와 같은 비구조화 정보를 별도로 판별해 내지 않고, 사용자에게 필요한 정보를 포함하고 있는 구조화된 부분의 정보를 추출하도록 하였다.

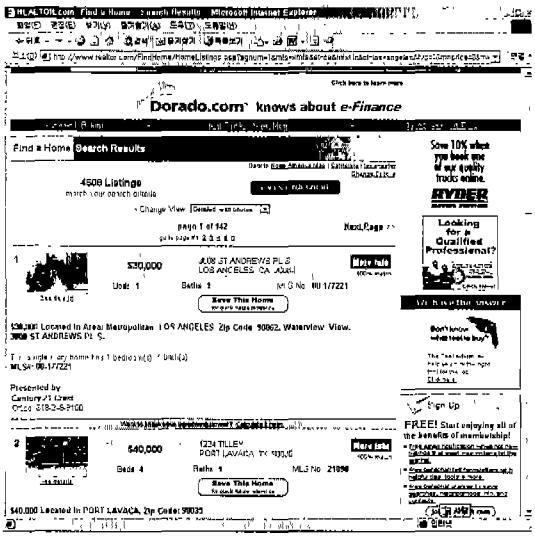

그림 1 부동산 매물검색 예

에이전트를 이용한 정보추출을 위해서는 각 문서에 대해서 추출하고자 하는 정보의 위치와 구조, 포맷 등을 나타내는 규칙이 필요하며 일반적으로 이러한 규칙을 *wrapper*라고 한다[4,5]. 기존의 정보추출 시스템은 대부분 wrapper를 수동으로 구성하여 적용하였기 때문에 추출의 정확성은 높일 수 있었지만 새로운 문서나 변경된 문서를 처리하기 위해서는 wrapper를 다시 수동으로 생성해야 하는 등 유연성, 확장성, 효율성의 측면에

서 문제점이 발생하였다. Wrapper를 자동으로 생성하는 일부 연구에서도 도메인 지식의 획득과 표현의 어려움, 그리고 여러 정보소스 사이에 나타나는 문서형태의 구조적 이질성 때문에 실세계에서 접하게 되는 복잡한 형태의 정보소스로부터 정확한 정보추출이 이루어지지 못했다. 또한 각 문서의 학습 시에 매번 사용자로부터 입력을 받아야만 한다는 단점을 가지고 있다.

본 논문에서는 이러한 문제점을 극복하여 이질적이고 복잡한 정보소스로부터의 정확한 정보추출을 위해 wrapper를 자동으로 생성하고 사용자의 번거로움을 줄인 정보추출 에이전트 XTROS를 제안한다. XTROS는 도메인 지식을 이용하여 라벨을 지닌 준구조화된 형태의 정보소스에서 제공하는 문서를 분석하고 학습하여 wrapper를 자동으로 생성하고, 이 wrapper를 이용하여 정보추출과 정보통합을 수행하는 시스템이다. 이 시스템은 최소한의 도메인 지식만을 사용자로부터 입력받아 문서의 정보추출 규칙을 자동으로 학습한다. 도메인 지식은 문서에서 추출해야 할 정보의 라벨과 위치를 파악하는데 이용된다.

Wrapper의 정보추출 규칙은 도메인 지식과 샘플 문서를 이용하여 자동으로 생성된다. 정보추출 규칙을 자동으로 생성하는 알고리즘의 핵심은 도메인 지식을 바탕으로 샘플 문서의 각 논리 라인에 의미를 부여하여 문서를 논리 라인의 의미표현의 나열로 표현하고, 이것으로부터 반복되는 패턴을 찾아내는 것이다. 이 패턴의 위치와 구조를 XML 문서로 표현한 것이 wrapper가 된다. 이 알고리즘은 이미 비교쇼핑 도메인에 적용하여 성능을 평가받은 바 있다.[6]

본 논문에서는 이 패턴 검색 알고리즘보다는 도메인 지식을 표현하고 적용하여 wrapper를 생성하고 정보추출을 수행하는 기법에 더 중점을 두어 기술하고자 한다. XTROS 시스템에서는 도메인 지식과 wrapper를 모두 XML 문서의 형태로 구성하는 새로운 표현기법을 제시함으로써 wrapper 규칙의 정형화된 표현을 가능하게 하였고, wrapper 해석기를 XML 해석기로 대치함으로써 구현의 간결화도 도모하였다. XML을 이용한 도메인 지식 표현 방법은 도메인이 변경되더라도 쉽게 새로운 도메인 지식을 작성할 수 있는 기반을 조성한다. 또한 도메인 지식을 이용하여 생성된 wrapper도 XML로 구성함으로써 표현의 통일성과 구현의 용이함, 그리고 XML에 내재된 이식성(portability)의 특성을 최대한 활용하여 하드웨어나 OS 플랫폼에 독립적인 정보추출 시스템의 개발에도 기여하고자 하였다. 이러한 시스템의 특성은 실험결과에서 나타나고 있는데, XTROS를 부동산 매물정보를 제공하는 다수의 실제 웹 정보소스에 대해서 테스트한 결과 이질성과 복잡성을 가진 정보소스로부터 비교적 정확한 정보추출이 가능하였다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 제안하는 XTROS 시스템의 역할과 구조를 살펴본다. 3장에서는 학습에 사용하는 도메인 지식의 표현방법을 설명한다. 4장에서는 도메인 지식을 바탕으로 학습을 통해 wrapper를 자동으로 생성하는 알고리즘을 살펴본다. 5장에서는 부동산 매물정보 사이트에 시스템을 적용한 실험 결과를 기술한다. 6장에서는 웹 문서에 대한 정보추출을 다루는 기존의 여러 연구들과의 비교를 기술한다. 7장에서는 결론 및 향후 연구과제 등을 살펴본다.

