사용자 인터페이스 에이전트를 통한 정보추출 규칙의 자동 생성

(Automatic Generation of Information Extraction Rules Through User-interface Agents)

김 용 기 † 양 재 영 †† 최 중 민 †††
(Yongkee Kim) (Jaeyoung Yang) (Joongmin Choi)

요 약 정보추출은 한 문서에서 그 문서의 중심적 의미를 나타내는 특정 구성요소를 인식하여 추출하는 작업으로서, 이질적인 여러 정보소스로부터 균일화된 정보추출을 수행하기 위해서는 각 정보소스에 맞는 정보추출 규칙을 생성해야 한다. 기존 정보추출 규칙의 생성 방법에는 전문가에 의한 수동 생성 방법과 에이전트 프로그램에 의한 자동 생성 방법이 있는데, 수동 생성은 규칙의 정확성은 보장되나 확장성과 효율성에 문제가 있고, 자동 생성은 확장성은 있으나 규칙 생성 자체의 어려움과 생성된 규칙의 신뢰성이 문제점으로 대두된다.

본 논문에서는 이러한 두 가지 방법의 문제점을 보완하여 추출 규칙의 정확성과 확장성을 동시에 제공하기 위해 지도 학습(supervised learning)을 적용한 정보추출 규칙 생성 기법을 제안한다. 본 논문에서 제시하는 방법은 사용자 인터페이스 에이전트를 사용하여 정보추출 규칙 생성을 위한 단서 정보를 사용자로부터 받고 이 정보를 바탕으로 에이전트가 XML로 표현된 규칙을 생성하는 것이다. 결과적으로 정보추출 규칙의 수동 생성과 자동 생성을 혼합한 형태가 된다. 사용자 인터페이스 에이전트는 규칙의 생성 뿐 아니라 기존의 규칙을 수정하거나 확장하는데도 이용된다. 구인 광고와 논문모집 공고와 관련된 정보소스에 대해 이 방법을 테스트한 결과 다른 기법에서 추출하지 못했던 정보를 추출할 수 있었고, 성능 면에서도 80% 이상의 정확도와 재현율을 보였다. 본 시스템은 추후 정보 중재자 에이전트와 같은 응용 분야에 적용시킬 수 있을 것으로 기대한다.

키워드 : 인터페이스 에이전트, 정보추출, 기계학습

Abstract Information extraction is a process of recognizing and fetching particular information fragments from a document. In order to extract information uniformly from many heterogeneous information sources, it is necessary to produce information extraction rules called a wrapper for each source. Previous methods of information extraction can be categorized into manual wrapper generation and automatic wrapper generation. In the manual method, since the wrapper is manually generated by a human expert who analyzes documents and writes rules, the precision of the wrapper is very high whereas it reveals problems in scalability and efficiency. In the automatic method, the agent program analyzes a set of example documents and produces a wrapper through learning. Although it is very scalable, this method has difficulty in generating correct rules per se, and also the generated rules are sometimes unreliable.

This paper tries to combine both manual and automatic methods by proposing a new method of learning information extraction rules. We adopt the scheme of supervised learning in which a user-interface agent is designed to get information from the user regarding what to extract from a document, and eventually XML-based information extraction rules are generated through learning according to these inputs. The interface agent is used not only to generate new extraction rules but also to modify and extend existing ones to enhance the precision and the recall measures of the

· 본 연구는 한국과학재단 지역대학우수과학자 지원연구(과제번호 R05-2001-000-01005-0) 지원으로 수행되었음
† 비 회 원 : 한양대학교 컴퓨터공학과
ykkim@cse.hanyang.ac.kr
†† 비 회 원 : (주)오픈베이스 기술연구소 연구원
jyyang@openbase.co.kr
††† 종신회원 : 한양대학교 컴퓨터공학과 교수
jmchoi@cse.hanyang.ac.kr
논문접수 : 2003년 8월 12일
심사완료 : 2004년 1월 6일

extraction system. We have done a series of experiments to test the system, and the results are very promising. We hope that our system can be applied to practical systems such as information-mediator agents.

Key words : interface agent, information extraction, machine learning

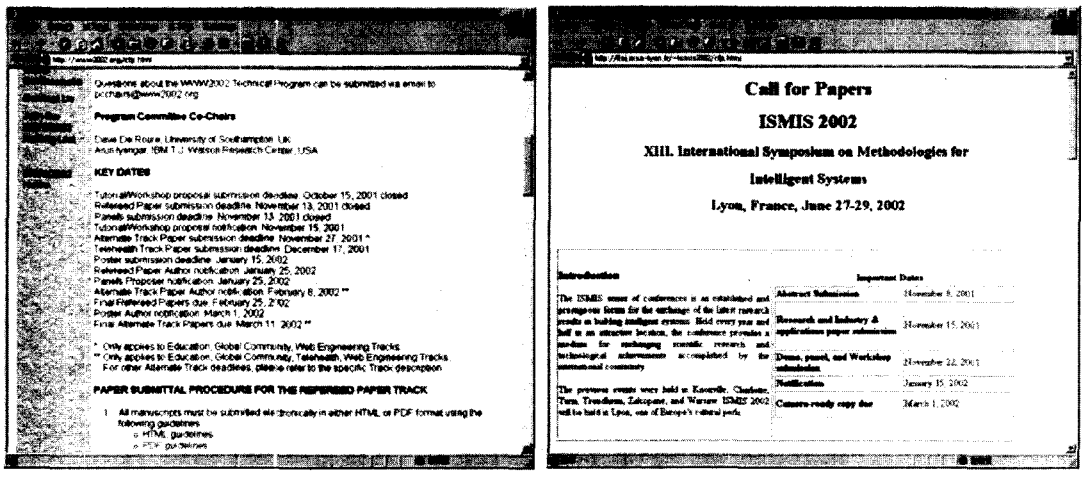
1. 서 론

정보추출은 관심을 가지고 있는 문서에서 필요한 데이타 부분을 추출해내는 작업이다[1]. 정보추출에서의 정보소스는 필요한 데이타를 추출해야 하는 대상 문서를 말하며, 본 논문에서는 웹 문서를 정보소스로 삼는다. 웹 문서는 준구조화된 문서(semi-structured document)의 형태를 가지며 주로 HTML로 작성되어 있기 때문에 같은 정보라 하더라도 정보를 제공하는 사이트 및 문서 작성자의 스타일에 따라서 사용하는 단어와 레이아웃이 서로 다르다. 예를 들어, 그림 1은 국제학술대회의 논문모집공고(Call-for-Paper, CFP)에 기술된 논문제출 마감일을 표현한 두 가지 서로 다른 방법을 나타내고 있다. 그림 1(a)는 자연어로 기술된 리스트 구조로 되어 있고, 그림 1(b)는 HTML 테이블 구조를 사용하여 작성되었다.