2. XTROS: Wrapper 자동생성 에이전트

XTROS는 웹에 존재하는 여러 정보소스로부터 필요한 정보를 수집하여 사용자에게 제공하는 에이전트이다. 문제는 수많은 정보소스들이 이질적(heterogeneous)이어서 각자 저마다의 방법으로 정보를 저장하고 출력한다는 것이다. Wrapper는 이러한 이질적 정보소스에 접근하여 그 정보를 이해하고 필요한 요소를 추출할 수 있도록 정보의 위치와 구조, 포맷 등을 기술하는 규칙 또는 이러한 규칙을 적용하여 다른 에이전트가 요구하는 정보 형태로 변환할 수 있는 능력을 가진 프로그램이라고 정의할 수 있다.[7] 따라서 하나의 정보소스에 대해 그 정보소스만을 이해할 수 있는 하나의 wrapper가 생성된다.

문서로부터 자동으로 wrapper를 생성하여 필요한 정보만을 추출하고, 이것을 사용자에게 제공하기 위한 에이전트의 구조는 <그림 2>와 같다.

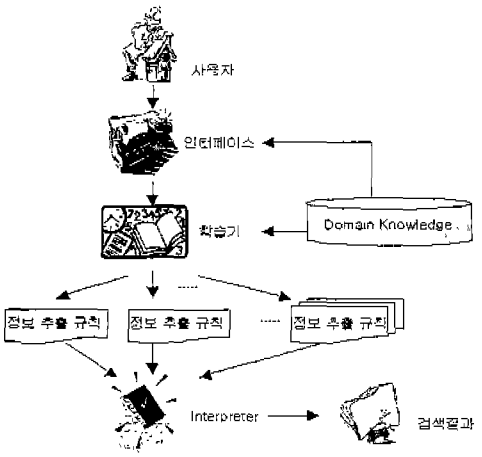

그림 2 Wrapper 자동생성 에이전트 구조

"인터페이스"는 학습한 정보소스에 접근하기 위한 사용자 질의 입력을 받고, 그 질의에 따른 결과 문서를 가져온다. 질의문은 도메인 지식의 input form부분에 사용자 입력의 추가를 통해 생성된다. 본 논문에서는 학습하고자 하는 문서의 도메인이 미국에 산재한 부동산이기 때문에 도시(city), 주(state), 가격(price) 등의 입력을 받고 검색사이트에 결과를 요청한다. "학습기"(Learner)는 인터페이스로부터 검색 결과를 받아서 지식 기반의 학습을 수행하고 wrapper를 생성한다. 즉, 도메인 지식을 이용해서 해당 문서의 반복되는 정보 패턴을 학습하여 XML 기반의 wrapper를 생성한다. "해석기"(Interpreter)는 생성된 여러 wrapper를 이용해서 각 정보소스의 URL에 질의를 요청하고, 질의결과 문서에서 정보를 추출하고, 추출된 여러 문서의 정보들을 통합해서 사용자에게 제공한다.

3. 도메인 지식의 표현

도메인 지식은 특정 응용 도메인에 국한된 지식을 표현한 것으로 어휘적(lexical) 분석만으로는 파악하기 어려운 문서의 의미를 보다 정확히 분석하기 위해 필수적인 요소가 된다. 본 논문에서 새롭게 제시하는 XML 기반의 도메인 지식의 형태는 <그림 3>과 같다.

하나의 도메인에 대한 지식은 <KNOWLEDGE>... </KNOWLEDGE> 안에 표현되며, 그 도메인에서 많이 사용되는 어휘나 사용 규칙 등이 기술된다. <OBJECTS>는 여러 OBJECT의 리스트를 나타내는데, OBJECT는 정보추출을 위해 학습의 대상이 되는 속성(attribute)이나 구성요소(component)를 가리킨다. 부동산 매물정보 사이트에 대한 OBJECT에는 PRICE(부동산 가격), BED(침실수), BATH(욕실수), CITY(소재지 도시), MLS(매물 고유번호), DETAIL(상세정보 링크) 등이 존재하며 <OBJECTS>안에 <OBJECT> 태그로 표현되어 있다. <OBJECT>..</OBJECT> 안에 기술된 각 OBJECT에 대해서는 하나의 XML 구조체가 구성된다. 예를 들어, PRICE에 대해 <PRICE>..</PRICE>의 구조가 존재하고, BED에 대해 <BED>..</BED>의 구조가 존재한다.

각 OBJECT에 대한 XML 구조는 <ONTOLOGY>와 <FORMAT>의 두 가지 요소로 구성되며 각 요소가 의미하는 내용과 표현 방법이 <그림 4>에 나타나 있다. 여기서 <ONTOLOGY>는 학습해야 하는 정보들이 존재하는지 판별하기 위한 데이타, 즉 온톨로지를 표현하고, <FORMAT>은 그 속성이 가질 수 있는 데이타 타입과 또한 ONTOLOGY와 데이타와의 관계를 표현한다. 예를 들어 <그림 3>에서 PRICE의 경우, <ONTOLOGY>에 "PRICE"와 "$"가 명시되었는데, 이것은 가격이 존재하는지 판별하기 위해서는 "PRICE"라는 스트링이나 "$"의 심볼이 포함되어 있는지를 검사하면 된다는 의미이다. 또한 <FORMAT>을 살펴보면 가격에 대한 데이타 형태는 DIGITS(0~9까지 숫자들의 나열)가 존재해야 하고 이 DIGITS가 온톨로지 바로 다음에 나타나거나([ONTOLOGY] DIGITS), 온톨로지 바로 이전에 나타나야 한다(DIGITS [ONTOLOGY])는 것을 알 수 있다.

```
<KNOWLEDGE>
    <OBJECTS>
        <OBJECT> PRICE </OBJECT>
        <OBJECT> BED </OBJECT>
        <OBJECT> BATH </OBJECT>
        <OBJECT> CITY </OBJECT>
        <OBJECT> MLS </OBJECT>
        <OBJECT> DETAIL </OBJECT>
    </OBJECTS>

    <PRICE>
        <ONTOLOGY>
            <TERM> PRICE </TERM>
            <TERM> $ </TERM>
        </ONTOLOGY>
        <FORMAT>
            <FORM> [ONTOLOGY] DIGITS </FORM>
            <FORM> DIGITS [ONTOLOGY] </FORM>
        </FORMAT>
    </PRICE>

    <BED>
        <ONTOLOGY>
            <TERM> BEDROOM </TERM>
            <TERM> BED </TERM>
            <TERM> BEDS: </TERM>
            <TERM> BR : </TERM>
            <TERM> BEDS </TERM>
            <TERM> BEDROOMS </TERM>
        </ONTOLOGY>
        <FORMAT>
            <FORM> [ONTOLOGY] DIGITS </FORM>
            <FORM> DIGITS [ONTOLOGY] </FORM>
        </FORMAT>
    </BED>
    .....
</KNOWLEDGE>
```

그림 3 도메인 지식의 표현

```
<OBJECT>
    <ONTOLOGY> [OBJECT가 존재하는지 판별할 데이타]
    </ONTOLOGY>
    <FORMAT> [OBJECT가 맞는지 판별할 데이타 타입과
              ONTOLOGY와 데이터의 관련성]
              </FORMAT>
</OBJECT>
```

그림 4 도메인 지식의 OBJECT 표현

4. 정보추출 규칙 자동생성 알고리즘

4.1 논리 라인 생성

웹 문서는 대부분 HTML을 이용하여 작성되는데, 이 HTML 문서에서 사용자에게 의미 있는 정보는 일반 텍스트나 하이퍼링크, 이미지 등이고 나머지 HTML 태그들은 단지 브라우저에 출력하기 위한 규약을 나타낸다. 이 중 정보추출의 대상이 되는 부분은 일반 텍스트 문장이며 추출을 쉽게 하기 위해 불필요한 HTML 태그들은 학습하기 전에 제거해주는 전처리 작업이 필요하다. 이러한 전처리 작업 후의 문서의 각 라인을 논리 라인(logical line)이라고 한다. 논리 라인은 브라우저 상에 출력되는 형태처럼 눈에 보이지 않는 HTML 태그를 제거하고 테이블 관련 태그(예를 들어, TR, TH 등)나 라인을 분리할 때 사용되는 리스트형 태그(예를 들어, BR, P, LI)를 기준으로 라인을 분리한 것이다. <그림 5>는 부동산 정보사이트 중 하나인 HOMES의 검색결과 화면에 대한 전처리 전의 HTML 소스의 일부분과 전처리 후의 논리 라인을 나타내는 HTML 소스를 비교해 주고 있다.

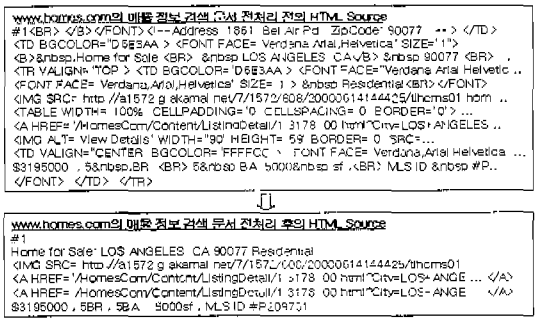

그림 5 논리 라인의 생성

논리 라인으로 변환한 HTML 문서에 대해 학습하는 것이 유리한 이유는 다음과 같다. 첫째, 논리 라인 변환 전의 문서에 비해 변환 후의 문서가 훨씬 간결하다. 둘째, 필요 없는 정보가 제거되었기 때문에 각 라인이 가지는 중요성이 높아진다. 셋째, 추출해야 할 정보가 존재할 가능성이 있는 라인만을 대상으로 학습하므로 학습이 더 효율적이다. 넷째, 문서마다 눈에 보이는 것과 실제 HTML 소스에 기술된 것이 서로 다른 표현 형식을 가지는 것 때문에 발생할 수 있는 학습 시 혼란을 사전에 방지할 수 있다.

4.2 도메인 지식을 이용한 논리 라인의 의미 분석

논리 라인에 대한 의미 분석은 XML로 정의된 도메인 지식에 포함된 OBJECT 중에 어떤 정보가 논리 라인에 존재하는지를 판별하는 작업이다. 이를 위해 각 OBJECT에 대한 <ONTOLOGY>와 <FORMAT> 정의를 이용한다. 하나의 논리 라인 *LL*에 어떤 OBJECT가 존재하는지, 즉 *LL*의 의미가 무엇인지 판별하기 위해 도메인 지식을 사용하는 알고리즘을 <그림 6>에 pseudo code로 나타내었다.

```
For (each <OBJECT> OBJ)                                          ①
    If LL includes a term in <ONTOLOGY> of OBJ                   ②
        If LL conforms to a <FORM> in <FORMAT> of OBJ            ③
            If LL includes other <OBJECT> of OBJ2                ④
                Separate LL into two lines L1 and L2             ⑤
                Recognize L1 as OBJ, and L2 as OBJ2              ⑥
            Else                                                 ⑦
                Recognize LL as OBJ                              ⑧
```

그림 6 논리 라인의 의미 분석 알고리즘

이 알고리즘은 우선 하나의 논리 라인 *LL*에 대하여 XML로 표현된 도메인 지식의 각 OBJECT에 대해 차례로 그 OBJECT의 <ONTOLOGY>에 정의된 term이 *LL*에 포함되어 있는지 확인한다(①②번). 한 OBJECT의 ONTOLOGY 중 여러 개가 한 라인에 존재하는 경우에는 가장 긴 ONTOLOGY를 선택하는데, 이것은 짧은 ONTOLOGY가 우연히 논리 라인에 나타날 확률이 그보다 긴 ONTOLOGY가 논리 라인에 나타날 확률보다 크며 따라서 긴 ONTOLOGY가 오류가 발생할 확률이 작기 때문이다.

ONTOLOGY 중 하나가 존재하면, 그 OBJECT의 <FORMAT>과 일치하는 가를 판별한다(③번). FORMAT의 형태는 ([ONTOLOGY] data-type) 또는 (data-type [ONTOLOGY])의 두 종류가 있다. Data type은 학습할 데이타의 종류에 따라서 다를 수 있는데, 현재 부동산 매물정보 사이트에 대한 학습의 경우 DIGITS(연속된 십진 숫자들), USER_CITY_QUERY (사용자가 입력한 도시명), 그리고 URL(하이퍼링크된 주소)의 data type이 존재한다.