그림 2는 단어를 서로 다르게 사용함으로써 발생할 수 있는 이질성의 예로서, 회사의 고용정보 사이트를 나타낸다. 이 경우, 고용하는 회사명을 나타내는 레이블이 사이트마다 "사업장명", "기업명", "회사명" 등 서로 다른 단어를 사용하는 경우가 많으며, 이러한 현상은 담당자, 연락처, 모집직종 등 다른 정보에 대해서도 마찬가지다.

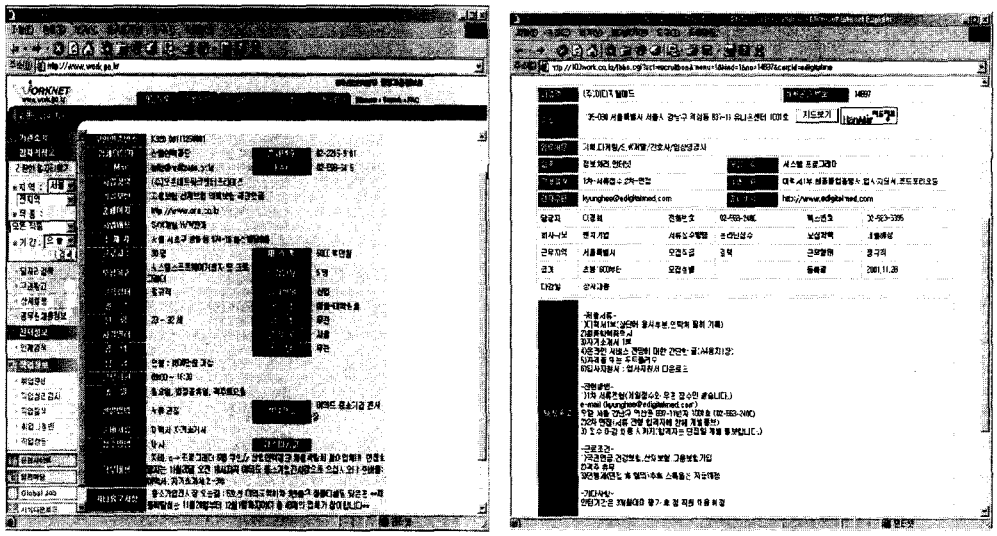
이러한 이질적인 여러 정보소스로부터 단일화된(uniform) 정보추출을 수행하기 위해서는 각 정보소스에 맞는 정보추출 규칙인 래퍼(wrapper)를 생성해야 한다[2]. 기존의 정보추출 규칙 생성 방법은 전문가에 의한 수동 생성 방법과 에이전트 프로그램에 의한 자동 규칙생성 방법으로 나눈다[3]. 수동 생성은 사람이 정보소스 문서의 구조를 파악한 후 각 정보소스에 대한 래퍼를 작성하는 것이다[4]. 이러한 수동 생성은 규칙의 정확성은 보장되나 새로운 정보소스가 있을 때마다 사람이 일일이 문서의 분석과 규칙 생성을 해야 하므로 확장성과 효율성에 문제가 있다. 반면에 자동 생성은 에이전트 프로그램이 기계학습을 통해 자동으로 문서로부터 규칙을 생성하므로 확장성과 효율성은 매우 좋아지나 규칙 생성 자체의 어려움과 생성된 규칙의 신뢰성이 문제점으로 대두된다[5-7].

본 논문에서는 이러한 두 가지 방법의 문제점을 보완하여 추출 규칙의 정확성과 확장성을 동시에 제공하기 위한 정보추출 규칙 생성 기법을 제안한다. 이 방법의 주된 특징은 사용자 인터페이스 에이전트를 사용하여 정보추출 규칙 생성을 위한 정보를 사용자로부터 받고 이 정보를 바탕으로 XML로 표현된 규칙을 생성하는 것이다. 인터페이스 에이전트를 통해 사용자가 학습에 필요한 단서(hint)를 제시함으로써 학습의 성능을 높여



(a) 자연어로 작성된 제출 마감일 (b) HTML로 작성된 제출 마감일

그림 1 사용자의 작성 스타일에 따른 이질적인 정보소스의 예



(a) 산업인력공단 고용정보 사이트 (b) 헤드뱅크 사이트

그림 2 단어 사용의 다양성에 따른 이질적인 정보소스의 예

주는 이 기법은 지도 학습(supervised learning)[8]으로 구분할 수 있으며, 결과적으로 정보추출 규칙의 수동 생성과 자동 생성을 혼합한 형태가 된다. 이를 통해 사용자가 원하는 정보가 무엇인지를 파악하고 정보추출기의 추출결과에 대한 사용자의 피드백을 추출규칙 학습에 이용하여 보다 정교한 정보추출 규칙을 생성하고자 한다. 사용자 인터페이스 에이전트는 규칙의 생성 뿐 아니라 기존의 규칙을 수정하거나 확장하는데도 이용된다.

2. 관련 연구

문법적 형식을 갖춘 문장, 즉 비구조화된(unstructured) 텍스트로 이루어진 문서에서 원하는 정보를 추출하려면 추출 규칙은 문장의 구문구조와 의미구조를 모두 이용하여 문서 안에서 관련된 정보를 식별해야 한다. 추출 패턴을 적용하기 전에 전처리 과정으로 구문 분석기(syntactic analyzer)와 의미 태거(semantic tagger)가 필요하다. AutoSlog[9]는 추출 규칙의 사전이라고 할 수 있는 개념 노드(concept nodes)를 자동으로 생성한다. 하지만 이 시스템은 하나의 문장에서 하나의 정보만을 추출할 수 있으며 미리 정의된 언어 패턴을 사용한다. PALKA[10]는 FP 구조(FP-structure)와 개념계층(concept hierarchy)을 사용하며, 문장의 구문 형식과 의미를 나누어서 분석하는 시스템이다.

웹의 폭발적인 증가로 문법 구조에 맞는 문장과 간결하지만 비문법적인 문장이 혼합된 형태의 문서가 많이 제공되었다. 따라서 구인 광고나 버스 시간표, 부동산 광고와 같은 혼합적인 구조의 웹 문서에 대해서는 위에서 제시된 비구조화된 문서에 대한 정보추출 방식이 더 이상 적합하지 않게 되었다. 이와 같은 웹 상의 온라인 문서에서 정보를 추출하는 시스템으로 WHISK[11], RAPIER[12], SRV[13] 등이 있다. WHISK는 구조화된 문서와 비구조화된 문서 모두에 적용할 수 있는 추출 알고리즘을 제시하고 있고, 유일하게 모든 형식의 문서에 대해 다중 슬롯 규칙을 생성한다. WHISK의 추출 패턴은 두 가지로 구성된 정규 표현식의 특수한 형태인데, 하나는 관련된 구를 만드는 문맥을 기술하고 다른 하나는 추출될 구의 구분자(delimiter)를 설명한다. RAPIER는 의미 클래스와 POS(part of speech, 품사) 태거의 결과인 제한된 문장 성분에 대한 정보를 이용하여 하나의 슬롯을 채우는 추출 패턴을 학습한다. SRV는 문서의 관계 구조에 기초를 둔 일차 논리(first-order logic) 추출 패턴을 생성한다.

자동으로 정보추출 규칙을 생성하는 가장 널리 알려진 방법은 래퍼 유도(wrapper induction)이며 여러 웹 문서의 데이타를 통합하고 추출하기 위한 필요성에 의해 나타났다. 래퍼는 특정 웹 소스에서 여러 종류의 데이타를 추출할 수 있도록 작성된 규칙 또는 프로시저를 말한다. 래퍼 유도 시스템은 대상이 되는 문서에서 사용되는 구분자에 기초한 규칙을 추출하며 언어적인 규칙은 전혀 사용하지 않는다. 대표적인 래퍼 유도 시스템으로는 WIEN[2], SoftMealy[14], STALKER[15] 등이 있다. WIEN은 WHISK와 유사하지만 구분자만을 사용

하는 규칙을 생성한다는 것이 다르다. 또한 모든 문서에 적용 가능한 하나의 다중 슬롯 규칙이 있음을 가정하고 의미 클래스를 사용하지 않는다. SoftMealy는 WIEN보다 더 표현력이 있는 추출 패턴을 생성하여 다양한 형식이 혼재하는 문서에서 추출 규칙을 생성하는데 유용하다. STALKER는 계층적인 정보추출 래퍼 유도 시스템으로서, 계층적인 내용을 지니고 있는 문서에서 정보를 추출한다. 이것은 문서의 계층적 구조를 기술하는 ECT(Embedded Catalog Tree) 형식을 도입함으로써 가능하다. ECT는 정보추출 작업에 의해 생성되는 결과물의 구조를 기술하며, STALKER는 트리의 각 노드에 하나의 추출 규칙을 생성한다.

이상에서 설명한 관련 연구를 살펴보면 초기의 AutoSlog와 같은 규칙의 수동 생성 시스템에서 출발하여 래퍼 유도 시스템과 같은 규칙의 자동 생성 시스템으로 발전해 온 것을 알 수 있다. 하지만 두 방법 모두 장단점을 지니고 있으며 이를 동시에 보완하기 위한 작업에 대한 연구는 미진한 상태이다. 본 논문에서는 인터페이스 에이전트를 이용하여 사용자의 의도를 입력받아서 이를 바탕으로 학습을 통해 정보추출 규칙을 생성하는 수동/자동 혼용 시스템을 제시하고자 한다.

3. 정보추출 에이전트 시스템 구조

본 논문에서 제안하는 정보추출 에이전트 시스템의 전체구조는 그림 3과 같다. 전체 시스템은 인터페이스 에이전트, 규칙 학습기, 규칙 해석기로 구분되며 규칙을 갱신, 확장하기 위해서 WEKA의 결정트리(decision tree) 모듈을 이용하였다.

인터페이스 에이전트를 통한 사용자 입력이 데이타와 훈련예제로 변환되고, 훈련예제에 대한 학습이 이루어지며, 이 학습의 결과가 추출규칙으로 저장된다. 규칙 해석기는 추출 규칙을 토대로 웹 문서에서 슬롯별로 정보를 추출한다.

3.1 인터페이스 에이전트

인터페이스 에이전트는 사용자와 학습기의 중간에 위치해 사용자와 상호 작용하도록 설계되었다[16]. 인터페이스 에이전트의 주요 기능은 "추출할 정보의 카테고리 인식"에 있다. 인터페이스 에이전트는 그림 4와 같은 드래깅 앤 마우스 클릭(dragging and mouse click) 입력방식을 취하고 있으며, 이를 통해서 사용자가 추출을 원하는 데이타 항목을 선택하고 마우스 조작을 통해 그 항목의 카테고리 정보를 메뉴에서 선택하여 입력할 수 있다.

사용자는 우선 입력하고자 하는 정보를 마우스로 드래그하여 선택한 후 오른쪽 마우스 버튼을 누른다. 이때 그림 4와 같은 팝업 메뉴가 나타나며 첫 번째 메뉴인 "Slot Name"을 클릭하여 추출하고자 하는 정보의 이름(예를 들어 "전화번호" 또는 "제출기간")을 입력한다. 그리고 두 번째 메뉴인 "Target Data"를 클릭하면 선택된 부분의 정보가 자동으로 입력되고 데이타의 입력이 정상적으로 이루어졌다는 대화 창이 뜨게 된다. 인터페이스 에이전트는 사용자의 마우스 입력에 대한 내용을 이해하고 선택된 문자열, 트리거, 슬롯 이름 등 관련 정보를 학습기에 전달한다. 트리거나 슬롯 이름과 같은 내용은 도메인에 따라 정해진다.