논리 라인에 어떤 OBJECT에 대한 ONTOLOGY가 존재하고 FORMAT과 일치하면 그 논리 라인은 그 OBJECT를 나타낸다고 의미 분석을 할 수 있다(⑦⑧번). 만일 그 논리 라인에 다른 OBJECT가 함께 존재할 경우 그 논리 라인을 두개의 라인으로 분리하고 각 라인에 해당하는 OBJECT를 위와 같은 방법으로 분석하여 두 개의 의미를 부여한다. 그런 다음 논리 라인이 해당 OBJECT를 의미하고 있다는 것을 표시한다(④⑤⑥번).

논리 라인에 대한 의미 분석의 결과는 <그림 7>과 같은 데이타 구조로 표현된다. 하나의 논리 라인은 pattern과 oneLine의 두 개의 스트링으로 표현된다. pattern은 그 라인이 포함하고 있는 정보의 의미가 무엇인지를 표현하고, oneLine은 pattern에 표현된 정보가 있던 원래 논리 라인에 ONTOLOGY와 data type 정보를 구분해서 표현한다. patternExp은 그 논리 라인의 의미를 미리 정의된 숫자 카테고리로 표현한다. 부동산 매물 정보 검색의 경우 <그림 8>과 같이 각 의미 카테고리에 따라 번호를 부여하였다. type은 데이타 타입을 표시하고, format은 사용된 도메인 지식의 FORMAT 원형을 나타낸다.

```
String pattern : · {[OBJECT_NAME]} ·
String oneLine . · {[· [ONTOLOGY]· ]}{{OBJECT_NAME}}
·
int patternExp . 0, 1, 2, ·
String type : DIGITS, STRING, URL, ·
String format : DIGITS [ONTOLOGY], [ONTOLOGY] DIGITS
```

그림 7 논리 라인을 표현하기 위한 데이타 구조

```
0 : 가 격 (PRICE)
1 . 침 실 (BED)
2 · 욕 실 (BATH)
3 · 주 소 (ADDRESS)
4 MLS ID NUMBER
5 . 상 세 페 이 지 (DETAIL)
6 : 매 물 이 미 지 (IMAGE)
9 · 기 타 텍 스 트 (TEXT)
```

그림 8 OBJECT에 할당된 의미 번호

예를 들어, 논리 라인이 "$48,000 10813 MONA BL LOS ANGELES, CA 90059"와 같은 경우는 "PRICE"로 판별되며 의미 분석 결과는 <그림 9>와 같이 표현된다.

```
pattern · {[PRICE}] 10813 MONA BL LOS ANGELES, CA
90059
oneLine : {[[{$}]48,000]}{{PRICE}} 10813 MONA BL LOS
ANGELES, CA 90059
patternExp : 0
type : DIGITS
format : [ONTOLOGY] DIGITS
```

그림 9 논리 라인 표현 예제 : PRICE로 인식된 경우

학습된 각 논리 라인이 <그림 7>의 구조로 표시된 후 patternExp에 나타낸 각 논리 라인의 의미 정보를 바탕으로 반복되는 패턴을 생성한다. 페이지에 존재하는 반복되는 패턴을 찾아내는 문제는 검색(search)문제로 귀결된다. 본 연구에서 패턴을 찾기 위한 검색 공간(search space)은 패턴 데이타가 포함되지 않은 헤더 정보(H)와 테일 정보(T), 그리고 실제 찾아야 하는 패턴 정보(P)로 구성된다. 따라서 검색 공간 $P = \langle H, P_1, \ldots, P_n, T \rangle$로 표현할 수 있다. 찾고자 하는 패턴 P는 실제 찾고자 하는 데이타의 집합으로 $P = \langle ATTR_1, \ldots, ATTR_m \rangle$과 같이 구성된다. 페이지에서 패턴을 찾아내기 위해 Bottom up 방식을 이용하여 패턴의 시작 속성과 끝 속성을 파악한다. 반복되는 패턴을 찾아내는 알고리즘은 이 논문의 초점은 아니며, 이에 대한 자세한 내용은 [6,8]에 기술되어 있다.

4.3 XML 규칙 생성

논리 라인의 의미 분석과 반복되는 패턴 검색이 수행된 후 찾아진 패턴을 이용하여 XML 기반의 정보추출 규칙을 만든다. 이러한 규칙은 하나의 정보 사이트에 대해 하나씩 생성되어 사용자 질의의 결과로 생성되는 페이지에서 원하는 정보만 추출할 때 이용된다. 부동산 매물 사이트에 대한 XML 기반의 정보추출 규칙의 형태는 <그림 10>과 같다.

```
<Start>
    <Form>
        Query 관련 규칙
    </Form>
    <Home>
        <Price>
            <Ontology> $ </Ontology>
            <Ident> NULL </Ident>
            <Format> Digits </Format>
            <Operation> Ontology*Format </Operation>
        </Price>
        <Bed>
            <Ontology> BR </Ontology>
            <Ident> NULL </Ident>
            <Format> Digits </Format>
            <Operation> Format*Ontology </Operation>
        </Bed>
        ..
    </Home>
</Start>
```

그림 10 XML 기반의 정보추출 규칙

여기서 <Form>...</Form> 부분은 샘플 키워드를 사용자 질의어처럼 입력해 질의결과 페이지를 가져오기 위한 입력 폼에 대한 template을 나타내는 규칙인데, 본 시스템에서는 이 부분은 수동으로 생성하기 때문에 자세한 설명은 생략한다.

특정 사이트의 질의 결과 페이지로부터 원하는 정보를 추출하기 위한 규칙은 <Home>...</Home>에 표현된다. 각 OBJECT의 <Ontology>...</Ontology> 내에 있는 term이 그 OBJECT를 추출하기 위해 맨 먼저 검

색하는 정보이고, <Format>은 추출할 정보의 type을 나타낸다. <Ident>는 다른 OBJECT와 구별시키는 구분기호(delimiter)를 표시하는데, 이러한 구분기호가 없는 경우 <Ident>는 NULL이 된다. <Operation>은 추출해야 하는 정보의 위치를 나타낸다. 즉, 정보가 Ontology의 앞에 존재하는지 혹은 뒤에 존재하는지, Ontology와 Ident 사이에 존재하는지 등을 나타낸다.