인터페이스 에이전트는 사용자의 마우스 입력에 대한 이해, 웹 문서의 정확한 표현, 사용자 피드백의 수용, 확장의 용이성 등을 갖추고 있어야 한다. 이러한 요구사항을 만족시키기 위해 별도의 인터페이스 에이전트를 구현하는 것은 어려운 일이다. 특히 웹 문서의 정확한 표현과 확장의 용이성을 지원하는 별도의 브라우저를 단기간에 개발할 수는 없다. 따라서 본 논문에서는 마이크로소프트사의 인터넷 익스플로러와 프락시 서버를 이용하여 인터페이스 에이전트를 구현하였다. 인터페이스 에이전트는 추출된 데이타를 자바 스크립트의 팝업 윈도

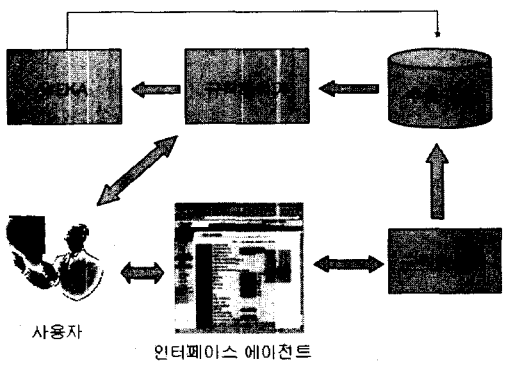


그림 3 인터페이스 에이전트 기반 정보추출 시스템의 구조

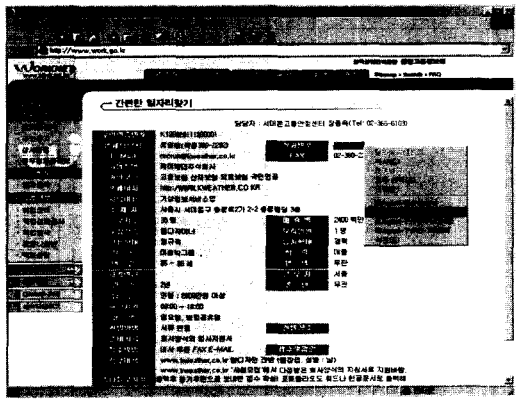


그림 4 학습을 위한 인터페이스 에이전트의 입력방식

우를 이용하여 사용자에게 제공한다. 사용자는 팝업 윈도우에 나타난 정보가 추출대상의 정보가 맞는지에 대한 피드백을 인터페이스 에이전트에게 제공할 수 있다.

3.2 정보추출 규칙의 형식

정보추출 규칙은 사람이 쉽게 이해할 수 있고 유지보수가 용이하도록 XML을 사용하여 작성하였다[17]. 그림 5는 본 논문에서 사용하는 정보추출 규칙의 형식을 보여준다. 정보추출 규칙은 도메인마다 따로 정의되며 각 도메인의 규칙은 추출하고자 하는 슬롯의 집합으로 구성되고 또한 각 슬롯은 정보추출 패턴의 집합으로 이루어져 있다. 패턴은 가장 작은 단위의 규칙이라고 할 수 있으며 하나의 학습 문서에서 사용자가 입력한 규칙의 단위가 된다. 사용자는 각 슬롯에 대한 패턴을 계속해서 추가함으로써 시스템을 학습시킨다. (그림 5에서 이탤릭체로 표현된 <Domain>과 <Slot> 태그는 해당 도메인과 슬롯에 따라 태그 이름이 정의된다는 의미이다. 예를 들면, "구인광고" 도메인의 "제출시기" 슬롯에 대한 태그는 <Job_offer>와 <From_to>로 작성될 수 있다. 나머지 태그는 그림에 있는 태그 이름 그대로 사용된다.)

```
<Domain>
   <Slot>
      <pattern>
         <id> String </id>
         <type> TOKEN | PHRASE </type>
         <trigger> String </trigger>
         <format> [TRIG] String* TARG {String}* String*
                  </format>
      </pattern>
      .....
   </Slot>
   .....
</Domain>
```

그림 5 정보추출 규칙의 형식

<pattern> 구조는 <id>, <type>, <trigger>, <format>의 구성요소를 가진다. 실제로 사용자로부터 받는 정보는 트리거와 추출하고자 하는 데이타뿐이며 다른 값들은 인터페이스 에이전트가 자동으로 데이타를 분석하여 구한다. <id>는 패턴의 식별자 역할을 한다. <type> 요소는 추출하고자 하는 데이타의 구조적 형식을 의미하며 TOKEN과 PHRASE의 두 가지로 정의된다. <type>의 값이 TOKEN인 경우는 추출할 데이타가 문장에서 하나의 토큰으로 분리되어야 한다는 것을 의미하며 PHRASE인 경우는 여러 단어로 구성되므로 문장에서 부분 매칭을 통해서 만족되는 부분만을 추출한다는 것을 의미한다. TOKEN에 비해 PHRASE를 이용하면 재현율(recall)은 높아질 수 있지만 TOKEN을 써야 하는 부분에 PHRASE를 잘못 사용하면 추출대상이 아닌 데이타를 추출할 수도 있어서 정확도(precision)가 떨어진다. <trigger> 요소는 대상 문장에 대해 이 규칙을 적용할 것인지를 결정할 때 사용되는 구분자 역할을 수행한다. 하나의 패턴에는 하나의 트리거만 올 수 있지만 한 슬롯에 여러 패턴이 존재할 수 있으므로 마치 한 슬롯에 여러 트리거가 올 수 있는 효과가 되어 결국 특정 정보가 다양한 단어로 표현될 수 있는 상황을 표현할 수 있다. <format> 요소는 실제 문장에서 추출할 데이타의 위치 및 구조적인 특징을 기술한다. [TRIG]는 현재 패턴 내의 <trigger> 요소의 참조이고, TARG은 추출하고자 하는 타겟 정보, {String}은 String에 포함된 문자열이 문장 전체나 혹은 추출하고자 하는 목표 데이타에 반드시 포함되어야 함을 나타낸다. 여기서 "*"는 옵션으로서 건너뛸 수 있음을 의미한다. <format> 요소는 규칙 해석기에서 문자열 분석을 위해 사용되는 부분이며 여러 <format>을 정의함으로써 다양한 범위의 문서에서 필요한 데이타를 추출할 수 있다.