XML 기반의 정보추출 규칙 생성은 앞 절의 알고리즘을 이용해 구한 논리 라인의 패턴(각 논리 라인은 <그림 7>의 데이타 구조로 표현됨)으로부터 <그림 10>과 같은 규칙을 자동으로 만드는 것이며 이 생성과정은 <그림 11>에 나타나있다.

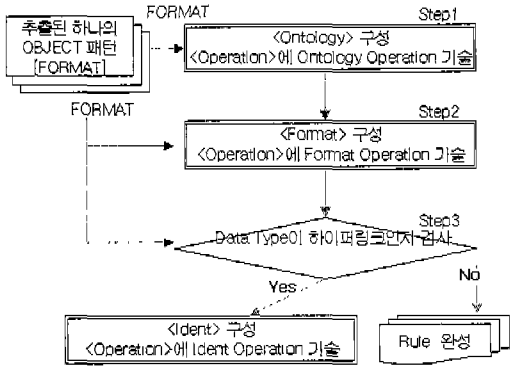

그림 11 XML 정보추출 규칙 생성 과정

이 과정을 살펴보면 먼저 추출된 하나의 OBJECT 패턴의 FORMAT을 참조하여 규칙의 <Format>과 <Operation>을 생성한다. 다음으로 oneLine의 {[... [ONTOLOGY] ...]}부분을 이용해 규칙의 <Ontology>를 생성한다. Data type이 하이퍼링크(URL)인 경우는 일반 텍스트 정보와 구분하기 위해 <Ident>를 "A HREF"로 지정하여 정보추출에 이용하고, URL이 아닌 다른 type의 경우 <Ident>는 NULL로 지정한다. 모든 OBJECT에 대해서 이러한 작업을 수행하면 XML 기반의 규칙이 생성된다. 예를 들어서 <그림 12>와 같이 부

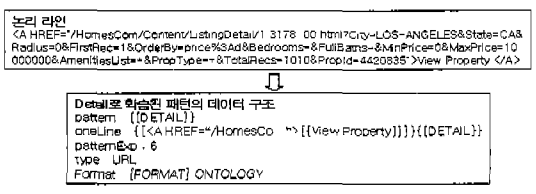

그림 12 DETAIL로 판별된 논리라인 예제의 데이타 구조

동산 매물의 상세 페이지를 기술하는 "DETAIL" OBJECT의 논리 라인 정보로부터 생성된 정보추출 규칙은 <그림 13>과 같다.

```
<Detail>
        <Ontology> View Property </Ontology>
        <Ident> A HREF </Ident>
        <Format> URL </Format>
        <Operation> Ident*Format*Ontology
        </Operation>
</Detail>
```

그림 13 DETAIL에 대한 정보추출 규칙

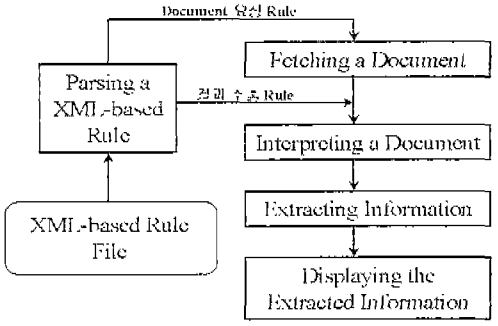

그림 14 Wrapper를 적용한 정보추출 과정

생성된 XML 정보추출 규칙은 한 사이트의 wrapper가 된다. XTROS는 이 wrapper를 이용해서 사용자로부터 도시, 주, 가격 등의 입력을 받고 XML 규칙의 <Form>...</Form>부분의 질의 규칙에 따라 질의문에 추가하여 결과를 요청한다. 그리고 나서 이에 대한 결과 문서들에 대해 wrapper의 <Home>...</Home>부분의 규칙을 적용하여 필요한 정보를 추출해 낸다. 추출된 정보들을 통합해서 사용자에게 제공하면 사용자는 하나의 인터페이스에서 여러 문서의 정보를 비교할 수 있다. XML wrapper를 이용한 정보추출 과정은 <그림 14>에 도식화되어 있다.

5. 구현과 성능 분석

XTROS는 Java를 이용하여 구현되었으며, 2장의 시스템 구조에서 소개한 대로 인터페이스, 학습기, 해석기의 3 모듈에 대해서 구현사항을 설명하고자 한다.

XTROS의 인터페이스는 <그림 15>와 같이 구성되어 있으며, 사용자가 원하는 매물의 세부사항(주, 도시, 가격범위, 침실수, 욕실수 등)을 입력하고, 학습된 사이트의 리스트에서 일부만 선택하여 정보를 얻을 수 있다.

인터페이스 모듈의 기능은 사용자 세부요청 사항을 질의 template에 추가하여 해당 사이트에 요청을 하고 검색결과로 얻어진 문서를 다음 단계에서 필요로 하는 벡터로 변환하는 것이다.

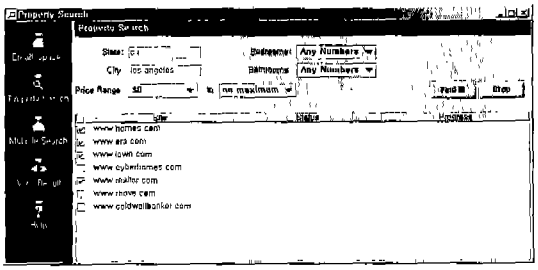

그림 15 XTROS의 인터페이스

인터페이스 모듈은 다음과 같은 4개의 Java 패키지로 구성되어 있다.