3.3 정보추출 규칙 학습기

규칙 학습기는 인터페이스 에이전트에서 사용자가 입력한 정보를 추출 규칙의 패턴으로 변환시켜준다. 슬롯의 이름과 추출할 정보, 트리거 정보 등이 인터페이스 에이전트를 통해 사용자로부터 제공된다. 먼저 이 데이타를 분석하여 하나의 단어로 구성된 TOKEN 타입인지, 아니면 여러 개의 단어로 구성된 PHRASE 타입인지 구별하여 <type> 요소의 값으로 한다. 그리고 나서 트리거와 입력 데이타를 문서 내에서 찾아 이들의 관계를 알아낸 후 <format> 요소의 값으로 채운다. 예를 들어, 트리거가 "제출기간", 데이타가 "2002. 4. 6~4. 19"이었고 트리거가 데이타의 앞에 나타났다면 <format>의 값은 "[TRIG]: TARG"가 된다.

규칙 학습기는 기본적으로 Covering 알고리즘을 사용하였다[8]. 즉, 학습의 대상이 되는 문서의 집합을 구성하고 각 패턴이 생성될 때마다 규칙 해석기를 수행시켜 그 패턴을 이용하여 얼마나 많은 문서에서 성공적으로 원하는 정보를 가져올 수 있는지, 즉 얼마나 많은 문서를 커버할 수 있는지를 구한다. 이 커버 값은 새로 생성된 규칙이 의미가 있는지 또는 무시하고 버려야 할 것인지를 판단하는 기준이 된다. 본 시스템에서는 커버 값이 2 이하인 규칙은 무시한다. 그림 6은 Covering 알고리즘을 이용하여 패턴 집합을 학습하기 위한 알고리즘을 pseudo-code로 나타낸 것이다.

그림 7은 그림 5에서 정의한 규칙 형식을 적용하여 실제 구인정보를 제공하는 사이트에서 사용자가 관심을 두고 있는 고용회사명, 모집기간, 전화번호, 이메일주소 등의 슬롯 정보를 추출하기 위해 생성된 규칙 중 "회사

```
procedure Learner(examples)
  let rule_set = { };
  for each slot:
    for each example in examples:
      patterns = pattern_learner(examples);
      add pattern to rule_set;
  return rule_set;

procedure pattern_learner(examples)
  let patterns = { };
  repeat for each example in examples
    receive input_data from user;
    if input_data is not null
    then find covers
        build pattern
        add pattern to patterns
    remove those examples in examples covered by pattern
  until examples is null
  return patterns
```

그림 6 패턴 집합을 학습하기 위한 학습 알고리즘

명"과 "모집기간"에 대한 규칙 패턴의 일부를 보여준다. 여기서 볼 수 있듯이 <trigger> 요소는 문장 분석의 시작을 알리는 역할도 있지만 <format> 요소에 다시 사용됨으로써 준구조화된 문서나 구조화된 문서에서 특정 슬롯의 존재를 인식할 수 있는 중요한 구분자 역할을 담당하기도 한다.

3.4 정보추출 규칙 해석기

정보추출 규칙 해석기는 실제로 정보추출을 수행하는 시스템을 의미한다. 정보추출 규칙 해석기에서 규칙을 해석하기 위해 먼저 정보소스를 일정한 길이의 문장 집합으로 재구성한다. 재구성된 문장에서 각 슬롯을 채우기 위해 <trigger> 요소의 값들이 실제 문장에서 발생하는지를 검사한다. <trigger> 값이 존재하는 문장만을 선택하여 <format> 요소의 내용을 적용하여 추출하고자 하는 데이타의 구조적인 특징이 맞는지를 판단한다. 매칭이 성공적으로 이루어지면 추출할 타겟 데이타를 가리키는 TARG 부분을 추출하게 된다.

정보추출 규칙 해석기는 XML의 DOM 파서와 JAVA를 이용하여 구현하였다. 그림 8은 구현한 정보추출 규칙 해석기의 실행화면을 캡처한 것이다. 그림 8에서 왼쪽 창의 트리는 생성된 XML 규칙의 계층구조를 나타내며 오른쪽 창 상단의 텍스트는 문자열 처리후의 정보소스를 나타낸다. 오른쪽 창 하단은 규칙을 이용하여 정보소스로부터 정보추출을 수행한 결과를 나타낸다.

3.5 WEKA 모듈

규칙을 검증하고 갱신하기 위한 모듈로서 학습 기법 중 하나인 결정트리를 이용하는 WEKA를 사용하였다 [18]. 사용자가 입력한 규칙은 결정트리를 통해 재구성되며 먼저 입력한 패턴이 결과로 다시 나타나게 된다.

```
<?xml version="1.0" encoding="euc-kr" ?>
<Job_offer>
  <Company>
    <pattern>
      <id> companyName01 </id>
      <type> TOKEN </type>
      <trigger> (주) </trigger>
      <format> [TRIG] TARG </format>
      <format> TARG [TRIG] </format>
    </pattern>
    <pattern>
      <id> companyName02 </id>
      <type> TOKEN </type>
      <trigger> 회사명 </trigger>
      <format> [TRIG] : TARG </format>
      <format> [TRIG] TARG </format>
    </pattern>
  </Company>
  <From_to>
    <pattern>
      <id> from_to01 </id>
      <type> PHRASE </type>
      <trigger> 접수기간 </trigger>
      <format> [TRIG] : TARG {DATE} </format>
    </pattern>
    <pattern>
      <id> from_to02 </id>
      <type> PHRASE </type>
      <trigger> 마감일 </trigger>
      <format> [TRIG] * TARG {DATE} </format>
    </pattern>
    .....
  </From_to>
  .....
</Job_offer>
```

그림 7 고용정보 추출을 위한 정보추출 규칙 예

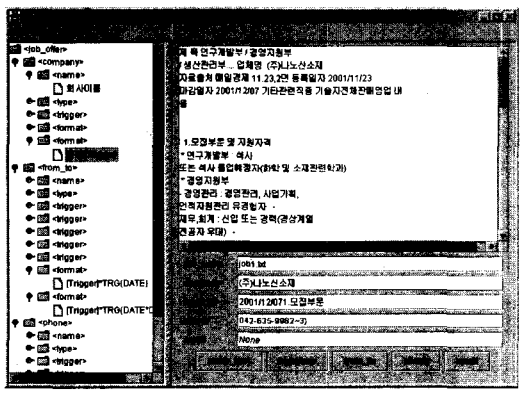


그림 8 정보추출 규칙 해석기의 실행화면

사용자가 입력하지 않은 규칙도 생성될 수 있다. 결정트리에 의해 생성된 규칙을 패턴으로 추가하면 정확도와 재현율을 높일 수 있으며, 학습기를 통해 입력한 규칙을 다시 구할 수도 있어 검증이 가능하다. 또한 학습된 규칙이 결정트리에 의해 생성된 규칙에서 벗어나는 경우

학습된 규칙을 수정할 수도 있다. 그림 9는 WEKA를 사용하기 위한 규칙과 데이타의 입력형태이다. 추출기에서 데이타를 추출할 때 어느 패턴을 사용하여 추출되는지 기록으로 남길 수 있다. 속성값으로 사용되는 값들은 이미 학습된 패턴에서 현재 슬롯의 값을 추출하는데 사용된 값이며, 데이타는 추출기에서 데이타를 추출할 때 생성된다.

```
@relation job.phone

@attribute type {TOKEN, PHRASE}
@attribute trigger {Tel, 문의, 전화}
@attribute format {[TRIG]TARG{PN}, [TRIG]*TARG{PN},
    [TRIG]*:TARG{PN}}
@attribute rightAccess {yes, no}

@data

phrase,전화,[TRIG]*:TRG{PN},no
token,전화,[TRIG]*TRG{PN},yes
phrase,전화,[TRIG]TRG{PN},yes
.....
```

그림 9 WEKA 포맷으로 작성된 정보추출 규칙과 데이타

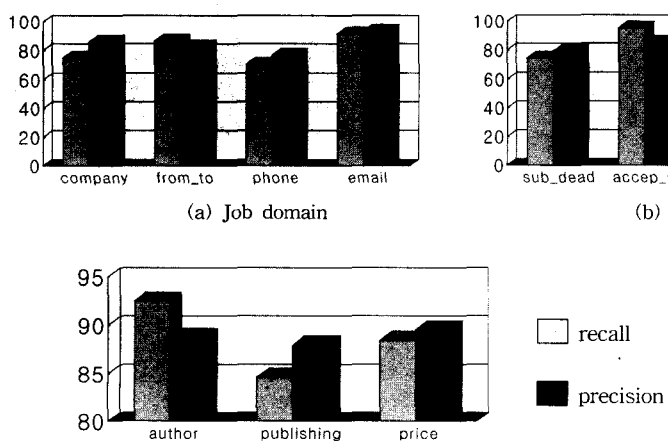
4. 실험결과

정보추출 인터페이스 에이전트의 성능을 평가하기 위해 인터넷 서점(Book), 구인광고(Job), 논문모집공고(CFP)의 세 가지 도메인에 대한 웹 문서 데이타를 수집하여 실험하였다. 인터넷 서점은 네이버 검색엔진을 이용하여 찾은 인터넷 서점을 순서대로 32개 선정하여 각 사이트에서 최소 한 개에서 최대 다섯 개 정도의 페이지를 수집하여 훈련 데이타와 실험 데이타로 같은 수만큼 분리하여 실험을 수행하였다. 인터넷 서점에 대해서는 책 정보 중 저자(author), 출판사(publishing), 가격정보(price)를 추출하였다. 구인광고의 경우도 마찬가지로 하나의 사이트에서 소수의 문서를 찾고 모아진 전체문서를 훈련 데이타와 실험 데이타로 분리하였다. 구인광고에 대해서는 회사이름(company), 모집기간(from_to), 전화번호(phone), 이메일주소(email) 정보를 추출하였다. CFP의 경우 구글 검색엔진에서 질의어를 주었을 때 가져온 순서대로 100개의 문서를 선택하여 50개씩 훈련 데이타와 실험 데이타로 나누었다. CFP에 대해서는 논문제출시한(sub_dead), 통과논문 고지일(accept_no), 최종버전 제출일(camera-re)의 정보를 추출하였다.

데이타의 특성상 인터넷 서점과 구인광고는 각 사이트별로 문서가 정해진 형식을 가지고 있다. 따라서 하나의 사이트에서 많은 수의 문서를 가져오는 것보다는 많은 수의 사이트에서 문서를 가져오는 것이 추출 규칙의 정확도를 높일 수 있다. 하지만 CFP의 경우는 학회나 워크샵이 모두 다른 형식의 문서구조를 가지고 있으며 어느 정도의 공통점은 있으나 같은 내용에 대해서 서로 다른 형식과 어휘를 사용하는 경우가 많다는 것을 고려하였다. 실험은 우선 훈련 데이타를 이용하여 인터페이스 에이전트를 통해 입력한 규칙이 실험 데이타에 대해 얼마만큼의 데이타를 정확하게 추출할 수 있는지를 측정하였다.

그림 10은 세 도메인에 대한 재현율과 정확도를 그래프로 나타낸 것이다.