- DKInputParser.java: XML 기반의 wrapper에서 <FORM>..</FORM> 부분을 파싱하여 질의문 template을 생성한다.
- DomParser.java: DKInputParser.java에서 호출되면 <FORM>..</FORM>을 파싱한다.
- ConstructQueryForm.java: 생성된 질의문 template에 사용자가 지정한 내용을 추가한다.
- Robot.java: 구성된 질의를 해당 사이트에 보내어 결과를 요청한다.

XTROS의 학습기 모듈은 부동산 매물 결과의 샘플 페이지와 도메인 지식을 바탕으로 wrapper를 생성하는

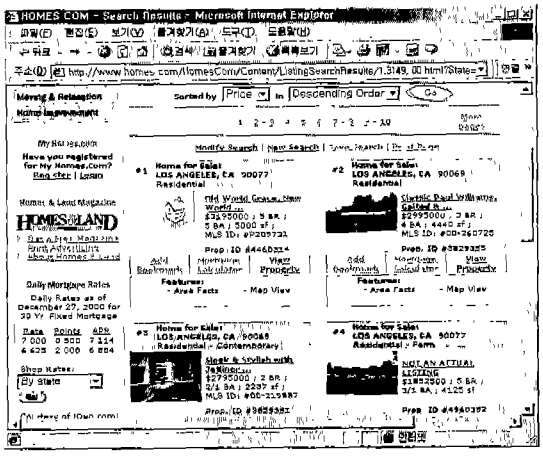

그림 16 HOMES 사이트의 검색결과 샘플 페이지

기능을 수행하며, <그림 16>의 HOMES 사이트(www.homes.com)에 대한 샘플 페이지로부터 <그림 17>과 같은 wrapper가 생성된다.

```
<Start>
<Form>
  <RobotOperation> 1 </RobotOperation>
  <URL>
www.homes.com/Content/ListingSearchResults.cfm
</URL>
  <Method> GET </Method>
  <InputNumber> 9 </InputNumber>
  <Input>
      <Input1>
          <Semantic> Dissatisfied </Semantic>
          <Ontology> STATE_INPUT </Ontology>
          <Name> State </Name>
          <Value> NULL </Value>
      </Input1>
      --- 중략 ------
  </Input>
</Form>
<Home>
  <Price>
      <Ontology> $ </Ontology>
      <Ident> NULL </Ident>
      <Format> Digits </Format>
      <Operation> Ontology*Format </Operation>
  </Price>
  <Detail>
      <Ontology> MoreInfo </Ontology>
      <Ident> A HREF </Ident>
      <Format> URL </Format>
      <Operation> Ident*Format*Ontology </Operation>
  </Detail>
  <Bed>
      <Ontology> Beds: </Ontology>
      <Ident> NULL </Ident>
      <Format> Digits </Format>
      <Operation> Ontology*Format </Operation>
  </Bed>
  --- 중략 ------
</Home>
</Start>
```

그림 17 HOMES 사이트에 대해 생성된 XML wrapper

학습기 모듈은 다음과 같은 5개의 Java 패키지로 구성되어 있다.

- devideintoLogical.java: HTML 문서를 논리라인으로 변환한다.
- DKParser.java: XML 기반의 도메인 지식을 파싱한다.
- LogicalLineAnalyzer.java: 도메인 지식을 이용하여 논리라인의 의미를 파악한다.
- RecognizePattern.java: 의미가 파악된 논리라인에서 반복되는 패턴을 인식한다.
- makeRuleFile.java: 패턴에 대한 XML 기반의 wrapper를 생성한다.

XTROS의 "해석기" 모듈은 wrapper와 실제 부동산 매물결과를 대상으로 가격, 침실수 등의 값을 추출하는 기능을 수행하며, 다음과 같이 3개의 Java 패키지로 구성되어 있다.

- RuleParser.java: XML 기반의 wrapper를 파싱한다.
- DocumentInterpreter.java: 파싱된 wrapper를 이용하여 문서에서 필요한 정보의 위치를 파악하고 추출한다.
- ExtractInfoStructure.java: 추출된 정보를 저장한다.

HOMES의 wrapper를 이용해서 실제 매물검색을 수행한 결과는 <그림 18>과 같다. 이 결과는 <그림 16>의 화면에서 가격, 침실수, 욕실수, MLS 번호만을 추출한 것과 동일하다.

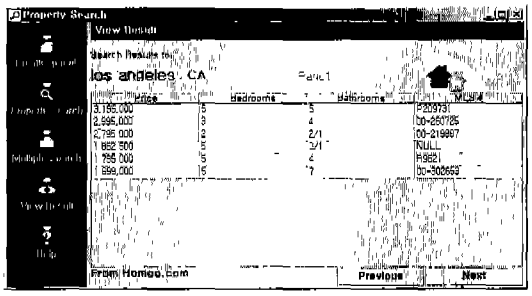

그림 18 HOMES에 대한 해석기 수행 후 정보추출한 결과

이와 같이 XTROS 시스템을 이용하면 각 사이트마다 별개의 검색을 통해서 확인할 수 있었던 여러 문서의 정보를 한번의 질의로 하나의 인터페이스를 통해서 볼 수 있다.

XTROS의 성능 평가를 위하여 우선 미국의 주택, 건물을 대상으로 하는 10개의 부동산 매물 검색 사이트에 적용하였다.(<표 1>)

표 1 학습에 적용한 부동산 사이트

테이블형 결과 제공 사이트
http://www.homes.com http://www.realtor.com http://realstate.yahoo.com http://www.iown.com http://www.cyberhomes.com http://www.listinglink.com
리스트형 결과 제공 사이트
http://www.move com http://www.era.com http://www.coldwellbanker.com http://www.homeseekers.com

10개의 사이트 중 4개의 사이트가 리스트형으로 매물정보를 출력하고, 6개의 사이트가 테이블형으로 데이타를 제공한다. (여기서 리스트형은 테이블 형태가 아닌 모든 준구조화된 데이타 나열 형태를 가리킨다.) 이들 중 학습이 불가능한 사이트는 2개의 사이트(realstate.yahoo.com와 www.listinglink.com)인데, 이들 사이트는 검색결과가 완전한 테이블 형태로 표현되며, 이 경우 각 논리 라인에서 데이타만 존재하고 라벨이 존재하지 않기 때문에 논문에서 제시하는 학습 알고리즘을 적용할 수 없었다. 학습이 가능했던 8개의 사이트에 대해서는 필요한 정보를 추출하는 검색률이 100%에 달하였다. 이것의 의미는 만일 학습이 가능한 사이트의 문서에 매물 정보가 10개가 존재할 경우 XTROS가 10개의 매물에 대한 정보를 모두 추출해 온다는 것을 나타낸다. 본 논문에서는 10개의 적은 수의 사이트에 대한 wrapper 학습을 적용하였지만, 좀 더 복잡하고 이질적인 사이트에 대한 학습 능력을 향상시키기 위하여 현재 수천 개에 달하는 미국의 부동산 관련 사이트 중 100개 정도에 대한 성능 평가를 진행 중에 있다.