여기서 재현율은 관련된 정보를 얼마나 빠뜨리지 않고 가져오는 가하는 비율이고 정확도는 얼마나 정확히



(a) Job domain

(b) CFP domain

(c) Book domain

그림 10 세 개의 도메인에 대한 정보추출 실험결과

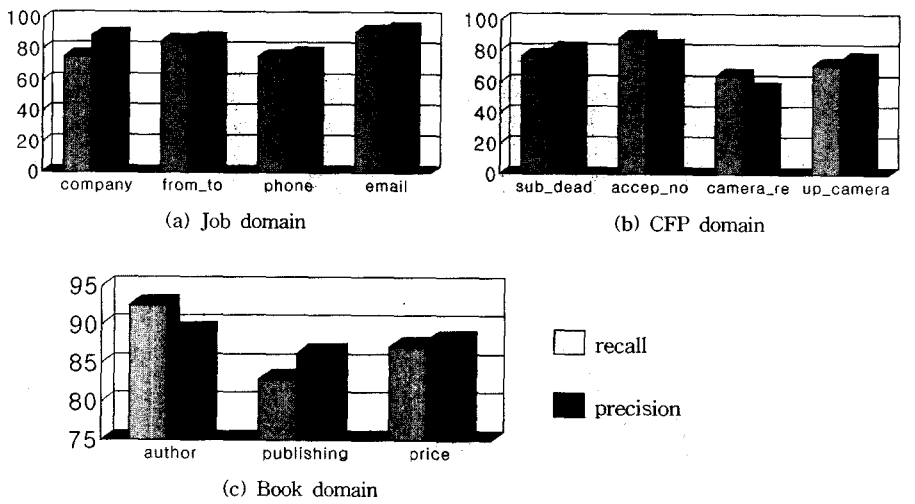
추출하는가의 비율이므로 예를 들어, 50개의 구직광고 문서에 서류제출 마감기간의 데이타가 모두 포함되어 있을때 이 시스템이 40개의 문서에서만 마감기간을 추출해을 수 있었다면 재현율이 80%가 되는 것이고, 이 가져온 40개 마감기간 데이타 중에서 실제로 정확한 데이타가 30개였다면 정확도는 30/40 = 75%가 되는 것이다. 물론 추출해온 데이타에 약간의 noise가 포함될 수도 있는데, 필요한 데이타만 들어있다면 정확한 것으로 평가하였다.

(a)의 구인광고 도메인에서는 재현율과 정확도가 평균 80% 이상이었으며 각 슬롯에 따라 약간의 차이를 나타내었다. 특히, 이메일주소의 경우 90%를 넘는 수치를 나타냈는데 이것은 사용하는 트리거와 데이타가 한정되어 있어서 인식이 쉽기 때문이다. 트리거의 선택에 따라서 정확도와 재현율은 조금씩 차이를 보이게 된다. 하나의 패턴 안에 하나의 트리거가 존재하므로 패턴의 수가 많을수록 정확도와 재현율은 높아진다. 전화번호는 숫자로만 표현되는 경우가 많으므로 내부의 데이타가 제출기간과 유사한 형태로 분석될 수 있고 이때는 트리거를 이용해서 두 종류의 데이타를 구분하게 된다.

(b)의 CFP의 경우를 보면 논문제출시한은 비교적 양호한 결과를 보여주며, 통과논문 고지일도 높은 정확도와 재현율을 보여주지만 논문의 최종버전 제출일은 낮은 정확도와 재현율을 보여주고 있다. 이것은 훈련 데이타에서 거의 발견되지 않았던 "final version"이란 단어가 실험 데이타에서 빈번히 나타났기 때문이며 학습시의 편향 현상에 의한 결과라고 할 수 있다. 그림에서 up_camera라고 표시된 것은 인터페이스 에이전트와 WEKA를 이용해 재학습을 한 후 같은 테스트 집합에 대해 다시 실험을 한 결과이다. 이때 트리거로 "final"을 추가하고 데이타를 입력한 결과 재현율과 정확도 모두 camera_re보다 높고 다른 슬롯과 거의 유사한 결과를 얻을 수 있었다.

갱신된 규칙은 결정트리에서 만들어진 규칙 중에서 이미 학습된 규칙과 현재의 트리거에서 생성되기 어려운 포맷을 제거하면 통상적으로 하나에서 두 개 정도의 패턴이 추가된다. 이 패턴은 트리거와 패턴의 새로운 조합이라고 할 수 있으며 이를 이용해서 비슷한 형태를 지니면서도 추출되지 못했던 정보를 추출할 수 있다. 따라서 갱신된 규칙은 갱신되기 전보다 정확도와 재현율에 있어서 통상적으로 약간 높게 나타나며 이에 대한 실험 통계를 그림 11에 나타내었다. 이 실험에서 재현율의 경우 평균 6%, 정확도는 1%가 높게 나타났다. 재현율이 높다는 것은 사용자에게 더 많은 정보를 가져다준다는 것을 의미한다.

정보추출 분야에서는 정보검색의 TREC과 같은 benchmark 데이타가 없어서 다른 관련 연구와의 직접적인 성능비교는 쉽지 않은 편이다. 하지만 2장의 관련 연구에서 다룬 몇몇 시스템과 비교할 때 수동 생성이나 자동 생성의 한 가지 방법만을 이용한 시스템에서 추출이 불가능한 정보(즉, 구조 정보가 없거나 trigger가 없는 경우)에 대해서 본 시스템은 추출이 가능하였고, 성능면에서도 평균 80% 이상, 일부 도메인에 대해서는 95%이상의 정확도와 재현율을 보임으로써 만족할 만한 수치를 보여주었다. 물론 실제의 응용 제품으로 가기 위해서는 거의 100%에 가까운 성능을 보여야 하므로 이에 대



(a) Job domain

(b) CFP domain

(c) Book domain

그림 11 WEKA를 통해 갱신된 규칙에 의한 정보추출 실험결과

한 보완이 필요하다. 앞으로 구조적 정보가 없는 free text에 대한 좀 더 심화된 테스팅을 통해 알고리즘의 개선을 시도하고자 한다.

5. 결 론

본 논문에서는 정보추출의 수동 생성과 자동 생성의 두 가지 방법의 문제점을 보완하여 추출 규칙의 정확성과 확장성을 동시에 제공하기 위한 정보추출 규칙 학습 방법을 제안하였다. 이 방법의 주된 특징은 사용자 인터페이스 에이전트를 사용하여 정보추출 규칙 생성을 위한 정보를 사용자로부터 받고 이 정보를 바탕으로 학습을 통해 XML로 표현된 규칙을 생성하는 것이다. 결과적으로 정보추출 규칙의 수동 생성과 자동 생성을 혼합한 형태가 되었다.