6. Wrapper 생성에 대한 관련연구

Wrapper 생성에 대한 초기의 연구는 특정 도메인에서 정보를 추출하기 위한 규칙을 수동으로 생성하였다[9,10]. 수동의 규칙 생성 방법은 사람이 하는 작업이기 때문에 시간이 많이 걸리고 새로운 정보소스에 대해서 확장성과 유연성이 결여되는 문제점이 발생하였다.

정보추출 규칙의 수동생성에 따른 문제점을 최소화하기 위해 XWRAP은 wrapper를 학습하기 위해 최소한의 사용자 입력을 받는 방안을 소개하였다.[11] XWRAP은 HTML 문서를 계층 구조의 트리로 구성하고, 의미 부분에 대한 사용자 입력을 받도록 하고 있다. 이 시스템의 문제점은 반드시 사용자 입력을 받아야 하는데 있다. 사용자에게 친근한 인터페이스를 제공한다고 하더라도 시스템의 동작 상황을 정확히 알지 못하는 사용자가 정확한 입력을 준다는 보장은 없다. 또한 계층화된 트리로 구성되지 못하는 HTML 문서의 경우는 학습할 수 없는 문제를 가지고 있다.

이와 같은 수동이나 반자동(semi-autonomous) 방법의 문제점을 극복하기 위해 Wrapper의 자동생성에 대한 연구들이 수행되어 왔다. 그 중 자연어 처리를 통한 wrapper 자동생성 시스템은 전통적인 자연어 처리와는 달리 관련 있는 데이타 이외의 다른 텍스트들은 무시한다. 자연어로 구성된 문서에서 정보추출 패턴을 학습하

는 시스템에는 AutoSlog[12], LIEP[13], CRYSTAL [14,15,16] 등이 있다. 이 시스템들은 모두 자연어 처리에 초점을 맞추었기 때문에 웹의 문서를 학습하기에는 제약조건이 많으며 이러한 제약조건을 줄이는 것이 웹의 문서를 학습할 수 있는 관건이 된다.

자연어로 이루어진 문장이 아닌 경우에는 자연어 처리를 위한 알고리즘을 적용할 수 없으며 각 문서 형태에 대한 분석 연구가 필요하다. 이러한 연구들에는 WIEN[4], WHISK[17]등이 있다. WIEN은 인터넷상의 많은 정보들이 상관관계가 있는 데이타로 존재하고 있다고 보고, 라벨이 포함된 데이타로부터 wrapper를 자동으로 생성하기 위한 wrapper 귀납법(induction)을 제안하였다. WIEN은 훈련 문서에 오류가 없어야 정확한 학습이 가능하고, HLRT와 같은 wrapper 클래스 인식에 있어서 HTML 태그에 의존적이기 때문에 HTML 태그는 같지만 의미는 다른 정보의 경우 추출 오류가 발생할 수 있다는 문제점이 있다.

WHISK는 본 논문에서 소개하는 시스템과 가장 비슷한 시스템으로 웹의 구조화 문서에서 비구조화 문서까지 다양한 형태의 문서로부터 추출 패턴을 생성하는 학습 시스템이다. 학습된 패턴은 정규식(regular expression)으로 표현된다. 이 시스템은 광범위한 문서 형식을 학습하지만, 이를 위해서는 학습을 위한 지식을 사람이 수동으로 입력해주어야 한다. 많은 문서 형식을 포함하기 위해서는 사람이 입력해야 하는 지식 또한 많아진다. 따라서 WHISK는 수동으로 수행하는 작업의 양이 너무 많다는 단점이 있다.

본 논문과 WHISK는 정보추출을 위해 학습한 결과의 표현 방식이 비슷하고, 라벨을 가진 문서의 학습을 수행한다는 점에서도 비슷하다. 하지만, WHISK의 경우는 자연어 처리의 문맥 분석을 통해 정보추출 규칙을 학습하는데 비해 본 논문의 XTROS 시스템은 도메인 지식만을 이용해서 패턴을 생성하고 반복되는 패턴의 학습을 통해 정보추출 규칙을 생성한다는데 다른 점이 있다. 또한 WHISK는 학습을 수행할 때마다 사용자로부터 어떤 정보를 추출해야 할지 입력을 받아야만 하는 지도학습(supervised learning)을 수행하지만, 본 논문의 시스템은 학습 시에 이러한 사용자 입력이 필요 없이 도메인 지식만을 이용해 학습을 수행하는 지식 기반 학습(knowledge-based learning)을 수행한다.

7. 결론 및 향후 연구방향

본 논문에서는 도메인 지식과 wrapper를 XML 문서로 표현하고, wrapper를 도메인 지식과 준구조화 문서로부터 자동으로 생성하는 정보추출 에이전트에 대하여 기술하였다. 현재 구현된 정보추출 에이전트를 10개의 부동산 매물 검색사이트에 적용한 결과 8개의 검색 사이트에서 자동으로 100% 정확한 정보추출 규칙을 학습할 수 있었으며, 특정 도메인의 문서 학습에서 높은 성능을 보이는 것을 알 수 있었다. 본 논문의 학습 알고리즘은 라벨을 가진 어떠한 문서에도 적용이 가능하다고 판단되며, 다른 도메인의 문서에도 도메인 지식의 변경만으로 적용할 수 있다고 생각한다.