사용자 인터페이스 에이전트는 규칙의 생성 뿐 아니라 WEKA 모듈을 통해 기존의 규칙을 수정하거나 확장하는데도 이용되었다. 사용자는 추출결과를 보면서 현재 추출결과가 적정한지를 판단하여 피드백 정보를 추출된 데이타와 함께 입력한다. WEKA 모듈의 결정트리를 이용하여 입력한 정보에 대한 규칙을 생성하고 이 규칙은 학습기에서 학습된 규칙을 포함하며 입력한 데이타로 생성 가능한 모든 가설 공간을 구성한다. 그러므로 트리거와 포맷의 새로운 조합이 생성되고 새로운 규칙은 규칙집합에 추가된다. 이러한 방식으로 학습기에서 학습한 규칙에 대한 검증과 갱신이 이루어진다. 갱신된 규칙을 같은 테스트 집합에 다시 적용한 경우 정확도와 재현율이 높게 나타나는 것을 볼 수 있었다.

실험 중에 도출된 문제점으로는 인터넷 서점에서 책 제목을 트리거와 포맷 형식으로 표현할 수 없었다는 것이다. 책 제목에 대한 표현은 트리거를 사용하지 않고 문서의 맨 앞에 책 제목을 내세우므로 본 연구의 규칙 표현 방식과는 맞지 않았다. 규칙 표현의 이러한 위치 정보에 의한 의미도 표현할 수 있도록 규칙의 표현방식을 확장할 필요가 있다. CFP 도메인의 최종버전 제출일에 대해서와 같이 데이타의 편향 현상이 발생한 경우 본 논문에서는 인터페이스 에이전트를 통해 규칙을 다시 입력하였다. 이를 보완하기 위해 향후 과제로는 양성 예제(positive example) 뿐 아니라 음성 예제(negative example)도 같이 학습 예제에 포함시키는 것이 필요할 것으로 판단되며, 이를 통해 "날짜"처럼 같은 형식으로 서로 다른 내용을 표현할 수 있는 경우에 대한 처리가 가능해질 것이다.

참 고 문 헌

[1] S. Huffman, "Learning information extraction pattern from examples," IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing, pp.127-142. 1995.

[2] N. Kushmerick, "Wrapper induction for information extraction," Proc. IJCAI-95, pp.729-735, 1995.

[3] S. Soderland, D. Fisher, and W. Lehnert, "Automatically learned vs. hand-crafted text analysis rules," Tech. Rep. TE-44, Center for Intelligent Information Retrieval, Univ. of Massachusetts, 1997.

[4] J. Hammer, H. Garcia-Molina, S. Nestorov, R. Yerneni, M. Breunig, and V. Vassalos, "Template-based wrappers in the TSIMMIS system," Proc. ACM SIGMOD Int. Conf. on Management of Data, Tucson, pp.532-535, 1997.

[5] N. Kushmerick, "Wrapper induction: efficiency and expressiveness," *Artif. Intell.* vol.118, pp.15-68, 2000.

[6] J. Yang, J. Kim, K. Doh, and J. Choi, "Wrapper generation by using XML-based domain knowledge for intelligent information extraction," *Lecture Notes in AI* vol.2417, pp.472-481, 2002.

[7] 서희경, 양재영, 최중민, "준구조화 정보소스에 대한 지식기반 Wrapper 학습 에이전트", 정보과학회 논문지: 소프트웨어 및 응용, 제29권 1-2호, pp.42-52, 2002.

[8] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[9] E. Riloff, "Automatically constructing a dictionary for information extraction tasks," Proc. AAAI-93, pp.811-816, 1993.

[10] J. Kim and D. Moldovan, "Acquisition of linguistic patterns for knowledge-based information extraction," *IEEE Trans. Know. and Data Eng.*, vol.7, no.5, pp.713-724, 1995.

[11] S. Soderland, "Learning information extraction rules for semi-structured and free text," *Machine Learning* vol.34 no.1-3, pp.233-272, 1999.

[12] M. Califf and R. Mooney, "Relational learning of pattern-match rules for information extraction," Working Papers of ACL-97 Workshop on Natural Language Learning, pp.9-15, 1997.

[13] D. Freitag, "Information extraction from HTML: Application of a general learning approach," Proc. AAAI-98, pp.517-523, 1998.

[14] C. Hsu and M. Dung, "Generating finite-state transducers for semi-structured data extraction from the Web," *J. of Inf. Sys.*, vol.23, no.8, pp.521-538, 1998.

[15] I. Muslea, S. Minton, and C. Knoblock, "A hierarchical approach to wrapper induction", Proc. Agents-99, pp.190-197, 1999.

[16] 김용기, 양재영, 최중민, "정보추출을 위한 학습 가능한 인터페이스 에이전트", 정보과학회 2001 가을 학술발표논문집(II), pp.61-63, 2001.

[17] L. Liu, C. Pu, and W. Han, "XWRAP: An XML-enabled wrapper construction system for Web information sources", Proc. 16th Int. Conf. on

Data Eng., pp.611-621, 2000.
[18] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann, 1999.

김 용 기
1999년 한양대학교 전자컴퓨터공학부 졸업(학사). 2002년 한양대학교 대학원 컴퓨터공학과 졸업(석사).

양 재 영
1997년 한양대학교 전자계산학과 졸업(학사). 1999년 한양대학교 대학원 컴퓨터공학과 졸업(석사). 2002년 한양대학교 대학원 컴퓨터공학과 졸업(박사). 2003년~현재, (주)오픈베이스 기술연구소 선임연구원. 관심분야는 인공지능, 지능형 에이전트, 기계학습, 정보추출

최 중 민
1984년 서울대학교 컴퓨터공학과 졸업(학사). 1986년 서울대학교 대학원 컴퓨터공학과 졸업(석사). 1993년 뉴욕주립대(Buffalo) 전산학과 졸업(박사). 1993년~1995년, 전자통신연구원(ETRI) 선임연구원. 1995년~현재, 한양대학교 컴퓨터공학과 교수. 관심분야는 인공지능, 에이전트, 웹 정보처리, 시맨틱 웹