Wrapper를 생성하지 못한 완전한 테이블형의 문서는 라벨이 존재하지 않기 때문에 학습할 기준을 찾기 힘들었다. 따라서 본 논문의 알고리즘을 이용해 라벨이 없는 문서에 대한 wrapper의 생성은 부적합한 것으로 생각된다. 이러한 문제를 해결하기 위해서 테이블형일 경우 문서를 계층 구조의 트리로 구성하고, 이 트리의 상위 노드의 의미를 파악함으로써wrapper를 생성하는 알고리즘을 적용하는 방안을 현재 진행 중에 있다.

본 논문에서 제시한 XTROS 시스템의 또 하나의 제한점은 사용자 질의를 위한 규칙 파일의 <Form>... </Form>부분을 수동으로 생성해야 했다는 것이다. 그 이유는 웹에서 학습에 필요한 샘플 문서를 얻기 위해서는 복잡한 질의 과정이 요구되며 결과적으로 자동 학습이 어려웠기 때문이다. 현재 부동산 매물검색 사이트의 경우도 사용자 질의로 입력되는 정보의 개수나 형태가 각 사이트마다 달라 자동 학습은 어려웠다. 이 부분은 향후 연구과제로 남겨두고자 한다. 궁극적으로는 입력폼의 자동 분석을 통해서 일반 사용자가 사이트의 URL만을 입력해 주면 자동으로 해당 사이트를 학습하여 정보를 얻어서 사용자에게 제공하는 다중 도메인에 대한 개인화된 자동학습 에이전트의 연구를 추구하고자 한다.

참 고 문 헌

[1] 최중민, 인터넷 정보추출 에이전트, *정보과학회지* 18권 5호, pp. 48-53, 2000.

[2] N. Kushmerick, Gleaning the Web, *IEEE Intelligent Systems*, vol.14, no.2, pp. 20-22, 1999.

[3] Avrim Blum and Tom Mitchell, Combining Labeled and Unlabeled Data with Co-Training, *Proceedings of the 1998 Conference on Computational Learning Theory*, 1998.

[4] Nicholas Kushmerick, Wrapper Induction for Information Extraction, *Proceedings of 15th International Conference on Artificial Intelligence (IJCAI-95)*, pp. 729-735, 1995.

[5] Marti A. Hearst, Information Integration, *IEEE*

Intellegent Systems, vol.13, no.5, pp. 12-24, 1998.
[6] J. Yang, E. Lee, and J. Choi, A Shopping Agent that Automatically Constructs Wrappers for Semi-Structured Online Vendors, *Lecture Notes in Computer Science*, vol. 1983, pp. 368-373, 2000.
[7] 양재영, 전자상거래에서 상점 Wrapper 생성을 위한 지능형 에이전트의 학습방안 연구, *한양대학교 전자계산학과 석사학위 논문*, 2000.
[8] J. Yang, H. Seo, N. Koo, J. Choi, J. Kim, S. Kim, K. Lee, and H. Ham, A More Scalable Comparison Shopping Agent, *Engineering of Intelligent Systems(EIS 2000)*, pp.766-772, Paisely, Scotland, 2000.
[9] P. Atzeni, G. Mecca, and P. Merialdo, Semi-structured and Structured Data in the Web: Going Back and Forth, *ACM SIGMOD Workshop on Management of Semi-structured Data*, pp. 1-9, 1997.
[10] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, V. Vassalos, Template-based Wrappers in the TSIMMIS System, *ACM SIGMOD International Conference on Management of Data*, pp. 532-535, 1997.
[11] Ling Liu, Calton Pu, and Wei Han, XWRAP: An XML-based Wrapper Construction System for Web Information Sources, *Proceedings of the 16th International Conference on Data Engineering*, 2000.
[12] E. Riloff, Automatically Constructing a Dictionary for Information Extraction Tasks, *Proceedings of the Eleventh Annual Conference on Artificial Intelligence (AAAI-93)*, pp. 811-816, 1993.
[13] S. Huffman, Learning Information Extraction Patterns from Examples, *Workshop on New Approaches to Learning for Natural Language Processing, IJCAI-95*, pp. 127-142, 1995.
[14] S. Solderland, CRYSTAL: Inducing a Conceptual Dictionary, *Preceedings of 15th International Conference on Artificial Intelligence(IJCAI-95)*, pp. 1314-1319, 1995.
[15] S. Soderland, D. Fisher, and W. Lehnert., Automatically Learned vs. Hand-crafted Text Analysis Rules, *Technical Report TE-44 at Center for Intelligent Information Retrieval*, University of Massachusetts, 1997.
[16] S. Soderland, Learning Text Analysis Rules For Domain-Specific Natural Language Processing, *University Massachusetts Amherst, Department of Computer Science Ph.D thesis*, 1997.
[17] S. Solderland, Learning Information Extraction Rules for Semi-structured and Free Text, http://www.cs.washington.edu/homes/solderland/WHISK.ps

서 희 경
1999년 한양대학교 전자계산학과 졸업(학사). 2001년 한양대학교 대학원 전자계산학과 졸업(석사). 2001년 ~ 현재 삼성종합기술원 HCI Lab. 연구원. 관심분야는 지능형 에이전트 시스템, 정보추출, 데이타 마이닝, 기계학습

양 재 영
1998년 한양대학교 전자계산학과 졸업(학사). 2000년 한양대학교 대학원 전자계산학과 졸업(석사). 2000년 ~ 현재 한양대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 지능형 에이전트 시스템, 기계학습, 데이타 마이닝, 정보검색, 정보추출

최 중 민
1984년 서울대학교 컴퓨터공학과 졸업(학사). 1986년 서울대학교 대학원 컴퓨터공학과 졸업(석사). 1993년 State University of New York at Buffalo, Computer Science 졸업(박사). 1993년 ~ 1995년 한국전자통신연구원 인공지능연구실 선임연구원. 1995년 ~ 현재 한양대학교 컴퓨터공학과 부교수. 관심분야는 지능형 에이전트 시스템, 인공지능, 정보검색, 데이타베이스, HCI